*Journal of **AI** and **D**ata **M**ining*
Vol 3, No 2, 2015, 157-168.
10.5829/idosi.JAIDM.2015.03.02.05

Web pages ranking algorithm based on reinforcement learning and user feedback

V. Derhami[1*], J. Paksima[2] and H. Khajeh[3]

1. School of Electrical and Computer Engineering, Yazd University, Yazd, Iran.
2. Department of Engineering, Payame Noor Yazd University, Yazd, Iran.
3. Department of Engineering, Science & Art University, Yazd, Iran.

Received 6 November 2014; Accepted 2 June 2015
**Corresponding author: vderhami@yazd.ac.ir (V. Derhami).*

Abstract
The main challenge of a search engine is ranking web documents to provide the best response to a user`s query. Despite the huge number of the extracted results for user`s query, only a small number of the first results are examined by users; therefore, the insertion of the related results in the first ranks is of great importance. In this paper, a ranking algorithm based on the reinforcement learning and user`s feedback called RL3F are considered. In the proposed algorithm, the ranking system has been considered to be the agent of learning system and selecting documents to display to the user is as the agents' action. The reinforcement signal in the system is calculated according to a user`s clicks on documents. Action-value values of the proposed algorithm are computed for each feature. In each learning cycle, the documents are sorted out for the next query, and according to the document in the ranked list, documents are selected at random to show the user. Learning process continues until the training is completed. LETOR3 benchmark is used to evaluate the proposed method. Evaluation results indicated that the proposed method is more effective than other methods mentioned for comparison in this paper. The superiority of the proposed algorithm is using several features of document and user`s feedback simultaneously.

Keywords: *Search Engine, Ranking, Reinforcement Learning, User Feedback, Web Documents.*

1. Introduction

Information search without tools for information retrieval is currently very difficult. Search engines as information retrieval tools are used to find information. A search procedure begins with a user-provided query. Then, the user receives a list of results and looks for more promising results. Search engines aim to create the best possible outcomes for all users at any time. Different people with different goals and expectations have created different search behaviors. One of the major challenges for the search engines is proper ranking of web documents. Therefore, the ranking algorithms are of particular interest. The method is efficient for ranking extracts of user's query results with the results obtained and sorting them in descending order according to their relevance.
Several methods have been proposed in the context of ranking web pages. Ranking methods are divided into two main categories based on content and connectivity. Content-based methods have a spamming problem [1], and the web page is dependent on its creator. Connectivity-based methods are divided into methods, independent on query and dependent on query. These methods suffer from the problem of the rich-get-richer [2]. This means that the popular pages become more popular and new relevant pages are not seen by the user or take a long time to be at the user's perspective. Combination methods can reduce these problems because of the simultaneous attention to content and connectivity.
It is clear that the intention of the users in expressing a similar query can be different. To address the ambiguity of queries, we can use the user's interests. The algorithms based on user feedback can provide good results.
Recent studies show that users tend to click on the documents in the first ranks of the ranking list. The study results of Agichtein et al. on frequency distribution of relevance of users' clicks on a web

search results have shown that the relative number of clicks on documents decreases with lower rank [3], and users click on documents in the second, third and fourth rank, respectively, with probability about 60%, 50% and 30% [4]. This shows that users click on high rank documents in the results, even if the documents are irrelevant. This fact is called eye tracking, as researched by Granka et al. and shows that the reason for the wrong results by users is high rank. Thus, clicking on the documents as user behavior is inherently noisy, and users often click on results of low quality [5]. However, 82% of clicked-on documents are relevant to the query topic [6]. Therefore, user's click can be the useful knowledge in the ranking of documents involved. On the other hand, machine learning as a powerful tool creates better ranking results than the basic method. Bottleneck in this category is to produce a model that is appropriate for the new query ranking after training.

The general approach of this paper is a learning method with a list-wise approach. This paper proposes ranking algorithm that is able to meet user's need. For this purpose, we need to know the user's interests. The user should be involved in the learning process. For this reason, ranking has been paid attention to as the problem of reinforcement learning. Reinforcement learning is a way agent (ranker) can obtain knowledge of the environment without the basic knowledge required in the environment, selects the documents to display the user, and presents proper ranking for queries similar to the next one.

The rest of the paper is organized as follows: In section 2, a number of ranking methods based on learning are reviewed. In section 3, an overview of reinforcement learning is expressed. The proposed algorithm is presented in section 4. In addition to the criteria and the data of evaluation described in section 5, experimental results are evaluated. Finally conclusion and recommendations for future are explained.

2. Related works

In recent years, ranking based on learning has become a hot topic of research in the field of information retrieval [7]. Learning-based ranking methods are classified into three general categories: point-wise, pair-wise and list-wise methods. Point-wise approach is the simplest method to learn. The simple and common idea of this method is mapping each query-document pair to the numeric value assigned to the amount of their relevance.

Linear regression (called REG) is a point-wise method based on statistics. REG aims to learn a linear ranking function in which the feature vector (multiple items) is mapped into the real value [8]. McRank method is based on the class of probabilities including incremental tree algorithm of multiple classification, ordered multiple classification and regression. This method uses combined ranking to minimize the number of pairs, which are ranked incorrectly [9]. A3CRank algorithm combines the results of algorithms like PageRank [10], BM25 [11] and TF-IDF [12], using the user`s feedback. This method is adaptive based on three components: connectivity, content and user behavior. The goal of agent is to maximize the number of clicks on the pages of high quality [13]. DistanceRank algorithm is based on learning used to receive penalty the logarithmic distances of pages. Its purpose is to minimize the total penalty. Distance calculation process continues until it converges to a constant value [14]. In [15, 16], Authors considered the problem of ranking in a graph-based data representation and proposed RL_Rank algorithm. This algorithm is based on the generalization of the reinforcement learning concepts for learning the ranking functions on graphs. RLRAUC ranking algorithm is based on reinforcement learning with point-wise approach and uses user feedback was proposed by Derhami et al [17]. The main idea behind the proposed method is based on consider value for each single word of query that is able to distinguish between relevant queries. FPR-DLA algorithm based on fuzzy logic and DLA is proposed to rank web pages. The algorithm consists of three stages. First, the weight of the links between web pages is determined using a set of learning automata. Secondly, the weight of each page is calculated. After weighting the web pages and links, web pages rank is calculated using a recursive formula. The higher weight of pages shows the greater user’s interest in the pages [18]. Forsati and Meybodi presented a combination algorithm based on distributed learning automata and PageRank algorithm. The proposed algorithm uses simultaneously navigate information of user and link between pages in order to offer the pages to the user [19]. Yarahmadi proposed an algorithm in which a learning automata is assigned to each page the task of which is to learn the connection of each page with other pages. The proposed algorithm recursively obtains the rank of a learning automata derived from the ranks of the other automata [20]. Saati has proposed a method based on distributed learning automata. In this method, learning automata is assigned to each document the task of which is learning the connection of a document with other documents. Any document is displayed with a content vector of length equal to the number of subjects in the system [21].

In the pair-wise approach, document pairs are considered as sample of learning and problem of learning is formulated for ranking as a classifier. This approach extracts the document pairs from rank lists, and each document pair has a label which takes into account the partial correlation between the two documents. Then, the model trains classification with the labeled data and makes a classification model in ranking. Yule Freud et al. have proposed the RankBoost algorithm which is ranking based on learning that combines several ranking methods. This algorithm acts similar to AdaBoost algorithm for clustering document pairs. The only difference is their approach. RankBoost is defined on a pair of documents [22].

Joachims proposed a pair-wise algorithm (called SVMRank) using data click based on learning formulation for ranking as a problem of binary classification of document pairs carried out by SVM. The probability of clicking on ranked document is directly related to query-document relevance [23].

List-wise approach is more efficient than other two approaches [24]. It is assumed that all document pairs or document points have specific features, although the feature selection is not unbiased in information retrieval and features are dependent on queries, which are large in numbers.

There is a list-wise learning method called ListNet which optimizes loss function using a list of k high probabilities of ranking results. Neural network as a model and a gradient descent is used instead of optimization algorithm. ListNet acts similar to RankNet except for the samples used are in the documents list [24]. The problem of ListNet is high time complexity of O(m.m!).

LambdMART is the incremental tree version of LambdRank. LambdMART has inherited the major benefits of LambdRank and MART methods. In the implementation of LambdMART, MART is used to determine the appropriate gradient and Newton step [25].

Volkovs and Zemel have recommended BoltzRank method. Using the conditional probability distribution, it ranks documents to the user's query. The idea is to define a probability distribution for permutations of document and prediction of the performance evaluation of the distribution [26].

Diaz-Aviles et al. have provided PSO-based ranking algorithm of SwarmRank, which tries to learn linear combination of the many ranking functions. SwarmRank goal is to optimize the MAP assessment criteria [27].

Pen et al. have suggested a ranking method based on learning from the combination of rankings of web documents and relevant scores [28]. In this method, the efficient function is list-wise (which is called PERF) coded by the relevant scores. To achieve better results, the proposed ranking algorithm (derived from AdaRank) has been combined with MAP or NDCG evaluation functions.

Akbari has proposed a ranking algorithm based on learning automata LRUF that makes use of user`s feedback. LRUF significantly improves ranking by the accuracy of the rewards commensurate with the degree of relevance of each document. Moreover, the proposed ranking is carried out according to the location of each document in ranking list, and ranking updates score. In this method, the selected document whose choice probability is low and removes this and is replaced by one of the other documents reducing the effect of the rich-get-richer [29]. Due to calculating the probability of all the documents in each step, the algorithm computational complexity is high.

Hofmann supposes that there are two ranking lists. First, the probability distribution of documents is calculated and the list of the interleaved result is generated. Then, all permutations of documents, which have non-zero probability, are being observed. After the user's click on the documents, the results of all possible states are guessed. The proposed framework improves the performance of automatic learning search engine by establishing a balance between exploration and exploitation [30].

Learning automata-based algorithms are provided to find the best combination of individual rankings. In the algorithm LARF, learning automata A determines the final ranking function. Learning automata, based on the user's feedback, adjusts the weight of each data source [31].

The problem in this method is that the relevant document related to the selected source is considered although it is possible for document in the first source of information to be in a lower rank than any other source of information. The first source is likely to increase and other data sources are fined due to fewer numbers of user feedback.

Raman et al. have proposed online learning-based ranking algorithm to create a balance between relevance and diversity [32]. At each stage, a user is provided with the results of ranking algorithms. Theoretically, the efficiency of the algorithm has increased and the algorithm is resistant to noise [32].

DBGD ranking algorithm is online and based on reinforcement learning [33, 34]. The algorithm aims to establish a balance between exploration and exploitation of the experience to improve the effectiveness of information retrieval systems during learning. This method is resistant against noise. Random exploration cost is high.

3. Reinforcement learning

In a reinforcement-learning problem, we face the agent made to interact with the environment through trial and error and learns to select the optimal action to reach agent's goal. The agent's goal is an attempt to maximize the total amount of reward it receives from the environment [35]. Reinforcement Learning is a way to train agents to perform an act via considering reward or punishment, without a need to specify how the action must do for the agent [36]. Thus, there are two main strategies for doing it. One is using evolutionary algorithms seeking an action in behavior space, which may lead to the aim in the environment, and the other one is the use of statistical methods and dynamic programming.

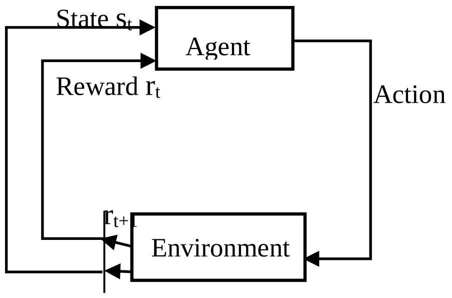

Figure1. The framework of Reinforcement learning [36].

The goal of Reinforcement Learning is to find the optimal policy so as to maximize the expected value for all states.

3.1. Q-learning

Q-Learning is one of the most widely used methods for reinforcement learning. In this method, the agent chooses the optimal policy with no perfect model of the environment.

Q-Learning actually works with values of the action-value. This amount is an estimate of the amount of reward that will be received for the current state of action in future.

Environment goes to the next state and gives the reinforcement signal r to the agent. Action-value values are updated using the following formula:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha_t \left[r_t + \gamma * \max_{b \in A} Q(s_{t+1}, b) - Q(s_t, a_t) \right] \quad (1)$$

where, the values of s_t and a_t, respectively, represent the state and action at time t. α_t is learning rate, r_t is immediate reward and γ is discount factor. $Q(s_{t+1}, b)$ represent the value of the next state. In learning, policy applied in selecting the next action does not affect the updating value of action-value and final value of action-value function is calculated independent of the policy made to take an action [36].

4. The proposed method

The proposed ranking algorithm is based on reinforcement learning and user's feedback. For this purpose, data sources (features) of query-document pair are used. The proposed method has a list-wise approach and is a member of query-dependent methods.

The idea of the proposed algorithm is valuing the features of a query-document pair. Each feature represents a certain aspect of a document or query. Using a number of features together covers imperfections; In other words, the page that is relevant in a user's view can have content relevant or links to other related pages or may be of interest for user because of relevance to other relevant page. Then, taking into account multiple relevant features provides additional results and the weaknesses of using each feature alone are reduced.

4.1. Formulation of the problem

In this subsection, the performance of a ranking system for the ranking of documents relevant to the user`s query is described.

Environment includes user and documents. The ranking system as an agent selects ten documents and shows them to the user. User clicks on documents and the important degree of features of the document, according to the terms stated in the article, receives awards or fines. During this process, each iteration causes the documents to be sorted in the list for a user who is the main purpose of the ranking system.

The first stage of the rankings starts with selecting ten documents from the documents and showing them to the user. This selection of documents is carried out similar to a semi-greedy method and rotating wheel. At first, it is performed heuristically and there is a probability of document selection at any location of the list and

over time (increase of (t)) it is likely to increase the probability of the first ten documents, and because at this time the relevant documents have been moved to the top of the list, the trend goes to the efficiency of the knowledge acquired and the exploration of environment is reduced. This was done with (2) and (3). Two important factors in these equations are time and the document situation so that the increase of the former makes use of the knowledge acquired.
The second one is the document situation, which is determined according to the degree of importance (value-action) of the document. Using (2), (3) and (4), the values of value-action are used to select documents.
Then, the user clicks on some documents (users do not click on all documents that are displayed to find the answers). The clicked-on the document is checked. If the document is relevant and the feature reflects the relevance of document or if the document is irrelevant and the feature shows irrelevance of the document, the degree of importance is awarded or it is fined.
If the document is relevant, the user`s feedback feature related to document-query pair is awarded, otherwise, it is fined. On each step, the importance degree of document features and the user`s feedback document (value-action values) of the last document seen by the user is not rewarded or fined. (Rate of change of value-action for the final state is zero.) Repeating the process ends when the time reaches the ultimatum or the user query has expired.
It is noteworthy that, due to the fact that in the ranking of web documents, if we have many more features of documents, we can better describe the document and the better description of document will help to distinguish the relevant and irrelevant documents from each other. To sort the ranking list, the importance degree values are multiplied in the feature values, which are normalized numbers. A user`s feature value has more importance than other features. Due to the fact that the features of user`s feedback contains users` opinion and there is an aim to provide ranking according to the user's wishes. Then, this value is multiplied in one; this value is considered in terms of other feature numbers; its value is multiplied in the number of other features of the document.

4.2. Determining the behavior of user’s click

Before stating the performance of the proposed method, we explained about the user’s feedback used in this paper. Overall, the behavior of user’s clicks is noisy and users cannot click on the relevant documents accurately. Many users still click on the irrelevant documents in top ranks. This shows the high frequency of clicks on documents in the first ranks of list, even if the documents are not relevant to the query [3]. The frequency of relevant click on documents is defined in terms of click frequency for queries with regard to the fact what rank the first documents in the list of documents shown to the user is located in [3].
In this case, according to the frequencies obtained, a statistical model for eleven cases and ten probabilities for a click on the ten documents of top list are considered. According to this model, the user clicks on documents. Clicking on the top ten of the documents of list continues according to the statistical model continues to be seen as a relevant document or ten first document according to statistical pattern until a relevant document is observed or ten first documents are clicked or left based on the probability.

4.3. RL3F: Reinforcement learning based function ranking using user feedback & web documents features

In RL3F algorithm, the reinforcement signal is assumed to be constant and is received in terms of the position of clicked-on document in the sorted documents list based on the same feature and relevance degree. Other features are effective in determining the importance degree of feature. Based on this fact that the features provide descriptions with a document better and it is their actions together which make their importance degree. In each learning cycle, only is a list based on the sum of values from user’s feedback and multiplication of features in their importance degree attained.
The RL3F method is as follows: First, the user enters the query. The ranking system is agent as a list of documents is shown to the user. The features are extracted from documents. Each list is arranged according to one of features of the document and there is a list for each in which position of documents do not change in learning process in lists of features. User clicks on the documents. Rewards or punishments are rewarded to feedback clicked-on document features and regarding the position of clicked-on document in the list, reward or punish is given to the feature in the corresponding list. Ranking system as an agent shows a main list of documents to user. Selecting documents to display to the user is the action of agent. The documents in the main list are selected by priority random method regarding the modified scores.

The selection method similar to ε-greedy method [36] is used so that over time the probability of selecting the first ten documents increases with the probability of the ε. At first, due to the lack of knowledge, selection probability is similar to roulette wheel action in which according to the document in the list, priority document is determined and the documents in upper positions of list have a greater priority.

The value of ε is zero in this case. As the user gets more knowledge about the environment, the relevant documents are listed on the top ranking and will have a higher priority. Then, ten top ranks will have high probability to be chosen and value of ε increased to one-tenth. On the other hand, over time, seeking knowledge from the environment makes the documents in the lower ranks be considered as irrelevant documents. Therefore, the selected probability of these documents is approaching zero.

This is done with the same incremental roulette wheel and an increased value of ε with the function of action-value. This will be more likely to choose other documents and list convergence to ranking optimization becomes faster. With this method, the documents will be selected for display to the user. This method is Action selection proposed for RL3F approach. Probability of document selection is calculated according to the following formula:

$$m(x) = \begin{cases} \frac{2*(\beta*(n+1-x)-(v*t))}{n*(\beta*(n+1)-2*v*t)} & 1 \le x \le \left\lceil n-\frac{v*t}{\beta} \right\rceil \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

$$p(x) = \begin{cases} 0.5*(m(x)+\varepsilon+\frac{1-\varepsilon}{n}) & 1 \le x \le 10 \quad 0 \le \varepsilon \le 0.1 \\ 0.5*(m(x)+\frac{1-\varepsilon}{n}) & 10 < x \le n \quad 0 \le \varepsilon \le 0.1 \end{cases} \quad (3)$$

where, m(x) represents a probability in incremental roulette wheel method. Over time, the probability of documents in the low ranks gets to zero. x is used to display the position document in the ordered list of documents. n is the number of documents per relevant query, and the initial value of v equal to one linearly increasing. t is time. After the user views all queries, the value of t is increased.

This procedure is repeated until the main list converges to a fixed list and the training phase is completed. Importance degree is considered to be the score of each feature. By applying the sum of the value of the feature multiplied by its weight, the score of each query-document pair of test phase is achieved.

4.3.1. Procedure of RL3F algorithm

This method consists of two training and testing phases. In the training phase, the proper ranking is based on reinforcement learning. At this stage, the degree of importance of each feature using rewards and punishments is determined. In the test phase by applying summation operation on multiplication of importance degrees in the feature values, we could assign a value score to the document and then sort the documents in the scores to produce the rank list.

4.3.1.1. Training phase

The procedure of training phase is as follows: n features are considered for each document. The goal is to find the convergence of sorted list based on features and user's feedback to a fixed list during which the importance degree of features is determined as shown by E. Initially, there is no knowledge about importance degree. The importance degree is taken to be same (E=1/n). Then, for each feature a list will be provided. The documents of each list are sorted in the value of corresponding feature and are used to determine the position of clicks on documents to score the corresponding feature. Then, a main list is provided for query related documents. The scores of documents are found from the following formula and sorted in descending format to be shown to user.

$$\text{Score}_{d,q} = \sum_{i=1}^{n} E_i * F_{i_{d,q}} + n * fe_{d,q} \quad (4)$$

where, $\text{Score}_{d,q}$ denotes the score of query-document pair. n is the number of features and $F_{i_{d,q}}$ is the value of ith feature related to query-document pair and E_i shows the corresponding importance degree of ith feature. $fe_{d,q}$ denotes the value of feature of user's feedback for paired query-document (d, q). Due to the important feature of user feedback, its score is multiplied in the number of other features.

Among the sorted documents, ten are chosen to be shown to the user, based on a priority random method (combination of roulette wheel and ε-greedy). The user will click on the displayed documents. If they are relevant, the feature of user's feedback will take reward, otherwise it takes punish. For other features, based position of clicked on document in the sorted list, the score is attributed. If the clicked-on document is relevant and in the first 50 documents, it takes reward. If the clicked-on document is irrelevant and on the top of the list (first 50 positions of list), the corresponding feature will be fined while located in the low ranks, it is rewarded. In other words, if the feature has sorted the document properly, it is given reward, otherwise, it is given a penalty. The importance degree is calculated as follows:

$$\alpha = e^{-\beta t} \tag{5}$$

$$e_i(t+1) = e_i(t) + \alpha \times [-e_i(t) \mp r] \tag{6}$$

$$fe_{d,q}(t+1) = fe_{d,q}(t) + \alpha \times [-fe_{d,q}(t) \mp r] \tag{7}$$

where, α denotes the learning rate. β shows the size-step and value is 0.01. t represents time for which at t = 0 learning rate is one and over time it is close to Zero and learning is completed. r shows the reward. The value of this parameter is constant. $fe_{d,q}(t)$ shows the value of user feedback for query-document pair at time t. $E_i(t)$ shows the weight corresponding to ith feature at time t-th. The learning procedure is repeated so that the document list converges to a fixed list. The pseudo-code and Module of RL3F algorithm are shown in figures 2 and 3.

4.3.1.2. Test phase

After the completion of training phase, the importance degree of each feature is specified in terms of the score; it has received and regarding the importance degree as weight, the score of each pair query-document is calculated as follows:

$$s_{d,q} = \sum_{i=1}^{m} E_i \times F_{i_{d,q}} \tag{8}$$

where, $s_{d,q}$ denotes the score of query-document pair. $F_{i_{d,q}}$ denotes the value of ith feature of query-document pair and E_i is the importance degree of ith feature.

4.4. Review of algorithm proposed

Ranking web documents as a bandit problem is formulated. Selecting ten documents from the list of documents related to query of the user to show the user is intended as agent`s action and reward is allocated to the importance degree of document features which causes the documents list to be sorted.

First, the selection is in a non-greedy way which is heuristic and over time, becomes greedy which means to use the prior knowledge.

(In this ranking algorithm according to the scores that are calculated in terms of the value-action, related documents will be moved to the top of the list and unrelated documents to the bottom of the list.)Each reinforcement signal decreases or increases the degree of importance of the features and user`s feedback characteristics of web documents so that the immediate reward is considered to be equal to a constant value in the range of ± [0.5, 0.85]. Value-action for feature importance degree and user`s feedback feature is calculated. If selecting a document is done from ten documents in the first rank of ranking list, a greedy action and current knowledge have been in operation, which is the value-actions. If, instead, a non-greedy act is selected, meaning the document is not selected in the first ten ranks then ranking system is exploring the environment and acquires knowledge that will adjust the values of action-value.

Each bandit problem is equivalent to a problem of MDPs (Markov decision processes). The issue in the article is the number of equal situations of document-query pairs. Value of action- value is calculated according to (6) and (7). Equation 6 is devoted to the values of feature importance degree that is considered for the number of features in the document.

Transition probability for each document is the feature value of the query-document pair between 0 and 1. Feature of user`s feedback has been calculated for each pair of query -document the transition probability of which is considered to be one because of this feature importance compared to other features, Its value is multiplied in the number of other features. Immediate reward values are considered to be constant value in the range of ± [0.5, 0.85]. The positive value is assigned to feature importance degree if the related document ranks first or unrelated document is displayed at the bottom of the ranking or immediate reward is considered negative. In other words, if the document is properly ranked, the feature receives award or it will be fined. On the other hand, if the clicked-on document is relevant, the user`s feedback features for document-query pair receives award or it is punished.

Parameters used in the pseudo-code RL3F

1: n: count of queries
2: t: number repeats or time
3: d: documents
4: q: queries
5: m: count of documents in dataset
6: g: count of features of document
7: vf: value of feedback user for pairs document-query
8: PTR: position of the first relevant document in the list
9: value_click is frequency distribution of relevance of users' clicks on web search results that this is a matrix 10*11
10: E: the importance degree of features in phase train and stored in array with length g
11: Rank_test$_i$: Ranking list for q_i in phase test

12: $v_{(d,q)i}$: vlaue of ith feaure for (d,q) th pair of document-query.

```
RL3F
Input
1:     d, // Matrix by dimensions of all documents for any query*query,
       q // query list,
       value_click // matrix by dimensions of 10*11 contain relevant frequency distribution of users' clicks on web search results
       v // matrix for value of features of document-query pairs
Output
2:     E, // importance degree of the feature,
       Rank_test //finally list Ranking
Assumption
3:          if $\exists(q_i,d_j)$ then $\exists d_{i,j}$
Initialize
4:   t=0, punish =-1*reward , $\beta$=0.01
5:   For all vf Set $vf_{(i,j)}$=0
6:   For i=1,2,...,g repeat Set E=1/g
7:   For i=1,2,...,g repeat provide a sorted list of documents relevant to query using feature i-th
Begin
8:   While t<N
9:       $\alpha=e^{-\beta*t}$ //learning rate $\alpha\in(0,1)$
10:      For i=1 to n
11:   Sort list documents ($d_{i,j}$) by $vf_{k,j}$*g+ sum($E_i$*$v_{(d,q)i}$) & Select 10 document by 2-3 & set in R list
12:       R Ranking list is shown to the User
13:   PTR is position of the first relevant document ($d_{i,h}$) in the R list
14:        User clicks on documents in R by probability $value\_click_{PTR,i}$
15: if document is relevant then // reward
16:  For p=1 to g //for all features
17:    If this document is at top position of the list sorted feature pth then
18:        $E_p$ (t+1)= $E_p$ (t)+ $\alpha$*[reward-$E_p$ (t)]
19:    else
20:        $E_p$ (t+1)= $E_p$ (t)+ $\alpha$*[punish-$E_p$ (t)]
21:    end
22:  end
23:        $vf_j$(t+1)= $vf_j$(t)+ $\alpha$ *[ reward -$vf_j$(t)]
24:        Break repeat for this query
25:  else if document is irrelevant then //punish
26:     For p=1 to g //for all features
27:          If this document is at the bottom position of the list sorted feature p-th then
28:        $E_p$ (t+1)= $E_p$ (t)+ $\alpha$*[reward-$E_p$ (t)]
29:     else
30:        $E_p$ (t+1)= $E_p$ (t)+ $\alpha$*[punish-$E_p$ (t)]
31:     end
32:    end
33:       $vf_j$(t+1)= $vf_j$(t)+ $\alpha$ *[ punish -$vf_j$(t)]
34:    end //if
35:     t=t+1
36:    until query session is expired
37:  end
38:      For i=1 to m   // for all documents
39:        For j=1 to n   // for all queries
40:          $rank\_test_{i,j}$= $\Sigma v_{(i,j,k)}$*$E_k$
41:        end
42:      end
```

Figure 2. pseudo-code of the proposed algorithm RL3F.

4.5. Proposed algorithm overview

In summary, the proposed algorithm is as follows:

1- Initial value

The immediate award rate is considered constant. First, because there is no knowledge, an initial value of feedback feature is considered zero. Importance degree values are also considered identical.

For each feature p, documents related to query are sorted in terms of feature p and list p is formed.

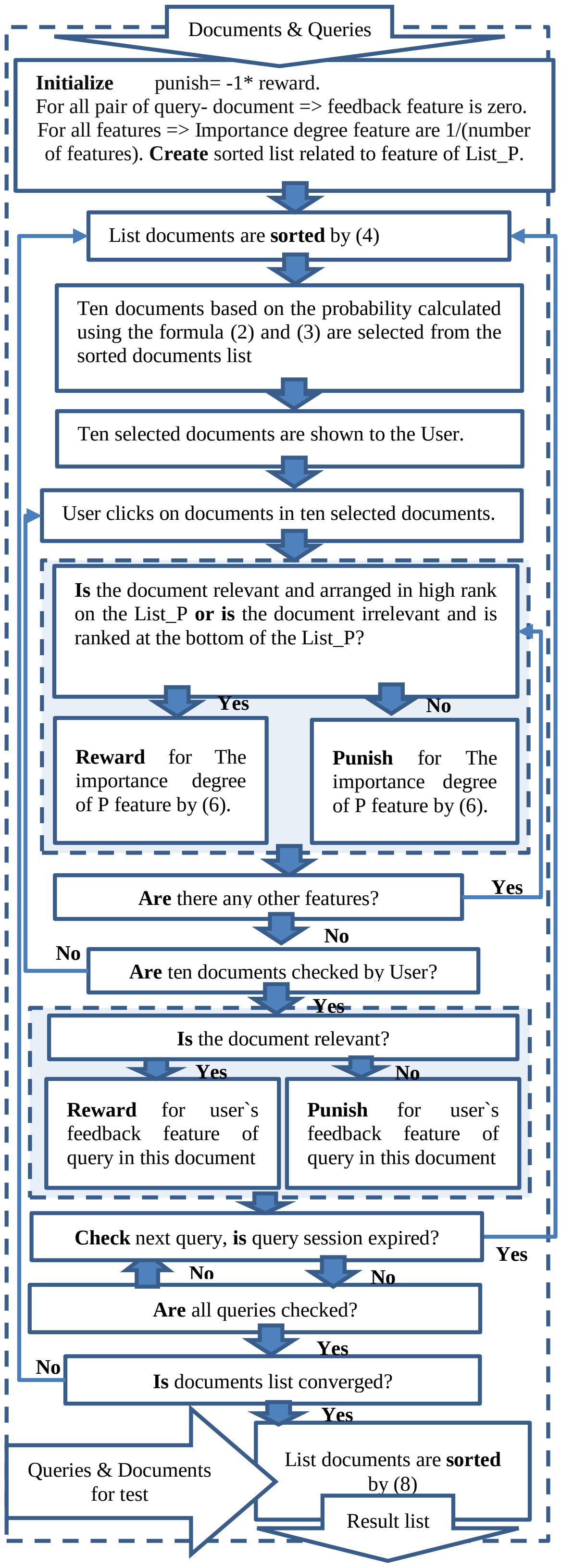
Documents & Queries
Initialize punish= -1* reward.
For all pair of query- document => feedback feature is zero.
For all features => Importance degree feature are 1/(number of features). Create sorted list related to feature of List_P.
List documents are sorted by (4)
Ten documents based on the probability calculated using the formula (2) and (3) are selected from the sorted documents list
Ten selected documents are shown to the User.
User clicks on documents in ten selected documents.
Is the document relevant and arranged in high rank on the List_P or is the document irrelevant and is ranked at the bottom of the List_P?
Yes
No
Reward for The importance degree of P feature by (6).
Punish for The importance degree of P feature by (6).
Are there any other features?
Yes
No
Are ten documents checked by User?
No
Yes
Is the document relevant?
Yes
No
Reward for user`s feedback feature of query in this document
Punish for user`s feedback feature of query in this document
Check next query, is query session expired?
Yes
No
No
Are all queries checked?
Yes
No
Is documents list converged?
Yes
Queries & Documents for test
List documents are sorted by (8)
Result list

Figure 3. Module of the proposed algorithm RL3F.

2- Begin the process

Step 1: Learning rate is calculated by (5) (as learning is completed, the learning rate also decreases exponentially.) (t increase decreases the learning rate)

Step 2: Given the value of document scores, which were calculated by (4), the list of documents is sorted in descending format.

Step 3: Ten documents based on the probability calculated using the formula (2) and (3) are selected from the sorted documents list.

Step 4: Ten selected documents are displayed to the user and the user clicks on the document d based the user`s click table (table containing the frequency of distribution of the user`s clicks on documents top of list seen [3].)

Step 5: If the document d is relevant and arranged in high rank on the sorted list related to feature of List_P or if the document d is irrelevant and is ranked at the bottom of the List_P, then according to equation the importance degree of P feature is awarded or according to (6), fine is allocated to importance degree of feature p.

Step 6: step 5 is repeated for all the features.

Step 7: If ten documents are not checked; then, based on the frequency of distribution table of the user`s clicks, the user clicks on the other documents, and step 4 is repeated. (Hint- the last document seen of ten is not considered punishing or rewarding.)

Step 8: If the document d is relevant, user`s feedback is given the reward, and or it is given the punishment.

Step 9: steps 2 through 7 are repeated to expire the user`s query or the number revolutions paved causing the learning rate to approach zero is obtained and learning is completed.

3- Test

Step 10: The document scores are calculated according to the (8).

Step 11: step 10 is repeated for all documents relevant to the query, and the final list is sorted based on scores in descending order.

5. Evaluation and analysis of experimental results

The evaluation and comparison of methods need the same conditions. Therefore, it is necessary to have a criterion to evaluate and test the different algorithm. The benchmark dataset related to ranking includes three components of document set, a class of information in query format and human’s judgment related to query-document pair [37].

5.1. Evaluation criteria

The evaluation criteria of P@n (Precision at nth place), MAP (Mean Average Precision), and NDCG (Normalized Discount Cumulative Gain) are widely used to assess the information retrieval. These criteria are supported by the tool set of team LETOR (Learning to Rank for Information Retrieval) [7]. Using these tools, we can have neutral and objective evaluation to get a fair comparison.

❖ **P@n**

This criterion shows the number of related documents in the first n documents of final ranking for each query. The formula for p@n is as follows.

$$\mathrm{P@n} = \frac{\mathrm{NoR_n}}{\mathrm{n}} \tag{9}$$

In which $\mathrm{NoR_n}$ shows the number of related documents to n top positions of final ranking.

❖ **MAP**

Map is presented as the average of AP values of all queries and AP is determined for each query as the average values of p@n for relevance documents [7].

$$\mathrm{AP} = \frac{\sum_{n=1}^{N}(\mathrm{P@n} \cdot \mathrm{R_e}(n))}{\mathrm{T_R}} \tag{10}$$

In which N, $\mathrm{T_R}$ and $\mathrm{R_e}(n)$ show the number of retrieved documents, number of relevance documents and binary function of nth relevant document, respectively, which show the relevant document with one and irrelevant document with zero.

❖ **NDCG**

The value of NDCG is a ranked list in the nth place.

$$\mathrm{NDCG} = Z_n \sum_{m=1}^{n} \frac{2^{r(m)} - 1}{\log(m+1)} \tag{11}$$

In which r(m) shows the relevance rate of mth document in the ranked list and Z_n is normalization constant. $2^{r(m)} - 1$ shows the gain of mth document. $\frac{2^{r(m)}-1}{\log(m+1)}$ shows the decreasing gain and $\sum_{m=1}^{N}\frac{2^{r(m)}-1}{\log(m+1)}$ shows the decreasing accumulative gain in nth position.

5.2. The benchmark dataset

In recent years, the data set of LETOR [38] has been developed for learning-based ranking investigation. The dataset applied in this paper is the dataset of OHSUMED from the version of LETOR3. OHSUMED is a subset of MEDLINE introduced as database of medical journals. This set consisted of 348566 records the field of which include title, abstract and Mesh indexed words, author, references and publication type. This set

consists of 106 queries and has 16140 query-document pairs with relevancy degree. There are three defined three levels for user`s judgment: relevant, a little relevant, and irrelevant. OHSUMED includes 45 features. These features are dependent on query-document pair. Some of these features are dependent on query-document pair.
Some are dependent on document and some are dependent on query [7,35]. In LETOR, each dataset consists of five data folds and each of which includes three subsets of test, train and validation [7].

5.3. Evaluation of RL3F algorithm

The conditions investigated are:
1. The learning rounds are 100.
2. The value of immediate reward is within [0.5, 0.85].
3. The number of features under consideration is 17.

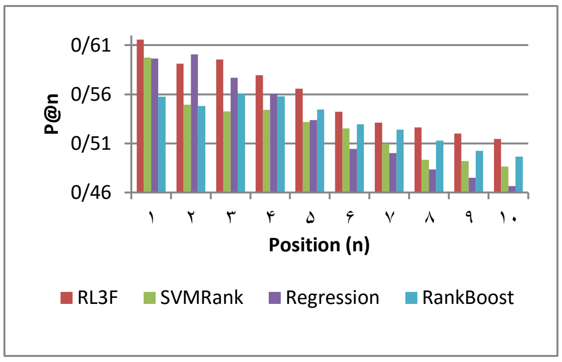

Figure 4. Comparison of proposed algorithm with algorithms expressed regarding the evaluation criterion P@n.

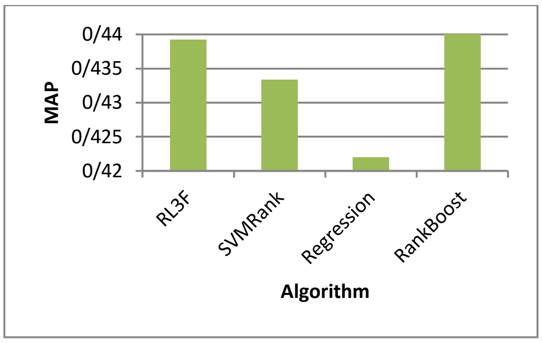

Figure 5. Comparison of proposed algorithm with algorithms expressed regarding the evaluation criterion MAP.

Figures 4-6 show the superiority of the proposed algorithm to RankBoost, SVMRank and Regression in terms of P@n and NDCG@n and superior to SVMRank and regression in terms of MAP. One of the reasons is the better performance of reinforcement learning than SVM in online case.

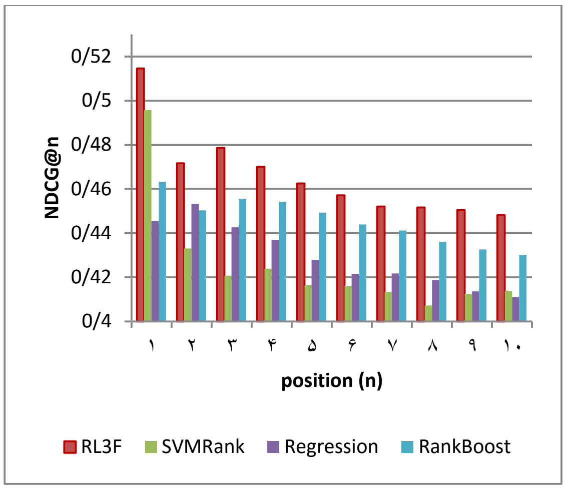

Figure 6. Comparison of proposed algorithm with algorithms expressed regarding the evaluation criterion NDCG@n.

The other reason is that the list-wise approach outperforms pair-wise approach. While in both methods user`s feedback and list features are used, there are presented better results in list-wise approach. One of the ranking problems of SVMRank is focusing low and middle ranks. In contrast, the focus of proposed algorithm of RL3F is on finding the related documents and inserting them on the top ranks which resolves the problem of algorithm SVMRank. The final ranking model in algorithm SVMRank is strongly influenced by a query with more relevance documents.
In proposed algorithm of RL3F, as the value is taken in feature, the values are calculated regardless of belonging to a query. Therefore, the proposed algorithm will not have the problem of SVMRank problem. As stated in section 2, the regression algorithm has a point-wise approach which is the indicator of its weak performance versus the proposed algorithms. As with RankBoost algorithm, the results show the superiority of proposed algorithm in terms of evaluation criteria of p@n and NDCG@n based on the better performance of reinforcement learning versus boosting. All in all, a suitable ranking of proposed algorithm versus algorithms of RankBoost, SVMRank, and regression is observed for experimental results.
In the proposed algorithm of RL3F, as the value is taken in feature, the values are calculated regardless of belonging to a query. Therefore, the proposed algorithm will not have the problem of SVMRank problem. As stated in section 2, the regression algorithm has a point-wise approach which is the indicator of its weak performance

versus the proposed algorithm. As with RankBoost algorithm, the results show the superiority of proposed algorithm in terms of evaluation criteria of P@n and NDCG@n based on the better performance of reinforcement learning versus boosting. All in all, suitable ranking of proposed algorithm versus algorithms of RankBoost, SVMRank, and regression is observed for experimental results.

6. Conclusion

In this paper, ranking web pages were studied as a reinforcement learning problem so that ranker uses user`s feedback as a reinforcement signal and reinforcement learning reduces the effect of user`s click innate noise and the suitable result for users even with low quality is provided.

Two methods of proposed ranking are dependent on query with a list-wise approach. In this paper, it was noted that each method based on content and connection has problems and uses any single one results in the low efficiency of search engine. Applying different features together is a solution to solve the problems of these two methods used in algorithm of RL3F.

The proposed algorithm becomes converged rapidly as it uses an action similar to ε-greedy one to choose the documents observed by users. Using user feedback in the method expressed shows the adaptability and reliability of the proposed algorithm using noisy clicking. To evaluate the dataset of criterion LETOR3, dataset of OHSUMED was especially used. The results show the superiority of the proposed algorithm to SVMRank algorithm.

In the proposed algorithm, a different combination of features and an increasing number of features can improve the algorithm performance.

References

[1] Henzinger, M. R., Motwani, R., & Silverstein, C. (2002). Challenges in web search engines. In ACM SIGIR Forum, vol. 36, no. 2, pp. 11-22.

[2] Barabási, A. L., & Albert, R. (1999). Emergence of scaling in random networks, science, vol. 286, no. 5439, pp. 509-512.

[3] Agichtein, E., Brill, E., & Dumais, S. (2006). Improving web search ranking by incorporating user behavior information. In Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval (pp. 19-26). ACM, 2006.

[4] Joachims, T., Granka, L., Pan, B., Hembrooke, H., Radlinski, F., & Gay, G. (2007). Evaluating the accuracy of implicit feedback from clicks and query reformulations in web search. ACM Transactions on Information Systems (TOIS), vol. 25, no. 2, pp. 7.

[5] Granka, L. A., Joachims, T., & Gay, G. (2004, July). Eye-tracking analysis of user behavior in WWW search. In Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval (pp. 478-479). ACM, 2004.

[6] Xue, G. R., Zeng, H. J., Chen, Z., Yu, Y., Ma, W. Y., Xi, W., & Fan, W. (2004, November). Optimizing web search using web click-through data. In Proceedings of the thirteenth ACM international conference on Information and knowledge management (pp. 118-126). ACM, 2004.

[7] Qin, T., Liu, T. Y., Xu, J., & Li, H. (2010). LETOR: A benchmark collection for research on learning to rank for information retrieval. Information Retrieval, vol. 13, no. 4, pp. 346-374.

[8] Li, L., & Lin, H. T. (2006). Ordinal regression by extended binary classification. In Advances in neural information processing systems (pp. 865-872), 2006.

[9] Li, P., Wu, Q., & Burges, C. J. (2007). Mcrank: Learning to rank using multiple classification and gradient boosting. In Advances in neural information processing systems (pp. 897-904), 2007.

[10] Page, L., Brin, S., Motwani, R., & Winograd, T. (1999). The PageRank citation ranking: Bringing order to the web, Technical Report. Stanford InfoLab, (pp. 1-17), 1999, Available: http://ilpubs.stanford.edu:8090/422/.

[11] Robertson, S. E., & Walker, S. (1994). Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval (pp. 232-241). Springer-Verlag New York, 1994.

[12] Salton, G., & Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. Information processing & management, vol. 24, no. 5, pp. 513-523.

[13] Zareh Bidoki, A. M., Ghodsnia, P., Yazdani, N., & Oroumchian, F. (2010). A3CRank: An adaptive ranking method based on connectivity, content and click-through data. Information processing & management, vol. 46, no. 2, pp. 159-169.

[14] Zareh Bidoki, A. M., & Yazdani, N. (2008). DistanceRank: An intelligent ranking algorithm for web pages. Information Processing & Management, vol. 44, no. 2, pp. 877-892.

[15] Derhami, V., Khodadadian, E., Ghasemzadeh, M., & Bidoki, A. M. Z. (2013). Applying reinforcement learning for web pages ranking algorithms. Applied Soft Computing, vol. 13, no. 4, pp. 1686-1692.

[16] Khodadadian, E., Ghasemzadeh, M., Derhami, V., & Mirsoleimani, S. A. (2012). A novel ranking algorithm based on Reinforcement Learning. In Artificial Intelligence and Signal Processing (AISP),

2012 16th CSI International Symposium on (pp. 546-551). IEEE, 2012.

[17] Derhami, V., Paksima, J., & Khajeh, H. (2014). RLRAUC: Reinforcement learning based ranking algorithm using user clicks. In Computer and Knowledge Engineering (ICCKE), 2014 4th International eConference on (pp. 29-34). IEEE, 2014.

[18] Anari, Z., Meybodi, M. R., & Anari, B. (2009). Web page ranking based on fuzzy and learning automata. In Proceedings of the International Conference on Management of Emergent Digital EcoSystems (p. 24). ACM, 2009.

[19] Forsati, R., Meybodi, M. R., & Mahdavi, M. (2007). Web page personalization based on distributed learning automata. Proc. IKT, Ferdowsi University of Mashad, Mashad, Iran, 2007.

[20] Yarahmadi, T., Torkestani, J. A., & Fatemeh, Z. (2012). A new method based on distributed learning automata for page ranking in web. International Journal of Physical Sciences, vol. 7, no. 13, pp. 2066-2075.

[21] Saati, S., & Meybodi, M. R. (2006, May). Document Ranking using Distributed Learning Automata. In Proceedings of 11th Annual CSI Computer Conference of Iran, Fundamental Science Research Center (IPM), Computer Science Research Lab, Tehran, Iran, 2006.

[22] Phophalia, A. (2011, December). A survey on learning to rank (letor) approaches in information retrieval. In Engineering (NUiCONE), 2011 Nirma University International Conference on (pp. 1-6). IEEE, 2011.

[23] Joachims, T. (2002, July). Optimizing search engines using clickthrough data. In Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 133-142). ACM, 2002.

[24] Cao, Z., Qin, T., Liu, T. Y., Tsai, M. F., & Li, H. (2007, June). Learning to rank: from pairwise approach to listwise approach. In Proceedings of the 24th international conference on Machine learning (pp. 129-136). ACM, 2007.

[25] Burges, C. J. C. (2010). From ranknet to lambdarank to lambdamart: An overview. Microsoft Research Technical Report MSR-TR-2010-82, pp. 1-19.

[26] Volkovs, M. N., & Zemel, R. S. (2009, June). Boltzrank: learning to maximize expected ranking gain. In Proceedings of the 26th Annual International Conference on Machine Learning (pp. 1089-1096). ACM, 2009.

[27] Diaz-Aviles, E., Nejdl, W., & Schmidt-Thieme, L. (2009). Swarming to rank for information retrieval. In Proceedings of the 11th Annual conference on Genetic and evolutionary computation (pp. 9-16). ACM, 2009.

[28] Pan, Y., Luo, H. X., Tang, Y., & Huang, C. Q. (2011). Learning to rank with document ranks and scores. Knowledge-Based Systems, vol. 24, no. 4, pp. 478-483.

[29] Akbari Torkestani, J. (2012). An adaptive learning to rank algorithm: Learning automata approach. Decision Support Systems, vol. 54, no. 1, pp. 574-583.

[30] Hofmann, K., Whiteston, S., Schuth, A. & de Rjike, M., (2014). Learning to rank for information retrieval from user interactions, (pp. 1-7), Sigweb, 2014.

[31] Torkestani, J. A. (2012). An adaptive learning automata-based ranking function discovery algorithm. Journal of intelligent information systems, vol. 39, no. 2, pp. 441-459.

[32] Raman, K., Shivaswamy, P., & Joachims, T. (2012, August). Online learning to diversify from implicit feedback. In Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 705-713). ACM, 2012.

[33] Hofmann, K., Whiteson, S., & de Rijke, M. (2011). Balancing exploration and exploitation in learning to rank online. In Advances in Information Retrieval (pp. 251-263). Springer Berlin Heidelberg, 2011.

[34] Hofmann, K., Whiteson, S., & de Rijke, M. (2013). Balancing exploration and exploitation in listwise and pairwise online learning to rank for information retrieval. Information Retrieval, vol. 16, no. 1, pp. 63-90.

[35] Wang, Y., Lu, J., Liang, J., Chen, J., & Liu, J. (2012). Selecting queries from sample to crawl deep web data sources. Web Intelligence and Agent Systems, vol. 10, no. 1, pp. 75-88.

[36] Sutton, R. S., & Barto, A. G. (1998). Introduction to reinforcement learning. MIT Press.

[37] dotIR collection website, (2010) Available: http://ece.ut.ac.ir/DBRG/webir/fa/index.html.

[38] LETOR: Learning to rank website, (2014) Available: http://research.microsoft.com/en-us/um/beijing/projects/letor.

نشریه هوش مصنوعی و داده کاوی

الگوریتم رتبه‌بندی صفحات وب بر اساس یادگیری تقویتی و بازخورد کاربر

ولی درهمی[۱،*]، جواد پاکسیما[۲] و هما خواجه[۳]

[۱] دانشکده برق و کامپیوتر، دانشگاه یزد، یزد، ایران.

[۲] دانشکده فنی و مهندسی، دانشگاه پیام نور یزد، یزد، ایران.

[۳] دانشکده فنی و مهندسی، دانشگاه علم و هنر، یزد، ایران.

ارسال ۲۰۱۴/۱۱/۰۶؛ پذیرش ۲۰۱۵/۰۶/۰۲

چکیده:

چالش اصلی موتورهای جستجو، رتبه‌بندی اسناد وب برای ارائه بهترین پاسخ به پرس‌وجوی کاربران است. با وجود حجم زیاد نتایج استخراجی به ازای پرس‌وجوی کاربر تنها تعداد کمی از اولین نتایج توسط کاربران مورد بررسی قرار می‌گیرند، از این‌رو قرار دهی نتایج مرتبط در رتبه‌های نخست اهمیت خاصی دارد. در این مقاله، یک الگوریتم رتبه‌بندی مبتنی بر یادگیری تقویتی و بازخورد کاربر ارائه شده است که RL3F نامیده شده است. در الگوریتم پیشنهادی، سیستم رتبه‌بندی به عنوان عامل سیستم یادگیری و انتخاب اسناد برای نمایش به کاربر به عنوان عمل عامل در نظر گرفته‌شده‌اند؛ سیگنال تقویتی در این سیستم با توجه به کلیک کاربر بر روی اسناد محاسبه می‌شود. مقادیر ارزش-عمل در الگوریتم پیشنهادی به ازای هر ویژگی محاسبه می‌شود. در هر چرخه‌ی یادگیری، اسناد بر حسب امتیازات برای ارائه پرس‌وجوی بعدی مرتب می‌شوند و با توجه به موقعیت سند در لیست رتبه‌بندی، اسنادی به صورت تصادفی برای نمایش به کاربر انتخاب می‌شوند. روند یادگیری تا تکمیل آموزش ادامه می‌یابد. برای ارزیابی روش پیشنهادی از مجموعه داده محک LETOR3 استفاده شده است. نتایج ارزیابی نشان‌دهنده‌ی مؤثرتر بودن روش پیشنهادی نسبت به سایر روش‌های بیان شده است. دلیل برتری الگوریتم پیشنهادی استفاده همزمان چندین ویژگی سند و بازخورد کاربر است.

کلمات کلیدی: موتور جستجو، رتبه‌بندی، یادگیری تقویتی، بازخورد کاربر.