The Surfer Model with a Hybrid Approach to Ranking the Web Pages

Javad Paksima
Department of Engineering, Payame Noor Yazd University, Yazd, Iran
Paksima@pnu.ac.ir
Homa Khajeh*
Department of Engineering, Science and Art University, Yazd, Iran
khajeh121@yahoo.com

Received: 31/Oct/2015 Revised: 19/Jun/2016 Accepted: 26/Jul/2016

Abstract

Users who seek results pertaining to their queries are at the first place. To meet users' needs, thousands of webpages must be ranked. This requires an efficient algorithm to place the relevant webpages at first ranks. Regarding information retrieval, it is highly important to design a ranking algorithm to provide the results pertaining to user's query due to the great deal of information on the World Wide Web. In this paper, a ranking method is proposed with a hybrid approach, which considers the content and connections of pages. The proposed model is a smart surfer that passes or hops from the current page to one of the externally linked pages with respect to their content. A probability, which is obtained using the learning automata along with content and links to pages, is used to select a webpage to hop. For a transition to another page, the content of pages linked to it are used. As the surfer moves about the pages, the PageRank score of a page is recursively calculated. Two standard datasets named TD2003 and TD2004 were used to evaluate and investigate the proposed method. They are the subsets of dataset LETOR3. The results indicated the superior performance of the proposed approach over other methods introduced in this area.

Keywords: Ranking; Web Pages; Surfer Model; Learning Automata; Information Retrieval.

1. Introduction

The content of the World Wide Web is increasingly growing. According to the studies reported in [1],[2], there exist more than 1 billion websites on the Web. In this regard, search engines are considered efficient tools used for recovering and extracting important information from this large set of data. Mostly, about 91% of users use search engines to find their desired information [3]; moreover, users stated that 73% of the information provided by search engines was valid [3]. Without appropriate ranking, search engines are not able to meet users' needs. Users' tendency to find the result of query at the high ranks of the ranking list indicates the importance of ranking algorithms efficiency. Ranking means placing relevant pages at the first rank so that users can find the answers to their queries in the shortest possible time.

Ranking methods fall into two general classes, namely, content-based and connection-based. Content-based methods use the content of webpages. Instances of such methods include models which are Boolean, probability (like BM25 method [4]) and vector (like TF-IDF method[1] [5]). These methods suffer from rank spamming [6]. Rank spamming means that the owners of some webpages usually add extra and irrelevant words, which are mostly invisible and blended in the background color, to their pages to be more selected by search engines. In connection-based methods, webpages are evaluated using other pages. Links indicate the quality of destination page from the perspective of source pages. Instances of connection-based methods are PageRank [7], and HostRank [8]. The main problem of such methods is called Rich-Get-Richer [9]. This problem is caused when search engines always place popular pages at the top of the list resulting from users' queries, and users usually visit the first ten results. Therefore, the popular pages become more popular, and the new relevant pages are less likely to be visited. Hybrid approaches have been proposed to solve this problem. They are based on connection and content. For instance, HITS [10] and TSPR [11] use content to improve ranking. The proposed method is also a hybrid approach.

Connection-based algorithms are divided into two main categories including query-independent and query-dependent methods. In query-independent methods such as PageRank and HostRank, ranking is done using the entire web graph offline. However, in query-based methods such as HITS, ranking is done only in a part of web graph which includes the query-related pages. In [7], out-link uniformly is chosen at random to determine the page to visit at the next time step on the graph, but in this paper, chosen page with respect to non-uniform distribution The proposed method is query-dependent, too. Thus, ranking is done among the query-related pages.

[1] TF-IDF as the vector-space model is utilized in page ranking and this ranking method is named TF-IDF.

* Corresponding Author

The intelligent surfer, proposed in this paper, would not select pages with respect to uniform distribution. However, it selects them with respect to their contents and connections. For this issue, the learning automata would be used to calculate the probability of selecting pages. A page was selected to hop by the learning automata along with the contents and connections of pages in each phase. Moreover, the surfer could also select a page for transition from the current page with respect to the content of the connected pages. While the surfer is moving about webpages, their ranking is selected recursively.

The rest of this paper is organized as follows: Part 2 reviews the research literature, which focuses on the studies which deal with only the connection and content of webpages. A definition of the learning automata, used in the proposed method, is presented and the proposed method is introduced in Part 3. Then the evaluation criteria are discussed in Part 4, and the results are investigated. Finally, the conclusion and suggestions for future studies are presented.

2. Literature Review

A review of the research works related to ranking webpages is presented here. Generally, ranking algorithms are trying to study the following problems: What criterion can be used to indicate whether the webpage is related to users' query or not? How are the results ranked so that they respond to user's query at any time? What algorithm is able to place more relevant results at first ranks? Ranking algorithms are classified with respect to the use of content and connection. Vector space and probabilistic models are among the most important content-based methods.

Salton introduced a method for ranking the documents corresponding queries. In this model, document and query are considered vectors; their length is equal to the number of words existing in the term. Cosine of the angle between the two vectors is considered the degree of their similarity [12]. Salton et al. [5] proposed a model named TF-IDF in 1988. This method used the frequency of document and query words to calculate the weight. The idea used in this method is that the most frequently repeated word describes the document better, and the words appearing in fewer documents have more information. The advantage of this vector model is its simplicity and flexibility; however, there was no official framework to find and display the degree of proposed relationship.

The probabilistic model is one of the content-based models whose objective is to find the probability of dependency of each document on each query. Unlike the vector model, it cannot find the similarity degree definitely. Robertson [4] proposed BM25, which is one of the best probabilistic methods. In BM25, weighting is considered to be based on an okapi. This highly accurate probability formula indicates the similarity between queries [4]. This method suffers from rank spamming. PageRank algorithm is a query-independent method, in which, ranking value of each page is equal to the weighted summation of its input pages' ranks. In other words, a page has a high ranking if many pages refer to it, or the referring pages have high rankings [7]. TSPR method is the developed version of PageRank, which considers N headlines in the entire Web. Using PageRank, the pages are ranked with each headline. Content methods retrieve the pages containing query words, and PageRank score is calculated on different topics [11].

Ghodsnia and Yazdani [13] proposed a penalty and reward-ranking algorithm named BPRR, which added a new dimension to PageRank model through the direct feedback from user. The visiting priority of search results was considered a vote for this document, and the score values were attributed to documents [13]. This method reduces the impact of Rich-Get-Richer problem. In PPR (Personalized PageRank), data-mining technology is used to extract user's automatic interests [14]. The proposed algorithm moves gradually towards user's interests to personalize ranking. The common filter is used when a new query is made in order to improve ranking accuracy and validity.

WordRank method [15] is similar to PageRank approach. Their difference is that the user waltzes through pages in the same way as the random surfer does in PageRank [15]. However, the user does not select the external links to a page with equal probabilities in this method; rather, he operates as a directed surfer. The user selects a page similar to the current page.

The WeightedPageRank method [16] is similar to PageRank. Their difference is that both the internal and external links are considered [16]. TrustRank [17] is a method to cope with rank spamming. It is a semi-automatic method to distinguish between good and bad pages. Good pages are the trusted ones (in which rank spamming did not occur). The idea of TrustRank algorithm is that when a page with trust degree of one is distributed to other pages, the impact of trust degree is reduced as the distance is increased [17].

Pandey [18] proposed a method by which a balance would be established between new quality pages and the available ones [18]. Using this method, new pages have the chance to be placed at the top of the ranking list. In HostRank method, the pages are first placed in a hierarchical structure of host directory, named the superior group, to obtain the connections on the graph. Then, the degree of each node is distributed among the pages containing the node by the hierarchical structure [8]. This method solves the problems of excessive distribution of webpages and Rich-Get-Richer.

In [19],[20], a ranking method named RL_Rank was proposed with a query-independent approach. In this algorithm, the user is a random surfer who moves between webpages. Moving between webpages is done by clicking on one of the external links of the current page. A reward is considered for the selected page, and the value function of each page indicates its rank. The results showed the superiority of the proposed method over PageRank.

Zareh et al. [21] introduced an algorithm based on reinforcement learning named DistanceRank. In this

method, the logarithm distance between pages was used as the received reprimand, and the objective is to minimize the total received reprimand. This algorithm is less sensitive to Rich-Get-Richer phenomenon [21]. In query-independent algorithms, all pages are competing, and irrelevant pages sometimes rank higher due to their popularity. HITS algorithm [10] is executed on a sub graph named root. It is then expanded to a bigger graph named the base graph. For each vector such as v, two variables of a(v) and h(v) named authority and hubness are defined. It means that a page of high authority is referred to by a number of pages with high hubness, and a page of high hubness refers to a number of high authorities [10]. SALSA [22] is a connection-based method. The idea of this method is to combine PageRank with HITS. It creates a repetitive graph between hubs and authority. At first, some nodes are haphazardly selected out of authority nodes. Then the values of variables are adjusted with repetition between previous and next steps [22].

Due to the problems existing in the two types of algorithms (connection-based and content-based), hybrid approaches were proposed to increase the accuracy of such algorithms. Shakeri and Zhai [23] introduced a hybrid ranking method, in which, a score named hyper-relevance was defined for each page. The rank of each page is calculated through the linear combination of three parameters such as the similarity between page and query, weighted functions for internal and external links [23]. The most important disadvantage of this model is that it should be online, a problem which reduces the response time. Zareh et al. [24] used the click data to combine some content and connection features of webpages. The simulated click data is used to allocate the weight to each feature; therefore, the best combination would be found [24].

In [25], a method from the combination of two methods; Enhanced-RatioRank and Page level keyword, is proposed. This method uses the concept of structure and keyword search at a page level. The problem which can be seen in method [25] is that each page must be clicked by the user once. This is possible for personal dictionaries or websites, but it will not be possible for web pages in a search engine. Method [26] is a developed model of PageRank, which forms some tables from the keywords of pages, and by considering the similarity between the titles and the user's query words, accords greater importance to a particular subject. Their aim is to personalize the ranking for the user, and they use the browser history data. Whereas, the method proposed in this paper can be used for the query of any user, and there is no need to receive information from the client side except the query phrase to rank pages by a search engines.

PageRank uses a model based on the random surfer agent to rank webpages, and selects one of the pages with an external link for transition at a uniform probability. Otherwise, it jumps to another page in a uniform way. An intelligent surfer is proposed in this paper. The selection of pages for hopping or transition is not based on the uniform distribution. The pages are selected with respect to their contents and connections. The learning automata were used to calculate the probability of selecting pages so that it could hop to other webpages. The contents features of referring pages were used for transition to one of the pages of the external link. The proposed method is query-dependent.

3. The Proposed Method

The idea of the proposed method is to create an intelligent surfer that moves between the retrieved webpages for query. This method considers the relationships between pages and their contents. The surfer has two approaches to transition from the current page to other pages. First, it goes to one of the linked pages. Then, it hops on one of the retrieved webpages. Given this reasoning, if the linked page is linked to another page, it can be stated that the page topic was appealing to the creator of the first page. Therefore, the link indicates the interest of another page in this page. The surfer hops on one of the pages to which it is linked. This page is selected with respect to the contents of the referring pages.

BM25 was particularly used as the best content feature in this paper. The reason the contents of referring pages were used is to consider the concept that relevant pages point to each other, and a relevant page can be relevant in terms of content. In the second case, the surfer considers the page rank, which is updated during movement in the graph with a source of content feature. The connection feature of webpage calculates the probability of selecting for hopping by using the learning automata. Exploring the webpages, the surfer calculates their ranks recursively. Put it another way, ranking webpages is converged to constant values. In the following statement, ranking score of web pages is calculated as section 3-1. Moreover, a definition of the learning automata, used in the proposed method, is presented in section 3-2. After that a page probability is calculated by Learning Automata and MB25 for proposed method as section 3-3. At last, section 3-4 presents proof of proposed method convergence.

3.1 Calculation Ranking of Web Page

In the proposed method, if the intelligent surfer is on page i, it hops on one of the pages related to external links or it hops on other pages with respect to their probabilities. The likelihood of selecting external links will not be the same. The surfer selects a link, which is more relevant in terms of content. The probability of a transition from the referring page to the referred page is equal to BM25 of the referred page divided by the summation of BM25's of all of the external links (Unlike reality, since it is possible that all of the external links are irrelevant in terms of the content feature of BM25, a very slight amount of ε was added to all BM25's to prevent the denominators from being zero.) Therefore, the score which page j receives due to a transition from page i to page j (with respect to

its external link) is equal to $\frac{R_{iq}(k)\times(BM25_{jq}+\varepsilon)}{\sum_{z\in O(i)}(BM25_{zq}+\varepsilon)}$, which is calculated recursively.

If the external links are not selected, the surfer hops on another page that will be selected with respect to probability. The page rank is calculated through the following equation:

$$R_{jq}(k+1) = (1-d)\times P_{jq}(k+1) + d \times \sum_{i\in B(j)} \frac{R_{iq}(k)\times(BM25_{jq}+\varepsilon)}{\sum_{z\in O(i)}(BM25_{zq}+\varepsilon)} \quad (1)$$

In which $R_{jq}(k+1)$ and $P_{jq}(k+1)$ indicate the rank and probability of page j for the query q in the $(k+1)^{th}$ step, respectively. $O(i)$ is equal to a set of pages of external links pertaining to page i. $B(j)$ indicates a set of pages referring to page j. d represents the damping factor. Parameter d is used to guarantee the convergence of the proposed method and to delete the impact of sink pages (the ones with no external links). $BM25_{zq}$ indicate content feature of BM25 of page z for query q. ε is the constant value of 0.05. All of the pages compete with each other with the same topic, and increase accuracy. This method would decrease the Rich-Get-Richer problem. The learning automata would be used to calculate the page probability. Description of the learning automata is used as follows:

3.2 Background of the Learning Automata (LA)

The idea of the learning automata was stated by Testlin in the early 1960. A learning automaton [27],[28] operates in a random environment. It is a comparative decision-making unit, which selects the optimal action out of a set of finitely authorized operations through repeated interactions with learning. It improves the efficiency, and the action is selected haphazardly. The selected action is the input of the random environment at each moment. The environment responds to the action with a reinforcement signal. The probability vector is updated through the reinforcement feedback from the environment. The aim of learning automata is to minimize the average penalty received from the environment. A learning automaton is useful for an environment with insufficient information [29]. A learning automaton is also well appropriate for an environment, which is complicated, dynamic and random with uncertainty. The reason for that is the use of learning automata in a wide range of issues such as optimization problems [30], computer network [31], grid computing [32], signal processing [33], information retrieval [34], and Web engineering [35].

The environment can be defined with {α,β,c} in which α≡{α1,α2,..,αr} indicates a finite set of inputs, while β≡{β1, β2,.., βm} refers to a set of variables which can be selected, and c≡{c1,c2,…, cr} represents a set of penalty probabilities in which ci depends on the given action αi. If the penalty probability is constant, the previously mentioned random environment turns to a constant random environment, and if it changes with time, the environment is named an inconstant one. Depending on the nature of reinforcing signal β, the environments can be categorized as p-model, S-model and Q-model. In P-model environments, the reinforcing signal can only be two binary values (0 or 1). In Q-model, a finite number of values ranging in [0, 1] can be selected for the reinforcing signal. In s-model, the reinforcing signal is in [a, b].

The learning automata can be divided into two main classes [27]: the learning automata with a constant structure and that with a changing structure. The first one is indicated with <β, α, L> in which β is the set of inputs while α is the set of actions, and L is the learning automata. The learning algorithm is used to change the probability vector. It allows αi(k)∈α and p(k) actions to be selected by the learning automata. The probability vector is defined for this set of actions at the moment of k. The parameters of reward and penalty are indicated with a and b. The number of actions that the learning automata can select is indicated with r. At each constant of k, the probability vector of p(k) is updated through the linear algorithm resulting from Eq. (2). If the selected activity of αi(k) is the reward given by the random environment, the updated penalty results from Eq. (3):

$$p_j(k+1) = \begin{cases} p_j(k) + a[1-p_j(k)] & j = i \\ (1-a)p_j(k) & \forall j \neq i \end{cases} \quad (2)$$

$$p_j(k+1) = \begin{cases} (1-b)p_j(k) & j = i \\ \left(\frac{b}{r-1}\right) + (1-b)p_j(k) & \forall j \neq i \end{cases} \quad (3)$$

If a=b, Equations (2) and (3) are named the reward of linear penalty (L_{R-P}). If a>>b, the above equations are named the reward of linear penalty ($L_{R-\epsilon P}$). However, if b=0, they are named the reward of linear inactivity (L_{R-I}). In the last case, if the action is fined by the environment, the probability vectors of the remaining action do not change [36].

The World Wide Web includes thousands of web pages, there is not sufficient information about which pages are relevant to the user query, and such information can only be obtained by surveying the environment. In this paper, learning automata is used as a tool that easily explores an environment about which there is no knowledge, and that acquires the required knowledge by interacting with the environment. This interaction is done by surveying the web graph, and the knowledge is obtained by updating the probability vector.

At each step, it selects one of the actions based on the action probability vector. This way of selection is based on the knowledge obtained from the environment, and it is better than selecting one of them randomly. The required knowledge is obtained from the web graph (environment).

3.3 Calculation Page Probability using LA

While calculating the page probability, it was assumed that the retrieved pages formed the state space of the learning automata, and the number of learning automata actions was equal to the number of the retrieved pages,

which are more than the threshold, except for the current page. Threshold is equal average double of HostRank and average double of pages rank in pervious step. In each cycle, only the pages with higher ranks from the previous step and with respect to HostRank are selected. The condition which should be met to select the page is as follows:

$$\mathrm{HostRank_i} > \mathrm{HostRank_{avg}} * \rho \text{ and } \mathrm{R_i} > \mathrm{R_{avg}} * \rho \quad (4)$$

In which R_i and $HostRank_i$ indicate the rank of page i which was calculated with the proposed method and HostRank [8], respectively. $\mathrm{HostRank_{avg}}$ and $\mathrm{R_{avg}}$ indicate the average values of HostRank for the retrieved pages for user's query and the average rank of the retrieved pages for user's query in the previous step, respectively. ρ is the constant value of 2. With coefficient 2, a top quarter of the ranked list is considered to be relevant. In this state, better results have been provided.

The probability is calculated according to Eq. (3). The probability of the selected page increases (it is rewarded), and those of other pages decrease. In Eq. (1), $P_j(k+1)$ represents the probability of page j in cycle $k+1$ while a indicates the reward according to the following relation.

$$a = e^{-\frac{\beta(T-t)}{t}} \quad (5)$$

In which, β is the constant value of 4.4 known as the step size. T indicates the entire number of execution cycles for convergence. It is assumed that the set of possible actions in each step is equal to the number of retrieved pages pertaining to user's query. In each step, if the page rank in the previous step is greater than twice of the average page ranks pertaining to the query, and the HostRank of the page is greater than twice of average HostRank of pages pertaining to the query, a reward will be received. The page ranks are calculated recursively. Finally, the pages are arranged in a descending order with respect to their ranks. The pseudo code and Module pertaining to the proposed algorithm are indicated in Figs. 1 and 2, respectively. Table 1 shows the parameters used in Fig. 1. Moreover, the convergence of the proposed method is empirically proved in the next part.

Table 1. Parameters used in the pseudo-code of Proposed Method

Parameter name	Represent
N	The total number of pages retrieved for query
t	The number repeats or time
T	The total number of execution cycles for convergence.
Beta	It is the constant value of 4.4 known as the step size.
D	It is the damping factor.
R_{iq}	rank of page i for the query q
P_{iq}	probability of selected page i for the query q
avg_hostrank	The average values of HostRank for the retrieved pages for user's query.
avg_R	The average rank of the retrieved pages for user's query.
$ParentCount_i$	The number of members in the set of pages pointed to page i.
Sum_val_links	The sum of BM25 of out-links.
$LinkCount_p$	out-degree of the page p

Algorithm

Input
graph_link
n: number of pages retrieved for query q^{th}
list of document-query pair for query q^{th}
Output
R: Ranking list
Initialize
t=0, T=50, beta=4.4, d=0.15;
For i=0 to n
R_{iq}=1/n; //R used for Rank of pages
P_{iq}=1/n;// P used for probability of pages
end
avg_hostrank = $averge_{i\in\{0,..,n\}}\ HostRank_i$
Begin
For t=0 to T
Phase1:
//Calculate probability of pages
a=exp(-beta*(T-t)/T);
avg_R= $averge_{i\in\{0,..,n\}}\ R_{iq}$
For i=0 to n
if($R_{iq} > 2 * avg\_R$ and $HostRank_i > 2 * avg\_hostrank$) then
$P_{iq}=P_{iq} + a*(1-P_{iq})$
else
$P_{iq}=(1-a)*P_{iq}$
end
end
Phase2:
//Calculate Ranking of pages
For i=0 to n
Value=0
For p=0 to $parentCount_i$
Sum_val_links=0
For j=0 to $linkCount_p$
Sum_val_links = Sum_val_links+$BM25_{jq}+\varepsilon$
end
Value=Value+ R_{pq}/ Sum_val_links
end
R_{iq}= (1-d)*P_{iq} + d*($BM25_{iq}+\varepsilon$)*Value
end
end

Fig. 1. The Proposed Method Pseudo Code

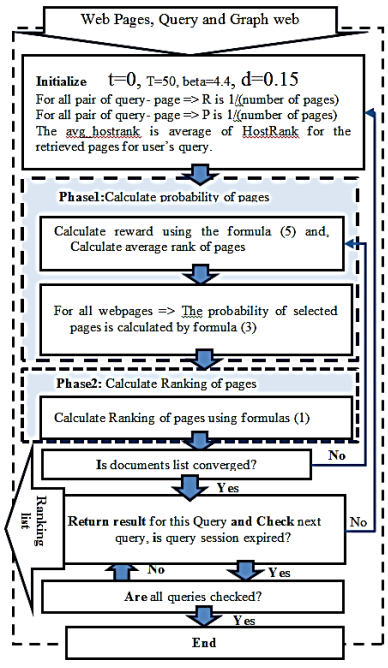

Fig. 2. Module of the proposed algorithm

3.4 Proof of Convergence

The convergence of the proposed method is empirically proven in this part.

A similarity test was conducted to prove the convergence of the proposed method empirically. The aim of ranking was to sort the pages out with respect to their scores [37]. Therefore, the results of different iterations are compared with each other to display the convergence. For this purpose, the similarity resulting from the 20th repetition was compared with a sorted list including 5th, 8th, 10th, 11th, 15th, 16th, 17th, and 19th repetitions. The similarity of two lists is calculated according to the following equation:

$$\text{Similarity} = \frac{|A \cap B|}{|A \cup B|} \quad (6)$$

In which A and B contain the N first pages on the sorted list resulting from two different repetitions. $|A \cup B|$ indicates the total number of pages which appeared on the N first pages of the two lists (list union), and $|A \cap B|$ indicates the number of pages which appeared on the N first pages of both lists (list intersection). Fig. 3 shows the similarity among different repetitions in comparison to the 20th repetition for 2 to 47200 pages. This test was conducted on the dataset TD2003. If the similarity of two lists resulting from two repetitions gets close to 1, it means that the list of pages proposed in two repetitions was almost the same and the order of pages were constant after these repetitions. In other words, the ranking order has converged.

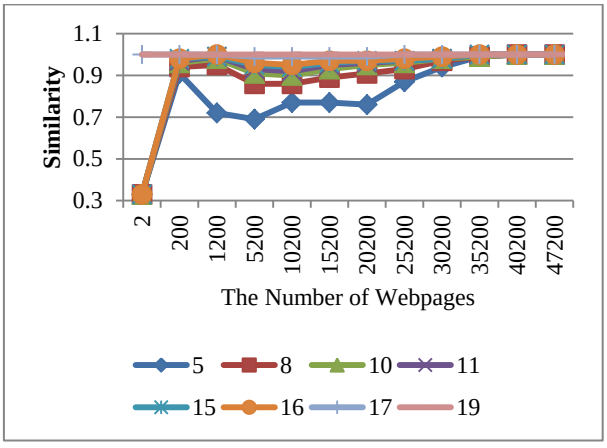

Fig. 3. The Convergence of the Proposed Method during the Successive Repetitions of Execution

4. Assessment of Empirical Results

In this part, the empirical results of evaluating the efficiency of the proposed algorithm on TD2003 and TD2004 datasets, taken from the standard set of ranking algorithms test LETOR [38], is presented.

Evaluation was made between the proposed method and the previous ones, using standard criteria such as MAP, P@n, and NDCG@n [39].

4.1 Assessment Criteria

In this evaluation, precision at the position of n (P@n), Mean Average Precision (MAP) and Normal Discount Cumulative Gain (NDCG) were used to evaluate the efficiency because they are comprehensively used in information recovery. They are defined as follows:

❖ **P@n**

Precision at n calculates the relevance of n webpages at the top of the list of ranking results with respect to a query.

For instance, if the first ten retrieved documents by the query are as {relevant, irrelevant, irrelevant, relevant, relevant, relevant, irrelevant, irrelevant, relevant, irrelevant}, P@1 to P@10 are as {1, 1.2, 1.3, 2.4, 3.5, 4.6, 4.7, 4.8, 5.9, 6.10}, respectively.

Precision at n and the relevance of n documents at the top of ranking list are calculated with respect to user's query [38].

$$P@n = \frac{\#\text{ of relevance docs in top n results}}{n} \quad (7)$$

❖ **MAP**

For an average query, precision is defined as the average P@n values for all relevant documents. The average precision (AP) [38] is calculated through the following equation. It is equal to the average value of P@n for all relevant documents.

$$AP = \frac{\sum_{i=1}^{N}(P@i * rel(i))}{\#\text{ total relevance docs for this query}} \quad (8)$$

In which N indicates the number of retrieved documents, and rel(n) shows the binary function. If the n^{th} document is relevant, it is 1; otherwise, it is zero. Finally, MAP is equal to the average precision (AP) for all queries [38].

❖ **NDCG@n**

It is the evaluation criteria of the cumulative gain which have been normalized. It represents the judgment on the multi-level relevance. The value of NDCG of ranking list at position n is calculated through the following relation for a query:

$$NDCG(n) = Z_n \sum_{j=1}^{n} \frac{2^{r_j}}{\log(j+1)} \quad (10)$$

In which r_j is the degree of relevance for document *j* on the ranking list. Z_n is the normalization constant, which is determined in a way that the highest value of NDCG would be one. For LETOR3, there are two degrees for relevance, {0 and 1}, from user's perspective. They indicate irrelevance and relevance, respectively [38].

❖ **NWN**

Another point states the most accuracy of the ranking methods on different datasets [38]. They propose a metric called Winning Number to evaluate the performance of ranking methods over the datasets included in the LETOR 3.0 collection. Winning Number is defined as the number of other algorithms which is better than they are. The Winning Number [40] is calculated according to Eq. (11).

$$WN_i(M) = \sum_{j=1}^{n} \sum_{k=1}^{m} I_{\{M_i(j) > M_k(j)\}} \quad (11)$$

In which, n and m are the number of datasets and algorithms in the comparison, respectively. *j* indicates the index of a dataset, *i* and *k* are indices of an algorithm, *M* is an assessment criterion (such as MAP or NDCG), $M_i(j)$ represents the performance of the i^{th} algorithm on the j^{th} dataset, and $I_{\{M_i(j) > M_k(j)\}}$ indicates an indicator function such that

$$I_{\{M_i(j) > M_k(j)\}} = \begin{cases} 1 & \text{if } M_i(j) > M_k(j) \\ 0 & \text{otherwise} \end{cases} \quad (12)$$

The Winning Number evaluation metric depends on the denseness of the evaluation results. This means that there were evaluation results for all rank algorithms on all datasets under comparison [41]. Therefore, the normalized Winning Number metric is proposed to enable comparison of a sparse set of evaluation results. This Normalized Winning Number takes each dataset into account, and an algorithm is evaluated on and divides this by the ideal Winning Number [41]. The indicator function I is defined in order to only take into account datasets on which it has been evaluated. If $M_i(j)$ and $M_k(j)$ are both defined and $M_i(j) > M_k(j)$ is true, it is 1; otherwise, it is zero.

The Normalized Winning Number is calculated according to the following equation:

$$NWN_i(M) = \frac{WN_i(M)}{IWN_i(M)} \quad (13)$$

In which, IWN_i is the Ideal Winning Number and, i is index of i^{th} algorithm. *IWN* theoretically is equal to the highest Winning Number.

4.2 Benchmark Datasets

Similar settings and conditions were required to evaluate the efficiency of the proposed method and to compare it with other approaches. Standard benchmark datasets of TD2003 and TD2004 were used in this paper. They were published at LETOR website [39] for the same purpose.

In addition, tests were carried out on a computer with an Intel i7 core 2.10 GHz CPU and 6 GB memory.

4.3 Empirical Results

Benchmark datasets of TD2003 and TD2004 were used to evaluate the proposed method. The results were stated with respect to the evaluation criteria of P@n, MAP, and NDCG@n. The proposed algorithm was compared with algorithms such as BM25, HostRank, PageRank, and HITS. In the figures, HITS_a and HITS_h meant HITS based on authority and hub, respectively. The proposed method has been called alg_automata in the figures. According to Figs. 4 to 5, the proposed algorithm showed a better performance on two benchmark datasets of TD2003 and TD2004 with respect to the evaluation criteria of P@n, MAP, and NDCG@n. It is noteworthy that Table (1) indicates the improvement percentage of the proposed algorithm in comparison to HostRank, BM25, HITS_h, HITS_a, and PageRank algorithms with respect to P@n, MAP, and NDCG@n. The highest improvement percentage was observed in three criteria on TD2003 compared to PageRank and on TD2004 compared to HITS_h.

The evaluation of empirical results of the proposed method can be seen in Figs. 4 to 5. They indicate the efficiency of the proposed method on two datasets of TD2004 and TD2003. They also showed the superiority of the proposed method over other algorithms. The proposed method worked better than PageRank because PageRank is in favor of the old pages, and new pages do not have many links, even if they are really good.

Table 2. The Improvement Percentage of the Proposed Method Compared to Other Algorithms

	Dataset	TD2003			TD2004		
	Evaluation Criteria	P@n	MAP	NDCG@n	P@n	MAP	NDCG@n
Algorithm	HITS_h	%124.82	%124.18	%165.84	%267.63	%161.71	%267.81
	HITS_a	%69.07	%55.13	%69.41	%47.21	%42.96	43.53%
	HostRank	%48.71	%61.52	%56.89	%14.30	%18.92	%17.42
	BM25_title	%54.49	%31.48	%51.	%66.46	%34.50	%60.48
	PageRank	%163.82	%148.38	172.37%	%80.57	62.33%	%78.57

Considering the contents of pages, the proposed method does not have this problem. It reduced Rich-Get-Richer problem. In PageRank, popular pages tend towards general popularity; however, the popularity of website is not guaranteed by taking enough information. Considering the content in the proposed method, this problem is solved; therefore, a better ranking is provided. The proposed method was better than HITS because of the following two reasons. First, HITS suffers from topic drift. It means that if irrelevant pages exist in the root set with strong connections, these irrelevant pages are reflected on the pages in the basic set. Moreover, the web graph is made up of the webpages of the basic set which will not have more relevant nodes, and the results of algorithm will not be able to find pages with high hubs and authorities for the query. The second reason is that HITS considers the same value for links although it may not provide user's query with the relevant topic. The proposed method considers a value for each link with respect to the content of pages, and this will result in its superiority. Compared to BM25, the proposed method also pays attention to the link between pages. This would also result in its superiority. It has been considerably improved over HostRank because of paying attention to the contents of pages.

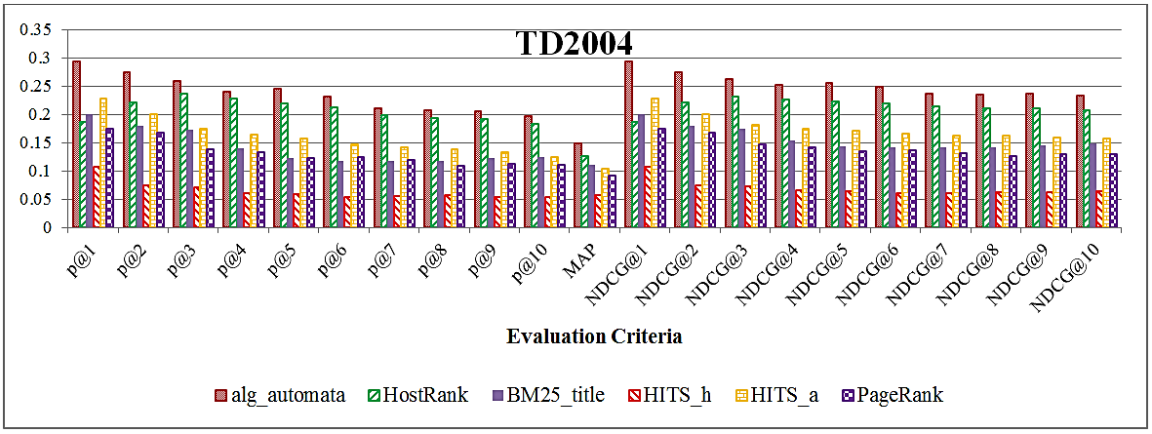

Fig. 4. Comparing the Proposed Method with Other Algorithms on TD2004

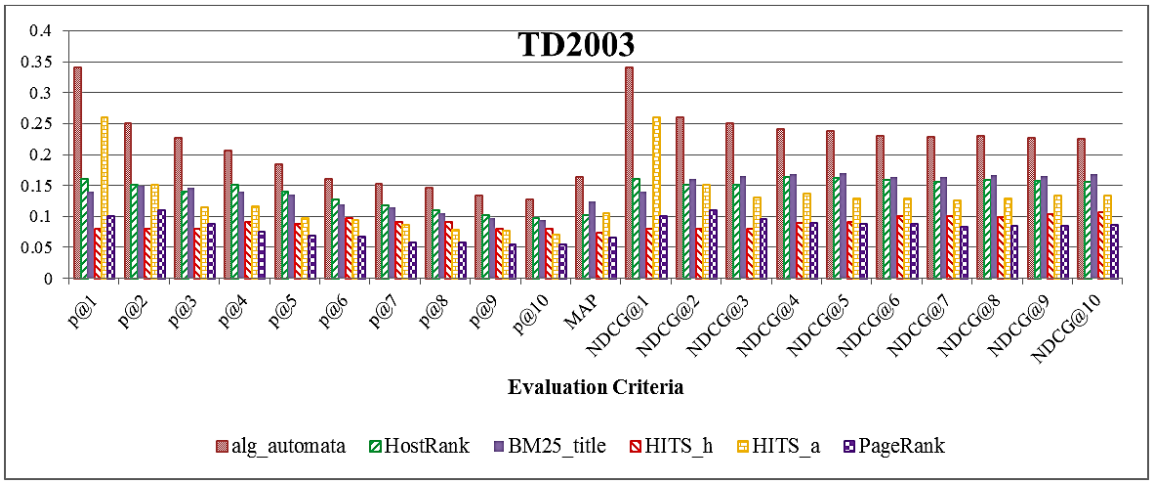

Fig. 5. Comparing the Proposed Method with Other Algorithms on TD2003

P@n, NDCG@n and MAP are the used evaluation metrics in the used datasets combined (for $n \in \{1, \ldots, 5\}$). Fig. 6 shows NWN as function of IWN for the considered methods in this paper. The proposed method scores very high NWN scores on two datasets on MAP, P@n and NDCG@n (for $n \in \{1, \ldots, 5\}$). HITS_a performed the NWN, around 0.8, on two datasets and also, performed well in both certainty and accuracy. BM25 is one of the best performers in the MAP comparison with a reasonable number of benchmark evaluations. There is a slight certainty on the accuracy of HITS_h and PageRank as both methods are evaluated on the two datasets included in the comparison for the NWN metric.

Fig. 7 shows the WN of methods based on results of MAP, P@n, and NDCG. The proposed method scores an IWN of 10 on two datasets, which is achieved by obtaining the highest score on the LETOR 3.0 TD2003 and TD2004 in this paper. HITS_a has a low WN value. The WN score of HITS_h is about zero. The WN and NWN scores for HITS_h and PageRank are lower than those for the other ranking methods, the certainty of their ranking performance is considered to be lower.

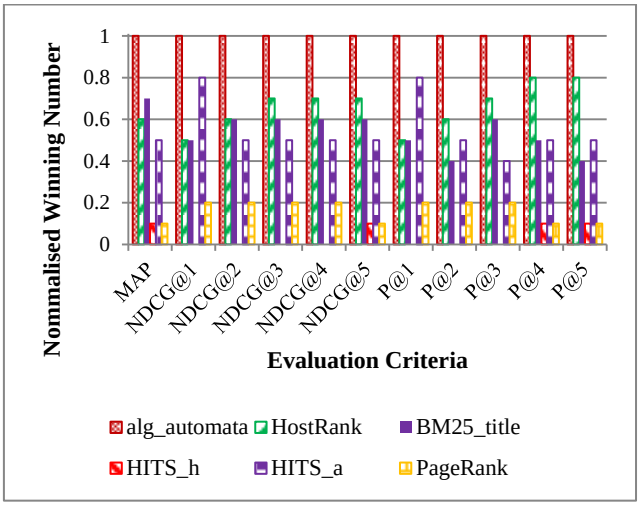

Fig. 6. Comparing the Proposed Method with Other Algorithms by Respect to Evaluation Criteria of NWN on TD2003 and TD2004

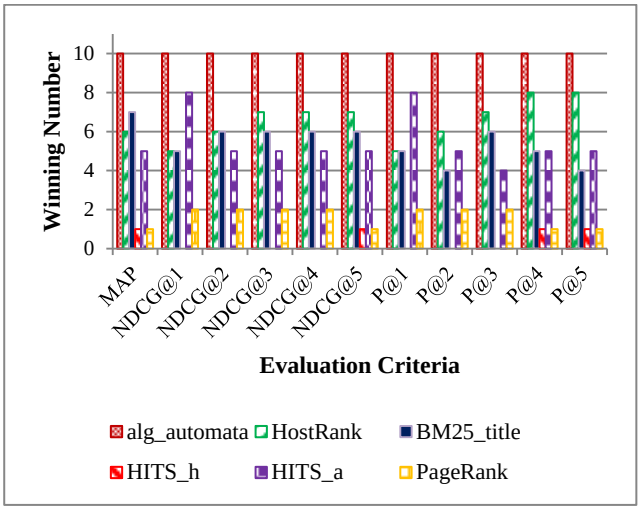

Fig 7. Comparing the Proposed Method with Other Algorithms by Respect to Evaluation Criteria of WN on TD2003 and TD2004

5. Conclusion and Future Suggestions

The proposed ranking method introduced an intelligent surfer that selects the pages with respect to their probability values. To calculate the probabilities, it was assumed that the retrieved pages were in the form of learning automata, and each page indicated a status. The number of actions of each learning automata was equal to the number of pages retrieved, except for the current page. Therefore, each page will have the chance of selection. Put it another way, the random surfer can hop to each page. Pages were selected for transition with respect to their scores, which were. This score was allocated to the page based on its content and connect. LETOR3 benchmark datasets, two standard datasets of TD2003 and TD2004 in particular, were used for evaluation. The empirical results indicated that the proposed method had better efficiency in comparison to content-based, connection-based, and hybrid methods such as BM25, HITS, PageRank and HostRank on TD2003 and TD2004 with respect to the evaluation criteria of P@n, MAP, and NDCG. The proposed method provided the users with the results of their queries due to being query-dependent. This method is based on content and connection; therefore, the proposed method could decrease the impact of problems such as rank spamming and Rich-Get-Richer. The proposed method has no other method superior to them in both IWN and NWN.

Using the methods of calculating probability of uniform distribution as the probability of hopping to webpages was postponed to the future along with the use of reinforcement learning methods.

References

[1] R. Albert, H. Jeong, and A.-L. Barabási. “Internet: Diameter of the world-wide web,” Nature, vol. 401, no. 6749, pp. 130–131, Sep. 1999.

[2] S. Lawrence and C. L. Giles. “Accessibility of information on the web,” Nature, vol. 400, no. 6740, p. 107, Jul. 1999.

[3] K. Purcell, J. Brenner, and L. Rainie. (2012, Oct.). “Search Engine Use 2012,” PEW Research Center, [Online]. p. 42, 2012 Available: http://www.pewinternet.org/2012/03/09/search-engine-use-2012/. [Sep. 1,2014].

[4] S. E. Robertson, and S. Walker. "Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval," in Proc. 17th annual international ACM SIGIR conf. Research and development information retrieval, Springer-Verlag New York, Inc, 1994, pp. 232-241.

[5] G. Salton and C. Buckley. “Term-weighting approaches in automatic text retrieval.” Information processing and management, vol. 24, no. 5, pp. 513–523, Dec. 1988.

[6] M. R. Henzinger, R. Motwani, and C. Silverstein. “Challenges in web search engines.” In ACM SIGIR Forum, vol. 36, no. 2, pp. 11–22, Sep. 2002.

[7] L. Page, S. Brin, R. Motwani, and T. Winograd. “The PageRank Citation Ranking: Bringing Order to the Web.” World Wide Web Internet Web Information System, vol. 54, no. 1999–66, pp. 1–17, Jan. 1998.

[8] G.-R. Xue, Q. Yang, H.-J. Zeng, Y. Yu, and Z. Chen. “Exploiting the hierarchical structure for link analysis,” in Proc. 28th annual international ACM SIGIR Conf. Research and development in information retrieval, 2005, pp. 186–193.

[9] A.-L. Barabási and R. Albert, “Emergence of scaling in random networks.” Science, vol. 286, pp. 509–512, Oct. 1999.

[10] J. M. Kleinberg. “Authoritative Sources in a Hyperlinked Environment.” Journal of the ACM (JACM), vol. 46, pp. 668–677, Sep. 1999.

[11] F. Qiu and J. Cho. “Automatic identification of user interest for personalized search,” in Proc. 15th int. conf. World Wide Web, 2006, pp. 727–736.

[12] G. Salton. The SMART retrieval system—experiments in automatic document processing. Amsterdam: IOS Press, 1971.

[13] P. Ghodsnia, A. M. Z. Bidoki, and N. Yazdani. “A punishment/reward based approach to ranking,” in Proc. 2nd int. conf. Scalable information systems, 2007, p. 58.

[14] W.-C. Peng and Y.-C. Lin. “Ranking web search results from personalized perspective,” in E-Commerce Technology, 2006. 8th IEEE Int. Conf. Enterprise Computing, E-Commerce, and E-Services, 3rd IEEE Int. Conf., 2006, p. 12.

[15] A. Kritikopoulos, M. Sideri, and I. Varlamis, “WordRank: A Method for Ranking Web Pages Based on Content Similarity,” presented at Databases, 2007. BNCOD’07. 24th British Nat. Conf., 2007, pp. 92–100.

[16] W. Xing and A. Ghorbani, “Weighted pagerank algorithm,” Commun. Networks and Services Research, 2004. in Proc.. 2nd Annual Conf., 2004, pp. 305–314.

[17] Z. Gyöngyi, H. Garcia-Molina, and J. Pedersen. “Combating web spam with trustrank,” in Proc. 30th Int. Conf. Very large data bases-Vol. 30, 2004, pp. 576–587.

[18] S. Pandey, S. Roy, C. Olston, J. Cho, and S. Chakrabarti. "Shuffling a stacked deck: the case for partially randomized ranking of search engine results," in Proc. 31st Int. Conf. Very large data bases, 2005, pp. 781–792.
[19] E. Khodadadian, M. Ghasemzadeh, V. Derhami, and S. A. Mirsoleimani, "A novel ranking algorithm based on Reinforcement Learning." presented at 16th CSI Int. Symposium on Artificial Intelligence and Signal Processing (AISP 2012), 2012, pp. 546–551.
[20] V. Derhami, E. Khodadadian, M. Ghasemzadeh, and A. M. Zareh Bidoki. "Applying reinforcement learning for web pages ranking algorithms." Applied Soft Computing, vol. 13, no. 4, pp. 1686–1692, Apr. 2013.
[21] A. M. Zareh Bidoki and N. Yazdani. "DistanceRank: An intelligent ranking algorithm for web pages." Information Processing and Management, vol. 44, no. 2, pp. 877–892, Mar. 2008.
[22] R. Lempel, and S. Moran. "SALSA: the stochastic approach for link-structure analysis." ACM Transactions on Information Systems (TOIS), vol. 19, no. 2, pp. 131-160, Apr. 2001.
[23] A. Shakery and C. Zhai. "Relevance Propagation for Topic Distillation UIUC TREC 2003 Web Track Experiments." in TREC 2003, 2003, pp. 673–677.
[24] A. M. Z. Bidoki and J. A. Thom. "Combination of documents features based on simulated click-through data." in Advances in Information Retrieval, Springer Berlin Heidelberg, 2009, pp. 538–545.
[25] L. Rodrigues and S. Jaswal. "An Efficient Page Ranking Approach Based on Hybrid Model." presented at Adv. in Computing and Communation Engeering (ICACCE), 2015 2nd Int. Conf., 2015, pp. 693–696.
[26] T. H. Haveliwala. "Topic-Sensitive PageRank." in Proc. 11th Int. Conf. World Wide Web, Www2002.Org, 2002, pp. 517–526.
[27] K. S. Narendra and M. A. L. Thathachar. Learning automata: an introduction. Courier Corporation, 2012.
[28] S. Lakshmivarahan and M. A. L. Thathachar. "Bounds on the convergence probabilities of learning automata." IEEE Transactions on Systems, Man, and Cybernetics, A Systems Humans, vol. 6, no. 11, pp. 756–763, Nov. 1976.
[29] E. Billard and S. Lakshmivarahan. "Learning in multilevel games with incomplete information. I." Systems Man, and Cybernetics Part B Cybernetics IEEE Trans., vol. 29, no. 3, pp. 329–339, Jun. 1999.
[30] J. A. Torkestani and M. R. Meybodi. "A New Vertex Coloring Algorithm Based on Variable Aaction-Set Learning Automata." Computing and Informatics, vol. 29, no. 3, pp. 447–466, Jan. 2012.
[31] N. Kumar, N. Chilamkurti, and J. J. P. C. Rodrigues. "Learning automata-based opportunistic data aggregation and forwarding scheme for alert generation in vehicular ad hoc networks." Computer Communications, vol. 39, pp. 22–32, Feb. 2014.
[32] M. Hasanzadeh and M. R. Meybodi. "Grid resource discovery based on distributed learning automata." Computing, vol. 96, no. 9, pp. 909–922, Sep. 2014.
[33] S. Bhattacharyya, A. Sengupta, T. Chakraborti, A. Konar, and D. N. Tibarewala. "Automatic feature selection of motor imagery EEG signals using differential evolution and learning automata." Medical & biological engineering & computing, vol. 52, no. 2, pp. 131–139, Feb. 2014.
[34] J. Akbari Torkestani. "An adaptive learning to rank algorithm: Learning automata approach," Decision Support Systems, vol. 54, no. 1, pp. 574–583, Dec. 2012.
[35] J. A. Torkestani. "An adaptive focused web crawling algorithm based on learning automata." Applied Intelligence, vol. 37, no. 4, pp. 586–601, Dec. 2012.
[36] H. Beigy and M. R. Meybodi. "A mathematical framework for cellular learning automata." Advances in Complex Systems, vol. 7, no. 03n04, pp. 295–319, Sep. 2004.
[37] T. H. Haveliwala. (1999,Oct.). "Efficient computation of PageRank," Technical Report, Database Group, Computer Science Department, Stanford University, [On-line]. 1999. Available: http://ilpubs.stanford.edu:8090/386/ [Dec., 1, 2014].
[38] T. Qin, T. Y. Liu, J. Xu, and H. Li. "LETOR: A benchmark collection for research on learning to rank for information retrieval," Information Retrieval, vol. 13, no. 4, pp. 346–374, Jan. 2010.
[39] J. Xu, T.-Y. Liu and H. Li. "The Evaluation Tool in LETOR," Microsoft Research Asia [Online]. Available: http://research.microsoft.com/en-us/um/beijing/projects/letor/LETOR3.0/EvaluationTool.zip [Dec. 1,2014]
[40] T.-Y. Liu. Learning to Rank for Information retrieval, Springer Science & Business Media, 2011.
[41] D. H. T. N. Sander Bockting. "A cross-benchmark comparison of 87 learning to rank methods." Information processing & management, vol. 51, no. 6, pp. 757–772, Nov. 2015.

Javad Paksima received his B.Sc. and M.Sc. degree in computer engineering from Sharif University, Tehran, IRAN, in 1996 and 1998. He is currently an Instructor in the Department of Computer engineering, Payame Noor University, Tehran, IRAN. He is currently a student in computer engineering in Yazd University and also is a member of Parsijoo team (Persian Search Engine). His research interests are in the areas of Computer Networks, and Information Retrieval.

Homa khajeh received the B.Sc. degree in Software Engineering from Islamic Azad University, Najafabad Branch (IAUN), in Isfahan, Iran, in 2009 and her M.Sc. degree of Software Engineering from Science and Art University in Yazd, Iran, in 2014. Her research interests are mainly in the field of Information Retrieval, Search Engine, Machine Learning, and Big Data.