Journal of AI and Data Mining
Vol 7, No 3, 2019, 421-442
DOI: 10.22044/JADM.2019.3547.1814

RRLUFF: Ranking function based on Reinforcement Learning using User Feedback and Web Document Features

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

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

Received 30 March 2018; Revised 28 December 2018; Accepted 26 February 2019
**Corresponding author: khajeh121@yahoo.com (H. Khajeh).*

Abstract
The principal aim of a search engine is to provide the sorted results according to the user's requirements. To achieve this aim, it employs the ranking methods to rank the web documents based on their significance and relevance to the user's query. The novelty of this work is to provide a user feedback-based ranking algorithm using reinforcement learning. The proposed algorithm is called RRLUFF, in which the ranking system is considered as the agent of the learning system and the selection of documents is displayed to the user as the agent's action. The reinforcement signal in this system is calculated based on the user's click on the documents. Action-values in the RRLUFF algorithm are calculated for each feature of the document-query pair. In the RRLUFF method, each feature is scored based on the number of the documents related to the query and their position in the ranked list of that feature. For learning, the documents are sorted according to the modified scores for the next query. Then according to the position of a document in the ranking list, some documents are selected based on the random distribution of their scores to display to the user. The OHSUMED and DOTIR benchmark datasets are used to evaluate the proposed method. The evaluation results indicate that the proposed method is more effective than the related methods in terms of P@n, NDCG@n, MAP, and NWN.

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

1. Introduction

Currently, a search engine is an information retrieval tool used to search for information. Each search procedure starts with obtaining a user's query. Then the list of these results is displayed to the user, and the user looks for the related results. The key challenge in how to get this procedure is a proper ranking of the Web documents. Hence, efficient ranking algorithms are important to arrange the results related to user queries based on their relevance in a descending order.

The ranking methods include two general categories [1]: content-based and connectivity-based. The former suffers from the spam problem [2] and the latter is divided into two groups: query-independent and query-dependent. This kind of method has the problem of rich-get-richer [3]. Combinatorial methods reduce the problems of the two sets of methods due to the simultaneous use of the content and the connection. However, there is still the problem of ambiguity in the query, meaning that the users can enter the same query for different purposes and look for different results. For solving this problem, one can use the user's preference. These batch methods are called user-feedback-based algorithms, which provide good results [4].

As an alternative to the traditional information retrieval systems, learning to rank has become very popular in the recent years [5]. The existing ranking methods show that methods of learning to rank [4] create better results than basic methods do. The ranking based on learning is useful for adaptive filtering [6], question answering [7], search engine [8], personalized recommendation [9], and many other applications.

The problem with this kind of training method is the model creation that provides an effective ranking for new queries. The supervised and

unsupervised learning to rank methods assume that a representative set of training data is available at the learning time so that models can be created from this set, for instance, SVMRank [10], RankBoost [11], RankNet [12], LambdaMART [13], and ES-Rank [14]. In contrast to supervised/unsupervised learning, using reinforcement learning, a ranking system learns directly from interactions with the user. Moreover, Reinforcement Learning (RL) collects labeled training data through interaction with the user. In addition, it requires no label to learn. Therefore, the proposed method uses an RL.

On the other hands, research studies indicate a tendency of the users to click on the results with top rankings in the ranking list [15]. For example, in Google, the user's click-through rate on documents in the first rank is 7.11%, which is 3.01% more than the second-highest and 2.19% more than the third-highest [16]. Overall, the click rate for the first ten documents is 52.32% [17]. The research results by Granka et al. [18] on eye tracking show that the reason for users' click on incorrect results is their high ranking. Therefore, clicking on the documents is inherently an intrusive behavior that results in a click on poor quality results [18]. The frequency distribution of relevance of user's clicks on search results, examined by Agichtein et al., shows that the number of user clicks on documents decreases with a decrease in rank [19]. For example, relatively 75% of users never scroll past the first page of a search engine [20], and in Google, they click on documents ranked the second, third, and fourth clicks, with a probability of 56.36%, 13.45%, and 9.85%, respectively [16].

By the same token, these cases highlight the fact that users often look for the answers to the queries in the first ranks and click on the top ranking, even if the documents are not relevant. Nevertheless, the desired result of user's queries in 66% of his/her search works is achieved with one or more clicks on the results [21]. In addition, researchers show that click models assist to improve the efficiency of search engines [22]. In other words, click models improve the ranking function [23]–[25]. Moreover, the accuracy of the ranking algorithms is improved over 31% than the original performance by using an implicit feedback (such as user click) [25]. In this paper, we can use the user's click model as an effective data in the ranking.

According to the discussion above, the general approach of this paper is to provide a learning-based ranking method. The purpose of this article is to provide an effective ranking algorithm in accordance with the user's needs. For this purpose, the user must be involved in the learning process in order to understand the user's performance. Thus in this paper, the ranking of web documents is considered a reinforcement learning issue. In this problem, the ranking system is the agent and the selection of documents to be displayed to the user is considered as an action. This selection of action is performed by a novel action selection method, which is called BoostRW. This is a method of incremental combination of Roulette wheel and ε-greedy methods. Without any initial knowledge, by selecting documents to display, the ranking system gives the user the necessary knowledge of the environment and provides a good ranking for the next queries.

The remainder of the paper is organized as what follows. In Section 2, the objectives of this work are introduced. Section 3 gives an overview of several learning-based ranking methods. Section 4 explains the general context of Reinforcement Learning (RL). In Section 5, the proposed algorithm is introduced. In Section 6, in addition to expressing measurement criteria and evaluation data, the experimental results are evaluated. Finally, in Section 7, the conclusions and future proposals are expressed.

2. Research Objectives

The main contribution of this research work is summarized as follows:

In this work, the goal is to provide a learning-based ranking method that uses RL for learning and ranking adaptation intended by the user.

A good ranking algorithm should place the desired results of the searcher in the first ranks. This paper models the ranking methodology as an RL problem. As far as we know, the baseline methods provide a poor ranking and are known as a document-query feature (information sources), provide a weak ranking on their own. To use these information resources, RL can determine a weight as the feature importance to utilize these features simultaneously and provide a more appropriate ranking for each feature individually. Moreover, the challenge can be a lack of information about the user's interests in the ranking-based learning, which can be collected by reinforcement learning because RL, as a learning method, interacts with the environment.

The following steps are taken to achieve this goal:

- ✓ Introducing a novel action selection method of incremental combination of Roulette Wheel and ε-greedy methods called BoostRW. It will allow selecting relevant documents for the user's query. In addition, it has the probability of selecting any document

even at a low rank, which prevents the rich-get-richer problem.
- ✓ Investigating the superiority and accuracy of the ranking of the proposed method
- ✓ Comparison and evaluation of the proposed method with several learning-based ranking methods with pointwise, pairwise, and listwise approaches

The aim is to give a ranking model using the user feedback, the features related to the document-query via RL. As the user feedback has information on the interest of the users, it is the most valuable ranking information that can increase ranking precision. Each feature contains information that increases the ranking precision. Several features are selected to reduce the amount of processed information and the time complexity. This feature selection is based on the knowledge of the expert according to the results of the individual ranking of the features.

3. Review of literature

The hot topic of research in data retrieval in the recent years is the learning-based ranking [26]. The learning-based ranking methods are divided into three general categories: pointwise, pairwise, and listwise.

3.1. Pointwise approach

The easiest method is pointwise learning. In this kind of method, mapping takes place between each pair of document-query pairs and their relationship. Linear regression (LREG) is a statistically-based pointwise method. In LREG, the feature vector is mapped to a numerical value [27]. RLRAUC is a learning-based ranking method with the pointwise approach. This method is based on RL and the user feedback. The authors of the article have used noisy data to test the accuracy of the method performance [28].

Wei et al. [29] proposed a pointwise ranking method and formalized learning to rank as a Markov decision process (MDP) [30], known as MDPRank. In the learning phase of MDPRank, the ranking system is considered as a sequential decision-making. Each action is a selecting document for the position. The policy gradient algorithm of REINFORCE [31] is used to train the model parameters. The NDCG criterion is utilized as the immediate reward for an action. MDPRank directly optimizes information retrieval (IR) measures with Monte-Carlo stochastic gradient [29].

3.1. Pairwise approach

Pairwise approaches take, as input, pairs of the document for a query, i.e. they act on document pairs [6, 7]. These pairs are recorded to binary labels, which indicate whether the two documents are presented in the right order or must be switched. The following methods have this approach:

RankNet is the first learning-based ranking algorithm used in the business search engine. The approach is a pairwise method. In this method, the neural network is considered as a model and the gradient descents as an optimization algorithm for learning the loss function [12]. Keyhanipour et al. [32] introduced a ranking method based on RL called QRC-Rank, which is a two-step recovery system. In the first step, the user's click data is generated. In the second step, the click features are combined. Then the QRC-Rank method builds an RL model based on the click features. In this model, the RL agent attempts to find the best label for the states related to the pairs of document-query seen [32].

Yole Freud et al. [11] introduced the RankBoost algorithm. In RankBoost, the ranking is obtained by combining several baseline ranking methods. This learning-based ranking algorithm acts similar to the AdaBoost algorithm, and the only difference between them is their approach. RankBoost is a pairwise approach. In this method, boosting is used to combine the base ranking. Joachims has proposed a pairwise ranking algorithm of SVMRank. The idea of SVMRank is that the ranking is considered as a binary classification problem for document pairs. This categorization is done using Support Vector Machine (SVM). The probability of clicking on the ranking documents provided is directly proportional to the relevance of the document to the query [10]. In this method, only the relevant documents are clicked but in reality, this is not the case, and the user's click is a noisy action.

3.2. 3.2. Listwise approach

The most promising approach among the existing approaches is the listwise approach [33], in which special properties or features are considered for all pairs of documents or all point-documents. Here, the problems are with the unbiased selection of the feature and the dependence of most features on the query. They obtain, as input, the multi-features of all documents for a query, and learn to forecast either the scores for all documents or perfect permutations of documents [5]. A few examples of the learning to rank with this approach are as follow:

ListNet [33] is a listwise learning method [34], the goal of whose cost function is to optimize the k list of the high probability of higher-ranking results. The cost function is defined using the probability distribution on the permutations. In this method, the neural network is considered as a model and gradient degradation instead of the optimization algorithm. The ListNet function is similar to the function of RankNet, except that its cost function is a listwise, while RankNet uses the pairwise cost function. The general optimum of the method is due to the use of the gradient descent method in the loss function [33]. The problem with ListNet is its high time complexity.

Master proposed a listwise ranking method [35], referred to as GeneticListMLE++ and GeneticListNet++. This method forms on the ListMLE [36] and ListNet [33] ranking algorithms and improves on them using incorporating multi-objective genetic algorithm, a regulation technique, and a non-linear neural network ranking method [35]. Their problem is a high time complexity.

LambdaMART is based on RankNet [37]. This method is the boosted tree form of a LambdaRank algorithm. The LambdaMART method has the benefits of the Multiple Additive Regression Tree (MART) and LambdaRank methods. In LambdaMART, MART is used to determine the appropriate gradient and the Newton's step [37]. MART is a boosted tree model; whose output is a combinational linear model of a regression tree. In addition, Burges et al. [13] were the winning entry from the Yahoo! learning to rank challenge for the LambdaMART method. LambdaMART and RankNet are the algorithms to solve the real-world ranking problems [37]; hence, in this paper, the proposed method is compared with these methods.

The BoltzRank method was proposed by Volkovs and Zernel. This method has a listwise approach. The idea of this method is to define a probability distribution for document permutations and to predict the performance evaluation under this distribution. In this method, using conditional probability distribution, they rank documents for the user's query [38].

Diaz-Aviles et al. [39] proposed a ranking algorithm based on particle swarm optimization (PSO) [40]. This algorithm is called SwarmRank, which uses numerous linear combinations of functions to learn. The function learns using PSO, and the degree of relevance is represented by the linear combination of the vectors of the property of the document-query pairs. The SwarmRank's goal is to optimize the MAP evaluation criterion [39]. A learning-based ranking method was proposed by Pan et al. based on a combination of document rankings and relevancy score [41]. This effective list method is called PERF and is derived from AdaRank. This method is characterized by coding the score of the relevancy ranking. To achieve better results, their method was combined with the MAP or NDCG evaluation functions. Yei et al. [42] proposed a ranking method based on a layered multi-population genetic programming called RankMGP. This method shows a ranking function as an individual in a population of genetic programming, and aims to optimize the average of NDCG evaluation measures in the training process. The best individual is obtained as the result of the ranking function [42].

Akbari [43] proposed a ranking algorithm based on a learning automaton called LRUF, which uses the user's feedback. The LRUF method improves the ranking precision by assigning rewards and punishments proportional to the relevance of the document-query, and updates the ranking score according to the position of each document in the ranking list. In this method, the documents with a low probability are removed from the list of results, and other documents replace the deleted documents to reduce rich-get-richer. Hoffmann et al. [44] considered the information retrieval system as a reinforcement learning system. In this method, first, the list of the results is inserted, and the documents are generated based on the probability of distribution. Then all permutations of documents that have a non-zero probability are observed. After the user's clicking on the documents, the result of all the possible states is guessed. The insertion method has made it possible to compare the ranking methods unbiased and accurately [44].

The LARF algorithm is based on the learning automata, which aims to find the best combination of distinct rankings [45]. This learning automaton determines the final ranking function. It adjusts the weight of each feature based on the user's feedback. The LARF method runs in three steps: ranking, user's feedback, and learning. In the ranking step, the learning automata generate a combination of information sources. In the feedback phase, the LARF method determines the degree of relevance of the final ranking based on the user's feedback [45]. The method problem is the influence of the selection order of the document on the score of information sources. Due to the calculation of the probability for all the features and the examination of all documents in the features to form the final ranking list, the LARF algorithm has a high run time.

Raman et al. proposed a DP ranking algorithm based on an online learning based upon the balance between diversity and relevance [46]. At each step,

the results of one ranking algorithm are presented to the user. A set of user-viewed documents is considered as the user's feedback, and the priority for ranking is determined by that. After receiving the feedback, the algorithm updates the model in the online method. Theoretically, the efficiency of the DP algorithm has risen, and it is resistant to noise [46].

The DBGD ranking algorithm is online and based on RL [57, 60]. The goal of this algorithm is to establish a balance between exploitation and exploration of experience to improve the efficiency of information retrieval systems during learning. In this method, the user's interaction with the Web environment is treated in the same way as reinforcement learning problems. This method is resistant to noise and has a high cost of exploration. The online learning ranking model is considered using reinforcement learning, and the user's feedback is limited to the position k. The purpose of the learners is to present the ranked list and the relevance of the judged documents. In this method, two adjustments are considered. The first one is a non-textual setting, and the list of ranking documents is fixed. The second adjustment is textual, i.e. the list of different documents in the form of the traditional document-query list [48].

Wang et al. presented a learning-based ranking approach based on sparse learning. The loss function is defined in accordance with the process of indexing optimization with a listwise approach. This method proposes four features including document and query similarity features, language model features, content-based features, and relevancy features according to the ranking features and research work. It has greatly reduced the redundancy of features. The accuracy and precision of the ranking have been improved effectively [49].

The RL3F ranking algorithm is based on RL, and uses the user's feedback. This method has a listwise approach. In this method, the reinforcement signal is a constant value, and the feature importance degree is generally given. In each repetition of learning, only one list is obtained based on the total value of the user's feedback and the multiplication of the features in their degree of importance [50]. In contrast, in this work, for the highest accuracy with each repetition of learning for each individual feature, a list is considered according to the user's feedback and the multiplication of the feature in its importance.

Ranking method, called ES-Rank, is based on an evolutionary strategy (1+1) [51] with a proper mutation process [14]. This technique is called the evolutionary strategy (ES) of ranking. The reason for the choice of ES is its ability to quickly converge and reduce the run time. The ES-Rank method has a listwise approach, which is an evolutionary strategy (1+1) evolving the unit vector more than a generation, and the output is a linear ranking function.

3. Background knowledge: Reinforcement learning (RL)

The distinctive feature of RL, distinguishing it from other types of learning, is the use of learning information in evaluating the accepted actions relative to learning to get the right actions [31], [52]. In an RL problem, we deal with an agent that engages with the environment by trial-and-error and learns the choice of optimal action [53]. RL is a method to train agents to perform an action by giving reward and punishment without the need to specify how the act is performed by the agent [30].

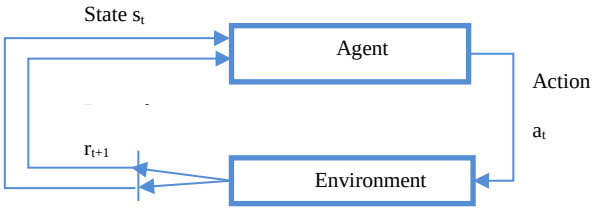


Figure 1. The framework of reinforcement learning [52].

In RL, the learning agent interacts with the dynamic environment through trial-and-error, and learns what to do in each state. The agent may receive r rewards for the action or a set of actions that it does. This reward may either be positive or negative (punishment) [31], [52].

4.1. Q-Learning

One of The RL methods is the widespread use of Q-learning, which works with Action-Value values. At each step, the agent selects the current state of an action and applies it to the environment. The environment goes to the next mode and gives Reinforcement Signal (RS) r_t to the agent. Updating the Action-Value values is performed using the following formula:

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

where s_t represents the state, a_t represents the action at time t, α_t is the learning rate, r_t is the immediate reward, and γ is the discount factor. $\max_{b \in A} Q(s_{t+1}, b)$ represents the maximum value of the next state value for A set of actions. In Q-learning, the policy used in the next practical

choice has no role in updating values of Action-Value, and the final value for the action value function is independent from the policy used to select the action [31], [52].

4. Proposed method

The proposed algorithm is based on RL and the user's feedback with a listwise approach. For this purpose, the ranking uses the document-query features. (There is a list of the features used in Appendix B.) The idea behind the proposed algorithm is to determine the importance of the document features to the query. Each feature represents a particular aspect of a document or query. The use of multiple features covers the shortcomings of each feature; in other words, a page that is relevant to the user's perspective can have the user's intended content or is targeted to the user due to a link to another linked page. For example, simple ranking methods are based on the isolated features (such as BM25 [54], PageRank [55], Vector Space [56], and Hits [57]), and create a weaker ranking than learning to rank the methods such as AdaRank [58] and RankBoost [11]. In addition, none of these features is not enough sophisticated in isolation to obtain the intricacies of most applications, for instance, the web search [35]. Thus considering the multiple features simultaneously provides the results that are relevant and reduce the ranking shortcomings of each feature.

4.1. User's decision to click

In this section, the user's feedback used in the proposed method is expressed. A model of the user's behavior can be a means of predicting the user mode on documents to improve the ranking results. The click behavior used in this paper is based on the position and cascade click patterns. In a position-based model, each position depends on a probability. The users click on the document if and only if the document position is attractive to them [59]. As we have seen, the users do not just click on the documents related to query. Therefore, it can be said that the behavior of the users' clicks is noisy. Most users also click on unrelated documents in the first rank, which indicates the high rate by clicking on documents in the top positions of the list even if the document is irrelevant [19].

Besides, the frequency of clicks is determined by the rank of the first relevant document in the list of documents shown to the user, and is called the frequency of the relevance of clicking on documents [19]. In this paper, the statistical pattern consists of eleven modes and ten probabilities of clicking on the ten first documents of the ranking list using their frequency (see Appendix A). This pattern is derived from the frequency of user-clicked distribution [19]. (Their analysis involved over 3,000 queries from more than 12 million user interactions with a search engine in [19].) Each mode is defined as the position of the first relevant document related to the query in the viewed documents via the user. Since the first relevant document is related to the query in which, of the top ten positions is or not, there are eleven modes. In addition, each mode includes probabilities of clicking on any of the ten top documents in the ranking list. Then there are ten probabilities. The user clicks on documents continue until the relevant document is seen or the first ten documents are clicked or dropped according to the probability.

4.2. RRLUFF algorithm

In this section, we describe the procedure of RRLUFF ranking algorithm for responding to the user's queries. In this method, first, the user enters the query. Then the ranking system shows multiple lists of documents to the user as the agent. Each listing is organized according to one of the features of the document-query pair (such as BM25, PageRank) (see Appendix B). The user has the detection power of the document list suitability. Each feature is assigned a score according to the list related to it. Then the user selects a more relevant list from the lists and clicks on the documents in that list. A reward or a punishment is awarded to the feedback feature of the clicked document. Moreover, the documents in the list are selected by the priority random method according to the modified scores. This procedure is repeated to converge the score of each feature to a fixed number, and the training phase ends. It is assumed that the score of each feature indicates its importance. The feature importance is considered as the weight of a feature for the ranking. By applying the total to the multiplication normalized feature value in its weight, the score of each document-query pair of the test stage is obtained.

Overall, The RRLUFF algorithm steps are summarized as follow:

1- Initializing

The initial value of the feedback feature is zero since there is no knowledge at first. The values of feature importance are the same. For each p-feature, the query-related documents are sorted by p-feature and the list (p) is formed. The criterion is selected for the formation of its normalized precision.

2- Beginning of process

Step 1: The value of learning rate is calculated according to (4) (learning rate decreases exponentially).

Step 2: For each feature, the list of documents is sorted according to the value of the score calculated by (8).

Step 3: For each feature, ten documents are selected from the list of ordered documents according to the probability calculated by Equations (2) and (3), and placed in the jth list related to feature j.

Step 4: All lists are displayed to the user. Then the user calculates the importance of each feature list using (5), and according to that, each feature receives a reward or punishment in accordance with (6).

Step 5: Among the lists, the list with the most ranking precision is displayed for the user to click, and the user will click on document D based on his/her click table. The table containing the user-click distribution frequency is displayed on the documents above the list.

Step 6: If document D is relevant, the reward will be redeemed according to (6) of the document-query pair, and if document D is irrelevant, the punishment will be given to the feedback feature.

Step 7: If the selected click mode is not completed, the user clicks based on the user-click distribution frequency table and repeats step 5.

Step 8: Steps 1 to 7 are iterated until the user's query expires or the number of laps is passed, causing the learning rate to go down to zero, and it is completed.

3- Test

Step 9: The ranking of the document is calculated according to Formula 9.

Step 10: Step 9 is iterated for all documents related to the query.

Step 11: The final list is sorted in descending order according to this.

Figure 2 shows a general overview of the proposed method described above. This illustrates the interactions between the ranking system and the environment (i.e. the user, document-query pairs, etc.) in the session search.

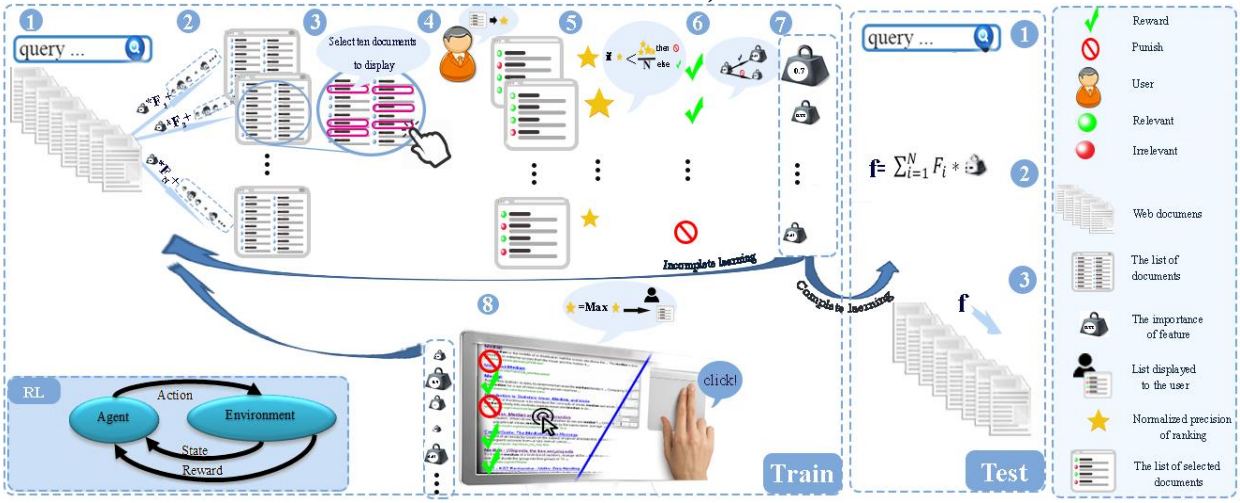


Figure 2. General overview of the RRLUFF method.

In what follows, the equation details are used in the proposed method, and a perfect description of the RRLUFF algorithm is expressed.

4.2.1. Problem formulation

In this sub-section, the process of functioning of the RRLUFF ranking system is formalized as an RL problem for providing a ranking of documents related to user queries. The construction of a document ranking can be considered as an RL problem. The method, referred to as RRLUFF, can be indicated by the environment, agent, action, state, reward, and Action-Value, which are, respectively, defined as follow:

Environment: The environment includes the user, documents, and queries.

Agent: The ranking system is considered as the agent. The agent chooses action, and the environment reacts to this action, giving a new state to the agent. The iteration of this interaction has made the documents arranged in a careful ranked list, which is the main purpose of each ranking system.

Action: The action is to select ten documents and display it to the user, and the ranking system, as an agent, performs the action.

State: The ranking system, as the agent, must know the position of the ranked document with a feature that the agent can choose and displays it to the user. Thus at the time step, the states are defined as a collection document-query pairs to determine the feature importance. In addition, to determine the

user's feedback importance, any document-query pair is considered as a state.

Reward: It is the immediate reward known as the reinforcement signal. As the environment, the user provides an RS for the feature ranking according to the relevance of the ten document list to the query when the user clicks on any document related to the query. If the document is relevant to the query, the reward is obtained; otherwise, the punishment is awarded. This RS is adaptive for the features, and its value is determined based on the position of the relevant documents to query but for the feedback feature, RS is a constant.

Action-Value: The importance of the feature is an Action-Value pair.

Action selection: In this part, the action-selection method used is introduced, in which, this action-selection method is presented for the ranking algorithm based on RL, known as BoostRW. Its function is based on the position of the document in the ranking list. The documents are arranged according to the value for Action-Value; therefore, the higher the document is in the list, the greater the value for Action-Value will be. This selection has a distribution similar to Roulette Wheel (RW). For the possibility to select all the documents in each rank, that rich-get-richer problem is not created. As we know, the user's desire is to get higher ranks; the probability of the document in all rankings is as RW. The method of selection of the action also uses an incremental method similar to ε-greedy [31], [52] so that overtime (increase of (t)), the probability of selecting the first ten documents increases with a decrease in the probability of ε. At first, due to the lack of knowledge, ε value was equal to one. Given that overtime, the user's knowledge of the environment increases, and the related documents are directed at the top of the list and will have a higher priority; therefore, the probability of selecting the top ten will be higher and the value of ε will be reduced to at least 0.9. On the other hand, overtime, by gaining knowledge of the environment, the related documents are ranked at high ranks. For this reason, the probability of selecting these documents increases with boosted RW and decreases with the value of ε. A boosted RW reduces the chances of selecting documents in low rankings overtime. The BoostRW selection type brings about faster convergence of the list to the optimum ranking. By this method, ten documents are selected for display to the user, and the reward or punishment is awarded to the feature based on whose Action-Value value the documents are arranged. The probability of selecting the list of documents is calculated according to the following equations:

$$p(X_C)=\begin{cases}1-\varepsilon+\dfrac{\varepsilon}{n} & X_C=\{1,2,...,10\}\\ \dfrac{\varepsilon}{n} & X_C=\{11,12,...,n\}\end{cases} \quad (2)$$

$$\varepsilon=1-\frac{0.1\times(t-1)}{T} \quad t\in\{1,2,...,T\}$$

$$m(x)=\begin{cases}\dfrac{2\times(\rho\times(n+1-x)-\frac{t^2}{T})}{n\times(\rho\times(n+1-x)-\frac{t^2}{T})} & x\in\{1,2,...,[n-\frac{t^2}{T\times\beta}]\}\end{cases} \quad (3)$$

where $p(X_C)$ represents the probability of incremental $\boldsymbol{\varepsilon}$-greedy for selecting the collection of the first ten documents of the ranking list and the set of documents ranked after the 11th. Overtime, the probability of selecting the first ten documents will increase. X_C, n, and T represent the selected document set, the total number of extracted documents per user query, and the maximum time, respectively, and t represents time. The value of t increases after viewing all queries by the user. $m(x)$ represents the selection probability in the BoostRW method. This selects a document from the X_C document set with ε-greedy probability. Overtime (increase of (t)), the chances of selecting low ranks go down to zero. X is used to indicate the position of the document in the ordered list of documents, and ρ is the constant adjustable parameter, here, and it is equal to ten. The numbers in the range {1, …, 100} have produced fit results. Overall, reinforcement learning is employed to set the model parameters (the importance of the feature). The first step of the ranking starts with selecting ten documents from among the ranked documents of feature, and displaying them to the user is intended as an action. At first, this selection is in a non-greedy mode that is exploratory, and with an increase in time, gets greedy, which means to use the previous knowledge. Then the user rewards or punishes according to the precision of the list ranked to the feature importance. In other words, the system ranking learns directly from feedback inferred from the user interactions such as user's click. Also the user's click is used for the given RS (reward or punish) in this process. Each reinforcement signal decreases or increases the importance of the features and user`s feedback feature that is recognized as the immediate reward. In the case of the feedback feature, if the clicked document is relevant, then the user feedback receives a reward for the document-query pair; otherwise, it is punished. At each stage, the importance of document features and user feedback feature (Action-Values) of the final document seen by the user are not punished or rewarded. The

iteration of the steps ends when either a certain time is past or the user's query has expired. The procedure of the RRLUFF method is as follows:

4.2.2. Procedure of RRLUFF algorithm

The RRLUFF algorithm consists of two general steps, each of which will be described in detail. In the training phase, the ranking is considered as the RL problem. At this stage, the importance of each feature is determined using rewards and punishments. At the test stage, the documents are ranked by the value of the linear relationship of the sum of the feature importance multiplication in the feature value of the document.

• Training Stage

In this process, n is considered as the number of document features. The goal is to determine the significance of each feature, and is shown by ' I '. First, the importance of the features is assumed to be the same ($I = \frac{1}{n}$). As there is no knowledge of the importance of the feature, a ranking list is provided for each feature. Ten documents are selected using the action selection method and are displayed to the user. Due to the value of ranking precision, the corresponding list of features is given a score. The precision of ranking is measured based on one of the two criteria to evaluate NDCG and P@n using (5).

$$\alpha = e^{-\beta \times t} \tag{4}$$

$$m_i = \begin{cases} \dfrac{\sum_{i=1}^{10} (\frac{NR_i}{i}) \times (10-i)^3}{\sum_{i=1}^{10} (\frac{R_i}{i}) \times (10-i)^3} & if\ criterion = P@n \\ \dfrac{\sum_{i=1}^{10} (NDCG@i) \times (10-i)^3}{\sum_{i=1}^{10} (\frac{(R_i-1)^2}{\log(i+1)}) \times (10-i)^3} & if\ criterion = NDCG \end{cases} \tag{5}$$

A feature with a ranking precision higher than the average ranking precision will receive a reward corresponding to its precision, and other features are punished in proportion to their low precision. The value of Action-Value is determined by the following equations. When the number of associated documents in the top ten positions of the list increases, the rewards will increase.

$$I_j = \begin{cases} I_j + \alpha \times [2 \times m_j + \\ \gamma \times \max_{b \in A} I_j - I_j] & m_j \geq \dfrac{\sum_{i=1}^{n} m_i}{n} \\ I_j + \alpha \times [(m_j - 1) + \\ \gamma \times \max_{b \in A} I_j - I_j] & otherwise \end{cases} \tag{6}$$

$$j \in \{1, 2, ..., n\}$$

where α and β represent the learning rate and the step, respectively. t represents time; $t = 1$ is the learning rate, and will be equal to one, and over time, the learning rate will go towards zero and learning will be completed. β is considered to be a constant value in the range of [0.01, 0.1]. NR_i shows the number of documents associated with query in the i top positions of the ranking list. $\frac{NR_i}{i}$ represents the precision of the document in the i[th] position of the document list. m_i shows the normalized precision of the documents in the first ten positions of the ranking list, which is selected based on one of the two criteria evaluation: NDCG and P@n. Its value is between zero and one. R_i denotes the level of relevance of the document in position i to the query (for example, irrelevant = 0, partially relevant = 1, and definitely relevant = 2) for the best ranked list. The denominators of the equation $\sum_{i=1}^{10} (\frac{R_i}{i}) \times (10-i)^3$ and $\sum_{i=1}^{10} (\frac{(R_i-1)^2}{\log(i+1)}) \times (10-i)^3$ are both equal to the highest value of the numerator of the equation, which means that when the list is best ranked, I_j represents the degree of importance of the j[th] feature. $(2 \times m_j)$ and $(m_j - 1)$, respectively, represent the rewards and punishments received, and are adaptive. This means that their values are set according to the normalized value of precision of the ranking. The value of I_i is calculated for all selected features. n represents the number of features considered, which is seventeen in this article. Selection of the seventeen features was done based on the expert knowledge and according to the individual results of the features ranking. γ represents the discount factor, and $\max_{b \in A} I_j$ represents the highest importance of feature j[th]. γ is a value in the range [0,1] that have produced good results. Then the most relevant j-m list ($m_j = \max(m_i) i \in \{1, 2, ..., n\}$) is displayed to the user, who clicks on the list documents, and RS (reward or a punish) is granted to the feedback feature of document-query pair. The score for each user feedback feature fe is determined by the following equation:

$$fe_{d,q}(t) = fe_{d,q}(t) + \alpha \times [\pm reward + \gamma \times \max_{b \in A} fe_{d,q}(t+1) - fe_{d,q}(t)] \tag{7}$$

where $fe_{d,q}(t)$ represents Action-Value for the user's feedback feature at time t. The reward

represents the gained RS. If the observed document is related to the query, the reward (positive value), and if it is irrelevant, the punishment (negative value) is considered. α is the learning rate calculated using Eq. (4). The initial value of Action-Value for each feature of user's feedback is zero. The documents for the features of each corresponding list are sorted according to the following equation:

$$Score_{d,q} = I_j \times F_{j_{d,q}} + fe_{d,q} \quad (8)$$

Where $Score_{d,q}$ represents the score of the document-query pair (d,q) for ranking the list of the j[th] feature to display to the user. $F_{j_{d,q}}$ indicates the value of the j[th] feature of the document-query pair and I_j represents the importance of the j[th] feature. $fe_{d,q}$ is the value of the feedback feature in the case of seeing a document related to query.

In the subsequent iterations of learning, among the documents sorted by $Score_{d,q}$, each feature should be selected for the ten documents to form each list. These choices are prioritized randomly (BoostRW). Among these lists, according to the user's view, a list is selected for the display (the list with the highest value of mj is selected). This process is repeated until the learning is completed.

• Test stage

After the end of the training, the importance of each feature (I) is determined by RS gained, and by applying this importance as the weight. The score of each document-query pair is calculated according to the following linear relation:

$$S_{d,q} = \sum_{j=1}^{m} I_j \times F_{j_{d,q}} \quad (9)$$

where $S_{d,q}$ and $F_{j_{d,q}}$ represent the ranking score and the value of the j[th] feature of the document-query pair (d, q), respectively. The importance of the j[th] feature is marked by I_j.

The pseudo-code and flowchart of the RRLUFF algorithm are shown in Algorithm 1 and figure 3, respectively.

Algorithm 1 RRLUFF

Input
d: document
q: query
value_click is frequency distribution of relevance of users' clicks on web search results that this is a matrix 10*11
F(j,d,q): value feature j[th] for pair document-query (d,q)
R_i: The user judgment is binary in P@n. NDCG have three levels.
Output
I: the importance of features in phase train and stored in an array
s(d,q): Ranking list for q_i in phase test //finally list Ranking
Parameters and Local variables
n: the number of queries in the dataset
t: number repeat or time
d: document
q: query
m: the number of documents in the dataset
I: the importance of features in phase train and stored in an array with size g
g: the number of features in pair document-query
fe: the value of feedback user for document-query pairs
PTR: the position of the first relevant document in the list
value_click is frequency distribution of relevance of users' clicks on web search results that this is a matrix 10*11
mi: Normalized precision Score is an array with size g (the importance of documents list)
criterion: The parameter value is P@n or NDCG@n
s(d,q): Ranking list for q_i in phase test
punish is a negative reinforcement signal.
reward is a positive reinforcement signal.
criterion is P@n or NDCG@n
R_i is a function of levels of the user's judgment
Assumption
if $\exists(d_j, q_i)$ **then** $\exists d_{i,j}$ **end**
Initialize
$t = 1$, $punish = -1 \times reward$
For i=1 to g
$$I_i = \frac{1}{g}$$
end//for line 27
$fe(i, j) = 0$
Begin
While $t < N$
$\alpha = e^{-\beta \times t}$ //learning rate α∈(0,1)
For i=1 **to** n //$q_{i,t}$
max=-1

For j=1 **to** g //feature
Sorting list[jth] documents $d_{i,j}$ by $Score_{d,q} = fe_{k,j} + F(d,q) \times I_j$
Selecting ten documents from list[jth] by Eqs. 2-3
$mi_j = 0$, M=0, Avg=0, max= -Inf
if criterion == P@n **then**
For k=1 **to** 10

$$mi_j = mi_j + (\frac{NR_k}{k}) \times (10-k)^3$$

$$M = M + (\frac{R_k}{k}) \times (10-k)^3$$

end//for line 40
else if criterion == NDCG@n **then**
For k=1 **to** 10

$$mi_j = mi_j + (NDCG@k) \times (10-k)^3$$

$$M = M + (\frac{(R_k - 1)^2}{\log(k+1)}) \times (10-k)^3$$

end//for line 45
end//if line 39

$$mi_j = \frac{mi_j}{M}$$

if $mi_j > \max$ **then**
list [jth] is choice for shown to User & set in R_i

$$\max = mi_j$$

end //if line 51

$$Avg = mi_j + Avg$$

end //for line 35

$$Avg = \frac{Avg}{g}$$

For j=1 **to** g //feature
if mi_j>= Avg **then**

$$I_j = I_j + \alpha \times [2 \times mi_j + \gamma \times \max_{b \in A} I_j - I_j] \quad \text{//reward}$$

else

$$I_j = I_j + \alpha \times [(mi_j - 1) + \gamma \times \max_{b \in A} I_j - I_j] \quad \text{//punish}$$

end //if line 59
end //for line 58
Ranking R_i is shown to the User
h=0
while h< m & don't find h
if h is position of top relevance $d_{i,h}$ **then**
PTR=h
end //if line 68
end //while line 67
For i=1 **to** 10
User clicks on documents in R_i by probability value_click$_{PTR,i}$
if document is relevant **then** //reward

$$fe_{d,q}(t) = fe_{d,q}(t) + \alpha \times [reward + \gamma \times \max_{b \in A} fe_{d,q}(t+1) - fe_{d,q}(t)]$$

Break repeat for this query
else if document is irrelevant **then** //punish

$$fe_{d,q}(t) = fe_{d,q}(t) + \alpha \times [punish + \gamma \times \max_{b \in A} fe_{d,q}(t+1) - fe_{d,q}(t)]$$

end //if line 74
end//for line 72
t=t+1
until query session is expired
end // while line 31
For i=1 **to** m //query
For j=1 **to** n //document
For k=1 **to** g //feature

$$S_{i,j} = I_k \times F(i,j,k) + S_{i,j}$$

end //for line 86
end //for line 85
end //for line 84

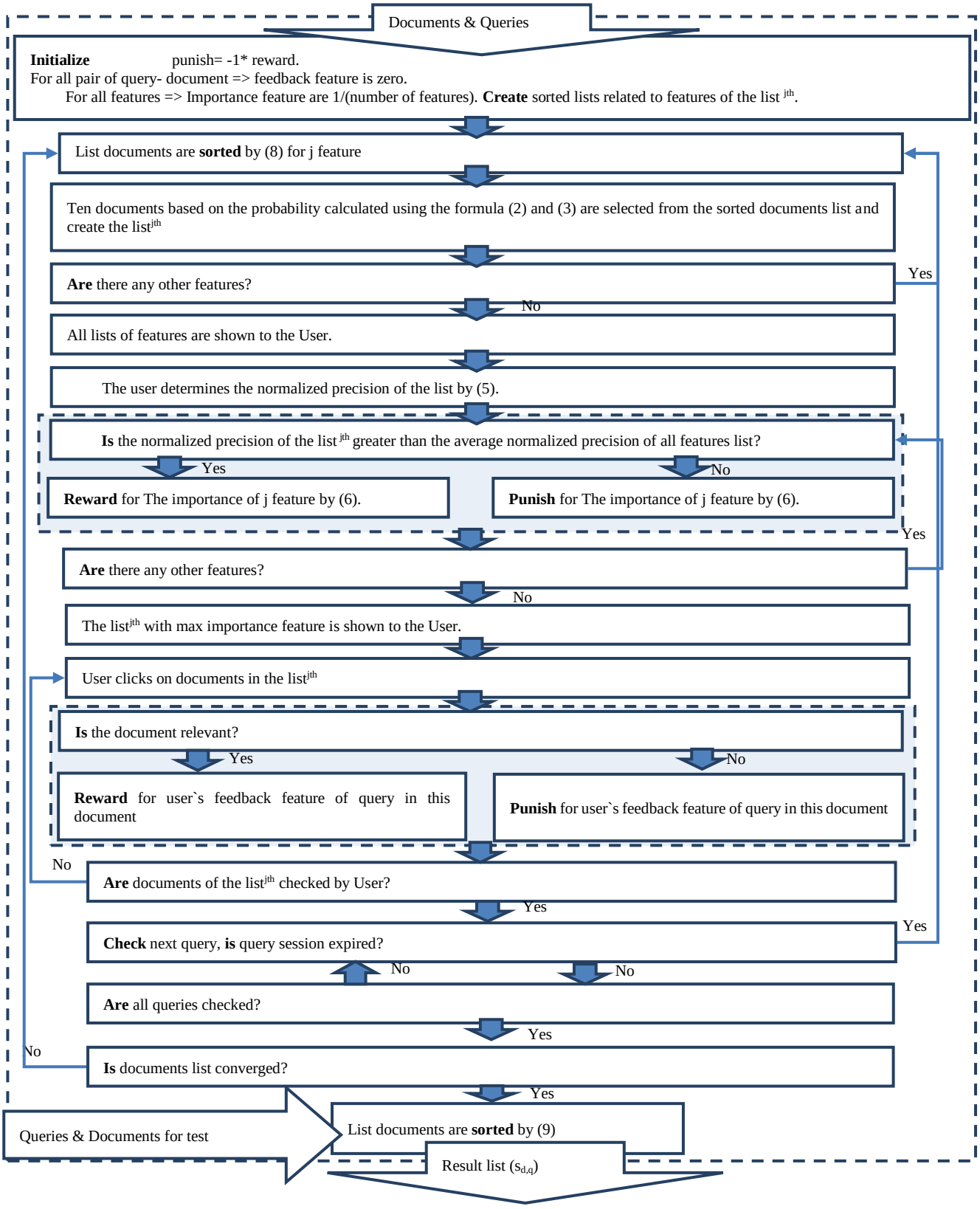


Figure 3. A flowchart of the proposed algorithm.

5. Evaluation and analysis of experimental results

The same conditions and context are the necessary conditions for evaluating and comparing the methods. To this end, there is a need for benchmarking data to evaluate and test different algorithms. The ranking dataset consists of three components of document sets, queries, and human judgments relating to the document-query pair [60].

5.1. Benchmark dataset

The datasets used in this article are OHSUMED dataset of the LETOR3 version and the DOTIR Persian dataset.

OHSUMED: LETOR dataset has been developed in the recent years due to the expansion of the learning-based ranking field of research. OHSUMED Dataset is a part of LETOR3 and a subset of MEDLINE. This set has 106 queries, 45 features, and 16140 document-query pairs judged by the user. In OHSUMED, the user judgment has three levels: definitely relevant, partially relevant, and irrelevant [26]. These features are related to the document-query pair, and some are only dependent on the query [26], [59]. Each dataset in LETOR consists of five-fold data, and each fold consists of three subsets of the train, validation, and test [26].

DOTIR: Department of Database Research of the University of Tehran has collected Farsi benchmark datasets. This collection is obtained from 8.5-million page crawls of the .ir domain, which includes about a million pages. This set has 50 queries [60]. This dataset consists of three sections of training, test, and validation. In addition, in a parallel effort, based on LETOR standard, Farsi web dataset that includes 56 features, 50 queries, and 50000 document-query pairs is processed.

In this work, a pre-processing stage was performed on the dataset. In this stage, if the feature values are the absence of normalization, then they are normalized, and the queries whose lists of retrieved documents do not have a relevant document are not involved in the training phase.

5.2. Evaluation criteria

The precision and quality of the ranking are measured by the criteria such as P@n, MAP, and NDCG. NDCG assumes different levels of relevance, and P@n and MAP consider the binary level in the evaluation. In measuring these criteria, the documents in a higher position than the list displayed for the user are given a higher weight. These measurements are discrete. In addition, information retrieval widely uses precision metrics for n (p@n) precision at n, mean-average accuracy (MAP), and normalized discount cumulative gain (NDCG) for precision evaluation. The LETOR Team Assessment Tools supports these criteria [61]. Those evaluation tools can be used as easy and unbiased tools for comparing information retrieval methods.

- **P@n**

This criterion indicates the number of relevant documents in the first n indices of the index of the retrieved ranking list for each query. Equation P@n is defined as follows:

$$P@n = \frac{NoR_n}{n} \tag{10}$$

where NoR_n represents the number of documents associated with the n positions above the ranking list.

- **MAP**

Mean average precision (MAP) is defined as the mean AP values for all queries [26], and is related to each AP query as the average of P@n values for all documents.

$$AP = \frac{\sum_{n=1}^{N}(P@n \times \mathrm{Re}(n))}{TR} \tag{11}$$

In this equation, N represents the number of recovered documents, and T_R is the number of documents. Re(n) represents the relevance level binary function of the nth document, where the function value for the relevant document is equal to one and the irrelevant document is zero.

- **NDCG**

NDCG value of a ranking list in the nth position for a query is expressed as follows:

$$NDCG@n = Z_n \times \sum_{m=1}^{n} \frac{2^{r(m)}-1}{\log(m+1)} \tag{12}$$

where r(m) represents the multi-level function of the document communication in the m-position of the ranking list; for example, $r(m) \in \{1,2,...,5\}$ Z_n is the constant of normalization. $2^{r(m)}-1$ denotes the gain of the document in the mth position. $\frac{2^{r(m)}-1}{\log(m+1)}$ represents the discount gain, and $\sum_{m=1}^{n} \frac{2^{r(m)}-1}{\log(m+1)}$ is the discount cumulative gain in the nth position.

- **NWN**

The normalized winning number is a type of the number of winners [62]. The NWN equation is as follows:

$$NWN_i(M) = \frac{WN_i(M)}{IWN_i(M)} = \frac{\sum_{j=1}^{n}\sum_{k=1}^{m} I\{M_i(j) > M_k(j)\}}{\sum_{j=1}^{n}\sum_{k=1}^{m} D\{M_i(j), M_k(j)\}} \tag{13}$$

where $IWN_i(M)$ represents the ideal winning number of the ith algorithm based on the M evaluation criterion (such as P@n, MAP, and NDCG), and the largest number is the winner. $WN_i(M)$ represents the winning number of the ith

algorithm. The winning number is equal to the number of algorithms that are worse than this set of data on this algorithm. $I\{M_i(j) > M_k(j)\}$ is a binary function. If both $M_i(j)$ and $M_k(j)$ are defined and $M_i(j) > M_k(j)$, it is equal to one; otherwise, it is equal to zero. n and m are the number of datasets and the number of algorithms, respectively. $D\{M_i(j), M_k(j)\}$ denotes the binary function, and if both $M_i(j)$ and $M_k(j)$ are defined, it is equal to one; otherwise, it is equal to zero.

5.3. Discussion and evaluation of RRLUFF algorithm

5.3.1. Experimental setup

We address a ranking based on learning in an offline setting, where labeled training data is provided and does not needs to be collected through interaction with the user. For these tests, the RRLUFF algorithm (which is implemented in C#) is calibrated as follows: its performance strongly pertains to the learning rate. In these tests, the learning rate reduces from one to zero. The amount of adaptive RS is another parameter that must be tuned in the learning steps. It is calculated according to (6). For another parameter, the number of learning iteration is set to 100 in these experiments. The number of features used is 17 features for all the documents related to the user queries (see Appendix B). (Those are selected by the knowledge of an expert and according to their ranking results.)

Meanwhile, to make the assessment and analyze, the relevance of the ranking results is recognized by the evaluation tool package readied by Microsoft research (LETOR) [26]. These tools have been created in Perl, and measure the P@n, NDCG, and MAP of the results of given ranking methods. In these tests, the "Eval-Score.pl" tool is used to appraise the ranking results. The inputs of this tool include three parameters: the first parameter is the information of the data test on that the experiment is executed, and the latter one is the scores given by the ranking algorithm. Finally, the output file includes the evaluation outcomes of P@n, NDCG, and MAP.

5.3.2. Investigating accuracy of RRLUFF algorithm

This set of simulation experiments is conducted to measure the accuracy of the ranking algorithms in terms of P@n, NDCG@n, and MAP, where n domain from 1 to 10. Those metrics advise the number of relevant documents for any query in the top n rank of the results obtained. The ranking results of the proposed method are summarized in figures 4-9 in comparison with the SVMRank, RankBoost, Regression, RankNet, MART, LambdaMART, RL3F, MDPRank, and ES-Rank.

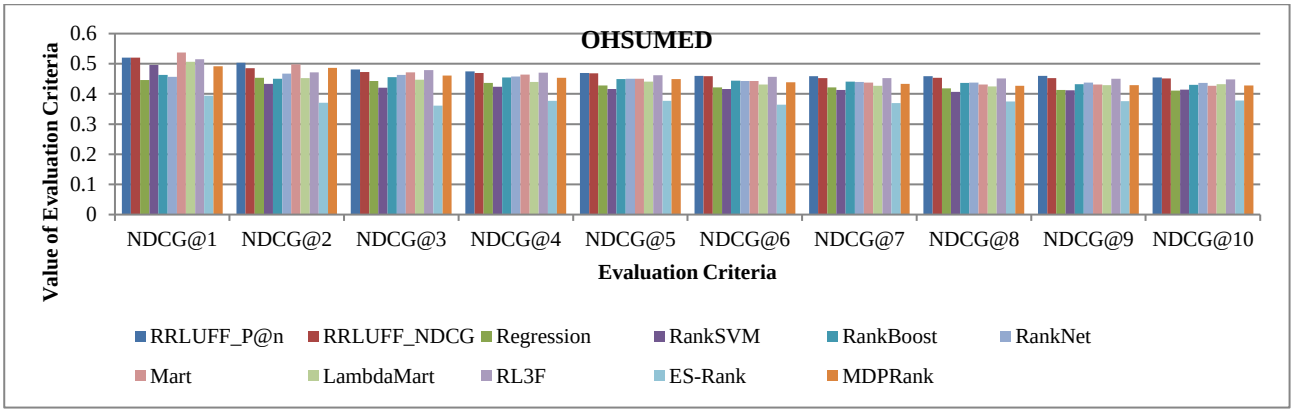
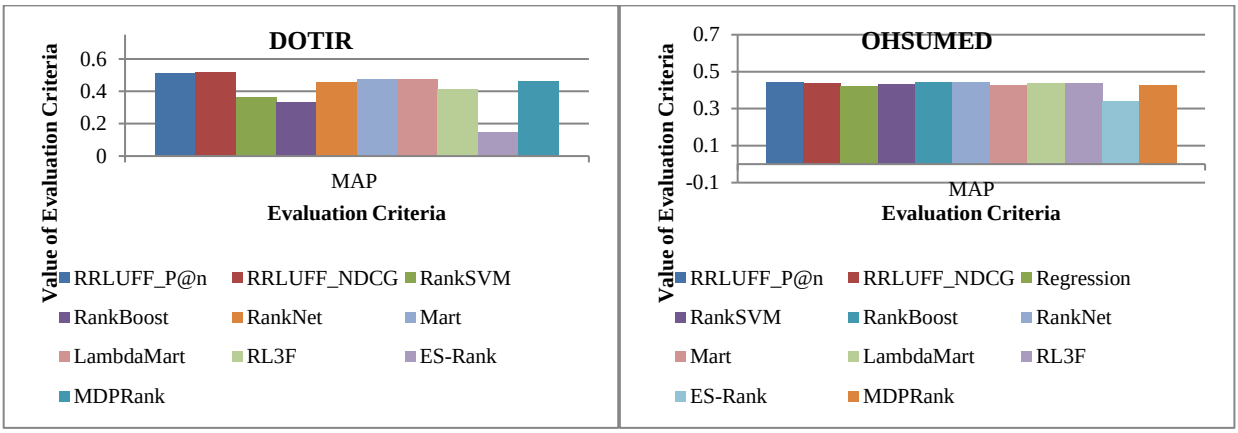


Figure 4. Comparison between the proposed algorithm and algorithms expressed regarding the evaluation criterion NDCG@n on OHSUMED Dataset.



(a) On DOTIR Dataset **(b) On OHSUMED Dataset**

Figure 5. Comparison between the proposed algorithm and algorithms expressed regarding the evaluation criterion MAP.

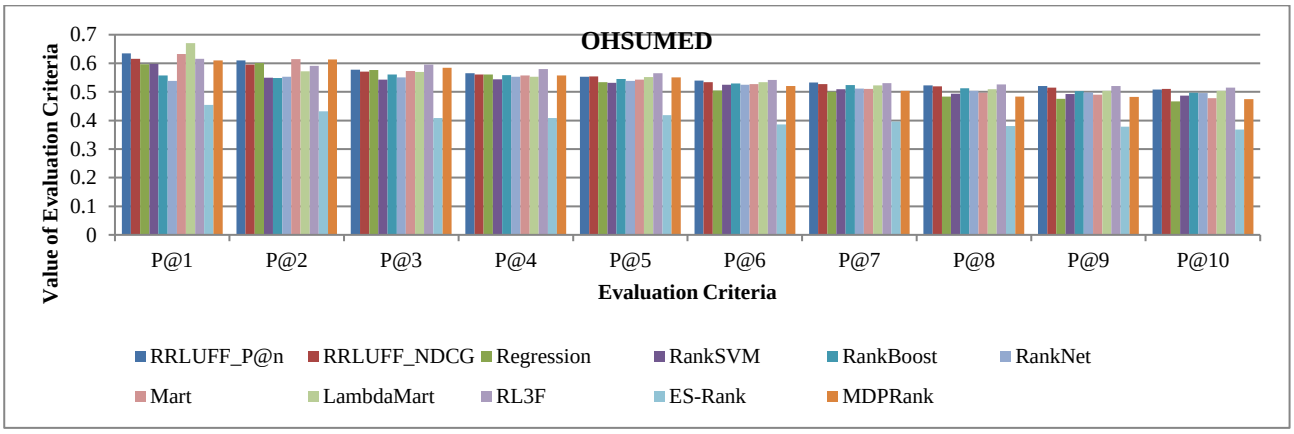


Figure 6. Comparison between the proposed algorithm and algorithms expressed regarding the evaluation criterion P@n on OHSUMED Dataset.

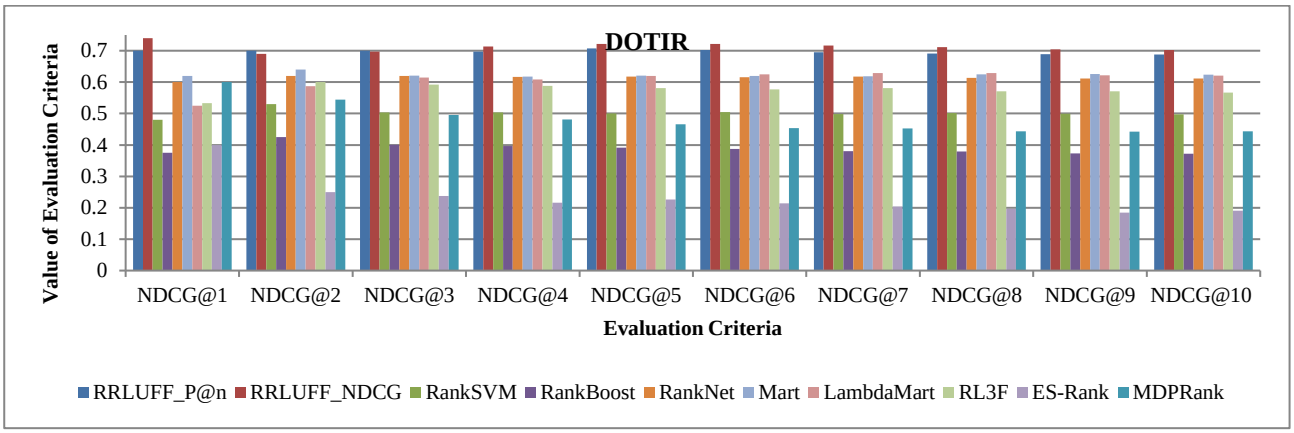


Figure 7. Comparison between the proposed algorithm and algorithms expressed regarding the evaluation criterion NDCG@n on DOTIR Dataset.

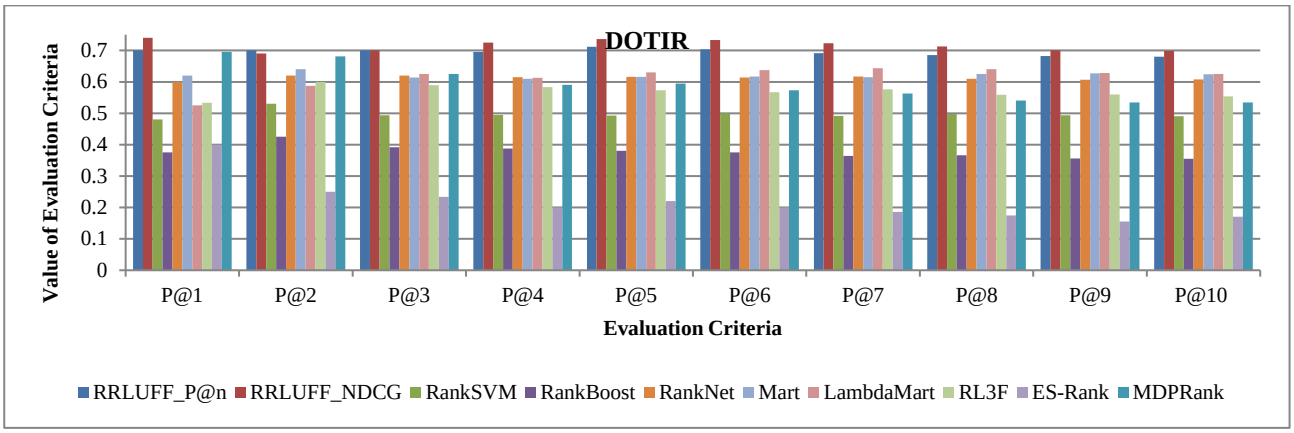


Figure 8. Comparison between the proposed algorithm and algorithms expressed regarding the evaluation criterion P@n on DOTIR Dataset.

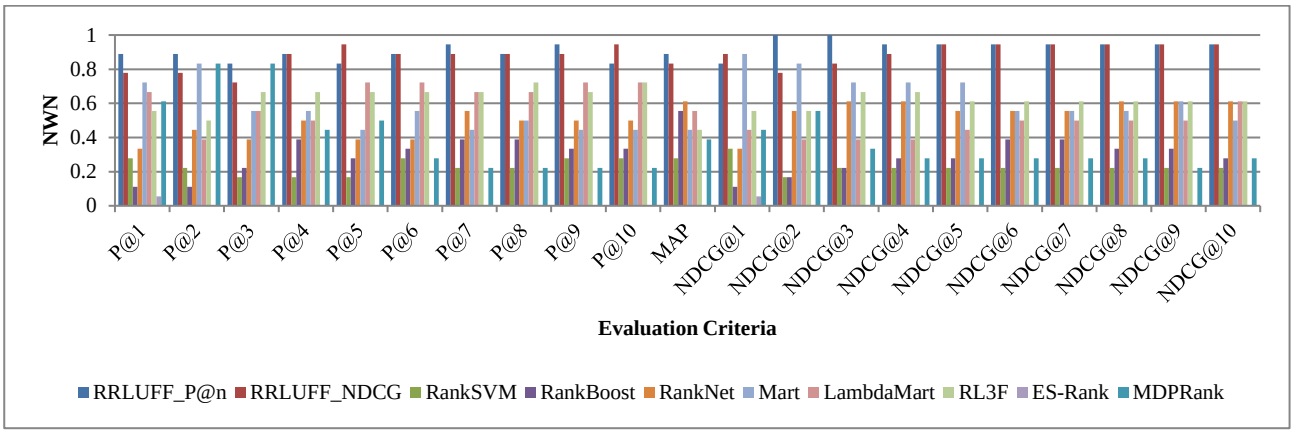


Figure 9. Comparing between the proposed method and Relevant Expressed Methods with Respect to Evaluation Criteria of NWN on OHSUMED and DOTIR Datasets.

Figures 4 to 6 show the superiority of the proposed algorithm with two versions of RRLUFF_P@n and RRLUFF_NDCG in terms of the NDCG@n evaluation criterion relative to RankBoost, SVMRank, Regression, RankNet, MART, LambdaMART, RL3F, MDPRank, and ES-Rank, and in terms of the MAP evaluation criterion. The proposed method performs better than all the compared algorithms. Regarding the P@1 LambdaMART evaluation benchmark, it has performed better and offered better results after LambdaMART. The MART method performs better in the P@2 evaluation criterion, and the proposed method has the best performance afterward. In other situations, P@n of the proposed method was better than the other methods. Among the methods, ES-Rank has shown the worst performance for all the evaluation criteria. Two versions of the proposed RRLUFF_P@n version of the OHSUMED dataset provide a better ranking.

Figures 7 and 8 show the better performance of both versions of the proposed method (RRLUFF_P@n and RRLUFF_NDCG) in the three evaluation criteria of P@n and NDCG@n in all situations compared to all methods on the DOTIR dataset. Figure 5b shows the superiority of the method in terms of the MAP evaluation criterion. From the two versions, RRLUFF_NDCG acts more appropriately than the RRLUFF_P@n version on the DOTIR dataset, and performs better on the OHSUMED dataset of the RRLUFF_P@n version. The reason for this small difference in the superiority of the two versions relative to each other is the difference in the type of data structure. The DOTIR dataset has a denser graph, and its user judgment is two-level, whereas the OHSUMED dataset is triple-level.

Figure 9 shows the comparison of the proposed method for NWN assessment criterion; where both proposed versions provide a better performance in all situations in all three-evaluation criteria P@n, MAP and NDCG on DOTIR and OHSUMED datasets. An RRLUFF_P@n method in the NDCG@2 and NDCG@3 evaluation criteria is one, which means that it has the best performance on both evaluated datasets compared to all methods. In the NWN, ten ranking models have been used. Moreover, the proposed method has provided better results than the ones using boosting, classifying, and regressing methods, and represents reinforcement learning power over those

learning methods. As well as showing that, the user's feedback is an important feature.

According to the above results, the proposed algorithm acts better than the RL3F technique that uses RL. In the proposed algorithm, as each feature has the ability to rank documents, the features have been treated independently in order to obtain its importance. In the RRLUFF algorithm, RS is adaptive, and is based on the normalized accuracy of the ten selected documents to be displayed to the user. However, in the RL3F method, RS has assumed a constant value, arranged according to the position of the clicked document based on the same feature and degree of relevance of the clicked document. Similarly, MDPRank is another method that uses the kind of RL. This method has good results but RRLUFF is better than MDPRank. The reason for the superiority of the proposed method has the listwise approach, which acts better than the pointwise approach. As well as the user's feedback, the proposed method is used and the MDPRank method does not use it. Another reason, the off-policy Q-learning technique, has navigated better than Monte-Carlo's stochastic gradient in this environment on both datasets.

Compared to the SVMRank algorithm, one of the reasons for these results is a better performance of RL compared to SVM in the online mode. Another reason is that the listwise approach works better than the pairwise approach, and although in both methods the user feedback and document features are used, the results are better because of the list approach. One of the problems with the SVMRank ranking is the emphasis on low and middle rankings. In contrast, the focus of the proposed algorithm is to find the related documents and to put them in a high rank, which solves this problem of the SVMRank algorithm. The final ranking model in the SVMRank algorithm is heavily influenced by the query with the number of related documents. However, in the proposed algorithm, RRLUFF, the value is considered for the feature and values are quantified regardless of belonging to query. Therefore, the proposed algorithm will not solve this problem of the SVMRank algorithm. As stated in Section 3, regression algorithm has a pointwise approach. This is the reason for its poor performance compared to the proposed algorithm.

Regarding the RankBoost algorithm, the results showing the superiority of the proposed method in the P@n and NDCG@n evaluation criteria, which is due to a better performance of the RL method compared to boosting. Some of the RankNet's problems are that its minimum cross-entropy loss function is non-zero, and the non-convex target is optimized with difficulty [63], and it suffers from the local optimum problem. The proposed method in this paper, due to the use of RL, does not have these problems, and shows better experimental results. LambdaMART is modeled with a boosted tree, where the empirical results show a better performance of Q-learning compared to the boosting tree. The ES-Rank method has a poor performance. The proposed method and ES-Rank both calculate outputs as a linear sum of their features and weights. The difference in results suggests a better performance of RL compared to the evolutionary strategy (1 + 1) in calculating the weight of the features. In general, the ranking of the proposed algorithm shows a better performance as compared to the above methods (such as RankBoost, SVMRank, Regression, RankNet, MART, LambdaMART, RL3F, MDPRank, and ES-Rank) in terms of the empirical results.

5.3.3. Investigating statistical significance of RRLUFF algorithm

In this section, the statistical significance test [64] is established to determine whether the difference between the RRLUFF method and the above ranking methods is significant. It is known as the paired t-test. Meanwhile, the paired t-test ($p < 0.04$) experiment is performed for these eight methods in order to determine significance by testing each evaluation criterion (MAP, NDCG@10, and P@10) on both datasets. The significance level is 5% and also the number of folds is five. As we can see in tables 1 and 2, t-test (p-value < 0.04) is with the results and detect that the performance ameliorations are significant as compared with all the above methods. As regards table 1, the test is performed on RRLUFF_P@n, RRLUFF_NDCG, and other algorithms and outcomes show that p-value in all measures is less than 0.039 and is greater than 0.0001. Therefore, RRLUFF performs significantly better than the ones do on the OHSUMED dataset. However, the results in OHSUMED are dramatically better than dotIR. Table 2 is related to the test between RRLUFF and other compared methods on DOTIR. The biggest p-value in the case of NDCG@10 is 0.0361 and is related to test between LambdaMART and RRLUFF_NDCG. Tables 1 and 2 show the t-test experiment between RRLUFF and the other ones; in all of them, p-values are less than 0.04, and the proposed method outperforms all the other methods significantly at the 0.04 level.

Table 1. The statistical significance of the p-value (paired t-test) between RRLUFF and other compared methods on OHSUMED dataset.

	P-value									
	RRLUFF_P@n					RRLUFF_NDCG				
	MAP	NDCG@10	NDCG@1	P@10	P@1	MAP	NDCG@10	NDCG@1	P@10	P@1
RankSVM	0.0029	0.0039	0.0092	0.0121	0.0071	0.0084	0.0079	0.03	0.0071	0.0043
RankBoost	0.0017	0.0001	0.0041	0.0001	0.0005	0.0009	0.0006	0.0001	0.0005	0.0037
ES-Rank	0.0087	0.0033	0.0107	0.001	0.0271	0.0209	0.0017	0.0021	0.041	0.007
RankNet	0.003	0.0001	0.0098	0.0181	0.0146	0.0181	0.0055	0.0062	0.0114	0.038
Mart	0.0037	0.0007	0.0112	0.0001	0.0008	0.0041	0.0001	0.0025	0.0114	0.0039
LambdaMART	0.0052	0.0031	0.0253	0.0178	0.0271	0.0308	0.0361	0.0148	0.0271	0.0012
RL3F	0.0037	0.0066	0.0021	0.0054	0.0043	0.0048	0.0091	0.0046	0.0094	0.0037
MDPRank	0.0006	0.0085	0.0001	0.0014	0.0027	0.0091	0.0025	0.0075	0.0012	0.023

Table 2. The statistical significance of the p-value (paired t-test) between RRLUFF and other compared methods on DOTIR dataset.

	P-value									
	RRLUFF_P@n					RRLUFF_NDCG				
	MAP	NDCG@10	NDCG@1	P@10	P@1	MAP	NDCG@10	NDCG@1	P@10	P@1
RankSVM	0.0034	0.0003	0.0001	0.0001	0.0013	0.0001	0.0005	0.0007	0.001	0.0012
RankBoost	0.0199	0.0005	0.0406	0.0352	0.0128	0.0092	0.0345	0.0452	0.0218	0.0047
ES-Rank	0.0046	0.0021	0.0051	0.0004	0.0031	0.0008	0.0001	0.0006	0.0004	0.009
RankNet	0.0219	0.0079	0.0008	0.0007	0.0008	0.0021	0.0007	0.0001	0.0001	0.052
Mart	0.0001	0.0012	0.0005	0.021	0.012	0.0016	0.0019	0.0013	0.0012	0.0064
LambdaMART	0.0052	0.0031	0.0253	0.0178	0.0271	0.0308	0.0361	0.0148	0.0271	0.0081
RL3F	0.0149	0.0085	0.0099	0.0013	0.0042	0.0008	0.0191	0.0333	0.0105	0.0034
MDPRank	0.0002	0.0051	0.0073	0.0101	0.0046	0.0111	0.0034	0.0001	0.0033	0.0031

7. Conclusion and Future Suggestions

In this paper, the ranking of Web documents was considered as an RL problem so that the ranking system, as agent, interacts with the environment, and provides a good ranking. The agent's action is to select and display documents to the user. Using the user feedback can bring about the interaction of the ranking system with the user, which provides the ranking system with useful knowledge for ranking. RL reduces the intrinsic impact of noise on a user's clicks, and provides good results for the users even with a low-quality feedback. The proposed ranking method is query-dependent with a listwise approach. This article has pointed out that each one of the content-based and connection-based methods has problems, and using each one alone results in a lower performance of the search engine. The simultaneous use of various features is a suitable solution to the problems of these two methods, which was used in the RRLUFF algorithm.

The proposed algorithm converges rapidly and reduces the rich-get-richer problem because of the use of the action selection of combined incremental ε-greedy and Roulette Wheel (BoostRW) to select the documents displayed for the user. For the proposed method, two versions are based on the two criteria of P@n and NDCG@n. To evaluate the DOTIR and LETOR3 benchmark datasets, the OHSUMED dataset was specifically used. The empirical results showed its superiority to RankBoost, SVMRank, Regression, RankNet, MART, LambdaMART, RL3F, ES-Rank, and MDPRank algorithms. For future directions, in the proposed algorithm, instead of Q-Learning, SARSA could be used to model rankings. In addition, a fuzzy system could be used for a hybrid of two criteria to achieve normalized precision to recognize the importance of lists.

References

[1] 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 and Management, vol. 46, no. 2, pp. 159–169.

[2] Henzinger, M. R., Motwani, R., & Silverstein, C., (2002). Challenges in Web Search Engines, SIGIR Forum, vol. 36, no. 2, pp. 11–22.

[3] Barabási, A.-L. & Albert, R., (1999). Emergence of Scaling in Random Networks, Science, vol. 286, no. October, pp. 509–512.

[4 Liu, T.-Y., (2011). Learning to Rank for Information retrieval. Springer Science & Business Media, Berlin, Heidelberg. New York: Springer-Verlag.

[5] Liu, T.-Y., (2009). Learning to Rank for Information Retrieval, Foundations and Trends® in Information Retrieval, vol. 3, no. 3, pp. 225–331.

[6] Zhang, T., Ghanem, B., Liu, S., & Ahuja, N., (2012). Low-rank sparse learning for robust visual tracking, in European conference on computer vision, ECCV 2012, Vol. 7577, pp. 470–484.

[7] Otsuka, A., Nishida, K., Bessho, K., Asano, H., & Tomita, J., (2018). Query Expansion with Neural Question-to-Answer Translation for FAQ-based Question Answering, in Companion of the The Web Conference 2018 on The Web Conference, WWW, Lyon, France, pp. 1063–1068.

[8] Hu, Y., Da, Q., Zeng, A., Yu, Y., & Xu, Y., (2018). Reinforcement Learning to Rank in E-Commerce Search Engine: Formalization, Analysis, and Application, in Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, London, United Kingdom, arXiv preprint arXiv:1803.00710, pp. 368-377.

[9] Liu, H., Wu, Z., & Zhang, X., (2018). CPLR: Collaborative pairwise learning to rank for personalized recommendation, Knowledge-Based Systems, vol. 148, pp. 31–40.

[10] Joachims, T., (2002). Optimizing search engines using clickthrough data, Kdd '02, in Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining, Edmonton, Alberta, Canada, pp. 133–142.

[11] Phophalia, A., (2011). A survey on Learning to Rank (LETOR) approaches in information retrieval, 2011 Nirma University International Conference on Engineering: Current Trends in Technology, NUiCONE 2011 - in Conference Proceedings, Ahmedabad, Gujarat, India, pp. 8–10.

[12] Burges, C. et al., (2005). Learning to rank using gradient descent, in ICML 2005 - in Proceedings of the 22nd International Conference on Machine Learning, Bonn, Germany, pp. 89–96.

[13] Chapelle, O., Chang, Y., & Liu, T., (2011). Future directions in learning to rank., in Yahoo! Learning to Rank, in Proceedings of the Learning to Rank Challenge, PMLR 14, Haifa, Israel, pp. 91–100.

[14] Ibrahim, O. A. S. & Landa-Silva, D., (2017). ES-Rank: evolution strategy learning to rank approach, in Proceedings of the Symposium on Applied Computing, Marrakech, Morocco, pp. 944–950.

[15] Zhang, Y., Jansen, B. J., & Spink, A., (2009). Time series analysis of a Web search engine transaction log, Information Processing & Management, vol. 45, no. 2, pp. 230–245.

[16] Allen, R., (2017). Search Engine Statistics, Smart Insights. Available: http://www.smartinsights.com/search-engine-marketing/search-engine-statistics/. [Accessed: 13-Apr-2017].

[17] Fishkin, R. & Staff, M., (2015). The beginner's guide to seo, MozBar. Available: https://moz.com/beginners-guide-to-seo. [Accessed: 18-Dec-2015].

[18] Granka, L. A., Joachims, T., & Gay, G., (2004). 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, Sheffield, United Kingdom, pp. 478–479.

[19] Agichtein, E., Brill, E., & Dumais, S., (2006). Improving web search ranking by incorporating user behavior information, SIGIR '06: in Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval, Seattle, Washington, USA, pp. 19–26.

[20] Ahern, P., (2017). How-To's 25 Mind-Bottling SEO Stats for 2017. Available: https://junto.digital/blog/seo-stats-2017/. [Accessed: 02-Mar-2017].

[21] Fishkin, R., (2017). The State of Searcher Behavior Revealed Through 23 Remarkable Statistics. Available: https://moz.com/blog/state-of-searcher-behavior-revealed. [Accessed: 14-Mar-2017].

[22] Liu, Y., Nie, J.-Y., & Chang, Y., (2017). Constructing click models for search users, Information Retrieval Journal, vol. 20, no. 1, pp. 1–3.

[23] Chapelle, O. & Zhang, Y., (2009). A dynamic bayesian network click model for web search ranking, in Proceedings of the 18th international conference on World wide web, ACM, Madrid, Spain, pp. 1–10.

[24] Dupret, G. & Liao, C., (2010). A model to estimate intrinsic document relevance from the clickthrough logs of a web search engine, in Proceedings of the third ACM international conference on Web search and data mining, New York, New York, USA, pp. 181–190.

[25] Borisov, A., Markov, I., de Rijke, M., & Serdyukov, P., (2016). A neural click model for web search, in Proceedings of the 25th International Conference on World Wide Web, Montréal, Québec, Canada, pp. 531–541.

[26] 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.

[27] Li, L. & Lin, H.-T., (2007). Ordinal regression by extended binary classification, Advances in neural information processing systems, Vancouver, B.C., Canada, vol. 19, p. 865.

[28] Derhami, V., Paksima, J., & Khajeh, H., (2014). RLRAUC: Reinforcement learning based ranking algorithm using user clicks, in Proceedings of the 4th International Conference on Computer and Knowledge Engineering, ICCKE 2014, Mashhad, Iran, pp. 29–34.

[29] Wei, Z. & Xu, J., (2017). Reinforcement Learning to Rank with Markov Decision Process, in Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, Tokyo, Japan, pp. 945–948.

[30] Montague, P. R., (1999). Reinforcement Learning: An Introduction, by Sutton, R.S. and Barto, A.G., Trends in Cognitive Sciences, vol. 3, no. 9, p. 360.

[31] Sutton, R. S. & Barto, A. G., (2018). Reinforcement Learning: An Introduction, 2nd ed. London, England: MIT press Cambridge.

[32] Keyhanipour, A. H. et al., (2016). Learning to rank with click-through features in a reinforcement learning framework, International Journal of Web Information Systems, vol. 12, no. 4, pp. 448–476.

[33] Cao, Z., Qin, T., Liu, T.-Y., Tsai, M.-F., & Li, H., (2007). Learning to Rank : From Pairwise Approach to Listwise Approach, in Proceedings of the 24th international conference on Machine learning, Oregon, USA, pp. 129–136.

[34] Xia, F., Liu, T.-Y., Wang, J., Zhang, W., & Li, H., (2008). Listwise approach to learning to rank, Proceedings of the 25th international conference on Machine learning - ICML '08, Helsinki, Finland, pp. 1192–1199.

[35] Master, L., (2017). Improving document ranking with genetic and optimization algorithms, International Journal of Numerical Modelling: Electronic Networks, Devices and Fields, vol. 31, no. 3. pp. e2310.

[36] Xia, F. & Wang, J., (2008). Listwise Approach to Learning to Rank - Theory and Algorithm, Analysis, in Proceedings of the 25th international conference on Machine learning. ACM, Helsinki, Finland, pp. 1192–1199.

[37] Burges, C. J. C., (2010). From ranknet to lambdarank to lambdamart: An overview, Learning, vol. 11, no. 23–581, p. 81.

[38] Volkovs, M. N. & Zemel, R. S., (2009). BoltzRank : Learning to Maximize Expected Ranking Gain, in proceedings of the 26th International Conference on Machine Learning (ICML), Quebec, Canada, pp. 1089–1096.

[39] 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 - GECCO '09, Québec, Canada, pp. 9-16.

[40] Shi, Y. & Eberhart, R. C., (1999). Empirical study of particle swarm optimization, in Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406), 1999, vol. 3, IEEE, pp. 1945–1950.

[41] 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.

[42] Yeh, J.-Y. & Lin, J.-Y., (2017). Learning Ranking Functions For Information Retrieval Using Layered Multi-Population Genetic Programming, Malaysian Journal of Computer Science, vol. 30, no. 1, pp. 27–47.

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

[44] 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.

[45] 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.

[46] Raman, K., Shivaswamy, P., & Joachims, T., (2012). Online learning to diversify from implicit feedback, in Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '12, Beijing, China, pp. 705-713.

[47] Hofmann, K., Whiteson, S., & De Rijke, M., (2011). Balancing Exploration and Exploitation in Learning to Rank Online, Advances in Information Retrieval Proceedings ECIR 2011, vol. 6611, Dublin, Ireland, pp. 251–263.

[48] Chaudhuri, S. & Tewari, A., (2017). Online Learning to Rank with Top-k Feedback, The Journal of Machine Learning Research, vol. 18, no. 1, pp. 3599-3648

[49] Wang, L., Yu, Z., Jin, T., Li, X., & Gao, S., (2016). Expert list-wise ranking method based on sparse learning, Neurocomputing, vol. 217, pp. 119–124.

[50] Derhami, V., Paksima, J., & Khajeh, H., (2015). Web pages ranking algorithm based on reinforcement learning and user feedback, Journal of AI and Data Mining, vol. 3, no. 2, pp. 157–168.

[51] Ibrahim, O. A. S. & Landa-Silva, D., (2017). (1+ 1)-evolutionary gradient strategy to evolve global term weights in information retrieval, in Advances in Computational Intelligence Systems, Springer, Cham, vol. 513, , pp. 387–405.

[52] Sutton, R. S. & Barto, A. G., (1998). Reinforcement learning: An introduction, vol. 1, no. 1. MIT press Cambridge, 1998.

[53] 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: An International Journal, vol. 10, no. 1, pp. 75–88.

[54] Sari, S. & Adriani, M., (2014). Learning to rank for determining relevant document in Indonesian-English cross language information retrieval using BM25, in Proceedings - ICACSIS 2014: 2014 International Conference on Advanced Computer Science and Information Systems, Jakarta, Indonesia, pp. 309–314.

[55] Page, L., Brin, S., Motwani, R., & Winograd, T., (1998). The PageRank Citation Ranking: Bringing Order to the Web, World Wide Web Internet And Web Information Systems, vol. 54, no. 1999–66, pp. 1–17.

[56] Lee, D. L. & Seamons, K., (1997). Document Ranking and the Vector Space Model, IEEE Software, vol. 14, no. 2, pp. 67–75.

[57] Ding, C., He, X., Husbands, P., Zha, H., & Simon, H. D., (2002). PageRank, HITS and a unified framework for link analysis, in Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval - SIGIR '02, 2002, no. 3, p. 353.

[58] Xu, J. & Li, H., (2007). AdaRank: a boosting algorithm for information retrieval, in Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval, vol. 49, Amsterdam, The netherkans, pp. 391–398.

[59] Chuklin, A., Markov, I., & Rijke, M. de, (2015). Click models for web search, Synthesis Lectures on Information Concepts, Retrieval, and Services, vol. 7, no. 3, pp. 1–115.

[60] (2010). dotIR collection. Available: http://ece.ut.ac.ir/DBRG/webir/fa/index.html. [Accessed: 05-May-2017].

[61] Xu, J., Liu, T. Y., & Li, H., (2009). The Evaluation Tool in LETOR, Microsoft Research Asia, pp. 1–4.

[62] Sander Bockting, D. H. T. N., (2015). A cross-benchmark comparison of 87 learning to rank methods, Information Processing & Management, vol. 51, no. 6, pp. 757–772.

[63] Tsai, M.-F., Liu, T.-Y., Qin, T., Chen, H.-H., & Ma, W.-Y., (2007). FRank: A Ranking Method with Fidelity Loss, in Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval - SIGIR '07, Amsterdam, The Netherlands, pp. 383-390.

[64] Smucker, M. D., Allan, J., & Carterette, B., (2007). A comparison of statistical significance tests for information retrieval evaluation, in Proceedings of the sixteenth ACM conference on Conference on information and knowledge management - CIKM '07, New York, New York, USA, 2007, p. 623.

Appendix A: Output of the user-click distribution frequency table.

Mode	**Position**									
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	1	0.28	0.19	0.16	0.13	0.116	0.102	0.088	0.074	0.06
2	0.47	0.62	0.29	0.205	0.12	0.112	0.104	0.096	0.088	0.08
3	0.53	0.37	0.45	0.305	0.16	0.142	0.124	0.106	0.088	0.07
4	0.52	0.3	0.35	0.225	0.155	0.138	0.121	0.104	0.087	0.07
5	0.51	0.23	0.26	0.205	0.15	0.134	0.118	0.102	0.086	0.07
6	0.508	0.2316	0.258	0.205	0.152	0.1364	0.1208	0.1052	0.886	0.074
7	0.506	0.2332	0.256	0.205	0.154	0.1388	0.1236	0.1084	0.0912	0.078
8	0.504	0.2348	0.256	0.205	0.154	0.1388	0.1236	0.1116	0.0938	0.082
9	0.502	0.2364	0.252	0.205	0.16	0.146	0.132	0.118	0.0964	0.086
10	0.5	0.31	0.25	0.205	0.16	0.146	0.32	0.118	0.104	0.09
11	0.59	0.32	0.28	0.225	0.17	0.152	0.134	0.116	0.098	0.08

Appendix B: The set of features used in the RRLUFF calculations on the OHSUMED and DOTIR datasets.

DOTIR		OHSUMED	
Feature ID	**Description**	**Feature ID**	**Description**
11	TF*IDF of body	2	$\sum_{q_i \in q \cap d} \log(c(q_i, d) + 1)$ in title
12	TF*IDF of anchor	3	$\sum_{q_i \in q \cap d} \frac{c(q_i, d)}{\mid d \mid}$ in title
13	TF*IDF of title	4	$\sum_{q_i \in q \cap d} \log(\frac{c(q_i, d)}{\mid d \mid} + 1)$ in title
14	TF*IDF of URL	5	$\sum_{q_i \in q} \log(\frac{\mid c \mid}{df(q_i)})$ in title
15	TF*IDF of the whole document	6	$\sum_{q_i \in q} \log(l\,\mathrm{og}(\frac{\mid c \mid}{df(q_i)}))$ in title
21	BM25 of body	7	$\sum_{q_i \in q} \log(\frac{\mid c \mid}{c(q_i, C)} + 1)$ in title
22	BM25 of anchor	8	$\sum_{q_i \in q \cap d} \log(\frac{c(q_i, d)}{\mid d \mid}.\log(\frac{\mid c \mid}{df(q_i)}) + 1)$ in title
23	BM25 of title	9	$\sum_{q_i \in q \cap d} c(q_i, d).\log(\frac{\mid c \mid}{df(q_i)})$ in title
26	LMIR.ABS of body	10	$\sum_{q_i \in q \cap d} \log(\frac{c(q_i, d)}{\mid d \mid}.\log(\frac{\mid c \mid}{c(q_i, C)}) + 1)$ in title
28	LMIR.ABS of title	11	BM25 of Title
31	LMIR.DIR of body	12	Log(BM25) of title
33	LMIR.DIR of title	13	LMIR.DIR of title
41	Sitemap based term propagation	14	LMIR.JM of title
42	Sitemap based score propagation	26	BM25 of abstract
49	HITS authority	27	Log(BM25) of abstract
50	HITS hub	28	LMIR.DIR of abstract
51	PageRank	41	BM25 of 'title+ abstract'