CHAPTER 5

Recommender systems for e-learning: towards non-intrusive web mining

O.R. Zaïane
Department of Computing Science, University of Alberta, Edmonton AB, Canada.

Abstract

Recommender systems are software agents that recommend options to users. They are becoming very popular in e-commerce applications to recommend the online purchase of some products. These agents can be very useful in an e-learning environment to recommend actions, resources or simply links to follow. However, most approaches to develop these intelligent agents are based on data explicitly collected from users to build profiles such as rankings, opinions and the like. This can be considered intrusive and a distraction by online learners. In this chapter we discuss methods to build recommender systems for e-learning that are non-intrusive and true to the choices of users.

1 Introduction

Using electronic and digital means to deliver courses is a very common trend, whether for distance learning courses or even for typical face-to-face courses as enhancements or supplements to what is delivered in classrooms. Today, there are many comprehensive online course management systems, some commercial and some distributed under the open-source license agreement. Typical web-based learning environments, such as Virtual-U [1, 2] and WebCT [3], include course content delivery tools, synchronous and asynchronous conferencing systems, polling and quiz modules, virtual workspaces for sharing resources, white boards, grade reporting systems, logbooks, assignment submission components, etc. These systems are commonly used to provide supplementary course material for typical classroom lectures or to implement student-centered learning approaches such as problem-based or evidence-based learning allowing sustained interaction between learners.

E-learning is becoming a reality with more and more learners and educators becoming versed with technology. However, the majority of current course management systems and web-based learning systems are closed learning environments [4]. Courses and learning materials are fixed for all learners and do not adapt to individuals. The course content and its delivery is static. Only the organization of the online material is sometimes dynamic. While it is widely recognized that learners have different preferred learning styles and learning paces, very few course management systems accommodate any dynamic component that can follow learners' progress, build intelligent profiles and provide contextual individual help.

With the advent of the World Wide Web, research in e-learning and web-based delivery of course material has gained attention. But again, very few course management systems incorporate intelligent agents that would allow to personalize the course delivery system or individualize the suggested learning material. Similar applications in business have flourished. The personalization of online catalogues and virtual stores to enhance the online buying experience, limit churning, entice purchase and attract new customers has been the focus of many research studies in computing science, business, communication and psychology. Today, many such applications are commercialized and are becoming popular. For example, some web sites may present customized product catalogues that would contain mainly items that are similar to the customer's previous likings or with high probability to be purchased based on the customer's profile avoiding annoying the user with lengthy and useless lists. Other commercial sites may provide automatic suggestions for products based on the current purchase and the likings and 'tastes' of the customer and the collective choices. The typical example is the recommender agent for books: 'Other people who bought this book also bought books A and B. Would you like to purchase them too?' Similar recommender systems exist to suggest music CDs, movies, clothes, and even jokes.

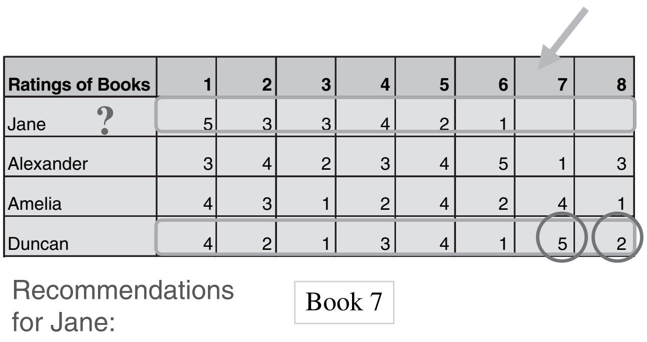
One natural question is, could such software agents be used in e-learning applications? The major difference between recommender systems in e-commerce and in e-learning is that the goals are different and are certainly not measured the same way. The obvious goal of recommender systems in e-commerce is to increase profit. Profit is tangible and can be measured in terms of amount of money, number of customers and customer loyalty. The money aspect is certainly the driver in these sophisticated implementations. In e-learning, the goals of a recommender system appear clear too: improving the learning. However, this goal is more subjective and is too subtle to be measured. Moreover, while querying a user and requesting opinions about products and rankings of products is somewhat acceptable in an e-commerce application to receive in return a better service, explicit surveys are considered intrusive and distracting in a learning environment.

In this chapter, we discuss the usefulness of recommender systems and how to design them as non-intrusive intelligent agents in the context of e-learning.

2 Collaborative filtering: how most systems work

Most recommender systems adopt the collaborative filtering approach [5, 6]. This approach is relatively simple and effective. Many variations of this algorithm exist but the principles are the same. User profiles are built based on past experience. The set of all user profiles constitutes the collective. When a user needs suggestions or a software agent is about to generate recommendations to a given user, the profile of the user is compared to the collective to find similar profiles. A selection from these similar profiles is used to produce recommendations. This is what is called collaborative filtering. Profiles collaborate to produce appropriate recommendations. The various approaches vary in the way profiles are built and modeled, in the way similarity between profiles is measured, in the way similar profiles are selected, and, finally, in the way recommendations are produced and ranked for presentation to the user. In general, user profiles are modeled in a vector space where each vector constitutes the ratings of a given user. Ratings are normalized numbers representing the opinion of the user vis-à-vis a given product (or object). Similarity between profiles could be measured in different ways using a variety of measures such as the cosine measure, correlations, Jaccard coefficient. Let us illustrate this approach with an example. Suppose we have an online recommender system for a library or a bookstore. Figure 1 shows the profiles for four people with regard to seven books. The matrix contains the ratings from the users for all books they read. These ratings are measured from 5 (high) to 1 (low). Empty cells mean an absence of opinion. We need to recommend a book to Jane. Using the correlation measure we find that Jane's ratings are highly correlated with Duncan's. Since Duncan read two books that Jane did not and book 7 was highly appreciated by Duncan, book 7 is the selected one to be recommended to Jane. This is obviously a imaginary situation. In reality more books could be selected (from different highly correlated profiles) and a ranking of these suggestions is necessary. Different criteria could be used for this ranking.

Collaborative filtering is commonly used in e-commerce applications to give suggestions. It is effective despite its simplicity. However, it has a significant drawback.



Ratings of Books	1	2	3	4	5	6	7	8
Jane ?	5	3	3	4	2	1		
Alexander	3	4	2	3	4	5	1	3
Amelia	4	3	1	2	4	2	4	1
Duncan	4	2	1	3	4	1	5	2

Recommendations for Jane: Book 7

Figure 1: An example of collaborative filtering.

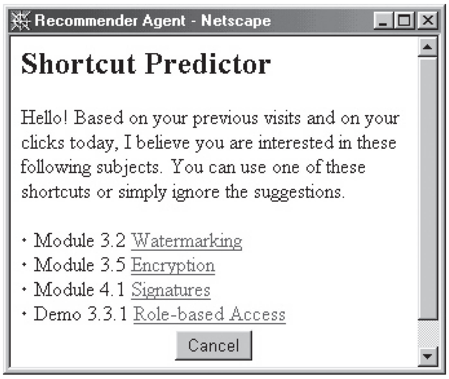
Building profiles necessitates the collection of opinions. The matrix of ratings needs to be large enough and trusted to be able to generate good recommendations. In reality, users do not take the effort to go back to the system to enter their opinions and if they do, the ratings they enter may not necessarily reflect their real opinion. In practice, the rating matrix ends up being very sparse and ineffective.

Can collaborative filtering be used in e-learning? Yes, but selectively. This is because without good and trusted ratings entered by the learners, the recommendations become useless and untrustworthy. Moreover, ratings may never be entered by the online learners as entering ratings could be considered intrusive and biased. To recommend learning activities, learning objects or simply online links to resources, it is better to use real past activities (history logs) by users as input for their profiles. This is not only non-intrusive but reflects the real choices of uses. Then again, collaborative filtering can still be used in an e-learning context. For example a software agent acting as an academic advisor recommending courses to take could rely on collaborative filtering if opinions of learners are collected after a course is taken. Constraints such as course prerequisites and requested curriculum paths in a program taken by a learner could be taken into account in the selection of courses to recommend. Another type of recommendation in the case of e-learning systems is the recommendation of research papers [7] or citations of papers [8].

3 Desired recommender systems in an online learning environment

In addition to the previously described course recommender system for which ratings are envisaged and intrusiveness is acceptable, other recommender systems can be foreseen in a web-based learning environment. These two types of recommender systems do not require any rating input. We refer to these as shortcut recommenders and action recommenders. Recommending shortcuts consists of providing to a learner a list of resources, typically Internet addresses (i.e. URL), that allows the user to directly 'jump' to the desired resource without having to look for it in the maze of hyperlinks in the sometimes complicated learning sites. In a site with a deep topology, a good and effective shortcut recommender system can save significant time and directly provide the needed learning resources. These resources, again, could be a web page, an applet, an image, a video, a simulation, or any learning object.

Figure 2A illustrates an example of recommendation for shortcuts. The software agent analyzes the click-stream of a user, builds a profile based on these initial visited pages, and compares this activity with actions in sessions of other users. This allows the prediction of pages to be visited by the current user. A model can be built based solely on previous history using the web access logs, or complemented, as we shall see later, by the connectivity in the site and even the content of the pages. In the example of Fig. 2A, the content is taken into account and the subject of interest in predicted before text web pages (modules) or simulations are suggested in a database security course web site.

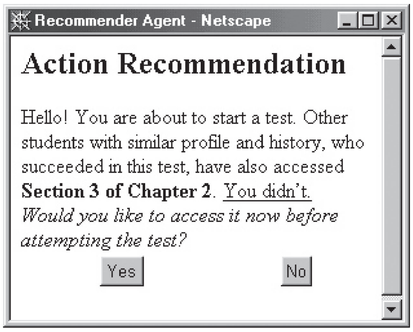


Shortcut Predictor

Hello! Based on your previous visits and on your clicks today, I believe you are interested in these following subjects. You can use one of these shortcuts or simply ignore the suggestions.

- Module 3.2 Watermarking
- Module 3.5 Encryption
- Module 4.1 Signatures
- Demo 3.3.1 Role-based Access

(A)



Action Recommendation

Hello! You are about to start a test. Other students with similar profile and history, who succeeded in this test, have also accessed **Section 3 of Chapter 2**. You didn't. *Would you like to access it now before attempting the test?*

(B)

Figure 2: Recommender system suggesting (A) a shortcut and (B) an action.

An action recommender system also suggests resources or learning objects, but often acts upon triggers. Typically, in web mining and web analytics, the canonical event is a click or request of a page. In e-learning the canonical event is an action such as the completion of polling, the answering of a quiz question, the execution of a simulation, the request of a text chapter for reading. The granularity and resolution of these actions could depend on a concept hierarchy previously defined by the educator. The recommendation of actions is triggered by other actions. The start of a test, the answering of a quiz question, the successful termination of a simulation, the incorrect response at an assessment stage, or the request of a module of a new chapter, all constitute good examples for actions that could trigger the action recommender system. Once the software agent is activated for suggestions, the recommender system not only considers past activity but other semantically significant information about successes and failures of the learner to build a profile. The profile is matched with a subset of other profiles of successful learners and constrained with desirable behavior imposed by the educator. Figure 2B shows an example of an action recommender activated by the request to start an online test. The action 'accessing Section 3 of Chapter 2' is suggested because the action is taken before the test by successful learners at the concerned test and probably not by the less successful ones.

4 Non-intrusive methods for recommendation

Since we do not have ratings for course material, and it is not desired to survey learners about their evaluations of the different learning objects and modules since we opt for a non-intrusive approach, collaborative filtering is not applicable for devising an accurate recommender agent for e-learning activities. We will build our models only based on logged online activities.

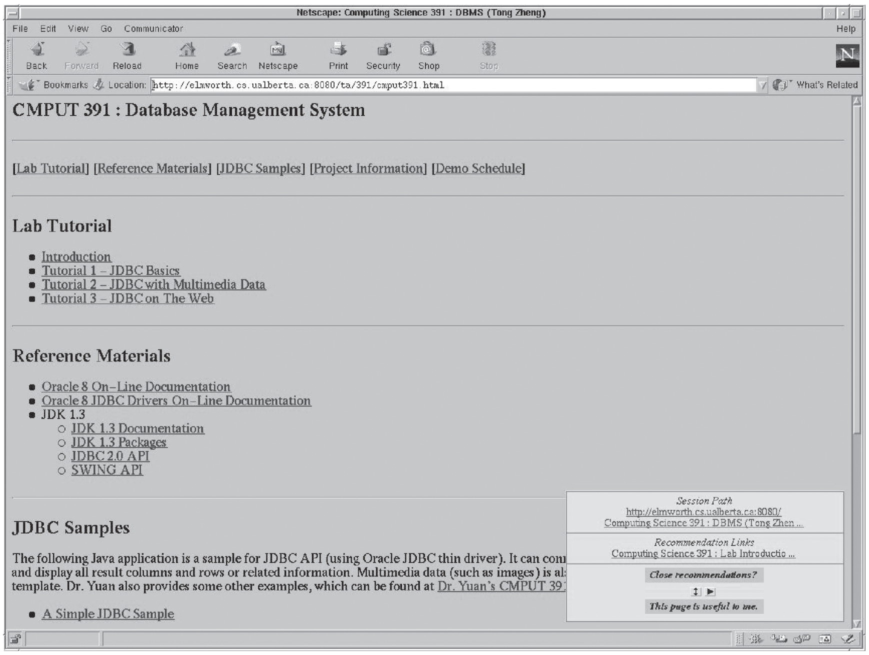
In recent years there has been an increasing interest in applying web usage mining techniques to build web recommender systems [9–12]. Web usage recommender systems take web server access logs as input, and make use of data mining techniques such as *association rule* and *clustering* to extract implicit, and potentially useful navigational patterns, which are then used to provide recommendations. Web server access logs record user browsing history, which contains plenty of hidden information regarding users and their navigation. They could, therefore, be a good alternative to the explicit user rating or feedback in deriving user models. In web usage recommender systems, navigational patterns are generally derived as an offline process. A significant cleaning and transformation phase needs to take place so as to prepare the information for data mining algorithms [13, 14].

4.1 E-learning recommender with association rules

Association rules are one of the typical rule patterns that data mining tools aim at discovering. They are very useful in many application domains, but are mainly applied in the business world as in market-basket analysis. In a transactional database where each transaction is a set of items bought together, association rules are rules associating items that are frequently bought together. A rule consists of an antecedent (left-hand side) and a consequent (right-hand side). Example: $I_1, I_2, \ldots, I_n \Rightarrow I_\alpha, I_\beta, \ldots, I_\gamma$. The intersection between the antecedent and the consequent is empty. If items in the antecedent are bought then there is a probability that the items in the consequent would be bought as well at the same time. An efficient algorithm to discover these association rules was first introduced in [15]. The algorithm constructs a candidate set of frequent itemsets of length k, counts the number of occurrences, keeps only the frequent ones, then constructs a candidate set of itemsets of length $k + 1$ from the frequent itemsets of smaller length. It continues iteratively until no candidate itemset can be constructed. In other words, every subset of a frequent itemset must also be frequent. The rules are then generated from the frequent itemsets with probabilities attached to them indicating the likelihood (called support) that the association occurs.

This idea of association rules is often used to train a recommender agent to build a model representing the web page access behavior or associations between online learning activities.

A recommender system suggests possible actions or web resources based on its understanding of the user's access. To do so the entries in the web log have to be translated into either known actions (i.e. learning activities such as accessing a course notes module, posting a message on the forum, doing a test, trying a simulation) or URLs of a web resource. This mapping is a significant processing phase that in itself presents a considerable challenge [16, 17]. Moreover, these identified actions and URLs are grouped into sessions which is yet another difficult and delicate task [13]. These sessions are then modeled into transactions as sets of actions and URLs. The association rule mining technique is applied on such transactions to discover associations between actions, associations between URLs and associations between actions and URLs, as well as associations between sequences of


Netscape: Computing Science 391 : DBMS (Tong Zheng)
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Shop Stop
Bookmarks Location: http://elmworth.cs.ualberta.ca:8080/ta/391/cmput391.html
What's Related

CMPUT 391 : Database Management System

[Lab Tutorial] [Reference Materials] [JDBC Samples] [Project Information] [Demo Schedule]

Lab Tutorial

- Introduction
- Tutorial 1 – JDBC Basics
- Tutorial 2 – JDBC with Multimedia Data
- Tutorial 3 – JDBC on The Web

Reference Materials

- Oracle 8 On-Line Documentation
- Oracle 8 JDBC Drivers On-Line Documentation
- JDK 1.3
  - JDK 1.3 Documentation
  - JDK 1.3 Packages
  - JDBC 2.0 API
  - SWING API

JDBC Samples

The following Java application is a sample for JDBC API (using Oracle JDBC thin driver). It can com
and display all result columns and rows or related information. Multimedia data (such as images) is al:
template. Dr. Yuan also provides some other examples, which can be found at Dr. Yuan's CMPUT 39

- A Simple JDBC Sample

Session Path
http://elmworth.cs.ualberta.ca:8080/
Computing Science 391 : DBMS (Tong Zhen ...

Recommendation Links
Computing Science 391 : Lab Introductio ...

Figure 3: Recommender system using reference feedback.

actions and/or URLs. This process usually leads to a very large number of association rules. A sophisticated pruning and filtering phase is required [18]. When the recommender agent is activated, the association rules are consulted to check for matches between the triggering event, or sequence of events, with the rule antecedents. When a match is found, the consequent of the rule is suggested. If more matches are found, the suggestions are ranked and only a small set (highest ranked) is displayed. The ranking is often based on some interestingness measures such as confidence. However, in some cases input from the user can interactively improve these measures. Figure 3 shows a prototype of a recommender system for pages in an online database course at the University of Alberta where the user rates the recommendation by reference feedback. This feedback adjusts the weights on the rules used for future recommendations.

4.2 A model with clustering

While using association rule mining is popular in building non-intrusive recommender systems [19, 20], clustering is also an accepted approach although not as effective. Indeed there are myriad clustering techniques not all as capable [21]. Clustering consists of grouping objects based on similarity or dissimilarity (i.e. distance function) by maximizing similarity between objects in a group and maximizing dissimilarity between objects in different groups. In the context of recommender systems in e-learning, the objects are either pages with their content (or learning objects) or sequences of clicks representing navigational behaviors. In both cases, page content or sequence of clicks, are modeled in a vector space. When the recommendation agent is activated, the current page (action or learning object) or the current sequence of event is compared to its neighbors in its cluster. The k-nearest neighbors are simply recommended to the user.

5 Hybrid methods for recommendations

Most non-intrusive recommendation agents rely either on access usage only, on content of visited pages or on connectivity between the visited pages. Very few combine these information channels. We have proposed a model to combine all these information channels [22] and showed the effectiveness in different contexts [23].

A few combined or hybrid web recommender systems have been proposed in the literature [24, 25]. The work in [24] adopts a clustering technique to obtain both site usage and site content profiles in the offline phase. In the online phase, a recommendation set is generated by matching the current active session and all usage profiles. Similarly, another recommendation set is generated by matching the current active session and all content profiles. Finally, a set of pages with the maximum recommendation value across the two recommendation sets is presented as recommendation. This is called a *weighted* hybridization method [26]. In [25], Nakagawa and Mobasher use association rule mining, sequential pattern mining, and contiguous sequential mining to generate three types of navigational patterns in the offline phase. In the online phase, recommendation sets are selected from the different navigational models, based on a localized degree of hyperlink connectivity with respect to a user's current location within the site. This is called a *switching* hybridization method [26].

5.1 Architecture of a hybrid recommender system

As most web usage recommender systems, our system is composed of two modules: an offline component, which pre-processes data to generate users' navigational models, and an online component which is a real-time recommendation engine. Figure 4 depicts the general architecture of our system.

Entries in a web server log are used to identify users and visit sessions, while web pages or resources in the site are clustered based on their content. These clusters of web documents are used to scrutinize the discovered web sessions in order to identify what we call *missions* [23]. A mission is a sub-session with a consistent goal. These missions are in turn clustered to generate navigational patterns, and augmented with their linked neighborhood and ranked based on resource connectivity, using the *hub* and *authority* idea [27]. These new clusters (i.e. augmented navigational patterns) are provided to the recommendation engine. When a visitor starts a new session, the session is matched with these clusters to generate a recommendation list.

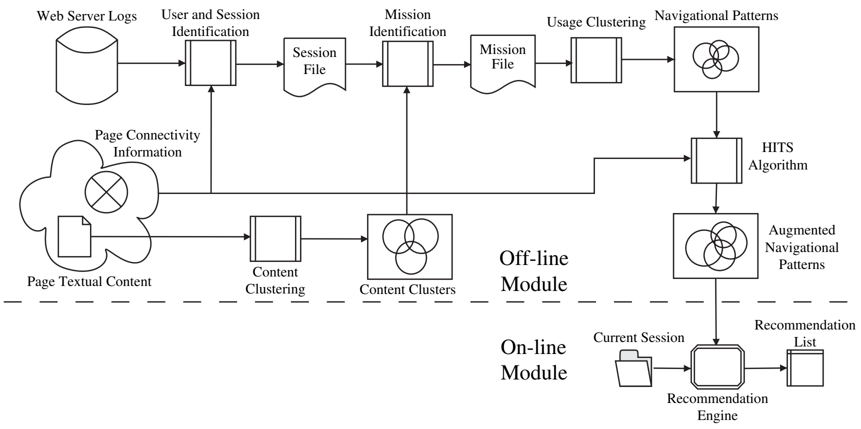


Figure 4: System architecture with all three channels available.

5.2 User and visit session identification

A web log is a text file which records information regarding users' requests to a web server. A typical web log entry contains a client address, the requested date address, a time-stamp, and other related information.

For any web access log data, several pre-processing tasks have to be performed before applying data mining techniques for pattern discovery. The pre-processing tasks usually include user identification, visit session identification, and transaction identification. We use similar pre-processing techniques as in [14] to identify individual users and sessions. To sessionize log entries, we chose an idle time of 30 min. Session-based access logs, however, like for a password protected e-learning system, logs have entries identified by users since the users have to login, and sessions are already identified since users may also have to logout.

5.3 Visit mission identification

The last data pre-processing step proposed in [14] is transaction identification, which divides individual visit sessions into transactions. Two transaction identification approaches are proposed: *reference length* approach and *maximal forward reference* approach, both of which have been widely applied in web mining. Rather than dividing sessions into arbitrary transactions, we identify sub-sessions with coherent information needs. We call these sub-sessions missions. We assume that a visitor may have different information needs to fulfill during a visit, but we make no assumption on the sequence in which these needs are fulfilled. In the case of transactions in [14], it is assumed that one information need is fulfilled after the other. A mission would model a sub-session related to one of these information needs, and would allow overlap between missions, which would represent a concurrent search in the site.

Now how do we identify missions? The first approach we proposed to identify missions is based on web content [22]. While in the transaction-based model, pages are labeled as *content* pages and *auxiliary* pages, and a transaction is simply a sequence of auxiliary pages that ends with a content page, in the mission-based model we propose, the identified sequence is based on the real content of pages. Indeed, a content page in the transaction-based model is identified simply based on the time spent on that page, or on backtracking in the visitor's navigation. We argue that missions could better model users' navigational behavior than transactions. In our model, users visit a web site with concurrent goals, i.e. different information needs. For example, a user could fulfill two goals in a visit session: a, b, c, d, in which pages a and c contribute to one goal, while pages b and d contribute to the other. Since pages related to a given goal in a visit session are generally supposed to be content coherent, whether they are neighboring each other or not, we use page content to identify missions within a visit session.

All web site pages are clustered based on their content, and these clusters are used to identify content coherent clicks in a session. Let us give an example to illustrate this point. Suppose the text clustering algorithm groups web pages a, b, c, and e, web pages a, b, c, and f, and web pages a, c and d into three different content clusters (please note that our text clustering algorithm is a soft clustering one, which allows a web page to be clustered into several clusters). Then for a visit session: a, b, c, d, e, f, our system identifies three missions as follows: mission 1: (a, b, c, e); mission 2: (a, b, c, f); and mission 3: (a, c, d). As seen in this example, mission identification in our system is different from transaction identification in that we can group web pages into one mission even if they are not sequential in a visit session. We can see that our mission-based model subsumes the transaction-based model, since missions could become transactions if visitors fulfill their information needs sequentially.

To cluster web pages based on their content, we use a modified version of the DC-tree algorithm [28]. Originally, the DC-tree algorithm was a hard clustering approach, prohibiting overlap of clusters. We modified the algorithm to allow web pages to belong to different clusters. Indeed, some web pages could cover different topics at the same time. In the algorithm, each web page is represented as a keyword vector, and organized in a tree structure called the DC-tree. The algorithm does not require the number of clusters to discover as a constraint, but allows the definition of cluster sizes. This was the appealing property which made us select the algorithm. Indeed, we do not want either too large or too small content cluster sizes. Very large clusters cannot help capture missions from sessions, while very small clusters may break potentially useful relations between pages in sessions.

The missions we extracted and clustered to generate navigational patterns are primarily based on the sessions from the web server logs. These sessions exclusively represent web pages or resources that were visited. It is conceivable that there are other resources not yet visited, even though they are relevant and could be interesting to have in the recommendation list. Such resources could be, for instance, newly added web pages or pages that have links to them not evidently presented due to bad design. Thus, these pages or resources are never presented in the missions previously discovered. Since the navigational patterns, represented by the clusters of pages in the missions, are used by the recommendation engine, we need to provide an opportunity for these rarely visited or newly added pages to be included in the clusters. Otherwise, they would never be recommended. To alleviate this problem, our general system model expands the clusters to include the connected neighborhood of every page in a mission cluster. The local neighborhood of a page, obtained by tracing a small number of links from the originating page, is a good approximation to the 'semantic neighborhood' of the page [29]. In our case, the connected neighborhood of a page *p* is the set of all the pages directly linked from *p* and having similar content of *p*, and all the pages that directly link to *p* also with similar content. In detail, this approach of expanding the neighborhood is performed as follows: we consider each previously discovered navigational pattern (i.e. a cluster of content coherent and visitation cohesive missions) as a set of seeds. Each seed is supplemented with pages it links to and pages that link to it as well as having similar content. The result is what is called a connectivity graph which now represents our augmented navigational pattern. This process of obtaining the connectivity graph is similar to the process used by the HITS algorithm [27] to find the authority and hub pages for a given topic. The difference is that we do not consider a given topic, but start from a mission cluster as our set of seeds.

Moreover, it was shown in [30] that HITS, using pure connectivity analysis, introduces a problem known as 'topic drift'. We eliminate this problem in our case by computing relevance weights of all supplementary pages. The relevance weight of a page equals the similarity of the page content to the corresponding mission cluster, which is represented by the cosine normalization of web pages and mission clusters keyword vectors. We then prune nodes with relevance weights below a threshold from the connectivity graph. For simplicity, we use *median weight* (i.e. the median of all relevance weights) as the pruning threshold [30]. The pruning process avoids augmenting the navigational patterns with pages that focus on other topics and guarantees that the augmented patterns are still coherent and focused. After expanding and pruning the clusters representing the navigational patterns, we also augment the keyword vectors that label the clusters. The new keyword vectors that represent the augmented navigational patterns have also the terms extracted from the content of the augmented pages.

We take advantage of the built connectivity graph by cluster to apply the HITS algorithm in order to identify the authority and hub pages within a given cluster. These measures of authority and hub allow us to rank the pages within the cluster. This is important because at real time during the recommendation, it is crucial to rank recommendations, especially if they are numerous. Long recommendation lists are not advisable.

Authority and hub are mutually reinforcing [27] concepts. Indeed, a good authority is a page pointed to by many good hub pages, and a good hub is a page that points to many good authority pages. Since we would like to be able to recommend pages newly added to the site, in our framework, we consider only the hub measure. This is because a newly added page would be unlikely to be a good authoritative page, since not many pages are linking to it. However, a good new page would probably link to many authority pages; it would, therefore, have the chance to be a good hub page. Consequently, we use the hub value to rank the candidate recommendation pages in the online module.

When a visitor starts a new session in the web site, we identify the navigation pattern after a few clicks and try to match on-the-fly with already captured navigational patterns. If they were matched, we recommend the most relevant pages in the matched cluster.

5.4 Evaluating hybrid recommenders

To demonstrate the effectiveness of the hybrid approach using all three information channels, usage, linkage and content, to build an effective recommender system for shortcuts and activities, we present herein a test using a generic web access log from the computing science department web site. The collected web access log is divided into months each averaging about 200,000 hits accessing on average more than 40,000 unique pages with on average 150,000 hyperlinks between them. Successively, each month is used to train a recommender system and the subsequent month is used to evaluate it.

We devised a methodology to assess the recommender system [23]. The evaluation is based on *recommendation accuracy* and *shortcut gain*. The recommendation accuracy ($RA = [\Sigma_s | \cup_p (T(p) \cap R(p))|/| \cup p\ R(p)|]/S$ where S is the visit session in the log, s is a given session, p is a page in s, $R(p)$ is the generated recommendation list from p, and $T(p)$ the tail (suffix) of s after p) is the ratio of correct recommendations among all recommendations, and the correct recommendation is the one that appears in the suffix of a session from which the prefix triggers the recommendation. The shortcut gain ($SG = [\Sigma_s |s| - |s'|/|s|]/S$ where s is the original session and s' is the improved session after the jump) measures how many clicks the recommendation allows users to save if the recommendation is followed. In addition, we compute the *coverage* of a recommender system, which measures the ability of a system to produce all pages that are likely to be visited by users ($RC = [\Sigma_s | \cup_p (T(p) \cap R(p))|/| \cup_p T(p)|]/S$). The concept is similar to what is called *recall* in information retrieval.

We first evaluated the performance of our system on the UofA CS web server dataset. Our first experiment varies the coverage to see the tendency of the recommendation accuracy, as depicted in Fig. 5A. For the purpose of comparison, we also implement an *association rule recommender system*, the most commonly used approach for web mining based recommender systems, and record its performance in the same figure. As expected, the accuracy decreases when we increase coverage. However, our system was consistently superior to the association rule system by at least 30%.

We next varied the coverage to test the shortcut gain, both with our system and with the association rule system, as illustrated in Fig. 5B.

From Fig. 5B, we can see that in the low boundary where the coverage is lower than 8%, the shortcut gain of our system is close to that of the association rule system. With the increase of the coverage, however, our system can achieve

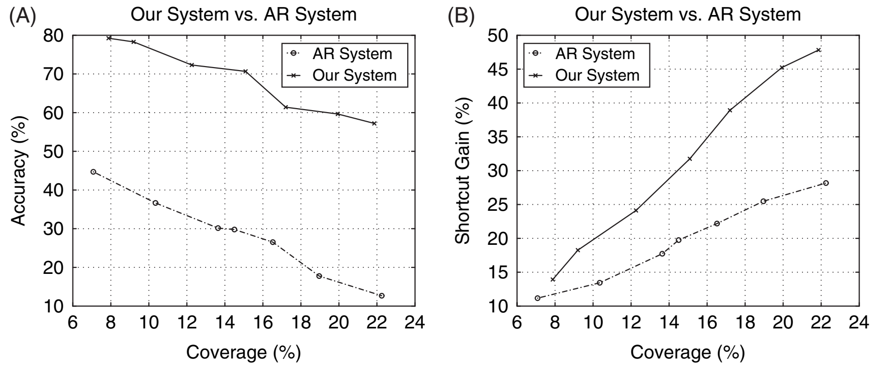


Figure 5: Performance comparison: our system vs. association rule Recommender System. (A) Recommendation accuracy; (B) shortcut gain.

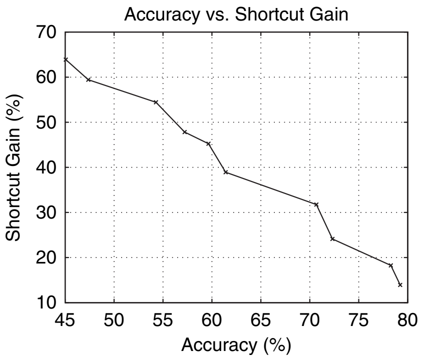


Figure 6: Accuracy vs. shortcut gain.

an increasingly superior shortcut gain than the latter, although the performance of both systems continues to improve.

Figure 6 depicts the relationship of recommendation accuracy and shortcut gain in our system. It shows that recommendation accuracy is inversely proportional to the shortcut gain. Our study draws the same conclusion from the association rule recommender system. We argue that this is an important property of a usage-based web recommender system, and therefore, how to adjust and balance between the accuracy and shortcut gain for a web recommender system to achieve the maximum benefit is a question that should be investigated. Some web sites, e.g. those with high link density, may favor a recommender system with high accuracy, while some others may favor a system with high shortcut gain.

In the above tests, the three distinctive information channels – usage, content, and structure – are provided to and used in our system. In a second battery of tests we measured the effect of the individual information channels. We first compared three recommender systems, one using all channels, one using only usage and

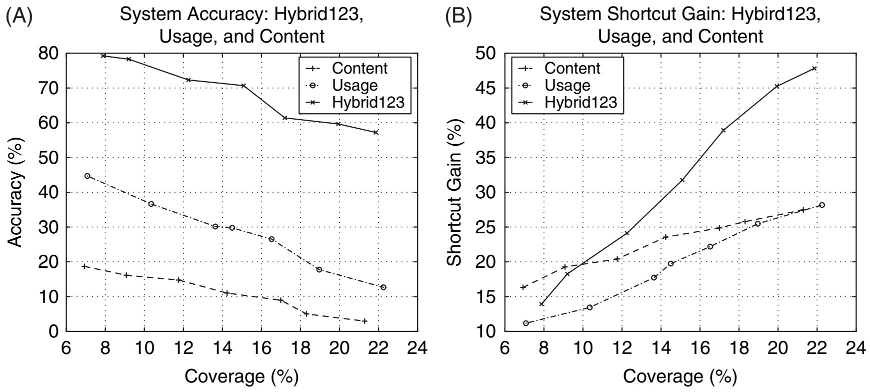


Figure 7: Hybrid123, Usage, and Content. (A) Recommendation accuracy; (B) shortcut gain.

one using only content. We refer to our recommender using the three channels as *Hybrid123*. For this comparison, we implemented an association rule-based usage recommender system as in the previous tests (referred to as *Usage*), as well as a web recommender system based purely on content similarity (referred to as *Content*). The Usage system works as follows: an efficient association rule algorithm [31] is applied to the access logs to generate a set of rules. Whenever the pages in the antecedent of an rule have appeared in the user's current session, those pages in its consequence are recommended. For the Content system, all pages in the web site are extracted and grouped into clusters solely based on their textual content similarity, using a high-quality content clustering algorithm [32]. If one or more pages in a cluster have been visited, the pages in the same clusters are selected to be recommended. The recommendation accuracy and shortcut gain of the three systems are depicted in Fig. 7. In the experiment, we varied the coverage to test the trend and consistency of the system quality.

Figure 7A shows the recommendation accuracy of the three contenders. As expected, the accuracy decreases when we increase coverage. However, Hybrid123 is consistently the best among the three systems, superior to Usage by at least 30% – while Usage always ranks second.

From Fig. 7B, we can see that in the low boundary, the shortcut gain of Content is the best of the three systems, and the other two are close. With the increase of coverage, the shortcut gain of all three systems continues to improve, but in different degrees. Hybrid123 can achieve an increasingly superior shortcut gain to that of Usage, and exceeds Content after coverage is larger than about 10%. The major reason that the shortcut gain improvement of Content is lowest is that with the increase of coverage, more and more pages containing only the same terms, but without any logical relationship are selected to be recommended.

In our next experiment, we illustrate the advantage of incorporating web content and web structure information in our system. To do so, we implemented additional two recommender prototypes. The first is similar to Hybrid123 but is stripped

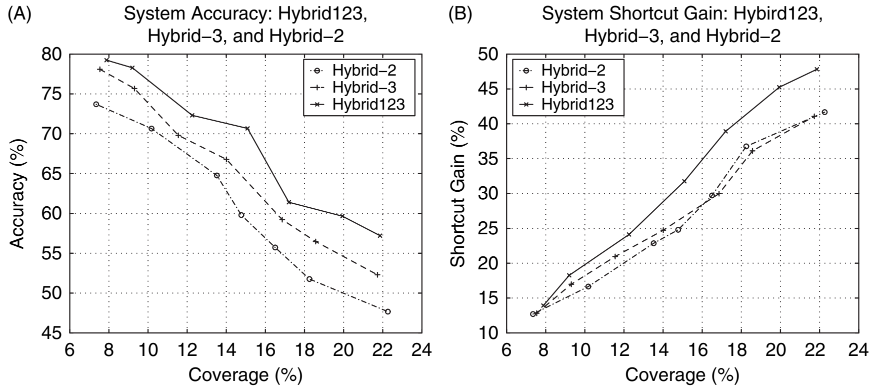


Figure 8: Hybrid123, Hybrid-3, and Hybrid-2. (A) Recommendation accuracy; (B) Shortcut gain.

from its connectivity information channel. That is, we do not make use of linkage information to augment and improve the navigational patterns built on usage and content information. We name this hybrid system *Hybrid-3*. The second is also a similar system to Hybrid123 but does not make use of content information to identify a mission. Rather, the navigational patterns in the system is built upon traditional transactions identified according to the approach in [14]. Then, the patterns are improved with structure information, as with Hybrid123. This hybrid system is called *Hybrid-2*. The recommendation accuracy and shortcut gain of the three systems are depicted in Fig. 8.

Figure 8A shows the recommendation accuracy of the three systems. The consistently best performance of Hybrid123 illustrates the validity of content and connectivity information to improve recommendations in our hybrid system, and also indicates that content is more useful for recommendation accuracy improvement. The shortcut gains of the three systems are depicted in Fig. 8B. We notice that with the increase of coverage, Hybrid123 can achieve an increasingly superior shortcut gain compared to both Hybrid-3 and Hybrid-2, while the two systems keep similar performance in terms of shortcut gain. This figure verifies our justification for using distinctive information channels in building a hybrid recommender system, and shows that content and structure information make a similar contribution to the improvement in shortcut gain in our system.

In summary, this experiment shows that our system can significantly improve the quality of web site recommendation by combining the three information channels, while each channel included contributes to this improvement.

6 Conclusion

Most recommender systems rely on ratings from users. We argue that this is intrusive and quickly loses effectiveness in an e-learning setting. Non-intrusive methods relying on web access logs are true to the real learning behavior of a user. A web usage-based recommender system which focuses solely on access history has its own problems:

- *Incomplete information problem*: One restriction with web server logs is that the information in them is very limited. Thus, a number of heuristic assumptions have to be made to identify individual users, visit sessions, and transactions in order to apply any data mining algorithm. One such assumption is that user information needs are fulfilled sequentially while in practice they are often in parallel.
- *Incorrect information problem*: When web site visitors are lost, the clicks made by them are recorded in the log, and may mislead future recommendations. This becomes more problematic when a web site is badly designed and more people end up visiting unsolicited pages, making them seem popular.
- *Persistence problem*: When new pages are added to a web site, because they have not been visited yet, the recommender system may not recommend them, even though they could be relevant. Moreover, the more a page is recommended, the more it may be visited, thus making it look popular and boost its candidacy for future recommendation.

To address these problems, we proposed a hybrid web recommender system, which attempts to use three information channels to model user navigational behavior: web access logs, the structure of a visited web site, and the content of visited web pages. In particular, the approach uses the terms within visited web pages to partition visit sessions into overlapping sub-sessions, called missions. Our preliminary experiments demonstrate that combining the different information channels has great potential for improving the quality of non-intrusive recommendation.

References

[1] Virtual-U, http://www.elearningsolutionsinc.com/productsvu.htm

[2] Groeneboer, C., Stockley, D. & Calvert, T., Virtual-U: a collaborative model for online learning environments. *Second Int. Conf. on Computer Support for Collaborative Learning*, Toronto, Canada, 1997.

[3] WebCT, http://www.webct.com

[4] Tang, T. & McCalla, G., Smart recommendation for an evolving e-learning system: Architecture and experiment. *International Journal on E-Learning*, **4(1)**, pp. 105–129, 2005.

[5] Chee, S., Han, J. & Wang, K., RecTree: an efficient collaborative filtering method. *3rd Int. Conf. on Data Warehousing and Knowledge Discovery (DAWAK 2001)*, Springer Verlag: Munich, Germany, LNCS 2114, pp. 141–151, 2001.

[6] Melville, P., Mooney, R. & Nagarajan, R., Content-boosted collaborative filtering for improved recommendation. *18th National Conf. on Artificial Intelligence (AAAI'02)*, Edmonton, Canada, pp. 187–192, 2002.

[7] Tang, T.Y. & McCalla, G., Towards pedagogy-oriented paper recommendation and adaptive annotations for a web-based learning system. *Workshop on Knowledge Representation and Automated Reasoning for E-Learning Systems (in IJCAI'03)*, Acapulco, Mexico, 2003.
[8] McNee, S., Alberta, I., Cosley, D., Gopalkrishnan, P., Lam, S., Rashid, A., Konstan, J. & Rield, J., On the recommending of citations for research papers. *ACM Int. Conf. on Computer Supported Collaborative Work (CSCW'02)*, pp. 116–125, 2002.
[9] Srivastava, J., Cooley, R., Deshpande, M. & Tan, P.N., Web usage mining: Discovery and applications of usage patterns from web data. *SIGKDD Explorations*, **1(2)**, pp. 12–23, 2000.
[10] Fu, X., Budzik, J. & Hammond, K.J., Mining navigation history for recommendation. *Intelligent User Interfaces*, pp. 106–112, 2000.
[11] Lin, C., Alvarez, S. & Ruiz, C., Collaborative recommendation via adaptive association rule mining, 2000.
[12] Yi-Hung Wu, A.L.C., Yong-Chuan Chen, Enabling personalized recommendation on the web based on user interests and behaviors. *11th Int. Workshop on Research Issues in Data Engineering*, 2001.
[13] Zaïane, O.R., Web usage mining for a better web-based learning environment. *Proc. of Conf. on Advanced Technology for Education*, Banff, AB, pp. 60–64, 2001.
[14] Cooley, R., Mobasher, B. & Srivastava, J., Data preparation for mining world wide web browsing patterns. *Knowledge and Information Systems*, **1(1)**, pp. 5–32, 1999.
[15] Agrawal, R., Imielinski, T. & Swami, A., Mining association rules between sets of items in large databases. *Proc. 1993 ACM-SIGMOD Int. Conf. Management of Data*, Washington, DC, pp. 207–216, 1993.
[16] Zaïane, O.R., Xin, M. & Han, J., Discovering web access patterns and trends by applying OLAP and data mining technology on web logs. *Proc. Advances in Digital Libraries ADL'98*, Santa Barbara, CA, USA, pp. 19–29, 1998.
[17] Zaïane, O.R. & Luo, J., Towards evaluating learners' behaviour in a web-based distance learning environment. *Proc. of IEEE Int. Conf. on Advanced Learning Technologies (ICALT01)*, Madison, WI, pp. 357–360, 2001.
[18] Zaïane, O.R., Building a recommender agent for e-learning systems. *Int. Conf. on Computers in Education*, 2001.
[19] Lin, W., Alvarez, S.A. & Ruiz, C., Efficient adaptive-support association rule mining for recommender systems. *Data Mining and Knowledge Discovery*, **6(1)**, pp. 83–105, 2002.
[20] Spertus, E. & Stein, L.A., A hyperlink-based recommender system written in squeal. *Proc. ACM CIKM'98 Workshop on Web Information and Data Management (WIDM'98)*, Washington, DC, pp. 1–4, 1998.
[21] Zaïane, O.R., Foss, A., Lee, C.H. & Wang, W., On data clustering analysis: Scalability, constraints and validation. *Sixth Pacific-Asia Conf. on Knowledge Discovery and Data Mining (PAKDD'02)*, Taipei, Taiwan, Lecture Notes in AI (LNAI 2336), pp. 28–39, 2002.

[22] Li, J. & Zaïane, O.R., Combining usage, content, and structure data to improve web site recommendation. *5th Int. Conf. on Electronic Commerce and Web Technologies (EC-Web 2004)*, 2004.

[23] Li, J. & Zaïane, O.R., Using distinct information channels for mission-based web recommender system. *Sixth ACM SIGKDD Workshop on Webmining and Web Analysis (WebKDD 2004)*, Seattle, WA, USA, pp. 35–46, 2004.

[24] Mobasher, B., Dai, H., Luo, T., Sun, Y. & Zhu, J., Integrating web usage and content mining for more effective personalization. *EC-Web*, pp. 165–176, 2000.

[25] Nakagawa, M. & Mobasher, B., A hybrid web personalization model based on site connectivity. *Fifth WebKDD Workshop*, pp. 59–70, 2003.

[26] Burke, R., Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted* Interaction, 2002.

[27] Kleinberg, J.M., Authoritative sources in a hyperlinked environment. *Journal of the ACM*, **46(5)**, pp. 604–632, 1999.

[28] Wong, W. & Fu, A., Incremental document clustering for web page classification, 2000.

[29] Lieberman, H., Autonomous interface agents. *Proc. of the ACM Conf. on Computers and Human Interface, CHI-97*, Atlanta, Georgia, 1997.

[30] Bharat, K. & Henzinger, M.R., Improved algorithms for topic distillation in a hyperlinked environment. *Proc. of SIGIR-98, 21st ACM Int. Conf. on Research and Development in Information Retrieval*, Melbourne, AU, pp. 104–111, 1998.

[31] Chen, M.S., Park, J.S. & Yu, P., Efficient data mining for path traversal patterns. *IEEE Transactions on Knowledge and Data Engineering*, **10(2)**, pp. 209–221, 1998.

[32] Pantel, P. & Lin, D., Document clustering with committees. *The 25th Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval*, 2002.