ACSIJ Advances in Computer Science: an International Journal, Vol. 4, Issue 3, No.15 , May 2015
ISSN : 2322-5157
www.ACSIJ.org

ACSIJ
WWW.ACSIJ.ORG

Determining the Similarity of Web Pages based on Learning Automata and Probabilistic Grammar

Zohreh Anari[1], Babak Anari[2]

[1] Department of Computer Engineering and Information Technology, Payame Noor University, I.R. of Iran
zanari323@yahoo.com

[2] Department of computer engineering, Shabestar Branch, Islamic Azad University
Shabestar Branch, Shabestar, Iran
anari322@yahoo.com

Abstract

As the number of web pages increases, search for useful information by users on web sites will become more significant. By determining the similarity of web pages, search quality can be improved; hence, users can easily find their relevant information. In this paper, distributed learning automata and probabilistic grammar were used to propose a new hybrid algorithm in order to specify the similarity of web pages by means of web usage data. In the proposed algorithm, a Learning Automata (LA) for each web page is assigned which its function is to evaluate association rules extracted by hypertext system. This learning process continues until the similarity of web pages are determined. Experimental results demonstrate the efficiency of the proposed algorithm over other existing techniques.

Keywords: *Web Mining, Association Rules, Learning Automata, Distributed Learning Automata, Hypertext Probabilistic Grammar.*

1. Introduction

The World Wide Web has remarkably become one of the most valuable resources for information retrievals and knowledge discoveries which are related to the permanent increase in the amount of online available data. Considering web dimension, it can be mentioned that users easily get lost in the rich hyper structure of webs. In particular, due to the two problems of low precision and low recall, web users usually suffer from the difficulties of finding desirable and accurate information on the web. Web (data) mining can be partly used to solve the above mentioned problems directly or indirectly. Indeed, web mining is considered to be the means for utilizing data mining methods to induce and extract useful information from web data. It can be further divided into three types: web structure mining, web content mining and web usage mining. Web structure mining discovers useful knowledge from hyperlinks, which represent the structure of the web. Web content mining extracts or mines useful information or knowledge from web page contents. Web usage mining is the process of extracting useful information from server logs [5]. Mining and web usage mining. Web structure mining discovers useful knowledge from hyperlinks, which represent the structure of the web. Web content mining extracts or mines useful information or knowledge from web page contents. Web usage mining is the process of extracting useful information from server logs [5].

Indeed determining the similarity between web pages is a key factor for the success of many web mining applications such as recommendation systems and adaptive web sites [7, 8, 10, 11, 12, 13]. Also, the information about the similarity of web pages can be used to provide similar documents to users so that the required information about their desired use can be found. The use of structure mining on the WWW, makes it possible to determine the similar structure of web pages by clustering through the identification of underlying structure. [14, 9, 15].

This information can be used to project the similarities of web content. On the World Wide Web, thus, it can be argued that the identified similarities enable us to maintain and improve the information of a web site; consequently, access to web spiders is accelerated significantly. The majority of existing techniques with mining on log files change them to a series of statistical information which can be used as a tool for exploring the structure of Web documents. For instance, some of these techniques include the use of learning rules such as using Hebbian learning law in Bollen method [16], using Ant system such as Ant Web [17], using Distributed Learning Automata (DLA) such as [18, 19, 20], using Markovian chain [21, 22, 23], using Hypertext Probabilistic Grammar [24, 25].

The first method, proposed by Heylighen and Bollen [16], utilizes users' browsing patterns between web pages. It can modify their data structure. The rationale behind this method is that if two web pages respond to an information necessity, then, the two web pages are similar. This method assumes that users who want to choose the next step are aware of the relative of content web pages. In fact, users create virtual data communication between web

pages and surf them. However, these relationships are user's mental model and may have no physical links.

AntWeb approach [17] was inspired by the foraging behavior ant colonies; it used ant theory as a metaphor to guide users' activities in a web site. In this method, the user moves between pages and all the maintained communications strengthen its motion path. In order to create a communication mechanism between users for each document, a list of documents relating to its highest hyperlinks are placed in the document which makes it possible to use the previous user's movement. Although the two methods of AntWeb and Bollen have more advantages than older methods, they use the matrix of relationships between web pages for use in large collections; hence, regarding these methods, extensions are not suitable.

Users' move between web pages in the Markov process was investigated in [21]. In this case, each web page represents a state and the link between web pages can be considered a one-step transition from one state to the other state. Based on this model, the next move to one or multiple web pages can be predicted. The disadvantage of using Markov chain is that high computational power is required by the n^{th}power transition matrix. Nevertheless, comparisons between the proposed algorithm and other algorithms were used with respect to the calculations which result in a partial reduction of the cost.

Borges et al [24, 25] proposed a generalization of the association rule to the concept of hypertext system such as WWW; it was aimed at capturing user patterns when accessing on-line services. Since iterations become more complex for larger graphs, the performance and efficiency of this algorithm regarding running time is still exponential in the number of nodes traversed.

In particular, Saati and Meybodi [20] presented a method based on self-organizing distributed learning automata to determine the similarity of documents in a digital library. This method used a distributed learning automata corresponding to communication graph documents in digital libraries. In this method, only those documents are considered to be similar which users are travelling directly in the surfing path. Anari et al [1, 2, 3, 4] presented several algorithms based on learning automata, these algorithms use both web structure mining and web usage mining. In these methods, as the relationship between web pages are determined based on users navigation path in the web site, operations such as clustering of web pages, clustering of web access patterns and ranking of web pages are accomplished. More recently, Motamedi Mehr [18] proposed web graph structure and graph partitioning to improve web mining performance. Web pages which are connected in graph structure have high priority for surfing users and the amount of similarity between documents are calculated when the user surfs from one document to another. This method does not use any information about the content of documents; rather, it only uses user behavior to calculate the amount of similar documents with each other. Inasmuch as distributed learning automata has been used in previous methods, consequently, correlation decreases with large number of pages which can lead to undesirable results. Nevertheless, it should be maintained that a significant feature of the algorithm proposed in this paper is that correlation value does not decrease as the number of pages increase.

In this paper, a new algorithm using the concept of hypertext probabilistic grammar based on learning automata for determining the similarity of web pages is proposed. In the proposed algorithm, for determining the similarity between web pages, learning automaton is assigned to each web page which its function is to discover the relationship of that page with all the pages which are relevant to that page. Also, the amount of similarity between web pages is calculated when users surf from one page to another. This feature causes the algorithm to be used online. To compare the proposed algorithm with other algorithms, we used the model introduced in [6] rather than using the actual web pages and web user's real data. The efficiency of the proposed algorithm was examined and analyzed by comparing it with other methods. Indeed, the results of the comparisons revealed that the correlation of the proposed method is more than distributed learning automata.

The rest of this paper is organized as follows: In Section 2, learning automata and hypertext probabilistic grammar which are used for modelling user sessions in hypertext system are introduced. Section 3 describes the proposed algorithm. In section 4, after introducing the model used for simulation, the evaluation measure is described. The computer simulations are given in section 4, and section 5, concludes the paper.

2. Learning Automata

Learning Automata (LA) are adaptive decision-making devices which operate in 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 a finite number of possible actions. In each decision process, the automata select 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 the applied action of automata. The random response of environment (i.e. grade of action) is used by automata in the selection of further action. By continuing this process, the automata learn to select an action with the best grade. Learning algorithm is used by automata to select the next action based on the response of the environment. An automaton which acts on

an unknown random environment and improves its performance in a specified manner is referred to as learning automata. LA can be classified into the following major categories: Fixed Structure Learning Automata (FSLA) and Variable Structure Learning Automata (VSLA) [26]. In the following section, the variable structure learning automata which has been used in this paper is described.

LA is represented by quintuple < α, β, *p*, T > where $\alpha = \{\alpha_1, \alpha_2, ..., \alpha_r\}$, $\beta = \{\beta_1, \beta_2, ..., \beta_r\}$ and $p = \{p_1, p_2, ..., p_r\}$ are an action set with *r* actions, an environment response set, and the probability set *p* containing probabilities, each being the probability of performing every action in the current internal automaton state, respectively. *T* is the reinforcement algorithm, which modifies the action probability vector p with respect to the performed action and received response. If the responses of the environment are binary values, LA model will be P − model; nevertheless, if it takes a finite output set with more than two elements, it will take values in the interval [0, 1]; hence, such a model is referred to as Q − model. Moreover, when the output of the environment is a continuous variable in the interval [0, 1], it will be referred to as S − model. Fig.1 shows the relationship between the environment and the learning automata.

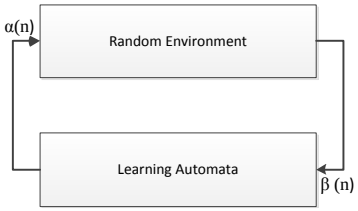

Fig. 1 The relationship between learning automata and the environment.

The linear reward-inaction algorithm is one of the learning schemas and its recurrence equation for updating action probability vector p is defined as Eq.1 and Eq.2.

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

$$p_j(n+1) = p_j(n) + a.\big(1-\beta(n)\big).p_j(n) + \frac{b.\beta(n)}{r-1} - b\beta(n).p_j(n) \quad if\ (j \neq i) \quad (2)$$

Where a ∈ [0, 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.

2.1 Distributed Learning Automata

A Distributed learning automata (DLA) shown in Fig. 2 is a network of interconnected learning automata 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 DLA activates another LA which corresponds to this action. Formally, a DLA can be defined by a graph DLA = (V, E) where $V = \{LA_1, LA_2, \dots, LA_n\}$ is the set of *n* LA and E ⊂ V ×V refers to the set of edges in the graph. The edge (i, j) represents the action j of automaton LA_i. In other words, LA_i is activated when action *j* of automaton LA_i is selected. The number of actions for particular automaton LA_k (k = 1, 2 , …, *n*) is equal to the out-degree of that node. If 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 automaton 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 automaton *j* [27, 28, 29]. For example, in Fig.2, every automaton has two actions. Selection of action α_3 by A_1 will activate automaton A_3. Activated automaton chooses one of its action which results in the activation of the LA corresponding to the selected action. At any given time, only one of the automaton in the network could be active.

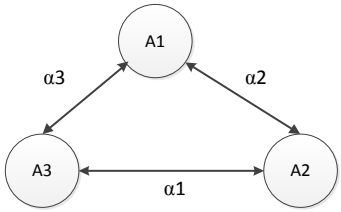

Fig. 2 Distributed Learning Automata.

2.2 Review of hypertext probabilistic grammar

A log file can be regarded as a per-user ordered set of web page requests from which it is possible to infer the user navigation sessions. The user navigation sessions inferred from the log data are modelled as a hypertext probabilistic language generated by a hypertext probabilistic grammar (or simply HPG) [30] which is a proper subclass of probabilistic regular grammars [31]. A HPG is a probabilistic regular grammar which has a one to one mapping between the set of non-terminal symbols and the set of terminal symbols. Each non-terminal symbol corresponds to a web page and a production rule corresponds to a link between pages. Moreover, there are two additional states *S* and *F* which represent the starting and finishing states of the navigation sessions. A hypertext system is a database of pages connected by oriented links

which provide a non-sequential method to access information. The hypertext system is modelled as a weighted directed graph $G = (N, E)$ constructed from the user's sessions log data where N consists of a set of nodes and E denotes a set of edges. A node, $A \in N$, refers to the entity which corresponds to a page. An edge is a pair $(A, B) \in E$, referring to the entity that corresponds to a link. The number of times a sequence of two pages appears in the sessions gives the number of times the corresponding link was traversed. As an example, a user's session can be seen in Figure 3. The weight of each edge is equal to the number of times that the edge is visited. For modelling user sessions in Hypertext system, we have [24]: ϕ *(H, M, S, F, Γ)*

H: Set of nodes, where $H = \{H_1, H_2, \dots, H_m\}$, each member would be a representative of a web page.

M: Is an $M \times M$ matrix where:

$M = p(h_i, h_j), (h_i, h_j) \in H, 0 \le p(h_i, h_j) \le 1,$
$\sum_{k=1}^{m} p(h_i, h_k) = 1$

S: Start state a user session

F: Final state a user session where $\forall\ h_i , h_j \in H$, $p(h_i , h_j) = \Gamma(h_i , h_j)$ In this case, the probability of moving from page h_i to page h_j is described as confidence and is defined as follows:

$$p(h_i, h_j) = \frac{R_j^i}{R^i} \qquad (3)$$

R_j^i: It refers to the number of times page h_i is located after page h_j.

R^i: It refers to the number of all pages after page h_i have been visited. If $< h_{i1}, h_{i2}, \dots, h_{ir} >$ is the string generated by the grammar, the probability of the string is determined by the following equation:

$$p(< h_{i1}, \dots, h_{ir} >) = \prod_{k=1}^{r-1} p(h_{ik}, h_{i(k+1)}) \quad (4)$$

In this paper, to model the user sessions in hypertext system, we used the model [24].

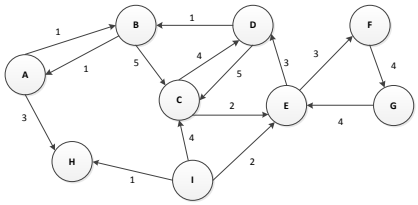

Fig. 3 Weighted graph for user session.

An Example

In the example illustrated in Fig.4 we have 6 user sessions with a total of 24 page requests. If the user moves a user session as shown in Table.1, the association rules generated by it would be: First, the user movements build a hypertext system which is illustrated in Fig.4 for each nodes we have:

In this step, mining association rules are conducted on the hypertext system and the values of all strings with probability $p(w)>$ cut-point are determined. In this example, cut-point value was considered to be 0.3 and association rules are extracted starting from A_1 is performed as follows:

From A_1 can be moved to A_2 , A_5. since $p(A_1 \to A_2) = \frac{1}{3} < \lambda$ then, the path $A_1 \to A_2$ is discarded. Then, we start fromA_5, we can go to pages A_2 and A_3. Also the path $A_1 \to A_5 \to A_3$ is discarded. Since we have $p(A_1 \to A_5 \to A_3) < {}^{2}/_{3} \times {}^{2}/_{5} < \lambda$. Finally, the extraction path starting from A_1 is as follows: $A_1 \to A_5 \to A_2$. Table 2 gives all the rules extracted for Fig.6 and Fig.5 shows the extraction paths from Table 3.

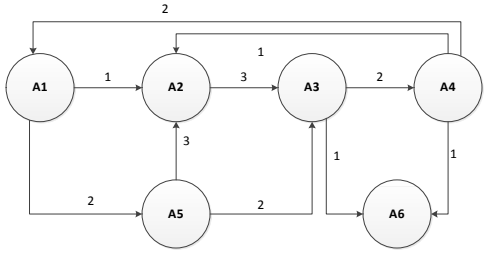

Fig. 4 Create a hypertext system from the user moves.

Table 1: An example set of user's trails

Session ID	User trail
1	$A_1 \to A_2 \to A_3 \to A_4$
2	$A_1 \to A_5 \to A_3 \to A_4 \to A_1$
3	$A_5 \to A_2 \to A_4 \to A_6$
4	$A_5 \to A_2 \to A_3$
5	$A_5 \to A_2 \to A_3 \to A_6$
6	$A_4 \to A_1 \to A_5 \to A_3$

Table 2: The probability of user's trails

A_1	A_2	A_3	A_4	A_5
$p(A_1 \to A_2) = \frac{1}{3}$	$p(A_2 \to A_3) = \frac{3}{4}$	$p(A_3 \to A_4) = \frac{2}{3}$	$p(A_4 \to A_1) = \frac{2}{3}$	$p(A_5 \to A_2) = \frac{3}{5}$
$p(A_1 \to A_5) = \frac{2}{3}$	$p(A_2 \to A_4) = \frac{1}{4}$	$p(A_3 \to A_6) = \frac{1}{3}$	$p(A_4 \to A_6) = \frac{1}{3}$	$p(A_5 \to A_3) = \frac{2}{5}$

Table 3: Extracted rules from hypertext system

Session ID	User Session
1	$A_4 \rightarrow A_1$
2	$A_5 \rightarrow A_2 \rightarrow A_3$
3	$A_5 \rightarrow A_3$
4	$A_3 \rightarrow A_4$
5	$A_2 \rightarrow A_3 \rightarrow A_4$
6	$A_1 \rightarrow A_5 \rightarrow A_2$

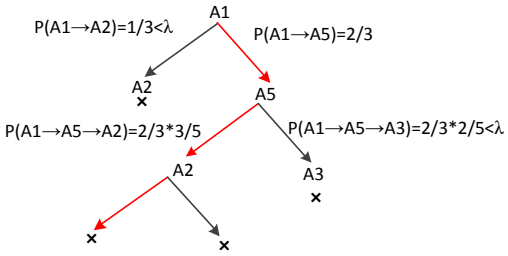

Fig. 5 Extraction of paths from Table.1

3. The Proposed Algorithm

If the number of users, number of consecutive pages requested, they might equally have responded to the needs of the pages and in this case, are related to each other. Using DLA, the relationship between the web pages according to the information the user moves between pages can be identified. As the relationship between web pages is determined, web pages can be ranked and clustered. In the proposed algorithm, to determine the relationship between web pages, LA is assigned to each web page which task is to learn the relationship of that page with all the pages which are relevant to it.

The proposed algorithm can be described as follows: web pages and users using it play the role of a random environment for LA in DLA. The output of DLA refers to a sequence of web pages browsed by a user which tracks user's movement towards a desired web page. Environment uses this sequence to generate a response to DLA. Given this response, the internal structure learning automata in DLA are updated according to learning algorithm.

In the proposed algorithm of the present study, a DLA was created for a web site with n pages. In the created DLA, there were n learning automaton. Each automaton corresponds to a particular page which has n actions. When a user moves from page p_i to p_j, action j of automaton LA_i in DLA is activated and is applied to the environment. If the selected action is the k^{th} action of LA_i automaton, then, the corresponding probability of this action, namely p_{ik} referring to the relationship between pages p_i and p_j are considered.

In the proposed algorithm, first we consider a log file to store users' movements. Then, the user moves are stored in this file. After filling the file, a hypertext system is made from it. Then, the association rules with the cut-point value are extracted from it. This extraction rules indicate the path intensity of the trail (probability value) is high. Finally, only the extracted rules will be rewarded by LA. Again, log file is cleaned; then, the new moves enter and the same procedure is performed again. The proposed algorithm is shown in the following pseudo-code and the flowchart of the proposed algorithm is shown in Fig.6.

The purpose of mining association rules is to generate those rules in the graph whose probability of extraction rules is greater than the cut-off value. From each link in the graph, a depth first search (DFS) tree is built in which each branch corresponds to a candidate composite association rule. Each DFS tree identifies all the candidate composite association rules which have the root link of the tree as their first link. The exploration of a branch ends when it finds a node such that all its outgoing-links have already been marked as visited, or when the confidence of the corresponding trail drops to a value below the specified threshold C. In addition, since each branch exploration corresponds to a new trail, it has its own independent list of visited links. Notations used in the algorithm are as follows: given a trail x, if CR represents the candidate rule, *forward neigh*(x) refers to its next non-visited forward neighbor. Given a trail x and a link e, $x + e$ denotes the concatenation of x and e.

Algorithm 1 $DLA-HPG$

```
Input:
Cut_point ← 0.01
Rules = {}
LogFile                                          ▷ this file save users sessions
a                                                ▷ Learning Parameter
N                                                ▷ Number Of Iteration
DLA          ▷ Distributed Learning Automata,each automaton has n-1 actions
for i=1:N do
    while Log File is not full do
        Add user's navigation path in Log File
    end while
    G ← Build hypertext system from Log File
    Rules ← Rules ∪ Modified_DFS(G,Cut_point)    ▷ Extract association rules
    for each extraction rule such as R_i do
        for each(Page_i → Page_j) do             ▷ Page_i is visited before Page_j
            p_m(n+1) = p_m(n) + a[1 − p_m(n)]
            p_j(n+1) = (1 − a)p_j(n), ∀j ≠ m
        end for
    end for
    Empty Log File
end for
```

Algorithm 2 $Modified\_DFS(G, C)$

```
for each e ∈ E : C_e ≥ C do
    Explore(e, C_e)
end for
```

Algorithm 3 $Explore(trail, C_{trail})$

```
CR = CR ∪ trail
for each forw_neigh(trail) do
    if C_fn × C_trail ≥ C then
        Explore(trail + f_n, C_trail)
    end if
end for
```

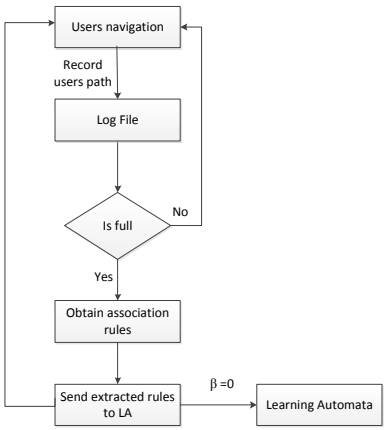

Fig. 6 Flowchart of the proposed algorithm.

4. Experimental Result

4.1 Experimental Result on Synthetic data

In this section, first, the model used for generating web usage data is explained. Then, the experimental results of the proposed algorithm are given and compared with the results of previous algorithms in the literature. In order to measure the performance of the proposed algorithm, we utilized the user log data generated by the web surfing model proposed by Liu et al [6]. In this model, which has been validated against some empirical web log data, users of a web site are considered as information foraging agents and the web page as the information resources. Each web page contains certain information contents, and each hyperlink between two web pages signifies certain content similarity between them. The contents contained in the page *i* can be characterized using a M-dimensional content vector, $cn = [cw_n^1, cw_n^2, \dots, cw_n^M,]$ where each component cw_n^t corresponds to the relative content weight of page *i* on topic *t*. The similarity between the two pages *i* and *j* is considered as the inverse of the distance between content vector of two pages *i* and *j* (d_{ij}), which is determined by the Euclidean distance of their content vectors Eq. 5.

$$d_{ij} = \sqrt{\sum_{k=1}^{M} \left(cw_i^k - cw_j^k\right)^2} \quad ,D = \{d_{ij} | \, i,j = 1,2,\dots,n\} \quad (5)$$

In this model, the distribution of topics among web pages as normal distribution and user interest profile as Power law distribution is considered. By changing distribution parameters and the number of web pages, different information systems can be created. In this model, motivation and users' movement strategy through the statistical distribution are modelled. By changing the parameters of statistical distribution, users with different interests, motivations, and strategies are established. In the conducted simulations, the number of parameters with respect to topics, number of users, interests, motivations, and their distribution parameters were kept constant throughout the simulation.

The users of the website were regarded as rational users who had specific interested topics in mind and forage so that they can locate the pages including information on those topics. When a rational user reaches a new page, he will try to decide whether or not the content sufficiently matches his information needs (denoted by his interest profile) and, if not, he predicts which page at the next level will be likely to become a more interesting one. In predicting the next page after page *i*, the user will select a page *j*, which has a link on page *i*, with a probability proportional to the similarity of his interest profile vector and content vector of page *j*. When a user does not find any interesting information after some foraging steps or has found enough contents satisfying his information needs, he will stop foraging and leave the web site. The values of the parameters of this model used in the following experiments are listed in table 4.

Table 4: Parameters of the web site and users' model

Parameter	Description	Value
r	Degree of coupling	0.7
Nu	Number of users	10000,20000
Np	Number of pages	30
Nt	Number of topics	5
T_c	The content increment offset on the topic	0.2
ΔM_t^v	The variable factor that dynamically changes at each time step	
α_u	The shape parameter of a power-law distribution	1
ϕ	Weights of reward	1.2
λ	Absorbing factor in [0,1]	0.5
μ_t	Mean of normally distributed offset T	
α_p	The shape parameter of a power-law distribution	3
σ_t	Variance of normally distributed offset T	0.25
θ	Weights of motivation	1
Mm	Minimum of motivation for the continued search	0.2

4.2 Evaluation metrics

In order to evaluate the performance of our algorithm, we used correlation. Correlation is a measure which obtains a linear dependence between two vectors. Correlation between the matrix of the distance between content vector of every two pages (i.e., X) and the matrix calculated by the proposed algorithm (i.e., Y) is measured by Eq.6, where *N* is the number of data.

$$Corr(X,Y)=\frac{Cov(X,Y)}{\sigma x \sigma y}=\frac{\sum XY-(\sum X\sum Y)/N}{\sqrt{(\sum X^2-(\sum X)^2/N)(\sum Y^2-(\sum y)^2/N)}} \quad (6)$$

$$X=\left\{P_{ij}\mid i,j=1,2,\cdots,n,\quad i\neq j\right\}, P_{ij}=\frac{(d_{ij})^{-1}}{\sum_{k=1}^{n}(d_{ik})^{-1}} \quad (7)$$

$$Y=\left\{P'_{ij}\mid i,j=1,2,\cdots,n,\quad i\neq j\right\}, P'_{ij}=\text{probability of action } j \text{ from } DLA(i)$$

4.3 Simulation Results

The results of simulation including 10000 and 20000 users are demonstrated in figures Fig.7 and Fig.8. Other parameters of the proposed algorithm are demonstrated in Table.4. The comparison of the results with other methods are shown in Fig.7 and Fig.8. As depicted in the following figures, the proposed algorithm performs better than the other three algorithms [16, 17, 19].

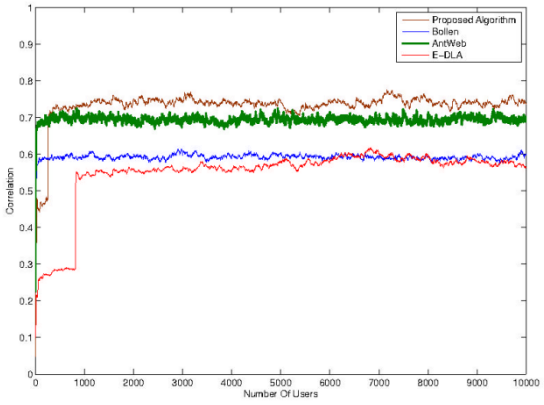

Fig. 7 Comparison of correlation between proposed and other algorithms with 10000 users.

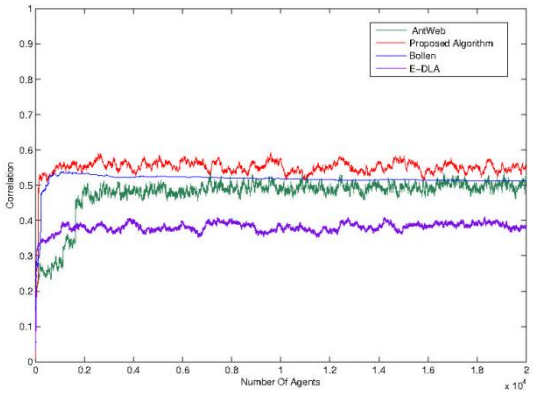

Fig. 8 Comparison of correlation between proposed and other algorithms with 20000 users.

Fig.9 illustrates the impact of cut-point parameter on correlation. As it is clearly observed in Fig.9, the cut-point value has significant impacts on correlation. In other words, as the cut-point value increases, correlation value decreases. The greater cut-point value, the shorter the length of the extracted rules. Conversely, as the cut-point value is less, the length of extracted rules is greater. The results of simulation are depicted in Fig.9. Furthermore, the impact of cut-point on time and efficiency parameters was examined and investigated in the proposed algorithm. The results of simulation are shown in Fig.10. This figure illustrates the impact of cut-point value on running time of the algorithm. As it is clearly demonstrated in Fig.10, cut-point value has a significant effect on the runtime of the algorithm. In other words, as the cut-pint value increases, the run time decreases.

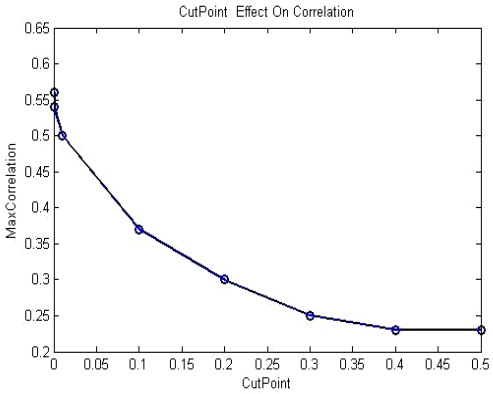

Fig. 9 The effect of cut-point on correlation.

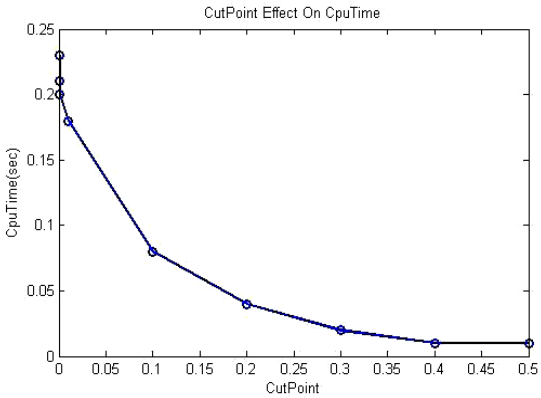

Fig. 10 The effect of cut-point on cpu time.

To evaluate the effectiveness of web pages and the number of users, another experiment was conducted. In this experiment, the maximum correlation in exchange for the number of web pages and the number of users are calculated.

Table 5 shows the maximum correlation in exchange for the number of web pages and the number of users. As it is clear from table 5, increasing the number of web pages has

an impact on performance and reduces the amount of correlation. In other words, it can be argued that as the number of pages increases, the length of automaton probability vector increases and the convergence rate decreases.

Table 5: Maximum correlation in exchange for the number of web pages and the number of users.

	# Web Pages				
# Users	20	60	100	200	300
10000	0.7641	0.7625	0.7605	0.7565	0.7510
15000	0.8511	0.8181	0.7741	0.7481	0.7427
30000	0.8632	0.8012	0.7498	0.7358	0.7135
50000	0.8511	0.8059	0.7375	0.7236	0.7111

4.4 Experimental Result on Real Data Set

The study reported in this paper was conducted on CTI server log files. This data set includes the session data for the DePaul University CTI web server, based on a random sample of users visiting the site for a two week period during April 2002 [32]. The data set included 611 distinct page views and 81745 distinct user sessions with a length which was more than one. The sessions were split into two non-overlapping time windows to form a training (2745 Sessions) and a test (47000 sessions) data set. This data set was referred to as CTI data set. After preprocessing, clean data was imported to the database and analyzed. The mining process was aimed at finding frequent item sets and discovering web user navigation patterns. To evaluate the effectiveness of web pages and the number of users, another experiment conducted. In this experiment, the maximum correlation in exchange for the number of web pages and the number of users were calculated.
Table 6 illustrates the maximum correlation in exchange for the number of web pages and the number of users. As it is clearly shown in Table 6, increasing the number of web pages significantly affects performance and reduces the degree of correlation. It should be noted that as the number of pages increases, the length of automaton probability vector increases and the convergence rate decreases.

Table 6: Maximum correlation in exchange for the number of web pages and the number of users.

	# Web Pages				
# Users	20	60	100	200	300
10000	0.6017	0.5911	0.4813	0.3142	0.2981
15000	0.6111	0.6218	0.4807	0.3159	0.2990
30000	0.6213	0.6220	0.4912	0.3201	0.3010
50000	0.6301	0.6222	0.4916	0.32206	0.30141

The simulation results including 10000 and 20000 users are depicted in Fig.11 and Fig.12. Other parameters of the proposed algorithm are shown in Table.4. The comparison of result with other methods are illustrated in Fig.11 and Fig.12. As it clearly shown in these figures, the proposed algorithm performs better than three methods [16, 17, 19].

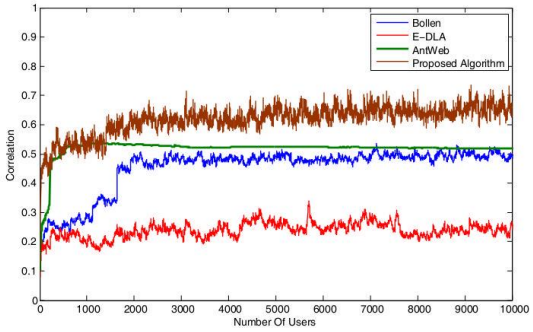

Fig. 11 Comparison of correlation between proposed and other algorithms with 10000 users.

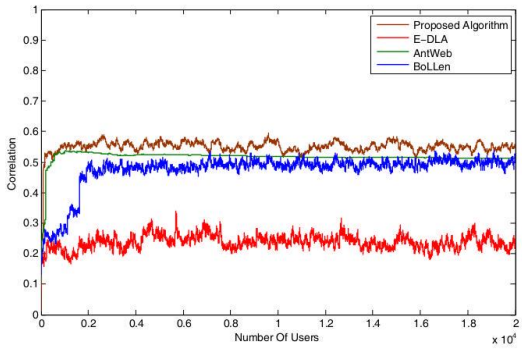

Fig.12 Comparison of correlation between proposed and other algorithms with 20000 users.

4.5 Time Complexity

It was found that by reducing the cut-point, the length of rules increases and by increasing the cut-point, the length of rules will decrease. Due to the depth first search tree (DFS), the proposed algorithm has a time complexity of $O(n)$.Thus, if the number of nodes which are in log file, n and k is the equal to the number of iterations, hence, the cost of the proposed algorithm will be equal to $O(nk)$.

5. Conclusions

In this paper, a new algorithm was proposed based on DLA and probabilistic grammar. In the proposed algorithm, a LA was used to determine the similarity of web pages. Patterns with low quality are not considered in the proposed algorithm. Consequently, it leads to a significant efficiency enhancement of the proposed algorithm in comparison with other algorithms. The results of simulations conducted in the study revealed that the degree of correlation in the proposed algorithm is higher

than those of distributed learning automata based algorithm E-DLA, Ant Web and Bollen. Moreover, the effect of the cut-point parameter on running time of the algorithm and correlation were investigated. We found that the proposed algorithm was affected by this parameter. In other words, the initial value of cut-point is a significant parameter which should be taken into consideration. The applications of the proposed algorithm include the followings: extraction association rules, predicting web pages, providing adaptive web sites for users, clustering and ranking of web pages.

Acknowledgments

This article has been supported by a grant from the Payame Noor University, I.R. of Iran.

References

[1] Z. Anari, M. R. Meybodi and B. Anari, "Web Page Ranking based on Fuzzy and Learning Automata", Proceedings of the International ACM Conference on Management of Emergent Digital EcoSystems (MEDES'81), 2009, Vol. 24, pp. 822-829.
[2] B. Anari, M. R. Meybodi and Z. Anari, "A New Method based on Distributed Learning Automata for Page Ranking in Web", Proceedings of the 13th Annual CSI Computer Conference of Iran, 2008.
[3] Z. Anari, M. R. Meybodi, and B. Anari, "Clustering Web Access Patterns Based on Learning Automata and Fuzzy Logic", Proceedings of the 3rd Iran Data Mining Conference (IDMC'81), 2009.
[4] B. Anari, M. R. Meybodi, and Z. Anari, "Clustering Web Access Patterns Based on Learning Automata", International Journal of Computer Science Issues, Vol. 8, Issues 5, No.1, 2011, pp. 60-65.
[5] B. Liu, Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data, Springer Berlin Heidelberg New York, ISBN-103-540-37881-2, 2008.
[6] J. Liu, S. Zhang, J. Yang, "Characterizing Web Usage Regularities with Information Foraging Agents", IEEE Transactions on Knowledge and Data Engineering, Vol. 16, No. 4, 2004, pp. 566-584.
[7] M. Eirinaki, M. Vazirgiannis, "Web Mining for Web Personalization", ACM Transactions on Internet Technology, Vol. 3, No. 1, 2003, pp. 1-27.
[8] B. Mobasher, O. Nasraoui, B. Liu, B. Masand, Advances in Web Mining and Web Usage Analysis, Berlin, Springer Berlin-Heidelberg, 2006.
[9] B. Mobasher, R. Cooley, J. Srivastava, "Creating Adaptive Web Sites Through Usage Based Clustering of URLs", Proceedings of the IEEE Knowledge and Data Engineering Work-shop Exchange (KDEX'11), 1999, pp. 19-29.
[10] M. Mulvenna, M. Anand, A. G. Bunchner, "Personalization on the Net Using Web mining", Communication of the ACM, 2000, pp. 843-890.
[11] M. Perkowitz, O. Etzioni, "Adaptive Web Sites", Communications of ACM, Vol. 43, No. 8, 2000, pp. 152-158.
[12] S. S. Anand, B. Mobasher, "Intelligent Techniques in Web Personalization", In Lecture notes in Artificial Intelligence, Vol. 3, 2005, pp. 137.
[13] G. Xu, Y. Zhang, L. Li, Web Mining and Social Networking, Techniques and Applications. New York, Springer-Verlag, London, 2010.
[14] S. Chakrabarti, B. Dom, D. Gibson, J. Kleinberg, R. Kumar, P. Raghavan, S. Rajagopalan, A.Tomkins, " Mining the Link Structure of the World Wide Web", IEEE Computer. Vol. 32, 1999, pp.60-67.
[15] W.W. Cohen, "Learning and Discovering Structure in Web Pages", Bulletin of the IEEE Computer Society Technical Committee on Data Engineering, Vol. 26, No.1, 2003, pp. 3-10.
[16] F. Heylighen, J. Bollen, "Hebbian Algorithms for a Digital Library Recommendation System", Proceedings of the International Conference on Parallel Processing Workshops (ICPPW83), 2002, pp. 439-446.
[17] W.Teles, Weigang, L., Ralha, C., "AntWeb-The adaptive web server based on the ants behavior", Procedings of the IEEE/WIC International Conference on Web Intelligence (WI'81), 2003, pp.558- 564.
[18] R. Motamedi Mehr, M. Taran, A. Baradaran Hashemi, M. R., Meybodi, "Determining Web Pages Similarity using Distributed Learning Automata and Graph Partitioning", Proceedings of the 16th Annual CSI Computer Conference of Iran, 2011.
[19] S. Saati, M.R. Meybodi, "Document Ranking using Distributed Learning Automata", Proceedings of the 11th Annual CSI Computer Conference of Iran, Fundamental Science Research Center(IPM), 2006, pp.467- 473.
[20] S. Saati, M.R. Meybodi, "A Self-organizing Model for Document Structure using Distributed Learning Automata", Proceedings of the Second International IEEE/WIC Conference on Information and knowege Technology (IKT2005), 2005.
[21] Z. Jianhan, Mining web site link structures for adaptive web site navigation and search, Ph.D. Thesis, University of Ulster at Jordanstown, 2003.
[22] J. Zhu, "Using Markov Chains for Structural Link Prediction in Adaptive Web Sites", Proceedings of the User Modeling, 8th International Conference, Lecture Notes in Computer Science, 2001, pp.298-300.
[23] R.R Sarukkai, "Link Prediction and path analysis using Markov chains", Computer Network 33, Elsevier Science, 2000 pp.377-386.
[24] J. Borges, M. Levene, "Data Mining of User Navigation Patterns", Proceedings of the Web Usage Analysis and User Profiling, Vol. 1, 1999, pp. 31-36.
[25] J. Borges, M. Levene, "Mining Association Rules in Hypertext Databases", Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining, 1998, pp. 149-153.
[26] K. Narendra, M. A. L. Thathachar, Learning automata, An Introduction, Englewood Cliffs, NJ: Prentice Hall, 1989.
[27] H. Beigy, M. R. Meybodi, "A New Distributed Learning Automata Based Algorithm for Solving Stochastic Shortest Path Problem", Proceedings of the 6th International Joint Conference on Information Science, 2002, pp.339343.
[28] M. R. Meybodi, H. Beigy, "Solving Stochastic Path Problem Using Distributed Learning Automata", Proceedings of the 6th Annual International CSI Computer Conference, CSICC2001, 2001, pp. 70-86.

[29] M. R. Meybodi, H. Beigy, "Solving Stochastic Shortest Path Problem Using Monte Carlo Sampling Method: A Distributed learning Automata Approach", In Lecture notes in advances in soft computing, Springer-Verlag, 2003, pp. 626-632.
[30] M. Levene, G. Loizou, "A Probabilistic Approach to Navigation in Hypertext", Information Sciences, Vol. 114, 1999, pp.165-186.
[31] C.S. Wetherell, "Probabilistic Languages: A Review and Some Open Questions", Computing Surveys, Vol. 12, No. 4, 1980, pp.361-379.
[32]http://facweb.cs.depaul.edu/mobasher/classes/ect584/resource.html.

Babak Anari received the B.Sc. degrees in computer engineering from Azad University of Shabestar, Iran in 2002, and M.Sc. degrees from Azad University of Arak, Iran in 2007, respectively. He has been a Ph.D. candidate in computer science from Islamic Azad University, Science and Research branch in Tehran, Iran from 2012. His research areas include learning automata, web mining and soft computing.

Zohreh Anari received the B.Sc. and M.Sc. degrees in computer engineering from Azad University of Shabestar, Iran in 2003 and 2009, respectively. She has some research papers in fuzzy web mining field. Her current research interests include learning automata, web mining and soft computing.