Effective Page Recommendation Algorithms Based on Distributed Learning Automata and Weighted Association Rules

R. Forsati[1*], M. R. Meybodi[2]

[1] Department of Computer Engineering, Islamic Azad University, Karaj Branch, Karaj, Iran

[2] Department of Computer Engineering, Amirkabir University of Technology, Tehran, Iran

[*]Corresponding author.

[1]*Rana Forsati, Msc

Department of Computer Engineering

Karaj Azad University, Karaj , Iran.

Tel: +98(21)22074369

Fax: +98(21)22074369

Email: forsati@kiau.ac.ir

[2]Mohammad Reza Meybodi, PhD

Department of Computer Engineering

Amirkabir University of Technology, Tehran, Iran.

Email: mmeybodi@aut.ac.ir

Abstract

Different efforts have been done to address the problem of information overload on the Internet. Recommender systems aim at directing users through this information space, toward the resources that best meet their needs and interests by extracting knowledge from the previous users' interactions. In this paper we propose three algorithms to solve the web page recommendation problem. In our first algorithm, we use distributed learning automata to learn the behavior of previous users' and recommend pages to the current user based on learned patterns. By introducing a novel Weighted Association Rule mining algorithm, we present our second algorithm for recommendation purpose. Also, a novel method is proposed to pure the current session window. One of the challenging problems in recommendation systems is dealing with unvisited or newly added pages. By considering this problem and improving the efficiency of first two algorithms we present a hybrid algorithm based on distributed learning automata and proposed weighted association rule mining algorithm. In the hybrid algorithm we employ the HITS algorithm to extend the recommendation set. Our experiments on real data set show that hybrid algorithm performs better than the other algorithms we compared to and, at the same time, it is less complex than other proposed algorithms with respect to memory usage and computational cost too.

Keywords:

Personalization, Machine Learning, Learning Automata, Web Mining

1. Introduction

World Wide Web has been growing rapidly in recent years and this has resulted in a huge volume of hyperlinked documents which contain no logical organization. Currently, Google indexes more than 3 billions web pages in the world which this number increases with the rate of 7.3 million pages per day. The massive influx of information onto World Wide Web has facilitated user, not only information retrieval, but also knowledge discovery. However, users are provided with more information and service options, it has become more difficult for them to find the "right" or "interesting" information, the problem commonly known as information overload due to the fact of significantly increasing and rapidly expanding growth in amount of information on the web.

Personalization techniques [1] are alternative, user-centric, promising approaches to tackle the problem of information overload by adapting the content and structure of websites to the needs of the users by taking advantage of the knowledge acquired from the analysis of the users' access behaviors. Personalization aims to provide users with what they want or need without explicitly ask them from it [2]. Typically, personalization focuses on the processes of identifying web users or objects, collecting information with respect to users' preference or interests as well as adapting its service to satisfy the users' needs. In short, web personalization can be used to provide better quality service and application of web to users during their browsing period. The actions can be made by highlighting the hyperlinks, inserting new hyperlinks that seem to be of interest for the current user dynamically, and the creation of new index pages.

Numerous approaches are introduced for personalization system which can be categorized into two major groups, which are content-based filtering agents and collaborative filtering systems [3]. In both cases a user model is created from data gathered explicitly and/or implicitly about user interests and used to recommend a set (referred to as the recommendation set) of items deemed to be of interest to the user. The most common form of implicit data about user interests takes the form of item ratings. Each item builds a model of user preferences using these content descriptions and the rating data of the user. The model is then used to predict the likelihood of items, not

currently viewed by the user, being of interest to the user. The most likely items of interest to the user constitute the recommendation set.

The main limitation of content-based filtering is the lack of diversity in the recommendations. As the recommendations are based only on items previously rated by the active user, the items in the recommendation set lack serendipity, tending to be very similar to previously rated items. User studies have shown that users find online recommenders most useful when they recommend unexpected items [4]; highlighting the fact that overspecialization by content-based filtering systems is indeed a serious drawback. Common approaches to dealing with this problem of overspecialization include explicitly injecting diversity into the recommendation set [5, 6, 7] and building hybrid recommendation systems by incorporating aspects of collaborative filtering into the recommendation generation process.

Collaborative filtering is traditionally a memory-based approach to recommendation generation, though model-based approaches have also been developed. The user model is generally in the form of an n-dimensional vector representing the ratings of the n items, in the item set, by the user. Hence, in contrast to content-based approaches, collaborative filtering does not traditionally use any item content descriptions. The recommendation process consists of discovering the neighborhood of the active user, that is, other users that have a similar rating vector to that of the active user, and predicting the ratings of items, not currently viewed by the active user, based on ratings of these items by users within the active user's neighborhood [8].

While collaborative filtering is commercially the most successful approach to recommendation generation, it suffers from a number of well-known problems of all, the collaborative filtering system has become the predominant approach in furnishing the e-commerce system with an intelligence to capture user profiles and recommending relevant pages to the users. However, it suffers from a number of well-known problems including the cold start/latency problem, sparseness within the rating matrix, scalability, and efficiency [9]. Item-based similarity [10] and dimension reduction was proposed by some researchers to overcome the drawback.

One research area that has recently contributed greatly for this problem is web mining. Most of the systems developed in this field are based on web usage mining (WUM) [11]. The term 'Web Usage Mining' [12] was introduced by Cooley et al, in which they define web usage mining as the 'automatic discovery of user access patterns from web Servers'. Web usage mining has gained much attention in the literature as a potential approach to fulfill the requirement of web personalization [13, 12, 14, 15, 3,16].

These systems are mainly concerned with analyzing web usage logs, discovering patterns from this data and making recommendations based on the extracted knowledge [14, 3, 17, 18]. Unlike traditional personalization techniques, which mainly recommend a set (referred to as the recommendation set) of items deemed to be of interest to the user base their decisions on user ratings on different items or other explicit feedbacks provided by the user [19,20]. These techniques discover user preferences from their implicit feedbacks, namely the web pages they have visited.

More recently, systems that take advantage of a combination of content, usage and even structural information of the websites have been introduced [51,52,53,54,55] and shown superior results in the web page recommendation problem.

In [42] the degree of connectivity based on the link structure of the website is used to evaluate usage based techniques. A new method for generating navigation models is presented in [54] which exploits the usage, content and structure data of the website. Eirikani et al. [52, 53] use the content of web pages to augment usage profiles with semantics using domain ontology.

A few combined or hybrid web recommender systems have been proposed in the literature [42] [55]. The work in [55] adopts a clustering technique to obtain both site usage and site content profiles in the off-line phase. In the on-line 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 [57].

In [42], Nakagawa and Mobasher use association rule mining, sequential pattern mining, and contiguous sequential mining to generate three types of navigational patterns in the off-line phase. In the on-line 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 [57]. An extensive study of web personalization based on web usage mining can be found in [21].

Some studies have considered the approach of using pages interesting to the user for the recommendation process. In [41], Mobasher et al use statistical significance testing to judge whether a page is interesting to a user. Its main idea is: A duration threshold is calculated for each page using the average duration and standard deviation of the visits to the page; if the duration of a page is longer than the threshold, that page is considered interesting to the user and vice versa. The drawback of such an approach is that it simply divides pages into interesting and uninteresting groups, and neglects the difference in the degrees of interest. For one thing, there isn't a clear division between interesting and uninteresting pages; for another, the degrees of interest are probably not the same for all the interesting (and uninteresting) pages.

Clustering and collaborative filtering approaches are ready to incorporate both binary and non-binary weights of pages, although binary weights are usually used for computing efficiency [21] [41]. Association Rule (AR) mining and Sequential Pattern (SP) mining [42] can lead to higher recommendation precision [21], and are easy to scale to large datasets, but how to incorporate page weight into the AR and the SP models has not been explored in previous studies.

Weighted Association Rule (WAR) mining allows different weights to be assigned to different items, and is a possible approach to improving the AR model in the web personalization process. Cai et al. [56] proposed assigning different weights to items to reflect their different importance. In their framework, two ways are proposed to calculate itemset weight: total weight and average weight.

Weighted support of an itemset is defined as the product of the itemset support and the itemset weight. Tao et al. [58] also proposed assigning different weights to items, the itemset/transaction weight is defined as the average weight of the items in the set/transaction, and weighted support of an itemset is the fraction of the weight of the transactions containing the itemset relative to the weight of all transactions. Both models attempt to give greater weights to more important items, facilitating the discovery of important but less frequent itemsets and association rules. However, both models assume a fixed weight for each item while in the context of web usage mining a page might have different importance in different sessions.

As the connectivity features of the web graph plays an important role in the process of the web personalization, on the other hand a page is important if many users have visited it before, in the context of navigating a web site, we propose a novel machine learning perspective toward the problem, which we believe is suitable to the nature of web page personalization problem and integrating web usage mining with link analysis techniques for assigning probabilities to the web pages based on their importance in the web site's navigational graph and makes recommendations primarily based on web usage logs and the structure of the web site.

In this paper, we first propose a page recommendation algorithm based on distributed learning automata to learn the behavior of previous users'. The proposed algorithm takes advantage of web usage data and link information to recommend pages to the current user base on learned pattern. In this work we try to assign a quantitative weight to each page, taking into account the degree of interest. We extend the traditional association rule mining algorithm by allowing that a weight to be associated with each item in a transaction for reflecting the interest of each item within the transaction.

By introducing a novel weighted association rule mining algorithm, we present our second algorithm for recommendation purpose. In the proposed weighted association rule miner, the time spent by each user on each page and visiting frequency of each page are used to assign a quantitative weight to the pages instead of traditional binary weights. Also, a novel method is proposed to pure the current session window.

One of the challenging problems in recommendation systems is dealing with unvisited or newly added pages. By considering this problem and improving the efficiency of first two algorithms we present a hybrid algorithm based on distributed learning automata and weighted association rule mining algorithm. In the hybrid algorithm we employ the HITS algorithm to extend the recommendation set.

We have applied these algorithms on standard data set and got very good results compared to the association rules, which is commonly known as one of the most successful approaches in web mining based recommender systems. The evaluation of the experimental results shows considerable improvements and their robustness. Our experiments on real data set show that hybrid algorithm performs better than the other algorithms we compared to and, at the same time, it is less complex than other proposed algorithms with respect to memory usage and computational cost too.

The rest of this paper is organized as follows. Section 2 provides distributed learning automata -based recommendation algorithm which is the basis of our method. In section 3 we present our weighting schema, weighted association rule and overview the weighted association rule based recommendation method. We represent our hybrid approach in section 4. Section 5 gives the performance evaluation of the proposed algorithms compared to association rule based method. Section 6 concludes the paper.

2. Web Page Recommendations based on Distributed Learning Automata

2.1 Preliminaries

2.1.1 Learning Automata

Learning Automata are adaptive decision-making devices operating on unknown random environments. The automata approach to learning involves the determination of an optimal action from a set of allowable actions. An automaton can be regarded as an abstract object which has finite number of possible actions. In each decision process, the automata selects an action from its finite set of actions. This action is applied to a random environment. The random environment evaluates the selected action and gives a grade to

applied action of automata. The random response of environment (i.e. grade of action) is used by automata in further action selection. By continuing this process, the automata learns to select an action with best grade. The learning algorithm used by automata to determine the selection of next action from the response of the environment. An automaton acting on unknown random environment and improves its performance is some specified manner, is referred to as learning automata (LA). Learning automata can be classified into main categories: fixed structure learning automata and variable structure learning automata [22]. In the following, the variable structure learning automata which will be used in this paper is described.

Variable structure learning automata is represented by quintuple $< \alpha, \beta, p, T(\alpha, \beta, p) >$, where $\alpha = \{\alpha_1, \alpha_2, \cdots, \alpha_r\}$, $\beta = \{\beta_1, \beta_2, \cdots, \beta_r\}$, and $p = \{p_1, p_2, \cdots, p_r\}$ are an action set with r actions, an environment response set, and the probability set p containing r probabilities, each being the probability of performing every action in the current internal automaton state, respectively. The function of T is the reinforcement algorithm, which modifies the action probability vector p with respect to the performed action and received response. If the response of the environment takes binary values learning automata model is P-model and if it takes finite output set with more than two elements that take values in the interval [0, 1], such a model is referred to as Q-model, and when the output of the environment is a continuous variable in the interval [0, 1], it is referred as S-model.

It is evident that the crucial factor affecting the performance of the variable structure learning automata is learning algorithm for updating the action probabilities. Various learning algorithms have been reported in the literature. Let α, be the action chosen at step n as a sample realization from probability distribution p. The linear reward-inaction algorithm is one of the learning schemas and its recurrence Equation for updating action probability vector p is defined as Equation(1).

$$p_i(n+1) = p_i(n) + a.(1-\beta(n)).(1-p_i(n)) - b.\beta(n).p_i(n) \quad (1)$$

$$p_j(n+1) = p_j(n) + a(1-\beta(n)).p_j(n) + \frac{b.\beta(n)}{r-1} - b.\beta(n).p_j(n) \qquad if\ j \neq i$$

Where $0<\alpha<1$ is called step length and determines the amount of increases (decreases) of the action probabilities.

The above mentioned learning automata has a fixed number of actions. In some applications, like our first proposed algorithm, we need that LA has a changing number of actions [23]. A LA with changing number of actions, at any time instance n selects its action from a set of active actions $V(n)$ and behaves like this. For selecting an action, the learning automata first computes the sum of its actions' probability $K(n)$ and then the vector $\hat{p}(n)$ is computed according to Equation (2). The automaton selects one of its active actions randomly based on actions probabilities, i.e. $\hat{p}(n)$. The automaton applies the selected action α_i to the environment and gets the response. For desirable responses, the $\hat{p}(n)$ vector is updated based on Equation (3) and for undesirable actions is updated based on Equation (4). Finally, the automaton updates the actions' probability vector $p(n)$ based on vector $\hat{p}(n+1)$ as shown in Equation (5).

$$K(n) = \sum_{\alpha_i \in V(n)} p_i(n) \quad (2)$$

$$\hat{p}_i(n) = prob[\alpha(n) = \alpha_i \mid \alpha_i \in V(n)] = \frac{p_i(n)}{K(n)}$$

$V(n)$ is the set of enabled actions

$$\hat{p}_i(n+1) = \hat{p}_i(n) + a.(1-\hat{p}_i(n)) \quad (3)$$
$$\hat{p}_j(n+1) = \hat{p}_j(n) - a.\hat{p}_i(n) \qquad \forall j \ \ j \neq i$$

$$\hat{p}_i(n+1) = (1-b).\hat{p}_i(n) \quad (4)$$
$$\hat{p}_j(n+1) = \frac{b}{\hat{r}-1} + (1-b)\hat{p}_j(n) \qquad \forall j \ \ j \neq i$$

$$p_i(n+1) = \hat{p}_i(n+1).K(n) \quad \text{for all i}, \alpha_i \in V(n)$$
$$p_j(n+1) = p_j(n) \quad \text{for all j}, \alpha_j \notin V(n) \tag{5}$$

2.1.2 Distributed Learning Automata

A distributed learning automata (DLA) is a network of LA which collectively cooperate to solve a particular problem. The number of actions for a particular LA in DLA is equal to the number of LA's that are connected to this LA. Selection of an action by a LA in the network activates one LA corresponding to the action. Formally, a distributed learning automata can be defined by a graph $DLA = (V, E)$, where the set $V = \{LA_1, LA_2, \cdots, LA_n\}$ is the set of n learning automata and $E \subset V \times V$ is the set of edges in the graph. The edge (i, j) represents the action j of automata LA_i. In other words, LA_j is activated when action j of automata LA_i is selected. The number of actions for particular automata $LA_k (k = 1, 2, \cdots, n)$ is equal to the out-degree of that node. If $\underline{p}^j$ corresponds to the probability distribution of actions of LA_j, then p_m^j shows the probability of selecting action α_m by automata A_j. In other words, we can assign a weight to each edge (i, j) in graph which is equal to the probability of selection of action i by automata j [24, 25, 26].

For example, in Fig. 1, every automata has two actions. Selection of action α_3 by A_1 will activate automata A_3. Activated automata choose one of its action which results in activation of the LA corresponding to the selected action. At any given time, only one of the automata in the network could be active.

2.1.3 PageRank Algorithm

The PageRank computation for ranking hypertext-linked web pages was originally outlined by Page and Brin [27]. Two intuitive explanations are offered for PageRank [28]. The first intuition of PageRank is based on an idea that if a page v of interest has many other pages u with high PageRank scores pointing to, then the authors of pages u are implicitly conferring some importance to page v. The second conceptual model of

PageRank is called the random surfer model. Consider a surfer who starts at a web page and picks one of the links on that page at random. On loading the next page, this process is repeated.

In the random surfer model, the web is represented by a graph $G = (V, E)$, with web pages as the vertices, V, and the links between web pages as the edges, E. If a link exists from page u to page v then $(u \to v) \in E$. To represent the following of hyperlinks, a transition matrix $\mathbf{P}$ from the web graph is constructed, setting:

$$p_{ij} = \begin{cases} \dfrac{1}{\deg(u_i)} : if \quad (u_i \to u_j) \in E \\ 0 \qquad\quad : otherwise \end{cases} \tag{6}$$

Where $\deg(u)$ is the out-degree of vertex u, i.e. the number of outbound links from page u. From this definition, we see that if a page has no out-links, then this corresponds to a zero row in the matrix $\mathbf{P}$. To model the surfer's jumping from dangling nodes, a second matrix $\mathbf{D} = \vec{\mathbf{d}}\,\vec{\mathbf{v}}^{\,\mathbf{T}}$, where $\vec{\mathbf{d}}$ and $\vec{\mathbf{v}}$ are both column vectors. If a page has no outgoing edge, its corresponding entry in the matrix $\vec{\mathbf{d}}$ will be zero. The $\vec{\mathbf{v}}$ is the personalization vector representing the probability distribution of destination pages when a random jump is made. Typically, this distribution is taken to be uniform, $p_i = 1/n$ for an n-page graph ($1 \le i \le n$). However, it need not be as many distinct personalization vectors may be used to represent different classes of user with different web browsing patterns. This flexibility comes at a cost, though, as each distinct personalization vector requires an additional PageRank calculation.

Putting together the surfer's following of hyperlinks and his/her random jumping from dangling pages yields the stochastic matrix $\mathbf{P'} = \mathbf{P} + \mathbf{D}$, where $\mathbf{P'}$ is a transition matrix of a discrete-time Markov chain (DTMC). To represent the surfer's decision not to follow any of the current page links, but to instead jump to a random web page, we construct a teleportation matrix $\mathbf{E}$, where $e_{ij} = p_j$ for all i, i.e. this random jump is also dictated by the personalization vector. Incorporating this matrix into the model gives:

$\mathbf{P}''=c\mathbf{P}'+(1-c)\mathbf{E}$, where $0<c<1$, and c represents the probability that the user chooses to follow one of the links on the current page, i.e. there is a probability of $(1-c)$ that the surfer randomly jumps to another page instead of following links on the current page. Having constructed $\mathbf{P}''$ we might attempt to find the PageRank vector of web pages by using power method approaches.

2.2 The Proposed Algorithm

Our first algorithm is based on DLA and PageRank introduced in the previous subsections. The proposed algorithm employs the web usage data and underlying site structure to recommend pages to the current user. In the proposed algorithm, the transition matrix $\mathbf{P}$ and personalization vector $\vec{\mathbf{v}}$ in the original PageRank algorithm are computed based on usage data instead of link structure. For this reason, a DLA learns the transition probability matrix $\mathbf{P}$ from the behavior of the visiting users which are available in the site's log file. In addition, the personalization vector $\vec{\mathbf{v}}$ is computed based on the visiting rate of pages preferring pages which are visited by more users. Having the transition matrix $\mathbf{P}$ and personalization vector $\vec{\mathbf{v}}$ which obtained from the knowledge acquired from previous users' visits, the PageRank algorithm is used to compute the rank of each page.. It is noticeable that the PageRank algorithm is used in a totally different context. In this context the PageRank is a usage-biased since it is based on the navigational behavior of previous visitors. The proposed algorithm, in addition to page recommendation, can be used to modify the links between the pages, (i.e. add new links or delete old links). In the following subsections, the steps of algorithm are described.

2.2.1 Computing the Transition Probability Matrix

In the original PageRank algorithm, the probability of following a link when the user is in a specified page is uniformly distributed between all of the outgoing links or favors certain pages in the personalized PageRank. In our algorithm, we bias the PageRank algorithm using the data acquired from previous users' visits, as they are discovered from the user sessions recorded in the web site's logs. The intuition behind our algorithm is as follows: a page is important in a web site if many users have visited it before. Suppose that a page is visited more than other pages considering the outgoing pages of a certain

page. The high visiting rate of a page indicates that the page is followed by more users and is important for them. So, it is better that the recommendation is biased to the page with high visiting rate.

To learn the transition probabilities based on precious users' behavior, we use a distributed learning automata with n LAs with variable number of actions. For each page in the site a LA with $n-1$ actions is added to the DLA. Each action corresponds to following a page. For each LA at each time a subset of its actions is active. The number of actions in the LA assigned to page i is equal to the number of pages that a user at page i can follow from that page. In the beginning, all of actions are inactive. When a user at page i go to page j, the action corresponds to page j is rewarded or penalized by the environment and so the probability of actions of LA is updated. These probabilities correspond to probability of transition between pages which are learned by the LA. In the following the rewarding and penalizing schema of actions in LA is described.

The rewarding and penalizing schema of actions is based on a learning algorithm which updates the actions' probabilities in each step. Since the employed LA has variable number of actions, Equation (3) and Equation (4) are employed to this intention. In using these equations, the parameter a, which is called rewarding parameter, is calculated from Equation (7):

$$a = \omega + \lambda \tag{7}$$

Where ω is a constant and λ is obtained by this intuition. If a user goes from page i to page j in the site and there is no link between these pages, the value of λ is set to constant value; otherwise it is set to zero. In other words, in the movement of a user from page i to page j the j^{th} action in i^{th} learning automate is more awarded when there is no link between pages i and j than there is a link between them. This intuition sounds reasonable and can be used to modify the underlying link structure statically exists between pages. Due to it is obvious that two pages which have hyperlinks together have more probability to be visited by a user than the pages without links. Specially, in comparing two pages with same visiting rate, the pages without link was more interesting for user.

If there is a cycle in users' navigation path, the actions in the cycle indicates the illegal movement of user, quandary of user, or the dissatisfaction of user from the visited pages and must be penalized. The penalization increases with the cycle length. So, the parameter b which is penalization factor is calculated from Equation (8):

$$b = (\textbf{steps in cycle containing k and l}) * \beta \quad (8)$$

Where β is a constant factor. As it is clear from Equation (8), the penalization factor has direct relation with the length of cycle traversed by the user.

For every user session in the log file, we begin with the first page. For each pair of consecutive pages in the session, the LA corresponding to the first page is used to update its probabilities if the action is already active; otherwise activates it. We assume that any consecutive pages' repetitions have been removed from the user sessions; on the other hand, we keep any pages that have been visited more than once, but not consecutively. This process is repeated till reaching the latest page in the session. After processing all of the sessions, the transition matrix $\mathbf{P}$ is generated based on probability of actions in DLA. Each entry p_{ij} in matrix $\mathbf{P}$ is set to the probability of action j in the LA corresponds to page i.

2.2.2 Page Ranking

The transition matrix $\mathbf{P}$ and personalization vector $\vec{\mathbf{v}}$ must be available to compute the importance of pages. As mentioned before, our objective is to generate a set of biased PageRank vectors using users' sessions. The key to creating usage-biased PageRank is that we can bias the computation to increase the effect of certain pages by using a nonuniform personalization vector for $\vec{\mathbf{v}}$. Note that the bias involves introducing additional rank to the appropriate pages in each iteration of the computation. The computation of matrix $\mathbf{P}$ based on distributed learning automata was described in previous subsection. In this subsection we describe computation of personalization vector $\vec{\mathbf{v}}$. We employ the visiting rate of pages as measure for personalization. Let $w(i)$ denote the number of users who visit the page i . The value of personalization vector for

page i is set to $\frac{w(i)}{\sum_{j \in V} w(j)}$. This setting exactly models the importance of pages for users and contribution of pages in recommendation. In this case the vector $\vec{\mathbf{v}}$ is a probability vector and sum of its all entries equal to 1.

2.2.3 Page Recommendation

As described before, the goal of personalization is to compute a set of pages unvisited by current user to recommend for him which has the maximum match with the users' interest [29, 30]. The recommendation phase is the only online phase of every recommendation algorithm and must have a satisfied performance. Suppose that a user is walking in the site and the path traversed by him is $p_1 \rightarrow p_2 \rightarrow p_3 \rightarrow ... \rightarrow p_k$. For a new user this path is empty. We use a fixed-size sliding window over the current active session to capture the current user's history path. Note that the sliding window of size $|w|$ over the active session allows only the last w visited pages to influence the recommendation set.

To recommend page p_{k+1} to the current user, we must model the navigational behavior of the users of a web site. Markov models provide a simple way to capture sequential dependency when modeling the navigational behavior of the users of a web site. The order of the Markov model indicates the "memory" of the prediction, i.e. denotes the number of previous user steps which are taken into consideration in the process of calculating the path probabilities. Therefore, in the case of Markov chains, the probability of visiting a page depends only on the previous one, in second-order Markov models it depends on the previous two, and so on. The selection of the order influences both the prediction accuracy and the complexity of the model while heavily depends on the application/data set. After computing the transition matrix $\mathbf{P}$ by using DLA, the path probabilities are computed for an m-order model using the chain rule as follows:

$$\Pr(p_1 \to p_2 \to p_3 \cdots \to p_k) = \Pr(p_1) \times \prod_{i=2}^{k} \Pr(p_i \mid p_{i-m} \cdots p_{i-1}) \tag{9}$$

equals to: $p_1 \to p_2 \to p_3$ For example the probability of path

$$\Pr(p_1 \to p_2 \to p_3) = \Pr(p_1)\Pr(p_2 \mid p_1)\Pr(p_3 \mid p_2) = \Pr(p_1)\frac{\Pr(p_1 \to p_2)\Pr(p_2 \to p_3)}{\Pr(p_1)\Pr(p_2)}$$

Where $\Pr(\bullet \to \bullet)$ represents the probability of transition between pages and $\Pr(\bullet)$ is the rank of page obtained from the biased PageRank algorithm.

Based on Equation (9), the prediction of the next most probable page p_{k+1} visit of a user is performed by computing the probabilities of all existing paths such $p_1 \to p_2 \to p_3 \to \ldots \to p_k \to p_{k+1}$ having the pages visited so far by the user as prefix and choosing the most probable one. The bounded probabilities' computation is straightforward since it reduces to a lookup on the transition probability matrix $\mathbf{P}$.

For all unvisited pages, this value is computed and sorted based on their probabilities. The number of recommended pages can be controlled based on the number of pages or based on determined threshold value for probability. Then the pages with highest rank are recommended to the current user.

3. Web Page Recommendations based on Weighted Association Rule

In this section we present our second algorithm for page recommendation. In the proposed algorithm, we extend the traditional association rule mining algorithm by allowing a weight to be associated with each item in a transaction to reflect the interest of each item within the transaction and develop a novel recommendation algorithm based on proposed weighted association rule mining approach. In the proposed weighted association rule miner, the time spent by each user on each page and visiting frequency of each page are used to assign a quantitative weight to the pages instead of traditional

binary weights. The intuition behind this idea is that the time spent on pages [31] and visiting frequency are good implicit interest indicator of a user on those pages.

The methodology is like this: first, the weighted association rules of each URL will be extracted from the web log data and similarity between active user sessions will be calculated upon the weighted rules instead of an exact match for finding the best rule. Finally, the recommendation engine will then find the most similar rules to the active user session with the highest weighted confidence by scoring each rule in terms of both its similarity to the active session and its weighted confidence. In the following, we first introduce our weighting schema. Then we describe the proposed weighted association rule mining algorithm and page recommendation mechanism.

3.1 Weighting Schema

Let $P = \{p_1, p_2, ..., p_m\}$ denote the set of web pages accessed by users in web server logs after the preprocessing phase [32], each of them is uniquely represented by its associated URL. Also let $T = \{t_1, t_2, ..., t_n\}$ be the set of user transactions where each $t_i \in T$ is a subset of P. To facilitate the high quality recommendation, we represent each transaction t as an m-dimensional vector over the space of web pages, $t = < (p_1, w_1), (p_2, w_2), ..., (p_m, w_m) >$, where w_i denotes the weight with the i^{th} web page $(1 \le i \le m)$ visited in a transaction t. The weight w_i in transaction t needs to be appropriately determined to capture a user's interest in i^{th} web page.

The weights can be determined in a number of ways; however in the context of personalization based on clickstream data, the primary sources of data are server access logs. This allows us to choose two types of weights for pages: weights can be binary, representing the existence or nonexistence of a page access in the transaction or they can be a function of parameters such as duration of the associated page in the user's session to represent the interest of page to a specific visiting user.

Since the recommendation process is based on the behavior of previous users, so the weighting schema must precisely model the user's interest. Recommendation approaches

proposed in previous works; however, do not distinguish the importance of different pages and all the visited pages are treated equally whatever their usefulness to the user. They neglect the difference in the importance of the pages and degree of interest in a users' session. It is quite probable that not all the pages visited by the user are of interest to him/her. A user might get into a page and find it is of no value to him/her, causing irrelevant page accesses to be recorded into the log file. Therefore, it is imperfect to use all the visited pages equally to capture user interest and predict user behavior. Although in usage-based recommendation systems we can't expect users to express their interests explicitly, we need a weight measure for approximating the interest degree of a web page to a user.

Inspired by Chan and coworkers [33, 34], we propose a weighting measure which is calculated from web logs to extract the interest of page for the visitor. In our weighting schema, both of time length of a page and visiting frequency of a page are used to estimate its importance in a transaction, in order to capture the user's interest more precisely instead of binary which is typically used in other researches. This approach try to give more consideration to more useful pages, in order to better capturing the user's information need and recommend more useful pages to the user.

Several reasons validate the idea of using pages visit duration as one of the weighting parameters. First, it reflects the relative importance of each page, because a user generally spend more time on a more useful page[31, 35], because if a user is not interested in a page, he/she do not spend much time on viewing the page and usually jumps to another page quickly [36]. However, a quick jump might also occur due to the short length of a web page so the size of a page may affect the actual visiting time. Hence, it is more appropriate to accordingly normalize duration by the length of the web page, that is, the total bytes of the page. The formula of duration is given in Equation (10). Second, the rates of most human beings getting information from web pages should not differ greatly [35]. If we assume a similar rate of acquiring information from pages for each user, the time a user spends on a page is proportional to the volume of information useful to him/her. As page duration can be calculated from web logs, it is a good choice for inferring user interest.

Frequency is the number of times that a page is accessed by different users. It seems natural to assume that web pages with a higher frequency are of stronger interest to users. A parameter that must be considered in the calculating the frequency of a page is the in-degree of that page (e.g. the number of incoming links to the page). It is obvious that a page with large in-degree has more probability to be visited by a user than a page with small one. Specially, in comparing two pages with same visiting rate, the page with small in-degree is more interesting. The formula of frequency is given in Equation (11).

We use time spent by a user for viewing a page and frequency of visiting as two very important pieces of information in measuring the user's interest on the page, so we assign a significant weight to each page in a transaction according to these definitions as Equation (12).

$$Duration(p) = \frac{\frac{Total\ Duration(p)}{Size(p)}}{\max_{Q \in T}(\frac{Total\ Duration(p)}{Size(p)})} \tag{10}$$

$$Frequency(p) = \frac{Number\ of\ visit(p)}{\sum_{Q \in T} Number\ of\ visit(Q)} * \frac{1}{\text{Indegree}(p)} \tag{11}$$

$$Weight(p) = \text{Frequency}(p) * \text{Duration}(p) \tag{12}$$

At the end, every user transaction is successfully transformed into a m-dimensional vector of weights of web pages, i.e., $t = <(p_1, w_1),(p_2, w_2),...,(p_m, w_m)>$, where m is the number of web pages visited in all users' sessions.

3.2 Weighted Association Rule Based Recommendation model

3.2.1. Association Rule Mining of Web Usage Log

Given a set of transactions where each transaction is a set of items (pages), an association rule implies the form $X \Rightarrow Y, where X \subset I, Y \subset I, X \cap Y = \phi$, where X and

Y are two sets of items; X is the body and Y is the head of the rule. The support for the association rule $X \Rightarrow Y$ is the percentage of transactions that contain both X and Y among all transactions. The confidence of the rule $X \Rightarrow Y$ is the percentage of transactions that contain Y among transaction that contain X. The support represents the usefulness of the discovered rule and the confidence represents certainty of the rule. The confidence is computed as follows:

$$Confidence = \frac{Support(X \cup Y)}{Support(X)} \tag{13}$$

Association rule mining is the discovery of all association rules that are above a user-specified minimum support and minimum confidence. Apriori algorithm is one of the prevalent techniques used to find association rules [37, 38]. Apriori operates in two phases. In the first phase, all itemsets with minimum support (frequent itemsets) are generated. This phase utilizes the downward closure property of support. In other words, if an itemset of size k is a frequent itemset, then all the itemsets below (k - 1) size must also be frequent itemsets. The second phase of the algorithm generates rules from the set of all frequent itemsets.

Association rules capture the relationships among items based on their patterns of co-occurrence across transactions. In the case of web transactions, association rules capture relationships among pages based on the navigational patterns of users. Each web page can be viewed as an item, and the set of web pages accessed by a user within a short period of time can be treated as a transaction so the purpose of mining association rules is to find out which web pages are usually visited together in different sessions.

However, the traditional association rules (ARM) model focus on binary attributes. In other words, this approach only considers whether an item is present in a transaction or not. Also it is supposed that all items have the same significance and does not take into account the weight of an item within a transaction and all pages in a transaction are treated uniformly. Also, in most previous approaches of applying ARM to web usage personalization they ignore the difference in the importance of the pages in a user session.

3.2.2 Mining Weighted Association Rules

As mentioned before, we first extend the traditional association rule problem by allowing a weight to be associated with each item in a transaction to reflect interest of each item within the transaction. In turn, this provides us with an opportunity to associate a weight parameter with each item in a resulting association rule, which we call a weighted association rule (WAR). Weighted association rule is useful in some sense. For example, the product, which has higher profit margin, should be paid more attention. Weighted Association Rule (WAR) mining allows different weights to be assigned to different items, and is a possible approach to improving the ARM model in the web personalization process.

In this model, greater weights are given to more important items, facilitating the discovery of important but less frequent itemsets and association rules. However, previous models assume a fixed weight for each item, while in the context of web usage mining, a page might have different importance in different sessions.

In the following we describe weighted rules with the definition of associated parameters. We extend the Apriori by adapting its parameters based on weighted items. In the next section we employ this algorithm for page recommendation.

3.2.2.1 Weight Settings

Given the transformation of user transactions into a m -dimensional space as vectors of weights of web pages, $t = <(p_1, w_1),(p_2, w_2),...,(p_m, w_m)>$ where each $p_i \in P$, the weight w_i associated to page p_i is a non-negative real number to reflect the importance of page p_i in transaction t according to Equation (12). Inspired by Tao[58], we modify the measures exist in Apriori algorithm in the following definitions to reflect the weighting schema.

Definition 1. *Weighted item:*

Item weight is a value attached to an item (page) representing its significance. We denote it as $w(p_i) = Weight(p_i)$, which is calculated using the Equation (12).

Definition 2. *Weight of an itemset in a transaction:*

Based on the item weight $w(p_i)$, the weight of an itemset X, denoted as **w(X, t)**, can be derived from the weights of its enclosing items. One simple way is to use the minimum weight of the all items in the itemset as the weight of whole itemset as shown in Equation (14).

$$\mathrm{w(X,t)} = \begin{cases} \min(w(p_1, p_2, ..., p_k) & X \subseteq t \\ 0 & X \not\subset t \end{cases} \tag{14}$$

Where k is the number of items in the itemset.

Alternatively, we can use the average weights of its enclosing items as the itemset weight. Our experiments show that the minimum weight has better quality.

Definition 3. *Transaction weight:*

By assigning a weight to each item and itemset, we also assign a weight to each transaction to be used in the calculation of the support of each itemset. Assigning weight to transactions gives us the possibility to distinguish between different transactions. Usually the higher a transaction weight, the more it contributes to the mining result. One simple way is to calculate the average weights of all items that enclosed in each transaction. The weight of each transaction $\mathbf{w(t_k)}$ is calculated as shown in Equation (15).

$$\mathrm{w(t_k)} = \frac{\sum_{i=1}^{|t_k|} w(p_i)}{|t_k|} \tag{15}$$

Definition 4. *Weighted support of an itemset across all transaction:*

We modify the support of an itemset, Weighted support $wsp(X)$ of an itemset X across all transactions is defined as follows:

$$wsp(X) = \frac{\sum_{t_i \in T} w(t_i) * w(X, t_i)}{\overline{w} * \sum_{k=1}^{|T|} w(t_k)} \quad (16)$$

Where $\overline{w}$ is the average weight of all the items across all transactions, and T is the set of all transactions.

3.2.2.2 Weighted Frequent Itemset

The problem of frequent pattern mining in the traditional association rule mining framework is to find the complete set of itemset satisfying a minimum support threshold in the database. In our model, we say an itemset is frequent if its weighted support is above a predefined weighted support threshold. Our approach to mining frequent itemsets is based on the Apriori [38] algorithm. To prune infrequent patterns, frequent pattern mining uses the downward closure property (anti-mono-tone property) [13,39]. That is, any subset of a significant itemset is also significant or if a pattern is infrequent pattern, all super patterns must be infrequent patterns. Using the downward closure property, infrequent patterns can be easily pruned. By our definition of weighted support and frequent itemsets, there is a property that any subset of a frequent itemset is also frequent, here called a weighted downward closure property [38]. The downward closure property of the support measure in the unweighted case longer exists. Therefore, the candidate itemsets having k items can be generated by joining large itemsets having k-1 items. This can result in much smaller number of candidate itemsets. For example, if we are looking for pairs of items with minsup, we can only consider those items that appear in the database having minsup. Provided minsup is high enough, the number of items for the next joining step will be small enough to speed up the computation significantly. Following theorem shows that our weighting schema holds the downward closure property.

Theorem. The proposed weighting schema holds the downward closure property and for any candidate itemset, all of its subitems also are candidate itemset.

Proof. Let I_1 and I_2 be two itemsets. Also suppose that $I_1 \subset I_2$, i.e. I_2 be a superset of I_1. For proving the validity of downward closure property in the proposed algorithm, we suppose that I_1 is not a significant itemset over all the transactions but I_2 is a significant itemset. Let T_1 denote a set of transactions which contains all the items in I_1 and similarly T_2 denote the set for I_2. Since I_2 is superset of I_1, so $T_2 \subset T_1$. Therefore $\sum_{t \in T_1} w(t) \geq \sum_{t \in T_2} w(t)$. According to the definition of weighted support of an itemset we have $wsp(I_1) = \frac{\sum_{t \in T_1} w(t) * w(I_1, t)}{\overline{w} * \sum_{t \in T_1} w(t)}$ and $wsp(I_2) = \frac{\sum_{t \in T_2} w(t) * w(I_1, t)}{\overline{w} * \sum_{t \in T_2} w(t)}$. By comparing $wsp(I_1)$ and $wsp(I_2)$ and considering the fact that $\sum_{t \in T_1} w(t) \geq \sum_{t \in T_2} w(t)$ we have that $wsp(I_1) \geq wsp(I_2)$. Because I_1 is not a significant itemset, its weighted support is less than the minimum threshold and since $wsp(I_1) \geq wsp(I_2)$ so the $wsp(I_2)$ is also less than the minimum support threshold and I_2 is not a significant itemset. In conclusion, if an itemset is a significant itemset, its subsets also are significant itemset and it proves that the downward closure property always valid in the proposed algorithm.

Definition 5. *Weighted confidence of the weighted association rule*

We define the weighted confidence of association rule for weighted rules as follows:

$$wconf(X \Rightarrow Y) = \frac{wsp(X \cup Y)}{wsp(X)} \tag{17}$$

Definition 6. *Weighted rules*

For each rule, besides the weighted confidence and weighted support, we also add the weight of each page. The result of weighted association rule mining conceptually described as follows: $r = <(p_1, p_2, ..., p_k), (q_{k+1}, q_{k+2}, ..., q_{k+m}), (w_1, w_2, ..., w_{k+m}), \delta, \alpha > \in R$, where $(p_1, p_2, ..., p_k), (q_{k+1}, q_2, ..., q_{k+m})$ present the body and head of the weighted rule

respectively, w_i represent the weight of i^{th} page in the rule, δ represent the weighted support and α represent the weighted confidence of the rule.

3.3. A Recommendation Engine Using Weighted Association Rules

The goal of personalization based on anonymous web usage data is to compute a recommendation set for the current user session, consisting of the objects (links, ads, etc.) that most closely match the current user session. These recommended pages are added to the last page in the active session accessed by the user before that page is sent to the browser. The methods based on association rule mining to compute a recommendation set for the current (active) user session, use a sliding window to control the number of session pages to be matched against the association rules [40]. So, maintaining a history depth may be important in the recommendation service to provide reasonable suggestions. In the following, we present our mechanisms for this purpose.

3.3.1. Modify User's Current Session

Maintaining a history depth may be important because most users navigate several paths leading to independent pieces of information within a session. Previous works [40, 21, 42] use a fixed-size sliding window over the current active session to capture the current user's history depth and generate the recommendations.

The sliding window of size n and go the right way over the active session allows only the last n visited pages to influence the recommendation value in the recommendation set because most of users go back and forth while navigating a site to find the desired information, and it may not be appropriate to use earlier portions of the user session to represent the user's current information need. However, this method does not distinguish the importance of different pages, and all the n last visited pages are treated equally whatever their usefulness to the user. A better approach would be to filter out uninteresting pages and use only the pages of interest to the user for the personalization process. Another parameter can also be used to associate an additional measure of significance with each page in the user's active session is weight of page. Although it seems that the recently visited pages by user are more appropriate to be used for the recommendation, but in many cases the user have a burst behavior. He navigates between

pages to find an interesting page and spent much of his time on that page and then repeats this process. So, the place of a page in the user session is not the only parameter influencing the selection of predictor pages. Hence we consider the freshness of a page and its weight simultaneously to choose the predictor pages.

In contrast to using a sliding window to preserve only the most recent session information for the matching work, inspired by Yan and Li [43], we propose a measure for approximating the user's current interest and filter out uninteresting pages by using a most simple method to capture the weight of interest of each page. We formulate the freshness of a page and its weight simultaneously to signify pages in user's current session as follows. First, the session is weighted as done for transactions. This guarantees that the time spent by user on each page and the frequency of page is reflected in the weight of each page. To apply the freshness of each page to its significance, we define the following parameter for each page:

$$Fresh(p_i)=\frac{i}{|w|} \qquad i=1,2,\cdots,|w| \tag{18}$$

Where $|w|$ is the size of sliding window and i is the place of page in the sliding window where 1 is assigned to the first visited page. In this Equation the last page is the freshest page. Also, the weighted vector should be normalized to effectively reflect the impact of freshness. The weight of each page is normalized as follows:

$$W_{\text{normalized}}(p_i)=\frac{w(p_i)}{\sum_{j=1}^{n} w(p_j)} \tag{19}$$

Therefore, in the weight measure we devised, fresh and Wnormalized are valued equally. We use the harmonic mean of Wnormalized and fresh to represent the interest degree of a web page to a user in the session. Equation (20) guarantees that Interest of a page is high only when Wnormalized and fresh are both high.

$$Interest(p_i) = \frac{2 * Fresh(p_i) * W_{\text{normalized}}(p_i)}{Fresh(p_i) + W_{\text{normalized}}(p_i)} \quad (20)$$

For example let $S = <(A,30),(B,20),(C,5),(D,5),(E,4),(F,10)>$ is an active user session after calculating the weight of each page according to Equation (12). Fig. 2 shows the comparison between our method and traditional sliding window. As we set the length of slide window to 3, the traditional method use the 3 latest pages from current session by choosing the page set $X = \{D,E,F\}$ but our method chooses the set $X = \{A,B,F\}$. Albeit the page A is visited first by user but as it has a large weight than D, E and F so it is the more interested for user and included in our window in contrast to the traditional method that escapes it.

3.3.2 Recommendation Mechanism

We developed a usage model for predictions based on weighted association rule. There are two phases in our system. First, the weighted association rules of each URL will be extracted from the web log data, the rules produced is representing the behavior of user's navigation on the web site. Secondly, the recommendation engine will search the top-n most similar weighted rules to the active user session before generating recommendation for the user. During the second phase instead of exact match between the active user and rules, we use a similarity measure for finding the most similar rules.

3.3.2.1. Similarity Measurement

Each of the weighted association rules $r = <(p_1, p_2, ..., p_k),(q_{k+1}, q_{k+2}, ..., q_{k+m}),(w_1, w_2, ..., w_{k+m}),\delta,\alpha> \in R$ obtained in the mining stage described in the previous section, are represented as a set of page-weight pairs. This will allow for both the active session and the association rules to be treated as m-dimensional vectors over the space of page in the site. Thus, given a weighted association rule r, we can represent the left-hand side of the each rule r_L as a vector: $r_L = \{w_1, w_2, ..., w_m\}$, where

$$w_i = \begin{cases} weight(p_i, r_{Li}), & if \quad p_i \in r_L \\ 0, & \text{otherwise} \end{cases} \tag{21}$$

Similarly, the current user session is also represented as a vector $S = \{s_1, s_2, ...s_m\}$ where s_i is a significance weight associated with the corresponding page reference, if the user has accessed p_i in this session, and $s_i = 0$, otherwise.

Then we compute the matching score between association rules that capture relationships among page based on their co-occurrence in navigational patterns of users and the current active session. The matching score between them is defined as:

$$\text{Dissimilarity}(\mathrm{S}, \mathrm{r_L}) = \sum_{i:r_{Li}>0} \left(\frac{2 * (w(s_i) - w(r_{Li}))}{w(s_i) + w(r_{Li})}\right)^2 \tag{22}$$

$$\text{Match Score}(\mathrm{S}, \mathrm{r_L}) = 1 - \frac{1}{4}\sqrt{\frac{\text{Dissimilarity}(\mathrm{S}, \mathrm{r_{Li}})}{\sum_{i:r_{Li}>0} 1}} \tag{23}$$

S and $\mathbf{r_L}$ represent the active user and left hand side of weighted association rule, respectively. The rationale behind this formulation is as follows: $\text{Dissimilarity}(\mathrm{S}, \mathrm{r_L})$ is a dissimilarity measure and have been applied to the experiments in literature and achieved success in solving different problems [44, 45, 46] that use average (arithmetic mean value $\_(w(s_i) + w(r_{Li}))/2$) weight as the normalization scheme. In order to have similarity measures between 0 and 1, it is necessary to normalize its distance by dividing it by the maximum discrepancy and then subtracting this normalized distance from 1.

Where a perfect match between active user and rule are found, the **Match Score** is equal to 1. As the algorithm tries to find rules that are similar to the active user session, the similarity measure between a rule and the active session is dependent on the magnitude of the left-hand side of the rule. Association rules might have multiple items on the right hand side of the rules but, due to the nature of the prediction problem in this paper recommendations are independent of one another and users will select only one of several recommendations so we only use rules that have singleton right-hand sides.

3.3.2.2. Recommendation Score

The recommendation engine is the online component of a usage-based personalization system in order to determine which items (not already visited by the user in the active session) are to be recommended, a recommendation score is computed for each page p_i. Two factors are used in determining this recommendation score: the overall matching score of the active session to the weighted rules as a whole, and the weighted confidence of the rule. The recommendation scores for the active user are computed by multiplying these factors. Given the weighted association rule and active session S, a recommendation scores for the active session, **Rec(S, X => p)**, is computed as follows:

$$\textbf{Rec(S, X => p)} = \textbf{Match Score(S, X)} * wconf\,(X \Rightarrow p) \quad (24)$$

Finally the top-n most similar pages are sorted then the highest recommendation score choose as the recommendation to the active user. The improvement of this approach is that instead of exact match between the active user and association rules, both of the similarity between rules and current session and the weighted confidence of X => p are used to determine the recommendation score, not just the confidence value as is used in previous works [3, 40]. We choose the highest recommendation score as the recommendation to the active session.

4. The Hybrid Algorithm

In this section we propose a hybrid efficient algorithm based on distributed learning automata and weighted association rule algorithms proposed before considering their weak and strong points. The algorithm solves the problem of recommending rarely visited or newly added pages. The steps in the algorithms could be briefly summarized as follows:

Step 1: Cluster the pages based on users' usage pattern.

Step 2: Generate the seed recommendation set.

Step 3: Extend the seed set by clusters to generate the candidate set.

Step 4: Apply the HITS algorithm to rank the candidate set and generate final recommendation set.

A general view of the Hybrid algorithm is depicted in Fig. 3. These steps are described in the next four subsections.

4.1. Cluster the Pages Based on Users' Usage Pattern

We propose an algorithm to cluster web pages not from the content of the pages but from the pattern of their usage, assuming that users have an intuitive grasp of what a page is about and how valuable it is, and this intuition guides their actions. The method clusters the pages based on how often they occur together across user sessions. On the other hand, page clusters tend to group together frequently co-occurring items across sessions, even if these items are themselves not deemed to be similar. This allows us to obtain clusters that potentially capture overlapping interests of different types of users.

The idea of clustering based on usage data is inspired by the functioning of the brain. In the brain, concepts that are activated simultaneously (co-activation) become more strongly associated. Since, users visiting a web site can be assumed to be looking for mutually relevant pages rather than a random assortment of unrelated pages, pages which are consulted by same user, are co-activated and have association with each other. In other words, documents develop stronger associations as they are more frequently co-activated. It is noticeable that this method is particularly useful for multimedia documents, which do not contain any searchable keywords.

To learn the associations implicitly exists between pages based on usage data, a DLA is employed as done in DLA Recommender algorithm. A distributed learning automata with n LAs with variable number of actions learns the association between pages using log data. In the DLA, the probability of action j in LA_i represents the association between i^{th} and j^{th} page. We create the association matrix $\mathbf{P}$ from the actions probability in DLA as follows. We set the a_{ij} to the probability of action j in LA_i. Since the learning process assumes ordered page access, so the learning process yields to an asymmetric association

matrix $(p_{ij} \neq p_{ji})$. By multiplying the (asymmetric) matrix **P** with its transpose we can create a new, symmetric matrix:

$$S = P \cdot P^T \qquad (25)$$
$$s_{ij} = \sum_k a_{ik} a_{kj}$$

Where s_{ij} represents the degree of similarity between the pages i and j. Indeed, s_{ij} is the dot product between the all the associations that the documents i and j have with other documents. The more the association vectors overlap, and thus the more i and j resemble each other in the way they relate to other documents, the larger the dot product, and therefore s_{ij}. This similarity measure can now be used as an input to a variety of clustering algorithms that put documents together in classes depending on how similar/dissimilar they are from each other.

Having the symmetric association matrix, the clustering phase is conducted in the following steps:

1. We create a similarity matrix between web pages where the distance (similarity) between pages is either zero, if the two pages are directly linked in the web site structure (i.e. there is a hyperlink from one to the other) or set to the co-occurrence frequency between the two pages in matrix S otherwise.

2. A graph G is created in which each page is a node and each nonzero cell in the similarity matrix is an edge. In order to reduce noise, we apply a threshold to remove edges corresponding to low co-occurrence frequency.

3. The graph created in previous step is partitioned using graph partitioning tool MeTiS for minimizing the number of cut edges. The generated clusters will be used to extend the recommendation set.

4.2. Generating the Seed Recommendation Set

The main drawback of first algorithm (DLA Recommender algorithm) is that the computation of recommendation set is time consuming and limits the algorithms'

performance. The same problem exists in the second algorithm where the process of matching current users' session with all of the generated rules needs a lot if time. So, we use another method based on idea in [124] to forward the process more intelligently. The idea behind our method is to limit the candidate set which must be considered for recommendation and decrease the online time of recommendation.

In order to facilitate the search for candidate recommendation set and improve the recommendation efficiency we use a Weighted Itemset Graph. Fig. 4 gives an example of the Weighted Itemset Graph. The idea comes from [40], in which the data structure is called the Frequent Itemset Graph because the itemsets stored in it are frequent itemsets. Each node stores an itemset along with its weighted support. The graph is organized into levels from 0 to k, where k is the maximum size among all frequent itemsets. Each node at depth d in the graph corresponds to an itemset, I, of size d. For a node N containing itemset I, each child node of N corresponds to a significant itemset $I \cup \{p\}$ at level d+1. The single root node at level 0 corresponds to the empty itemset. To be able to match different orderings of an active session with frequent itemsets, all itemsets are sorted in lexicographic order before being inserted into the graph. The user's active session is also sorted in the same manner before matching with patterns.

Given an active user session window w, we first modify the current session based on the method proposed in section 3.3.1 to obtain the modified window w', sorted in lexicographic order, and then a depth-first search of the Weighted Itemset Graph is performed to level $|w'|$. If a match is found, then the children of the matching node N containing w' are used to generate candidate recommendations. Each child node of N corresponds to a frequent itemset $w' \cup \{p\}$. In each case, the page p is added to the recommendation set if the weighted support ratio $\frac{wsp(w' \cup \{p\})}{wsp(w')}$ is greater than or equal to α, where α is a minimum confidence threshold. Note that $\frac{wsp(w' \cup \{p\})}{wsp(w')}$ is the weighted confidence of the association rule $w' \cup \{p\}$. Since the modification of active user session window changes the order of pages visited by the user, so we add the impact of transition probabilities between pages in the final score of each page p as follows:

$$score(p) = \frac{wsp(w' \cup \{p\})}{wsp(w')} \prod_{<u,v> \in w'} p(u,v) \quad (26)$$

4.3. Extending the Seed Set and Apply HITHS

The most of recommendation algorithms suffer from two major drawbacks. First, with increasing the size of recommendation set, the precision decreases significantly. Second, some resources such as rarely visited or newly added page are out of recommendation consideration. 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. We need to provide an opportunity for these rarely visited or newly added pages to be included in the recommendation set. Otherwise, they would never be recommended. To alleviate these problems, we propose a novel method.

We use the seed recommendation set generated in previous step as the input of this step. We extend the seed set to generate a candidate recommendation set. Initially, we put all of the pages in seed set in the candidate set. For each page p in the seed set, the candidate set is supplemented with pages that are in the same cluster with page p. The clusters generated in the subsection 4.1. Since the pages in each cluster have strong association based on users' behavior, this extension sounds good. We generate a graph from pages included in the candidate set by connecting them with links exist in the underlying site structure. 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 [50] to find the authority and hub pages. We take advantage of the built connectivity graph by clustering 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. Authority and hub are mutually reinforcing [50] 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 [47]. This is because a newly added page would be unlikely to be a good authoritative page, since not many pages are linked 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 on-line module to create the final recommendation set.

Generation of final set is the online process of recommendation system and must be conducted efficiently. The performance of this step strongly depends on the size of seed set. By increasing the size of seed set, the generated candidate set will be large and needs more time to compute the rank of pages. We set the size of seed set to 1 in our experiments to speedup this process.

5. Experimental Results

5.1 Data Sets

In this section we present a set of experiments that we performed for evaluating the impact of our proposed techniques on the prediction process. Overall our experiments have verified the effective of our proposed techniques in web page recommendation.

We are using the web access logs of the DePaul University CTI Web server [48], based on a random sample of users visiting the site for a 2 week period during April 2002 (DePaul Web Server Data). This dataset contains 13745 distinct user sessions of length more than 1 and 683 distinct pages. We treat each session as a transaction. Each transaction contains a sequence of pages along with their weights (durations). We split the data sets in two non-overlapping time windows to form training and a test data set. Randomly, 80% of the data set selected for training set whiles another 30% for testing.

For our evaluation, we presented each user session to the recommendation system, and the system recorded the recommendations it made after seeing each page the user had visited. The system was allowed to make n recommendations in each step with $n < 10 \quad and \quad n < \sqrt{l}$, where l is the number of outgoing links of the last page visited by the user. This limitation on number of recommendations is adopted from [49].

5.2 Experimental Methodology and Metrics

In order to evaluate the recommendation effectiveness for our method, we measured the performance of proposed method using 2 different standard measures, namely, Precision, Coverage [6], Recommendation precision and coverage are two metrics quite similar to the precision and recall metrics commonly used in information retrieval literature. Recommendation precision measures the ratio of correct recommendations (i.e., the proportion of relevant recommendations to the total number of recommendations), where correct recommendations are the ones that appear in the remaining of the user session. For each visit session after considering each page p, the system generates a set of recommendations $R(p)$. To compute the Precision, $R(p)$ is compared with the rest of the session $T(p)$ as follows:

$$\Pr ecision = \frac{T(p) \cap R(p)}{R(p)} \tag{27}$$

Recommendation coverage on the other hand shows the ratio of the pages in the user session that the system is able to predict(i.e., the proportion of relevant recommendations to all pages that should be recommended) before the user visits them:

$$Coverage = \frac{T(p) \cap R(p)}{T(p)} \tag{28}$$

5.3 Results and Discussions

In all experiments we measured both precision and coverage of recommendations against varying number of recommended pages from 1 to 11. To consider the impact of

window size (the portion of user histories used to produce recommendations), we performed all experiments using window sizes from 1 to 4.

5.3.1 Impact of Active Window Size on User Navigation Trail

In all experiments we measured both Precision and Coverage of recommendations against varying umber of recommended pages. In our state definition, we used the notion of N-Grams by putting a sliding window on user navigation paths. The implication of using a sliding window of size w is that we base the prediction of user future visits on his w past visits. The choice of this sliding window size can affect the system in several ways. To consider the impact of window size (the portion of user histories used to produce recommendations) on the DLA Personalization algorithm, we also vary window sizes $|w|$ from 1 to 4.

The impact of different window sizes on precision scores of recommendations against varying numbers of recommended pages from 1 to 12 is depicted in Fig. 5. A large sliding window seems to provide more information to the system while on the other hand causing a larger state space with sequences that occur less frequently in the usage logs. We evaluated our performance system with different window sizes on user trail as seen in Fig. 5.

As our experiments show the best results are achieved when using a window of size 3. It can be inferred form this diagram that a window of size 1 $|w|=1$ which considers only the user's last page visit does not hold enough information to make the recommendation, the accuracy of recommendations improve with increasing the window size and the best results are achieved with a window size of 3 $|w|=3$. As shown in Fig. 5 using a window size larger than 3 results in weaker performance, it seems to be due to the fact that, as mentioned above, in these models, states contain sequences of page visit that occurring less frequently in web usage logs, causing the system to make decisions based on weaker evidence.

Fig. 6 shows the impact of window size on precision of the weighted association rule Recommender algorithm. The results show clearly that precision increases as a larger

portion of user's history is used to generate recommendations. It can be inferred form this diagram although at higher number recommendation pages the difference between various window sizes becomes smaller.

5.3.2 A Comparison with Other Methods

As our experiments on the previous section show the algorithms are achieved the best results when using a window of size 3 and the other hand the mean transaction length of the data is 3, in these experiments we used a fixed window size of 3 on recommendation history (set the window size to 3).

We first compared three recommender systems, DLA Recommender algorithm, weighted association rule Recommender algorithm and Hybrid algorithm. The Recommendation Accuracy and coverage of the three systems are depicted in Fig. 7.

In the experiment, we varied the number of recommended pages to test the trend and consistency of the system quality. Fig. 7 shows the Recommendation Accuracy of the three contenders. As expected, the accuracy decreases when we increase the number of recommendation page. The consistent best performance of Hybrid illustrates the validity of usage and connectivity information to improve recommendations in our hybrid system, and also indicates that weighted association rule is more useful for recommendation accuracy improvement.

The coverage of the three systems are depicted in Fig. 8. We notice that with the increase of the number of recommended pages, Hybrid can achieve an increasingly superior result compared to both DLA and Weighted, while the two systems keep similar performance in terms of coverage. This figure verifies our justification for using two algorithms in building a hybrid recommender system.

We observed our system performance in comparison with association rules, which is commonly known as one of the most successful approaches in web mining based recommender systems [40]. Fig. 9 and Fig. 10 have shown the comparison of Hybrid system's performance with AR method in the sense of their accuracy and coverage in different number of recommended pages on CTI dataset. As the number of

recommendation page increases, naturally precision decreases in all systems, but our system gains much better results than the association rule algorithm. It can be seen the rate in which precision decreases in our algorithm is lower than traditional association rule algorithm. Experimental results show that the Hybrid model increases coverage and precision significantly and our system gains much better results than the traditional association rule algorithm.

It can be concluded that Hybrid approach is capable of making web recommendation more accurately and effectively against the conventional method. In summary, this experiment shows that our system can significantly improve the quality of web site recommendation by combining the two information channels, while each channel included contributes to this improvement.

By combining similarity between rules and active user and confidence of the weighted rules, the recommendation engine has selected only the most relevant pages. Therefore, it increases the effectiveness of the recommendation engine.

6. Conclusion

In this paper we proposed new methods for web page recommendation. First, we proposed an algorithm based on distributed learning automata to learn the behavior of previous users' and integrating web usage mining with link analysis techniques for assigning probabilities to the web pages based on their importance in the web site's navigational graph and makes recommendations primarily based on learned pattern and the structure of the web site. By introducing a novel Weighted Association Rule mining algorithm, we present our second algorithm for recommendation purpose. In which users' navigational patterns are automatically extracted from web usage data. These navigational patterns are then used to generate recommendations based on a user's current status. The pages in a recommendation list are ranked according to their importance and similarities, which is in turn computed based on web usage information. Also, a novel method is proposed to pure the current session window.

One of the challenging problems in recommendation systems is dealing with unvisited or newly added pages. So, third improvement was hybridization of the efficiency of first two algorithms.

We present a hybrid algorithm based on distributed learning automata and Weighted Association Rule mining algorithm. In the hybrid algorithm we employ the HITS algorithm to extend the recommendation set. Our experimental results illustrate that using this hybrid algorithms in a web recommender system has the potential to improve the quality of the system and can generate higher quality recommendations than using either the distributed learning automata recommendation or the Weighted Association Rule mining recommendation algorithm alone.

7. References

[1] S. S. Anand, B. Mobasher, "Intelligent Techniques in Web Personalization", Lecture Notes in Artificial Intelligence, Springer-Verlag, Berlin, Germany, vol. 3169, 2005, pp. 1–37.

[2] M. Mulvenna, M. Anand, A. G. Bunchner, " Personalization on The Net Using Web mining", Commun, ACM, 2000, pp. 8-43.

[3] B. Mobasher, R. Cooley, J. Srivastava, "Automatic Personalization based on Web Usage Mining", Communications of the ACM, vol. 43 no. 8, 2000, pp. 142-151.

[4] R. Sinha, K. Swearingen, " Comparing Recommendations Made by Online Systems and Friends", In Proceedings of the Delos-NSF Workshop on Personalization and Recommender Systems in Digital Libraries, 2001.

[5] B. Smyth, P. Mcclave, "Similarity Vs Diversity", In Proceedings of the 4th International Conference on Case-Based Reasoning: Case-Based Reasoning Research and Development, 2001, pp. 347–361.

[6] C. Ziegler, G. Lausen, L. Schmidt-Thieme, " Taxonomy-Driven Computation of Product Recommendations", In Proceedings of the ACM Conference on Information and Knowledge Management, 2004, pp. 406–415.

[7] C. Ziegler, S. M. Mcnee, J. A. Konstan, G. Lausen, "Improving Recommendation Lists Through Topic Diversification", In Proceedings of the 14th International Conference on the World Wide Web, 2005, pp. 22–32.

[8] P. Resnick, N. Iacovou, M. SUSHAK, P. Bergstrom, J. Riedl, "Grouplens: An Open Architecture for Collaborative Filtering of Netnews", In Proceedings of the 1994 Computer Supported Collaborative Work Conference, 1994.

[9] M. O. Mahony, N. Hurley, N. Kushmerick, G. Silverstre, "Collaborative Recommendations: A Robustness Analysis", ACM Trans, Internet Tech, vol. 4, no. 4, 2004, pp. 344–377.

[10]B. Sarwar, G. Karypis, J. Konstan, J. Riedl, "Item-Based Collaborative Filtering Recommendation Algorithms", In Proceedings of the 10th International World Wide Web Conference, Hong Kong, 2001.

[11] J. Srivastava, R. Cooley, M. Deshpande, P. Tan, "Web Usage Mining: Discovery and Applications of Usage Patterns from Web Data", SIGKDD Explorations, vol. 1, no. 2, 20001, pp.2-23.

[12] R. Cooley, B. Mobasher, J. Srivastava, "Web Mining: Information and Pattern Discovery on The World Wide Web", Proceedings of IEEE International Conference Tools With AI, 1997, pp. 558–567.

[13] M. Eirinaki, M. Vazirgiannis, "Web Mining for Web Personalization", ACM Transactions on Internet Technology, vol. 3, no. 1, 2003, pp. 1–27.

[14] X. Fu, J. Budzik, K. Hammond, "Mining Navigation History for Recommendation", In Proceedings of The Fifth International Conference on Intelligent User Iinterfaces, 2000, pp. 106–112.

[15] M. Gery, H. Haddad, "Evaluation of Web Usage Mining Approaches For User's Next Request Prediction", Proceedings of The Fifth ACM International Workshop on Web Information and Data Management, 2003, pp. 74–81.

[16] M. D. Mulvenna, S. S. Anand, A. G. Buchner, "Personalization On the Net Using Web Mining", Communications of the ACM, vol. 43, no. 8, 2000, pp.123–125.

[17] C. Shahabi, A. Zarkesh, J. Abidi, V. Shah, “Knowledge Discovery from User's Web-page Navigation”, In Proceedings of The 7th IEEE Intl, Workshop on Research Issues in Data Engineering, 1997.
[18] A. M. Wasfi, “Collecting User Access Patterns for Building User Profiles and Collaborative Filtering”, In: IUI ’99: Proceedings of The 1999 International Conference on Intelligent User Interfaces, 1999.
[19] M. Deshpande, G. Karypis, ”Item-Based Top-N Recommendation Algorithms”, ACM Transactions on Information Systems (TOIS), 2004.
[20]J. Herlocker, J. Konstan, A. Brochers, J. Riedel, “An Algorithmic Framework for Performing Collaborative Filtering”, Proceedings of 200 Conference on Research and Development in Information Retrieval, 2000.
[21] B. Mobasher, “Web Usage Mining and Personalization”, In Practical Handbook of Internet Computing, Munindar, P. Singh (ed.), CRC Press, 2005.
[22] K. Narendra, M. A. L. Thathachar, "Learning Automata: An Introduction", Prentice Hall, Englewood Cliffs, New Jersey, 1989.
[23] M. A. L. Thathachar, R. Harita Bhaskar, “Learning Automata with Changing Number of Actions”, IEEE Transactions on Systems Man and Cybernetics, vol. 17, no. 6, Nov. 1987, pp. 1095-1100.
[24] M. R. Meybodi, H. Beigy, "Solving Stochastic Path Problem Using Distributed Learning Automata", Proceedings of The Sixth Annual International CSI Computer Conference, CSICC2001, Isfahan, Iran, 2001, pp. 70-86.
[25] M. R. Meybodi, H. Beigy, "Solving Stochastic Shortest Path Problem Using Monte Carlo Sampling Method: A Distributed Learning Automata Approach", Springer-Verlag Lecture Notes in Advances in Soft Computing: Neural Networks and Soft Computing, 2003, pp. 626-632.
[26] H. Beigy, M. R. Meybodi, "A New Distributed Learning Automata Based Algorithm For Solving Stochastic Shortest Path Problem", Proceedings of the Sixth International Joint Conference on Information Science, Durham, USA, 2002, pp. 339-343.
[27] L. Page, S. Brin, R. Motwani, T. Wingord, “The PageRank Citation Ranking: Bringing Order to the Web”, Stanford University, 1998.
[28] A. N. Langville, C. D. Meyer, “Deeper Inside PageRank”, Internet Mathematics, 2004, pp. 335–400.
[29] B. Mobasher, H. Dai, T. Luo, M. Nakagawa, “Discovery and Evaluation of Aggregate Usage Profiles for Web Personalization”, Data Mining and Knowledge Discovery, 2002, pp. 61-82.
[30] H. Liue, V. Keselj, ”Combined Mining of Web Server Logs and Web Contents for Classifying User Navigation Patterns and Predicting Users’ Future Requests”, Data & Knowledge Engineering, 2007.
[31] C. Shahabi, A. Zarkesh, J. Abidi, V. Shah, “Knowledge Discovery from User's Web-page Navigation”, In Proceedings of The 7th IEEE Intl, Workshop on Research Issues in Data Engineering, 1997.
[32] R. Cooley, B. Mobasher, J. Srivastava, “Data Preparation for Mining World Wide Web Browsing Patterns”, In Journal of Knowledge and Information Systems, 1999, pp. 5–32.
[33] P.K. Chan, “A Non-Invasive Learning Approach to Building Web User Profiles”, In: Workshop on Web usage Analysis and User Profiling, Fifth International Conference on Knowledge Discovery and Data Mining, San Diego, 1999.
[34] S. Dumais, T. Joachims, K. Bharat, A. Weigend, “Implicit Measures of User Interests and Preferences”, 2003 Workshop Report: ACM SIGIR Forum, Fall 2003.
[35]Y. Liang, L. Chunping, “ Incorporating Pageview Weight into an Association-Rule-Based Web Recommendation System”, Springer-Verlag Berlin Heidelberg, AI 2006, LNAI 4304, 2006, pp. 577 – 586.
[36] M. Morita, Y. Shinoda, “Information Filtering Based on User Behavior Analysis and Best Match Text Retrieval”, In: Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Springer-Verlag, New York, Inc., Dublin, Ireland, 1994, pp. 272–281.
[37] R. Agrawal, T. Imielinski, A.Swami, “Mining Association Between Sets of Items in Massive Database”, International Proceedings of the ACM–SIGMOD International Conference on Management of Data, 1993, pp. 207–216.
[38] R. Agrawal, R. Srikant, “Fast Algorithms for Mining Association Rules in Large Databases”, In Proceedings of the 20th International Conference on Very Large Data Bases VLDB'94,Santiago, Chile, 1994, pp. 487-499.
[39] J. Srivastava, R. Cooley, M. Deshpande, P. Tan, “Web Usage Mining: Discovery and Applications of Usage Patterns from Web Data”, SIGKDD Explorations, vol. 1no. 2, 20001, pp.2-23.
[40] B. Mobasher, H. Dai, T. Luo, M. Nakagawa, “Effective Personalization Based on Association Rule Discovery from Web Usage Data”, In Proceedings of the 3rd ACM Workshop on Web Information and Data Management (WIDM01), Atlanta, Georgia, November 2001.

[41] B. Mobasher, H. Dai, T. Luo, M. Nakagawa, ”Improving the Effectiveness of Collaborative Filtering on Anonymous Web Usage Data”, In Proceedings of the IJCAI 2001Workshop on Intelligent Techniques for Web Personalization (ITWP01), August 2001.
[42] M. Nakagawa, B. Mobasher, “A Hybrid Web Personalization Model Based on Site Connectivity”, In The Fifth International WEBKDD Workshop: Web mining as a Premise to Effective and Intelligent Web Applications, 2003, pp. 59 – 70.

[43]Y. Liang, L. Chunping, "Incorporating Pageview Weight into an Association-Rule-Based Web Recommendation System", Springer-Verlag Berlin Heidelberg, AI 2006, LNAI 4304, 2006, pp. 577 – 586.
[44] V. Keselj, F. Peng, N. Cercone, C. Thomas, "N-gram-Based Author Profiles For Authorship Attribution In Proceedings of the Conference Pacific Association for Computational Linguistics, Nova Scotia, Canada, 2003.
[45] A. Tomovic, P. Janicic, V. Keselj, "N-gram-Based Classification and Hierarchical Clustering of Genome Sequences", Computer Methods and Programs in Biomedicine, 2005.
[46] Y. Miao, V. Kes˘elj, E. E. Milios, "Comparing Document Clustering Using N-grams Terms and Words", Master's thesis, Faculty of Computer Science, Dalhousie University, 2004.
[47] O. Za¨ıane, J. Li, R. Hayward, "Mission-Based Navigational Behavior Modeling for Web Recommender System", Springer-Verlag Berlin Heidelberg, 2007.
[48] http://maya.cs.depaul.edu/classes/ect584/data/cti-data.zip.
[49] J. Li, O. R. Zaiane, "Combining Usage Content and Structure Data to Improve Web Site Recommendation", 5th International Conference on Electronic Commerce and Web, 2004.
[50] J. M. Kleinberg," Authoritative Sources in a Hyperlinked Environment", Journal of
The ACM, vol. 46, no. 5, 1999, pp. 604–632.
[51] A. Bose, K. Beemanapalli, J. Srivastava, S. Sahar, "Incorporating Concept Hierarchies into Usage Mining based Recommendations", Proc. 8th WEBKDD workshop, 2006.
[52] M. Eirinaki, M.Vazirgiannis, I. Varlamis, "SEWeP: Using Site
Semantics and Taxonomy to Enhance the Web Personalization Process", in Proc: of the 9th SIGKDD Conf, 2003.
[53]M. Eirinaki, C. Lampos, S. Paulakis, M. Vazirgiannis, "Web Personalization Integrating Content Semantics and Navigational Patterns", In Proceedings of the sixth ACM workshop on Web Information and Data Management WIDM, 2004.
[54] J.Li, O. R. Zaiane, "Combining Usage, Content and Structure
Data to Improve Web Site Recommendation", 5th International Conference on Electronic Commerce and Web, 2004.
[55] B. Mobasher, H. Dai, T. Luo, Y. Sun, J. Zhu, "Integrating Web Usage and Content Mining for More Effective Personalization", In EC-Web, 2000, pp. 165–176.
[56] C.H. Cai, A.W.C. Fu, C.H. Cheng, W.W. Kwong," Mining Association Rules with Weighted Items", In Database Engineering and Applications Symposium, Proceedings IDEAS'98, July 1998, pp. 68 – 77.
[57] R. Burke," Hybrid Recommender Systems: Survey and Experiments", In User Modeling and User-Adapted Interaction, 2002.
[58] F. Tao, F. Murtagh, M. Farid," Weighted Association Rule Mining using Weighted Support and Significance Framework", In Proceedings of the 9th SIGKDD Conference, 2003.

Figures Captions:

Fig. 1. Distributed learning automata
Fig. 2. Comparison between our method and traditional sliding window

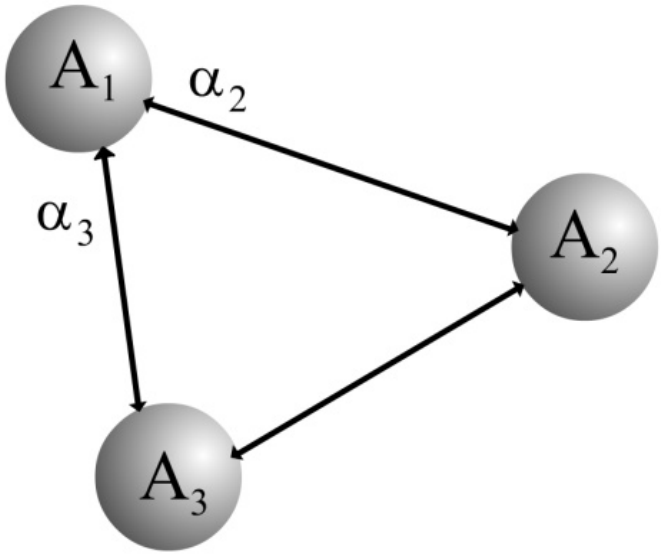

Fig. 1. Distributed learning automata

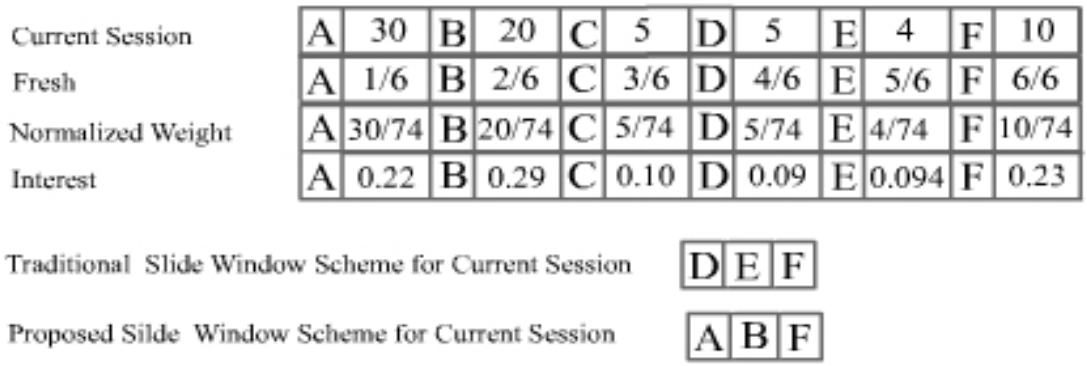

Fig. 2. Comparison between our method and traditional sliding window

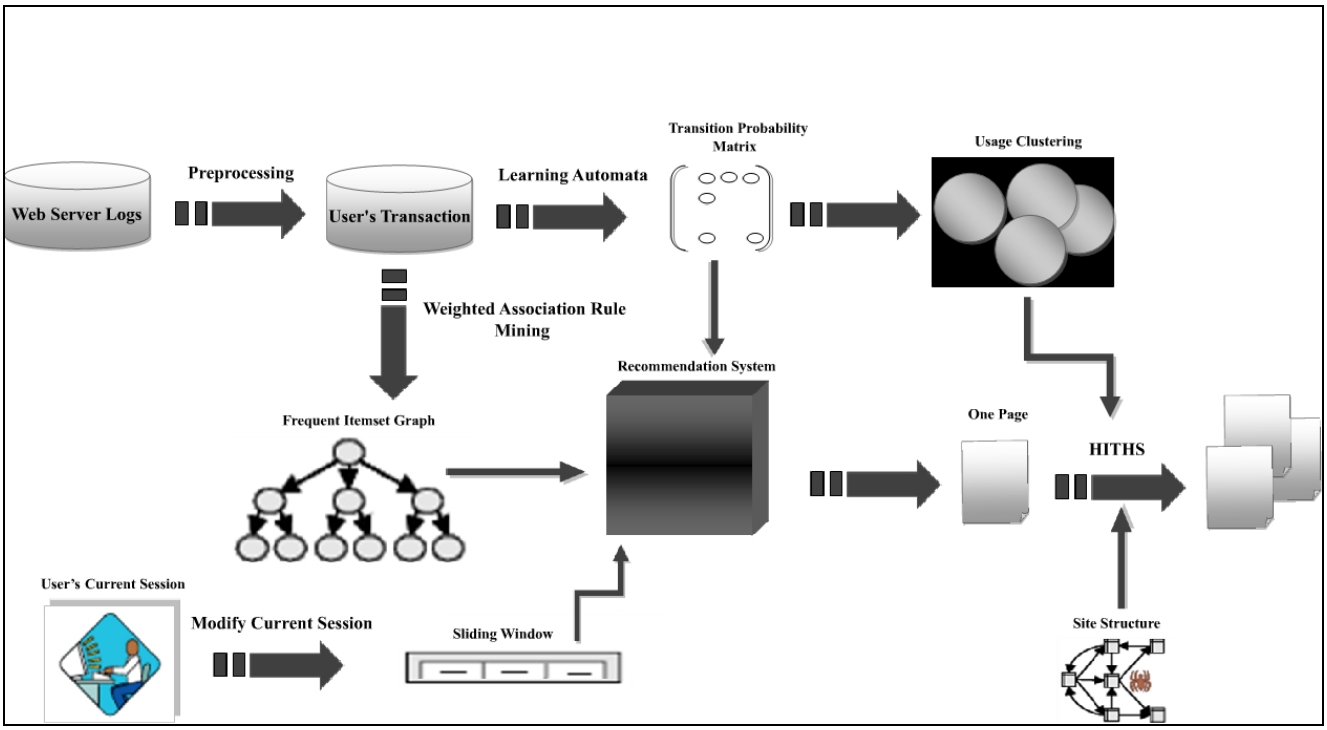

Fig. 3. Architecture of the hybrid recommender algorithm

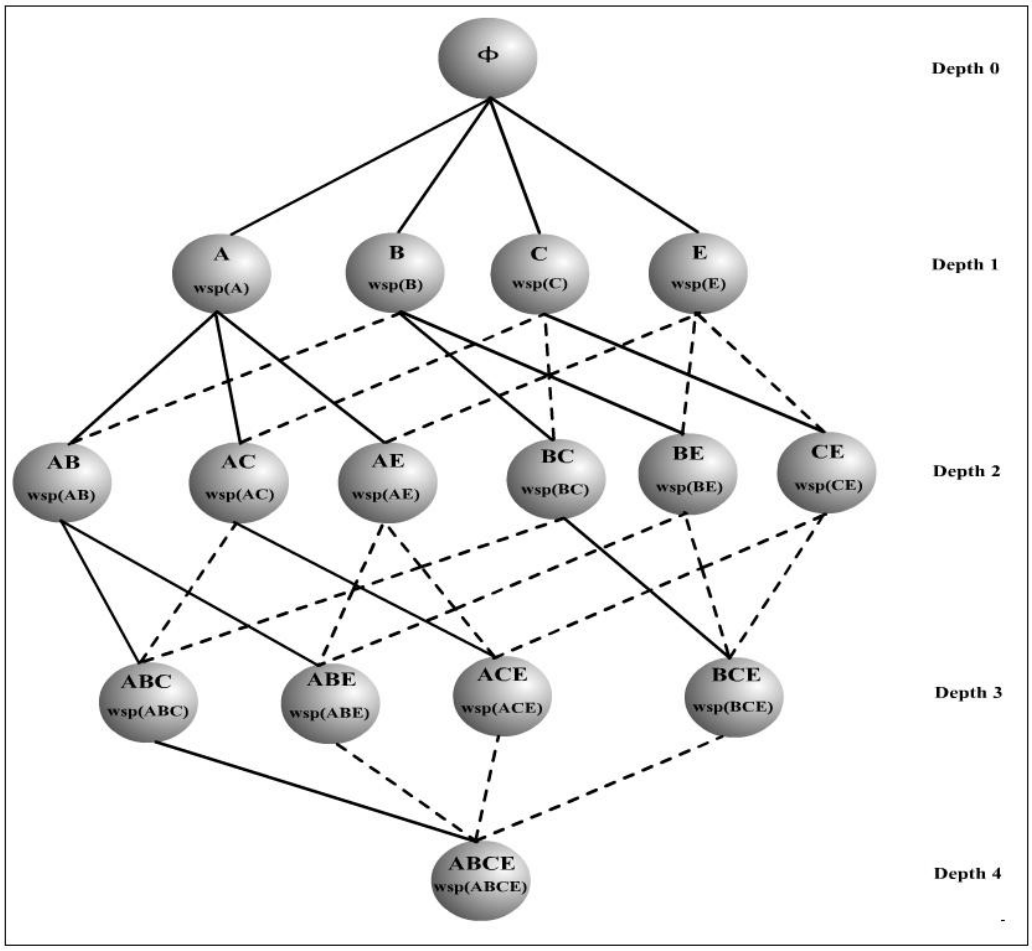

Fig. 4. The weighted itemset graph

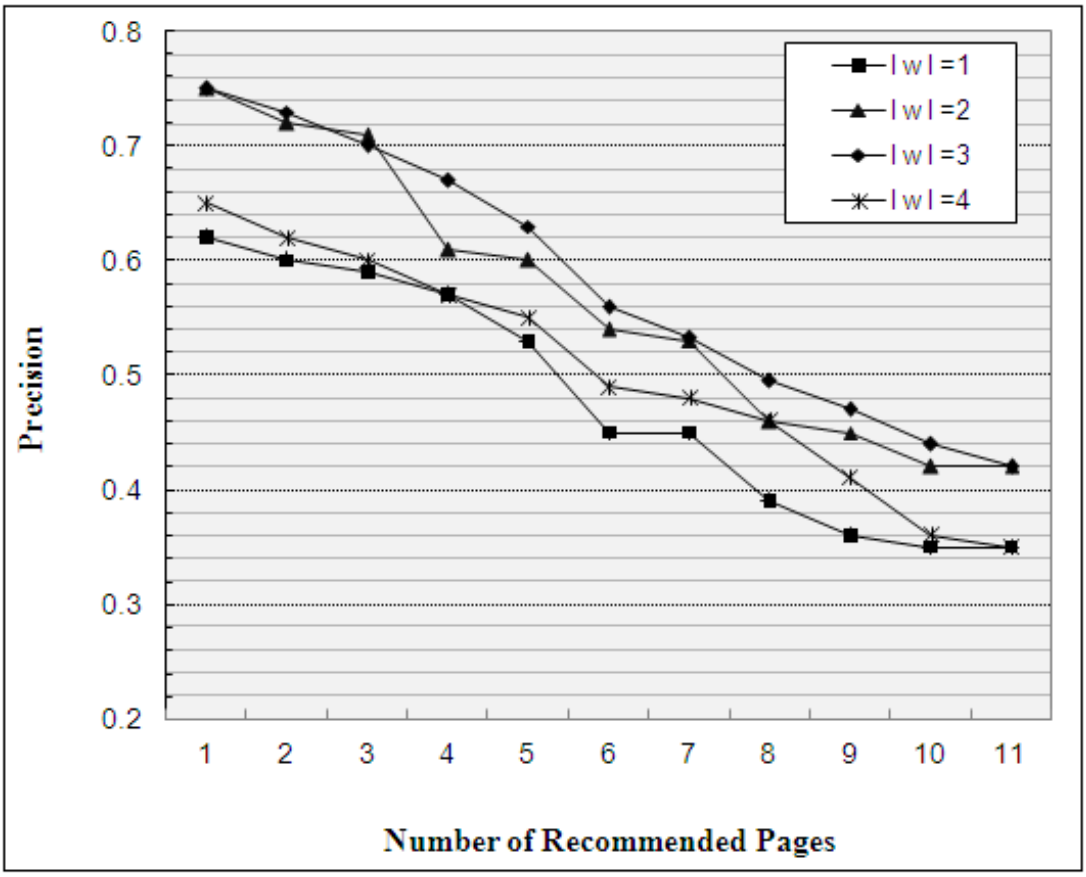

Fig. 5 DLA recommender algorithm performance with various user active windows size

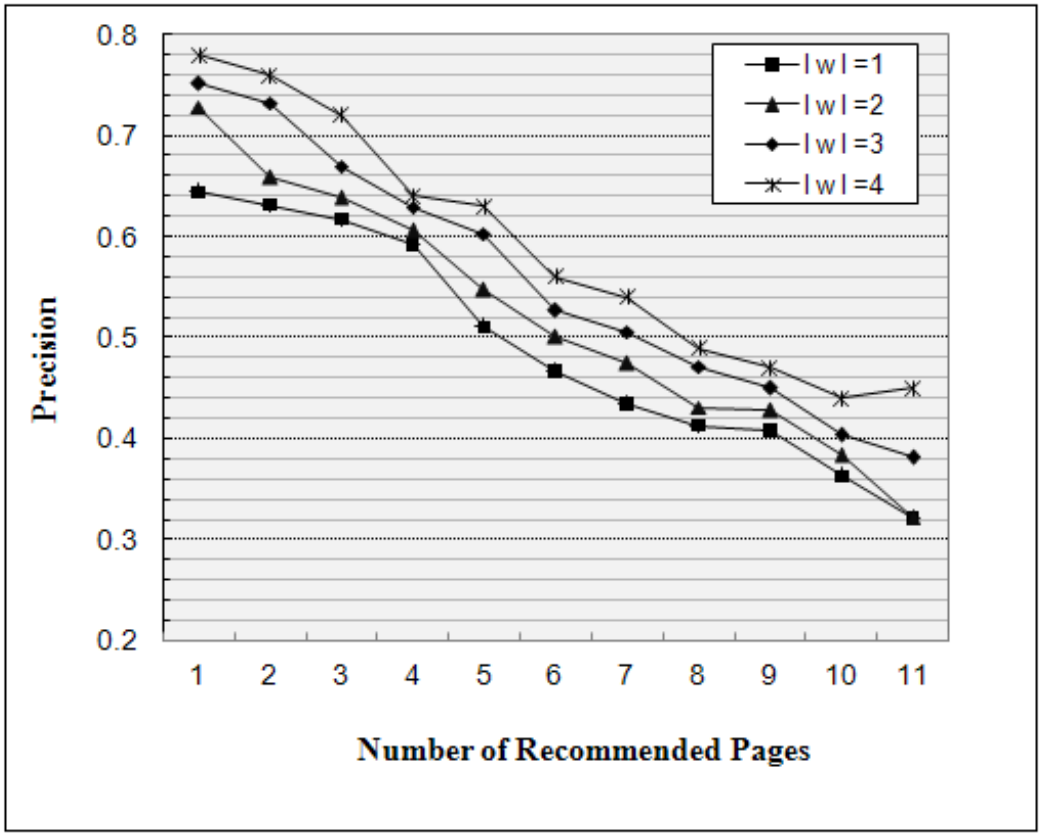

Fig. 6 Weighted association rule recommender algorithm performance with various user active windows size

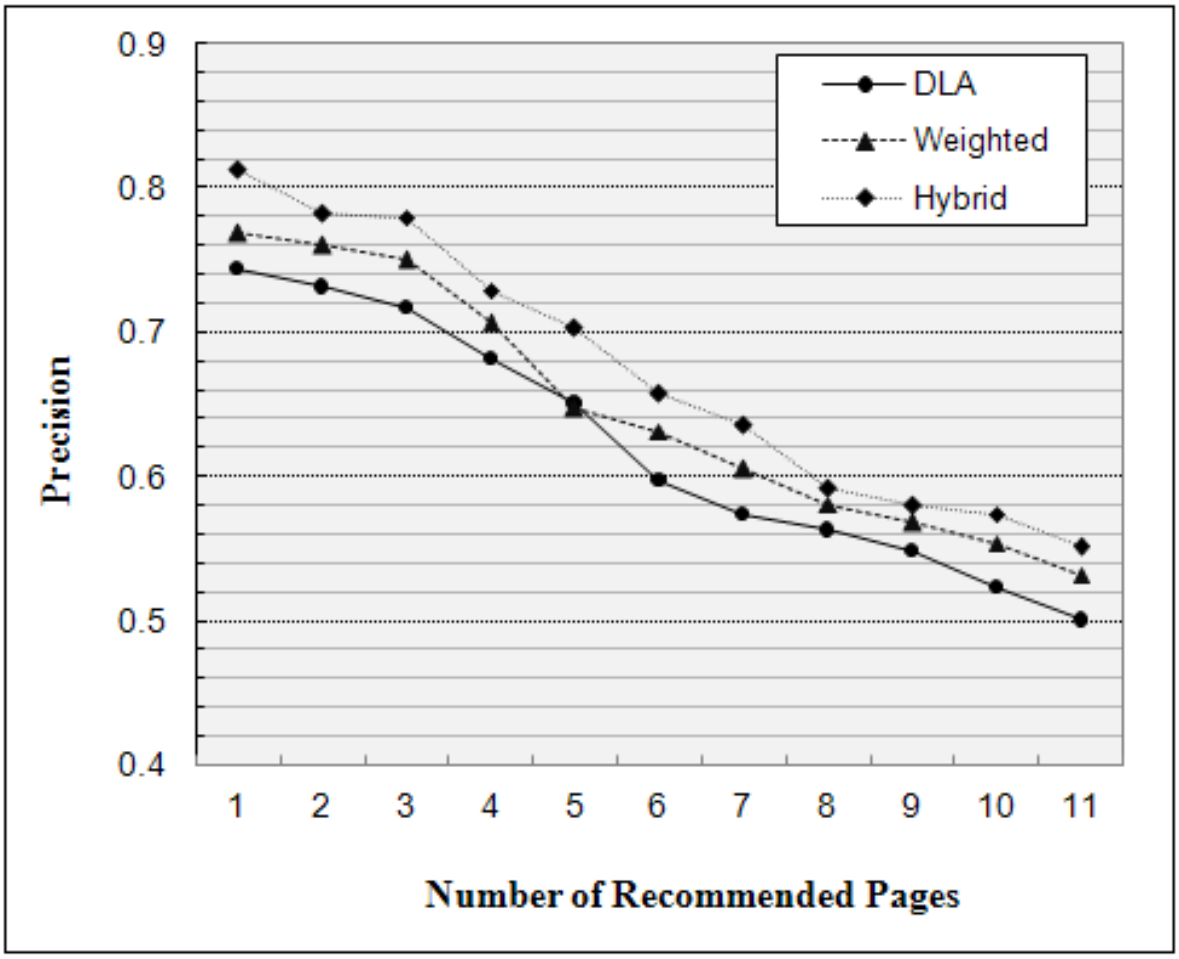

Fig. 7 Comparing the precision of proposed algorithms

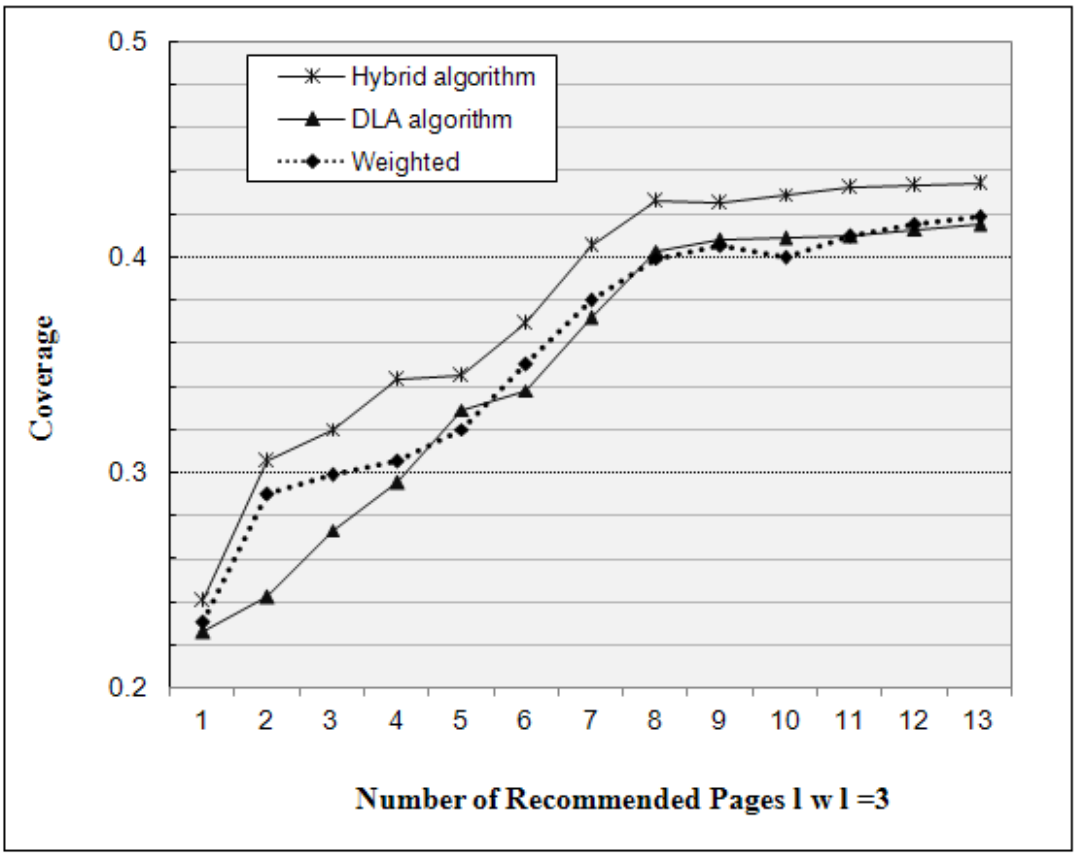

Fig. 8 Comparing the coverage of proposed algorithms

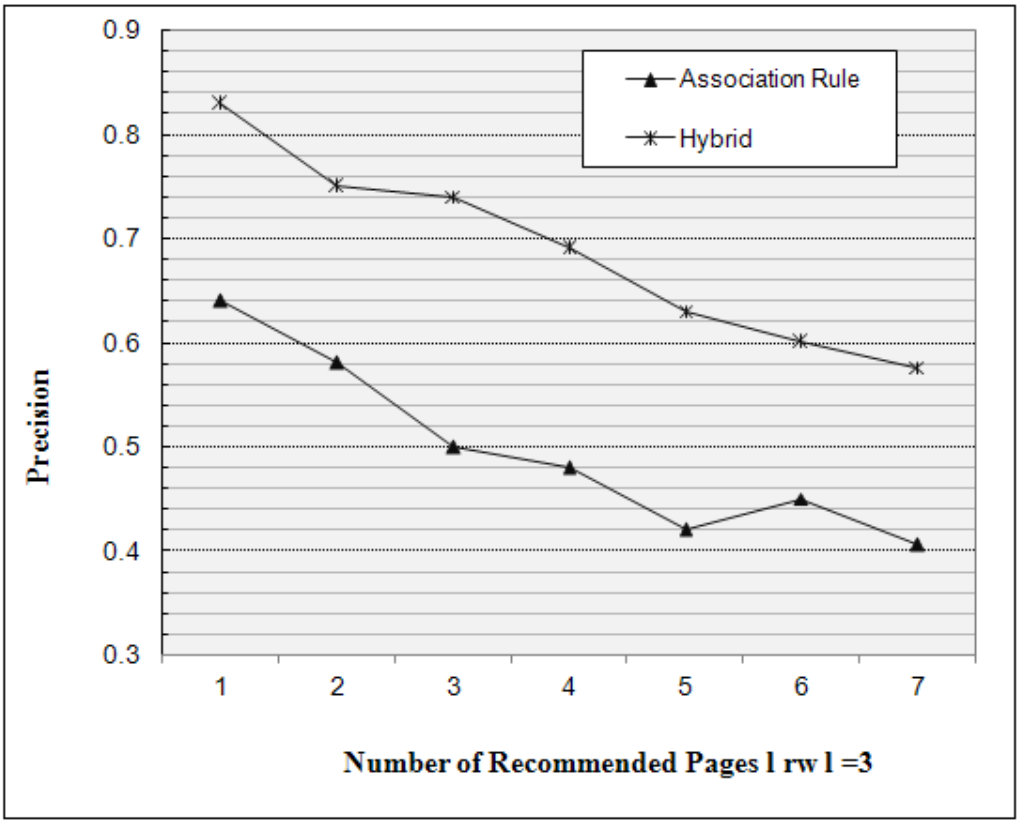

Fig. 9 Comparing our hybrid algorithm precision with association rule methods

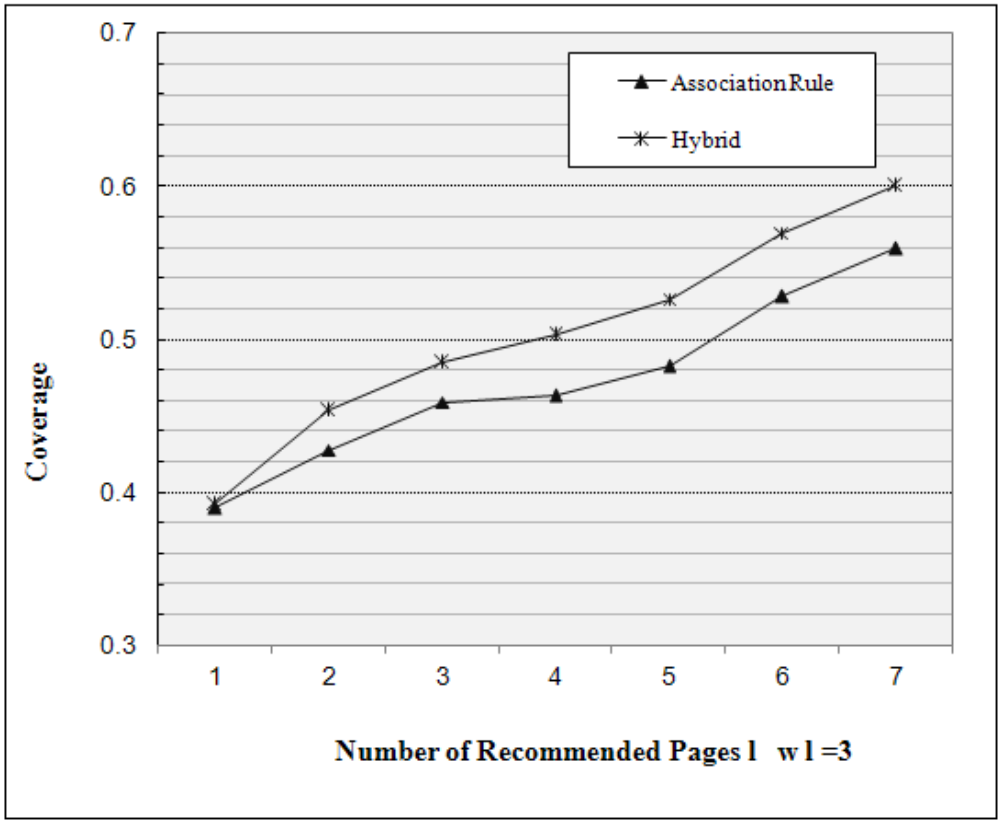

Fig. 10 Comparing our hybrid algorithm coverage with Association Rule methods