International Journal of Web Information Systems

Learning to rank with click-through features in a reinforcement learning framework

Amir Hosein Keyhanipour Behzad Moshiri Maryam Piroozmand Farhad Oroumchian Ali Moeini

Received 22 December 2015
Accepted 13 March 2016

Learning to rank with click-through features in a reinforcement learning framework

Amir Hosein Keyhanipour
Center of Excellence, School of Electrical and Computer Engineering, College of Engineering, University of Tehran, Tehran, Iran
Behzad Moshiri
Control and Intelligent Processing, Center of Excellence, School of ECE, University of Tehran, Tehran, Iran
Maryam Piroozmand
Computer Engineering Department, Amirkabir University of Technology, Tehran, Iran
Farhad Oroumchian
Knowledge Village, University of Wollongong, Dubai, United Arab Emirates, and
Ali Moeini
University of Tehran, Tehran, Iran

Abstract

Purpose – Learning to rank algorithms inherently faces many challenges. The most important challenges could be listed as high-dimensionality of the training data, the dynamic nature of Web information resources and lack of click-through data. High dimensionality of the training data affects effectiveness and efficiency of learning algorithms. Besides, most of learning to rank benchmark datasets do not include click-through data as a very rich source of information about the search behavior of users while dealing with the ranked lists of search results. To deal with these limitations, this paper aims to introduce a novel learning to rank algorithm by using a set of complex click-through features in a reinforcement learning (RL) model. These features are calculated from the existing click-through information in the data set or even from data sets without any explicit click-through information.

Design/methodology/approach – The proposed ranking algorithm (QRC-Rank) applies RL techniques on a set of calculated click-through features. QRC-Rank is as a two-steps process. In the first step, Transformation phase, a compact benchmark data set is created which contains a set of click-through features. These feature are calculated from the original click-through information available in the data set and constitute a compact representation of click-through information. To find most effective click-through feature, a number of scenarios are investigated. The second phase is Model-Generation, in which a RL model is built to rank the documents. This model is created by applying temporal difference learning methods such as Q-Learning and SARSA.

This research is financially supported by the University of Tehran; grant number: 8101004/1/02.

International Journal of Web Information Systems
Vol. 12 No. 4, 2016
pp. 448-476

1744-0084
DOI 10.1108/IJWIS-12-2015-0046

Findings – The proposed learning to rank method, QRC-rank, is evaluated on WCL2R and LETOR4.0 data sets. Experimental results demonstrate that QRC-Rank outperforms the state-of-the-art learning to rank methods such as SVMRank, RankBoost, ListNet and AdaRank based on the precision and normalized discount cumulative gain evaluation criteria. The use of the click-through features calculated from the training data set is a major contributor to the performance of the system.

Originality/value – In this paper, we have demonstrated the viability of the proposed features that provide a compact representation for the click through data in a learning to rank application. These compact click-through features are calculated from the original features of the learning to rank benchmark data set. In addition, a Markov Decision Process model is proposed for the learning to rank problem using RL, including the sets of states, actions, rewarding strategy and the transition function.

Keywords Click-through data, Learning to rank, Reinforcement learning

Paper type Research paper

1. Introduction

Because of the drastic growth of the Web information, Web search engines have become an essence of the information era. Information retrieval (IR) is defined as a ranking process in which a set of documents are ordered based on their relevance to the users' information need. In recent years, "Learning to Rank" has emerged as an active and growing area of research in both IR and machine learning research. Consequently, several learning to rank algorithms have been proposed, such as RankSVM (Herbrich *et al.*, 2000; Joachims, 2002), RankBoost (Freund *et al.*, 2003), AdaRank (Xu and Li, 2007) and ListNet (Cao *et al.*, 2007). Although these ranking methods have shown reasonable performance based on the evaluation criteria on benchmark data sets, but they have not taken advantage of the click-through data as a source of users' feedbacks (Dou *et al.*, 2008). One reason could be scarcity of explicit click-through data in the released and publicly available benchmark data sets.

Given lack of sufficient data sets with click-through information, one of the aims of this research is proposing a framework for generating click-through data from the information presented in the learning to rank data sets. We have also looked at effectiveness of various features in click-through data and experimentally proposed subsets of features that are more useful in learning to rank. This research also utilizes reinforcement learning (RL) methods to learn and adapt to the desired ranking for users.

The main contributions of this research could be summarized as:

- proposing a novel click-through feature generation framework from benchmark data sets that lack click-through information;
- analyzing the performance of the proposed click-through features using various scenarios on LETOR4.0 and WCLR benchmark data sets;
- designing a RL model with for temporal learning methods for ranking; and
- demonstrating the viability of using click-through features with the proposed method.

The rest of this paper is organized as follows: Section 2 provides an overview of the application of click-through data and RL methods in the learning to rank problem. Section 3 describes the fundamental ideas of the proposed method. Section 4 presents the details the evaluation settings and analytical discussion of the results. Finally, Section 5 provides the conclusion and future work.

2. Related works

Joachims (2002) for the first time introduced the application of click-through data as an alternative to the explicit relevance judgments in the RankSVM system. The RankSVM system still is one of the most powerful ranking methods. Later, it was observed that considering a user's queries as chains rather than considering each query individually produces more reliable inferred relevance judgments from the click-through data (Radlinski and Joachims, 2005; Macdonald and Ounis, 2009; Macdonald *et al.*, 2013).

The research in this area can be divided into three major categories. The first category includes those works that investigate the effect of the implicit feedback of users on the performance of learning to rank algorithms (Agichtein *et al.*, 2006a, 2006b; Dou *et al.*, 2008). The second category consists of research that intends to enhance the quality of click-through data. The last category includes those investigations that utilize click-through data to improve the performance of learning to rank algorithms.

Xu (2010) is an example research in second category. Xu (2010) tries to find what kind of input is required and how to obtain such an input using the implicit or explicit feedback for learning to rank approaches. Another example is Radlinski (Radlinski and Joachims, 2007), who presents an active exploitation strategy for collecting users' interaction records from search engine click-through logs. His proposed algorithm is a Bayesian approach for selecting rankings to present users so that interactions result into a more informative training data. In Xu *et al.* (2010), a method is proposed, which automatically detects judgment errors by using the click-through data. The sparseness of the click-through data is a major challenge in learning to rank approaches that have been investigated by researchers such as Gao *et al.* (2009). They have proposed two techniques for expanding click-through features to address the sparseness.

Most of research also has focused on using click-through data to improve the performance of the learning to rank methods. Ji *et al.* (2009) have chosen a minimalistic approach and by exploiting user click sequences based on a limited number of features have proposed a global ranking framework. Interestingly, Dupret and Liao (2010) have used click-through data exclusively for generating a relevance estimation model. The model was utilized to predict the document relevance. Click-through data are also utilized to provide deep structured latent semantic models for web search (Huang *et al.*, 2013). These models project queries and documents into a common low-dimensional space, where the relevance of a document given a query is readily computed from the distance between them. Click-through data have been successfully used in various areas of IR, including user modeling (Wang *et al.*, 2014; Agichtein *et al.*, 2006a, 2006b), query suggestion (Ma *et al.*, 2008) and image retrieval (Bai *et al.*, 2013; Jain and Varma, 2011). Hofmann *et al.* (2013) also have tried using historical data to speed up online learning. In the online learning to rank, the retrieval system learns directly from interactions with its users. This approach integrates estimations derived from historical data with a stochastic gradient descent algorithm for online learning to rank (Hofmann *et al.*, 2013).

RL methods are rarely applied to resolve the learning to rank problem. A related work is Derhami *et al.* (2013), in which based on the PageRank's random surfer model, a general ranking method is proposed in an RL structure. However, because this ranking algorithm does not deal with feature vectors of query-document pairs, it could not be categorized as a learning to rank algorithm. Another application of RL for the ranking problem is A3CRank algorithm, which aggregates the ranking results from a few ranking algorithms such as TF-IDF, BM25 and PageRank (Zareh Bidoki *et al.*, 2010). In a study by Hofmann *et al.* (2013), an RL model is proposed to assist IR systems to learn from users' interactions. Specifically, it presents an interleaved comparison method for online learning to rank problem.

This research also is concentrated on the application of RL techniques and learning to rank using the click-through data. In a study by Keyhanipour *et al.* (2007), a method called WebFusion is introduced, in which learning to rank from click-through data and information fusion have been successfully combined within an intelligent meta-search engine environment.

3. Proposed approach

The proposed learning to rank algorithm consists of two phases – *Transformation* and *Model-Generation* – that will be described in the next subsections. Briefly, within the *Transformation* phase, a feature generation mechanism will be applied to the benchmark data set and a compact representation will be generated as triplets of queries, results and a subset of clicks through features. Then, in the *Model-Generation* stage, a RL model is generated the learning to rank problem. During this step, temporal difference learning mechanisms such as Q-Learning and SARSA are used to find near-optimal solutions for the compact representation of the first phase. Following is the outline of the proposed learning to rank algorithm, which summarizes the proposed learning to rank method, which is called QRC-Rank:

The proposed learning to rank method: QRC-Rank

Input:

a learning to rank benchmark data set which consists of a set of query-document pairs with their feature vectors and relevance judgments (i.e., the training set, T)

Output:

an action table, A, which provides the most appropriate action (degree of relevance), for the state corresponding to a query-document pair

Procedure of the QRC-Rank:

Step 1. Transformation:

1. Selection of the scenarios needed for the calculation of click-through features from training set T.

2. Generation of click-through features from T based on the suggested scenarios. This process generates a secondary data set T' from T, which includes the generated click-through features corresponding to query-document pairs.

Step 2. Model-Generation:

3. Generating a Markov Decision Process Model for the learning to rank problem, including the sets of States, Actions, Rewarding Strategy and also the Transition Function.

4. Applying Temporal-Difference learning methods, Q-Learning and SARSA, on the proposed Markov Decision Process Model to realize the most relevance label for each query-document pair.

For better clarification, the same process is graphically illustrated in Figure 1.

3.1 Transformation phase

In the context of IR, ranking a set of documents in respect to a given query is influenced by a variety of features, which are related to this query-document pair. Some of these features are: term-frequency, inverse document-frequency and PageRank. Any given benchmark data set prepared for the learning to rank problem consists of the values of

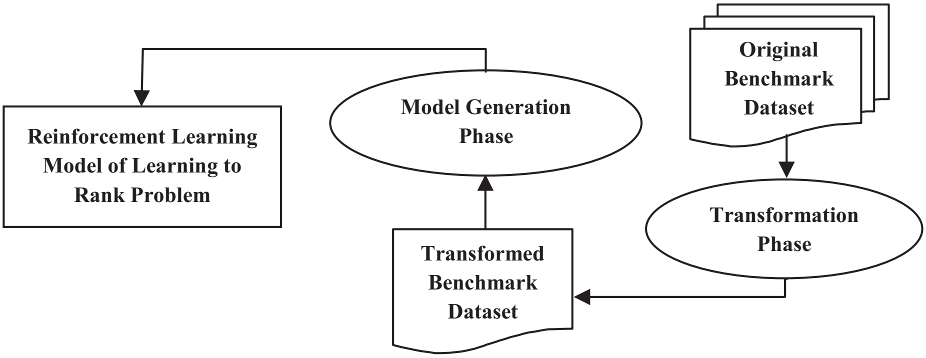

Figure 1.
Steps of the QRC-Rank algorithm

such features, which are calculated for some pairs of queries and documents, as well as the relevance degree of a document with respect to a specific query. There are two problems with these data sets to achieve a learning to rank algorithm. First, because of the presence of a large number of features, these data sets usually are high-dimensional. Usage of a large number of features leads to the inefficiency of the derived ranking algorithms in real-world situations. Second, these data sets usually do not contain the click-through data. Click-through data is an important source of implicit feedback of the users of Web search engines (Dou *et al.*, 2008; Xu *et al.*, 2010).

The goal of the Transformation phase is to generate click-through information for the benchmark data sets even if they lack such information. In this phase, a compact representation of original benchmark data set is produced based on a triplet of query (*Q*), ranked list of results (*R*) and features related to clicks of users (*C*) (Joachims, 2002). In this phase, eight features are defined in three groups: *Q*, *R* and *C*. These features are:

$$Q = \{Repetition, QScore, ResultsAmount\}$$
$$R = \{AbsoluteRank, StreamLength\}$$
$$C = \{Specificity, Attractiveness, ClickRate\}$$

Q contains features related to the nature of the queries of users. *Repetition* deals with the frequency of query terms in different parts of a Web document, including URL, title and content. *QScore* refers to the score of a document with respect to a given query. The *QScore* is generated by query-dependent ranking algorithms such as vector space and language models. Finally, *ResultAmount* indicates the number of results retrieved for a specified query.

In the same way, features of the category *R* highlight the characteristics of the Web documents independent of any query. In this category, *AbsoluteRank* shows the absolute rank of a given Web document. Undoubtedly, in calculation of this feature, query-independent specifications such as PageRank play an important role. *StreamLength* is a structure containing the length of document's URL length, its title length and the length of a document's content.

The category *C*, includes those features which deal with the users' click-through data. *Specificity* is an indication of the uniqueness of a given document for a set of queries. In other words, for a given Web document, *Specificity* shows how many users have clicked on this document for a given set of queries. The *Attractiveness* feature is an indicator of the number of Web users' attention to a given document during their search interactions. *Attractiveness* distinguishes between Web documents that are clicked first or last from those clicked during the rest of the search session. Surprisingly, these features could be calculated in the presence or even absence of click-through data.

The computation of the above-mentioned features is completely dependent on the amount of the information available in a specific benchmark data set. In the next section, a few scenarios would be presented for the calculation of the above features for two standard benchmarking data sets: LETOR4.0 and WCL2R.

3.2 Model-generation phase

In Model-Generation phase a model will be created for ranking web documents using RL techniques. The input data for this phase comes from the *Transformation* phase, which is an eight-dimensional data set, containing the generated click-through features in categories *Q*, *R* and *C*.

In this phase, a Markov Decision Process (MDP) model is generated as a triple of $\{States, Actions, Rewards\}$. The proposed MDP model is:

$$State = \{Q, R, C\} =$$

$$\left\{\begin{matrix} Repetition,\ QScore,\ Results\ Amount,\ AbsoluteRank,\ StreamLength, \\ Specificity,\ Attreactiveness,\ ClickRate \end{matrix}\right\}$$

$$Action = DifferentRelevanceLevels$$

$$Reward = -ABS(action - classLabel)$$

Based on the above definition of the learning to rank problem, any query-document pair specifies the current state of the learning agent as an eight-dimensional space of click-through features. In each state in this space, the learning agent may select an action from the set of possible actions (Relevant, Non-Relevant […]). Finally, the agent receives a numerical reward, which indicates the distance between the true relevance label of the corresponding query-document pair and the label, which was selected by the agent during its most recent action. For this definition, we can perceive that the Markov property, which is the independence of receiving a reward at a particular state from the previous states and actions, withholds (Sutton and Barto, 1998). This is because of our episode generation policy, in which data items are selected from the training set by the uniform distribution probability. Each data item belonging to an episode will be visited independent of other data items. Formally, we have:

$$\Pr\{s_{t+1} = s',\ r_{t+1} = r \mid s_t,\ a_t,\ r_t,\ s_{t-1}, a_{t-1}, \ldots, r_1, s_0, a_0\} = \Pr\{s_{t+1} = s',\ r_{t+1} = r \mid s_t, a_t\}$$

In the above equation, by doing action a_t in the state s_t at time-step *t*, the learning agent receives a reward r_{t+1}, and the surrounding environment transforms into the state s_{t+1}. Because the Markov property withholds in the proposed RL model, the learning agent can benefit from temporal-difference learning methods such as Q-Learning and SARSA. These methods use various updating mechanisms to bring up to date their estimations about the appropriateness of doing possible actions in different states (Szepesvari, 2010). Suppose $Q(s_t, a_t)$ is the estimation of the learning agent about the goodness of doing action a_t while being in state s_t at time-step *t*. SARSA estimates the values of the accomplished actions in visited states, based on the recently achieved reward, as well as its estimation about the goodness of doing next action in the new state, $Q(s_{t+1}, a_{t+1})$. In this way, SARSA is an *on-policy* RL algorithm with the below updating rule:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]$$

In contrast, using the Q-learning algorithm, the RL agent learns an optimal policy independent of its current action selection policy, provided that does enough exploration. In fact, Q-Learning renews its estimation about $Q(s_t, a_t)$ regarding the immediate reward, as well the goodness of the most suitable action in the next visiting state. Thus, for the Q-Learning algorithm, the updating rule is defined as:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\left[r_{t+1} + \gamma \max_{\alpha} Q(s_{t+1}, a) - Q(s_t, a_t)\right]$$

In the above formulae, α is a constant step-size parameter and $\gamma \in [0,1]$ is the discount rate. As it was mentioned previously, each training episode conations of a fixed number of data items (query-document pairs), which are selected by equal chance from the underlying benchmark data set. This strategy will guaranty the Markov property in the proposed representation of the learning to rank problem. In this framework, the RL agent tries to find the best action, which is the most suitable relevance label for each state, in an iterative manner.

4. Evaluation framework

4.1 Benchmark data sets

The main capability of our proposed QRC-Rank system is its ability to extract required click-through features from any given benchmark data set during the *Transformation* phase. We believe utilizing such click-through features are one of the contributors to higher performance of QRC-Rank in comparison to other well-known ranking methods. To evaluate the performance of the QRC-Rank system, we have used two benchmark data sets LETOR4.0 which does not include click-through features, as well as the WCL2R data set, which contains such features.

Microsoft's LETOR 4.0 is a set of benchmark data sets published for research on the learning to rank problem in July 2009 (LETOR4.0 Data sets, 2009). It consists of two data sets named as MQ2007 and MQ2008, which are designed for four different ranking settings: supervised, semi-supervised, list wise ranking and rank aggregation. There are about 1,700 queries in MQ2007 and 800 queries in MQ2008 with a number of human-labeled documents (Qin *et al.*, 2007). LETOR4.0 data set provides a feature vector containing 46 features for each pair of query-document. These features cover a wide range of common IR features and information such as term frequency, inverse document frequency, BM25, Language Models for IR (LMIR), PageRank and HITS. However, LETOR4.0 data sets do not contain any click-through data (Alcantara *et al.*, 2010). In this research, the "supervised ranking" part of LETOR4.0 is utilized, which is MQ2008. It is organized in fivefold structure, including training, validation and testing data and contains for each pair of query-document, a relevance label based on the human judgment in three relevance levels. The larger the relevance label, the more relevant the query-document pair. Each row of the LETOR4.0 data set is related to a query-document pair. The structure of a typical row of the LETOR4.0 is represented in Figure 2.

In the above figure, the first column is the relevance label of the document to that query. The second column is query Id; the following columns are Ids of features plus their values which are real values normalized between [0,1] for each feature. At the end of the row is comment about the pair, including Id of the document.

A second set of experiments also was conduct on WCL2R data set. WCL2R is released in October 2010 by a consortium of Federal University of Minas Gerais, Brazil, and the University of Pompeu Fabra, Spain (Alcantara *et al.*, 2010). WCL2R is intended to focus on the click-through data alongside traditional IR features. It contains two snapshots of the Chilean Web, which were crawled in August 2003 and January 2004 by the TodoCL (2002) search engine. The data is structured in ten folds, containing training, validation and testing data. Human judgments are presented in four relevance levels (WCL2R, 2010). The structure of each row of the WCL2R is similar with those of the LETOR4.0, which is depicted in Figure 2. However, the values of the features are not normalized in the WCL2R data set.

Table I provides an overview of LETOR4.0 and WCL2R collections. Training a ranking model in the LETOR4.0 data set is more difficult than those of the WCL2R data set. The main reason is that WCL2R has explicit click-through data, whereas such data are not available in the LETOR4.0. The second reason is the presence of only 6.13 per cent of total relevant documents per any given query in the LETOR4.0 data set, whereas this quantity is about 29.8 per cent in the WCL2R data set.

4.2 Experimental settings

The first phase of QRC-Rank system is computing the click-through features. In this research, we have looked at different scenarios for calculating these features. As explained below some of these calculations are based on smoothing of the values. Additionally, a binary discretization based on the mean of the values has been applied to the features of all of these scenarios.

Tables II and III list the scenarios that we have test for calculating the click-through features on WCL2R and LETOR4.0 data sets. In these scenarios, a limited number of features of WCL2R and LETOR4.0 data sets are used, and their list is presented in Appendix 1 (Alcantara *et al.*, 2010) and Appendix 2 (LETOR4.0's Features List, 2009). In Tables II and III, the primitive features are denoted by "F_i", where i stands for the ID of the feature in the corresponding Tables AI and AII.

Three different scenarios based on the click-through features of the WCL2R benchmark data set that have been experimented with are explained in Table II.

rel qid:*QID* 1:F_1 2:F_2… 46:F_{46}#docid:*DocID comments*

Figure 2. Structure of the LETOR4.0 data set

Table I. Summary of specifications of LETOR4.0 and WCL2R data collections

Data set	No. of features	No. of queries	No. of query-document pairs	Relevance levels	Average no. of documents per query	Average no. of relevant documents per query
LETOR4.0-MQ2008	46	784	15211	3	19.40	1.19
WCL2R	29	79	5200	4	61.94	18.01

Table II.
Click-through feature calculation scenarios for WCL2R benchmark data set

Scenario ID	Computation mechanism
WCL2R-DF1 and WCL2R-DF2	$Q:\begin{cases} Repetition = F_3 \times F_6 \times F_9, \\ QScore = F_{13} \times F_{15} \times F_{16}, \\ ResultsAmount = No.\ data\ items\ with: qid = q \end{cases}$ $R:\begin{cases} AbsoluteRank = F_{14}, \\ StreamLength = F_{10} \times F_{11} \end{cases}$ $C:\begin{cases} Specificity = F_{22}/F_{23}, \\ Attractiveness = F_{17}/F_{18}, \\ ClickRate = F_{20} \times F_{28} \times F_{29} \end{cases}$
WCL2R-DF3	$Q:\begin{cases} Repetition = F_3, \\ QScore = F_{13} \times F_{15} \times F_{16}, \\ ResultsAmount = No.\ data\ items\ with: qid = q \end{cases}$ $R:\begin{cases} AbsoluteRank = F_{14}, \\ StreamLength = F_{10} \end{cases}$ $C:\begin{cases} Specificity = 1/F_{23}, \\ Attractiveness = F_{17} \times F_{24}, \\ ClickRate = QScore \times AbsoluteRank \times Attractiveness \end{cases}$

Table III.
Click-through feature calculation scenarios proposed for LETOR4.0 benchmark data set

Scenario ID	Computation mechanism
LETOR4-DF1 and LETOR4-DF2	$Q:\begin{cases} Repetition = F_{15}, \\ QScore = \prod_{i=37}^{40} F_i, \\ ResultsAmount = No.\ data\ items\ with: qid = q \end{cases}$ $R:\begin{cases} AbsoluteRank = F_{41}, \\ StreamLength = F_{16} \end{cases}$ $C:\begin{cases} Specificity = F_{44}, \\ Attractiveness = F_{41}, \\ ClickRate = QScore \times AbsoluteRank \end{cases}$
LETOR4-DF3	$Q:\begin{cases} Repetition = F_{15}, \\ QScore = \prod_{i=37}^{40} F_i, \\ ResultsAmount = No.\ data\ items\ with: qid = q \end{cases}$ $R:\begin{cases} AbsoluteRank = F_{41}, \\ StreamLength = F_{20} \end{cases}$ $C:\begin{cases} Specificity = F_{44} \times F_{45}, \\ Attractiveness = F_{41} \times F_{42}, \\ ClickRate = Attractiveness \times QScore \times AbsoluteRank \end{cases}$

Each of these three scenarios provides an interpretation of the click-through features. For example, in the *WCL2R-DF1* scenario, a document's *Repetition* feature is calculated by the multiplication of TF-IDF values over the whole of that document, its title and its URL. For this scenario, the *QScore* of a given document is computed by the product of BM25 rank, HITS Hub and HITS Authority values of that document, which all of these rankings are query-dependent. *AbsoluteRank* score is equal to the PageRank score of the corresponding document, which is a query-independent ranking algorithm. A given document is assumed long if both of its content and its title are lengthy. This characteristic is stored in the *StreamLength* feature. A document is assumed specific, if for a few queries it was clicked by many users in many search sessions. In addition, a document is supposed to have a higher degree of *Attractiveness* if it was commonly clicked in the beginning of users' search sessions rather than being clicked at the end of search sessions. Finally, the *ClickRate* feature of a particular document is calculated by multiplying the total amount of users' clicks on it, number of non-single click sessions and number of non-single click queries.

The main difference between *WCL2R-DF1* and *WCL2R-DF2* scenarios is that in the former, smoothing is accomplished by:

$$F_i' = F_i + \varepsilon_i, \text{ where: } \varepsilon_i = 0.01 \times Average\,(F_i^{1:N})$$

In the above equation, ε_i is a fraction of average over all values of feature F_i. However, for the latter, the Dirichlet prior smoothing mechanism (Zhai and Lafferty, 2001) is used:

$$F_i' = F_i + \varepsilon_i, \text{ where: } \varepsilon_i = 0.01 \times SecondMin\,(F_i)$$

In *WCL2R-DF3* scenario, the *Specificity* of a document is defined as the inverse of the number of distinct queries for which that document was clicked. Besides, in this scenario, a given document is considered to achieve a higher *Attractiveness* value if it is the first clicked item in many search sessions, and it has received many single clicks in dissimilar sessions. Furthermore, the *ClickRate* of a given document is related to its attractiveness, query-dependent and query-independent ranking scores. Although *WCL2R-DF3* scenario uses only 31 per cent of original features, its performance is substantially better than those of best-known ranking methods.

In a similar way, three scenarios are defined for the LETOR4.0 benchmark data set, and they are listed in Table III. LETOR4.0's features is presented in the Appendix 2 (LETOR4.0's Features List, 2009).

The main difference between the *LETOR4-DF1* and *LETOR4-DF2* scenarios is that in the former, smoothing is done based on the above-mentioned Dirichlet prior smoothing (Zhai and Lafferty, 2001), whereas in the latter, no smoothing is done. Because there is no explicit feature in LETOR4.0 data set related to the click-through data, in *LETOR4-DF1* and *LETOR4-DF2* scenarios, it is assumed that *ClickRate* of a specific document is related to its query-dependent and query-independent ranking scores. This assumption is completed in the *LETOR4-DF3* scenario, by taking into account the effect of the *Attractiveness* feature. As it will be described in the next section, performance of these scenarios is related to the maturity of their interpretation from click-through features. It is worth mentioning that all of these scenarios use only a limited number of the original features of the data set, whereas according to the experimental results, their performances are comparable or even better than those of the well-known ranking algorithms.

Table IV provides a comparison of different scenarios based on the number of features generated and or used in each scenario. As it can be observed, these scenarios provide a very compact representation of the data set's features because they utilize only very few features from the data set plus eight features that they generate.

4.3 Evaluation metrics

Various measures have been used for the evaluation of performance of IR systems such as *Kendall-Tau* (Kendall, 1948), *P@n*, *NDCG@n* and *MAP* (Manning *et al.*, 2008). The following evaluation criteria are used in this research:

4.3.1 Precision at position n (P@n). This indicates the ratio of relevant documents in a list of the first *n* retrieved documents. The main aim of this metric is to calculate the precision of retrieval systems from users' perspective. Because users visit only top documents from the list of results, this evaluation criteria only consider the *n* top documents. Suppose, we have binary judgments about the relevance of documents with respect to a given query. In this way, each document may be either relevant or irrelevant with respect to a specific query. Then, *P@n* is defined as:

$$P@n = \frac{\#relevantdocs\ in\ top\ n\ results}{n}$$

4.3.2 Mean average precision. For a single query q, *Average Precision (AP)* is defined as the average of the *P@n* values for all relevant documents, where n goes from 1 to the number of retrieved documents:

$$AVG(q) = \frac{\sum_{j=1}^{|D_q|} (r(j) \times P@j)}{|R_q|}$$

In this formulation, r_j is the relevance score assigned to a document d_j with respect to a given query q, being one, if the document is relevant and zero otherwise; D_q is the set of retrieved documents and R_q is the set of relevant documents for the query q. Then, mean average precision (MAP) would be the mean of average precisions of all queries of the utilized benchmark data set as:

$$MAP = \frac{\sum_{q=1}^{|Q|} Avg(q)}{|Q|}$$

Table IV. Comparison of different scenarios of the proposed ranking method based on the number of consumed and generated features, which are first and second data items in each parenthesis

Dataset	No. of features	No. of features per QRC-Rank scenarios (consumed, generated)		
WCL2R	29	WCL2R-DF1 (16,8)	WCL2R-DF2 (16,8)	WCL2R-DF3 (9, 8)
LETOR4.0	46	LETOR4-DF1 (9, 8)	LETOR4-DF2 (9, 8)	LETOR4-DF3 (10,8)

The abovementioned ranking evaluation criterias (*P@n* and *MAP*) consider only binary degrees of relevance in the evaluation of query-document pairs.

4.3.3 Normalized discount cumulative gain at position n (NDCG@n). By assuming different levels of relevance degrees for data items, the *NDCG* of a ranked list at position *n* (*NDCG@n*) would be calculated as follows:

$$NDCG@n = 2^{r_1} - 1 + \sum_{j=2}^{n} \frac{2^{r_j} - 1}{\log(1 + j)}$$

In this formulation, r_j stands for the relevance degree of the j^{th} document in the ranked list.

5. Experimental results

In this section, the experimental results of applying the QRC-Rank algorithm on the WCL2R and LETOR4.0 benchmark data sets and the analytical comparison of the results with those of the well-known ranking algorithms are presented. All of the reported results for the LETOR4.0 data set are based upon the usage of the LETOR's Eval-Tool (Qin *et al.*, 2007). For the WCL2R experiments, based on the structure of this data set, an adapted copy of the Eval-Tool is utilized. It is noticeable that the results are achieved on a PC with a 2.0 GHz dual core processor, 2MB of cache and 3GB of RAM.

For each data set, the results of the QRC-Rank are compared with those reported for the baseline ranking algorithms. As it will be observed in the next subsections, the performance of the baseline algorithms on the WCL2R and LETOR4.0 benchmark data sets are different. This is mainly because of the nature of the ranking algorithms, as well as the structure of these data sets. As mentioned in Section 4, the utilized data sets provide different sets of features for the learning to rank problem. Specifically, in the WCL2R data set, some click-through data are available beside standard IR-related features, but the LETOR4.0 data set does not include click-through data. On the other hand, various ranking methods use different parts of evidence in their ranking functions. Consequently, successful ranking algorithms on these data sets are different.

5.1 WCL2R results

Table V demonstrates the performance of a few well-known ranking techniques on upon the precision evaluation criterion on the WCL2R data set (Alcantara *et al.*, 2010).

In the above table, the first baseline algorithm is SVMRank, which uses the support vector machine (SVM) technology for ranking documents (Joachims, 2002, 2006). The main idea of SVMRank is to formalize learning to rank as the binary classification on document pairs, where two classes are considered for applying SVM: correctly ranked and incorrectly ranked pairs of documents. The second baseline algorithm is LAC (Veloso *et al.*, 2008), a lazy associative classifier that uses association rules to learn

Baseline methods	P@1	P@3	P@10	MAP
SVMRank	*0.400*	*0.455*	*0.397*	*0.432*
LAC	0.383	0.449	0.385	0.427
GP	0.362	0.435	0.387	0.422
RankBoost	0.378	0.416	0.369	0.412

Table V. Comparison of well-known ranking methods based on precision criterion on the WCL2R data set ranking models at the query-time. By generating rules on a demand-driven basis, only the required information is extracted from the training data, resulting in a fast and effective ranking method. The third baseline method is called GP. This method is based on a genetic programming ranking algorithm (Almeida *et al.*, 2007). Finally, the last baseline algorithm, RankBoost, is a boosting algorithm that trains weak rankers and combines them to build the final rank function (Freund *et al.*, 2003).

Table VI demonstrates the precision achieved by the proposed ranking algorithm using different configurations on the WCL2R data set.

Table VII provides details of the settings of different configurations of Table VI. These settings include the parameters of the utilized temporal difference learning algorithms, which are: q_0, α, γ and ε. It must be noticed that the action selection policy for configuration *QRC.W3* is ε-greedy, whereas it is Softmax for the other three implementations of the QRC-Rank. For the Softmax action selection mechanism (Szepesvari, 2010), in which the probability of choosing an action within a given state is proportional to the current estimation of its goodness, the computational temperature, τ, is set to be 10.

The results that are reported in Table VI, illustrate that QRC-Rank has achieved higher precision and MAP values in comparison to the baseline methods on the WCL2R benchmark data set. A significant improvement of about 20.17 per cent is obtained for the proposed method in comparison to the best baseline algorithm, SVMRank on the P@1 criterion. The improvement is about 23.02 per cent for the P@2 measure. Also, the QRC-Rank has achieved a rise of about 2.36 per cent on the MAP criterion with comparison with the SVMRank. Our proposed method has outperformed the RankBoost algorithm by 7.33 per cent.

Moreover, the proposed method has achieved its best performance at the top of the ranked lists of results, which are usually mostly visited by the Web users rather than lower ranks that of less importance for the user. Based on the published results of the eye-tracking studies (Granka *et al.*, 2004; Miller, 2012), about 54 per cent of clicks of the users of Google (1998), as the most widely used Web search engine, were on its first

Table VI. Results of the evaluation of different configurations of the proposed method based on precision criterion on the WCL2R data set

QRC-Rank configuration	P@1	P@2	P@3	P@4	P@5	P@6	P@7	P@8	P@9	P@10	MAP
QRC.W1	0.4427	0.4737	0.4636	0.4499	0.4467	0.4235	0.4106	0.3961	0.3863	0.4103	0.4303
QRC.W2	0.3921	0.4104	0.4225	0.4246	0.4113	0.4088	0.4033	*0.4024*	*0.3919*	0.3985	0.4066
QRC.W3	0.4706	0.488	*0.4785*	*0.4603*	*0.4529*	*0.4424*	*0.4333*	0.398	0.3878	*0.4107*	*0.4422*
QRC.W4	*0.4807*	*0.4921*	0.451	0.4309	0.424	0.4151	0.3943	0.3819	0.3694	0.407	0.4246

Table VII. Configurations of the proposed method used for the evaluation on the WCL2R data set

QRC-Rank configuration	Method	Scenario	Parameters				No. of iterations	Episode length
			q_0	α	γ	ϵ		
QRC.W1	Q-Learning	WCL2R-DF1	10	1/iteration	0.1	Softmax, τ:10	1000	100
QRC.W2	Q-Learning	WCL2R-DF2	10	1/iteration	0.1	Softmax, τ:10	1000	100
QRC.W3	Q-Learning	WCL2R-DF3	100	1/iteration	0.1	0.1	1000	100
QRC.W4	Q-Learning	WCL2R-DF3	10	1/iteration	0.1	Softmax, τ:10	1000	100

search results, and about 80 per cent of clicks were accomplished only on the top three results.

Figure 3 depicts a comparison of the best configuration of the proposed algorithm, *QRC.W3*, with the baseline methods on the P@n criterion in WCL2R data set.

To have a more precise insight about the performance of the proposed ranking method, Tables VIII and IX present the comparison of its results with those of the well-known ranking algorithms based on the NDCG measure on WCL2R benchmark data set.

The above statistics show a reasonable improvement over the baseline methods based on the NDCG measure. This improvement is especially noticeable on the top positions of the ranked list. In this regard, in its best setting, the QRC-Rank algorithm has achieved an improvement of about 15.44 per cent compared with SVMRank on the NDCG@1 measure. The improvement for the NDCG@3 criterion is about 7.28 per cent and for the NDCG@10 criterion is about 6.4 per cent. Figure 4 illustrates a graphical representation of these statistics.

Figures 5 and 6, respectively, present the "Optimal Action Selection Rate" and "Average Received Rewards" per iteration for the SARSA and Q-Learning implementations of the QRC-Rank method on WCL2R data set. According to these diagrams, both of the utilized RL methods have an almost identical performance.

In these experimentations, the elapsed times for SARSA and Q-Learning methods are 29.766983 and 31.080834 s, respectively.

5.2 LETOR4.0 results

For the MQ2008 part of the LETOR4.0 data set, performance of some of some well-known ranking algorithms are reported based on the precision and NDCG criteria.

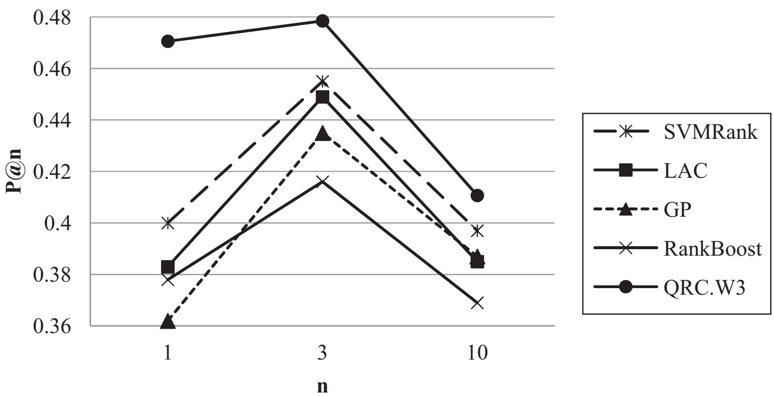

Figure 3. Comparison of the best configuration of the proposed algorithm against baseline methods on *P@n* criterion in the WCL2R data set

Table VIII. Evaluation results of baseline ranking methods based on NDCG criterion on the WCL2R data set

Baseline methods	NDCG@1	NDCG@3	NDCG@10
SVMRank	*0.314*	0.353	0.395
LAC	0.296	*0.360*	*0.403*
GP	0.288	0.344	0.396
RankBoost	0.295	0.328	0.375

Table IX.
Comparison of the performance of different configurations of the proposed method based on NDCG measure on the WCL2R data set

QRC-Rank configuration	NDCG @1	NDCG @2	NDCG @3	NDCG @4	NDCG @5	NDCG @6	NDCG @7	NDCG @8	NDCG @9	NDCG @10	Mean NDCG
QRC.W1	0.3613	*0.3845*	0.3648	0.3715	0.3758	0.3727	0.3727	0.3749	0.3758	0.3766	0.4764
QRC.W2	0.3324	0.3518	0.3568	0.3527	0.353	0.3597	0.3641	0.3703	0.3701	0.3705	0.4732
QRC.W3	0.3414	0.3792	*0.3862*	*0.3855*	*0.3802*	*0.3889*	*0.3878*	*0.3881*	*0.3903*	0.4114	*0.4859*
QRC.W4	*0.3625*	0.3792	0.3698	0.3741	0.369	0.3777	0.3736	0.3728	0.3741	*0.4203*	0.4758

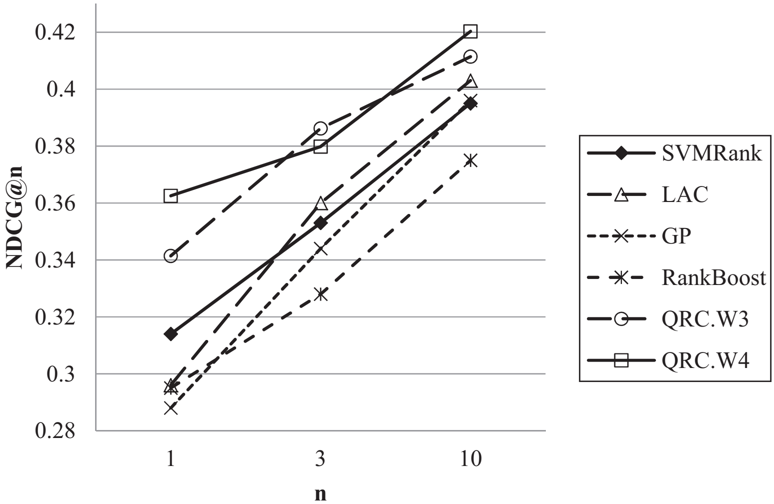

Figure 4.
Comparison of the best configuration of the proposed algorithm against baseline methods on *NDCG@n* criterion in the WCL2R data set

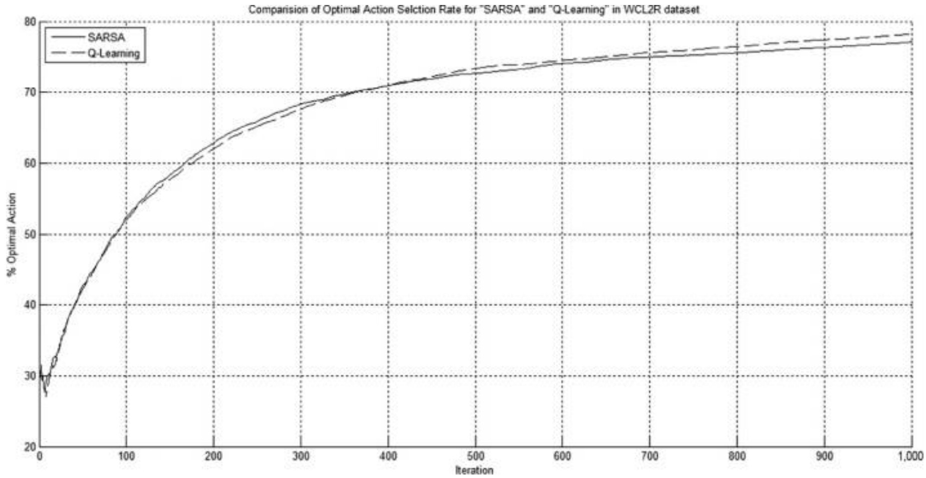

Figure 5.
Comparison of the performance of SARSA and Q-Learning methods based on the "Optimal action selection rate" in the WCL2R data set

Tables X and XIII present the performance of the baseline ranking methods based on the precision and NDCG criteria, respectively. It is noticeable that the reported performance of baseline methods and those of the proposed algorithm are based on the average of performance of five folds of the testing data Table X.

Table XI shows the detail settings of different implementations of the QRC-Rank used during its evaluation on the LETOR4.0 data set. For the *QRC.L3* setting, the Optimistic Initial Values technique is used, which lets the RL method to do an exhaustive exploration on possible actions in each state (Sutton and Barto, 1998).

In Table XII, precision of different configurations of the QRC-Rank is reported. It could be observed that the proposed algorithm outperforms baseline methods based on the precision measure. In comparison with the best baseline method, AdaRank-NDCG, the proposed algorithm has achieved an improvement of about 8.56 per cent based on the *MAP* criterion. This improvement is about 11.39 per cent compared with the RankSVM-Struct method. However, on the *P@n* measure, sometimes baseline methods have shown better performance than those of the proposed algorithm.

Figure 7 depicts the statistics presented in Table XII. As it can be observed, the proposed method was the fourth-best method in the *P@1* measure, but it reached the second the best at *P@2* by a negligible difference with top performer. However, after P@2 QRC-Rank has outperformed the other ranking methods. Moreover, the slope of degrading precision is smaller for QRC-Rank which means even in lower ranks, it is much better than the others.

Tables XIII and XIV provides the comparison of the QRC-Rank method in different settings based on the NCDG measure. As it can be seen in the table, the QRC-Rank's performance is slightly lower but comparable to those of baseline methods. This

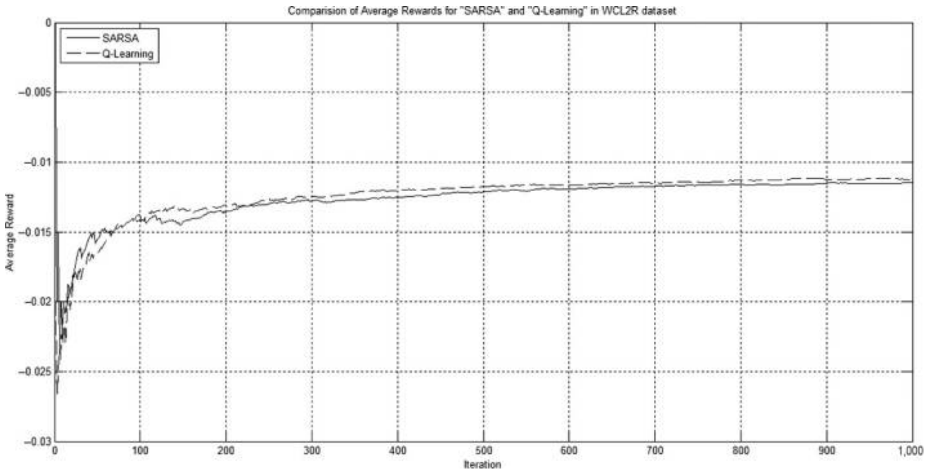

Figure 6. Comparison of the performance of SARSA and Q-Learning versions of QRC-Rank based on the "Average received rewards" in the WCL2R data set

Table X. Performance of baseline methods based on the precision criterion on the LETOR4.0 data set

Baseline method	P@1	P@2	P@3	P@4	P@5	P@6	P@7	P@8	P@9	P@10	MAP
AdaRank-MAP	0.443	0.417	0.390	0.368	0.345	0.322	0.299	0.280	0.262	0.245	0.476
AdaRank-NDCG	0.452	*0.422*	*0.395*	*0.370*	0.345	0.323	0.299	0.280	0.262	0.245	*0.482*
ListNet	0.445	0.412	0.384	0.365	0.343	0.320	0.301	0.279	0.263	0.248	0.478
RankBoost	*0.458*	0.411	0.392	0.364	0.340	0.321	*0.302*	*0.285*	*0.265*	*0.249*	0.478
RankSVM-Struct	0.427	0.407	0.390	*0.370*	*0.347*	*0.327*	*0.302*	*0.282*	*0.265*	*0.249*	0.470

Table XI. Configurations of the proposed method used during evaluation on the LETOR4.0 data set

QRC-Rank configuration	Method	Scenario	Parameters				No. of iterations	Episode length
			q_0	α	γ	ε		
QRC.L1	Q-Learning	LETOR4-DF1	100	1/iteration	0.1	Softmax,τ: 10	1000	100
QRC.L2	Q-Learning	LETOR4-DF2	100	1/iteration	0.01	0.1	1000	100
QRC.L3	Q-Learning	LETOR4-DF3	1E+10	1/iteration	0.1	0.1	1000	100

situation is mainly because of the absence of explicit click-through features in the LETOR4.0 data set. However, the drop in the performance is not alarming.

Figures 8 and 9 respectively depict the "Optimal Action Selection Rate" and "Average Received Rewards" per different iterations on using SARSA and Q-Learning methods in the implementation of the QRC-Rank on the LETOR4.0 data set. Based on these diagrams, both RL methods have shown similar performance in the rate of selecting best the action per iteration, as well as those of the average received rewards.

In this investigation, the elapsed time for SARSA was 30.50 s, but the same value is 31.91 s for the Q-Learning method.

5.3 Analytical discussion

As it was observed in the previous subsections, according to the MAP and NDCG criteria, the proposed method either outperforms baseline ranking methods or shows a very close performance in comparison with the well-known ranking methods. A closer look shows that the usage of the proposed click-through features have had a decisive role in the performance of the proposed ranking algorithm. In this regard, the informativeness of the proposed click through feature that make up the scenarios and act as a compact representation of the click-through features are compared with the original features in both WCL2R and LETOR4.0 data sets. Figures 10 and 11 show these comparisons on the WCL2R data set based on MAP and MeanNDCG criteria, respectively. In these figures, proposed click-through features used in the *QRC.W3* configuration are compared with the best feature of the WCL2R data set, F22 "Number of Sessions Clicked" (Appendix 1). F22 has the highest contribution to the ranking based on the MAP criteria among all original features in WCL2R data set.

Table XII. Performance of different variants of the proposed method based on the precision evaluation criterion on the LETOR4.0 data set

QRC-Rank configuration	P@1	P@2	P@3	P@4	P@5	P@6	P@7	P@8	P@9	P@10	MAP
QRC.L1	0.4233	0.3997	0.3761	0.3623	0.3506	0.3418	0.3329	0.323	0.3073	0.2953	0.49423
QRC.L2	0.4322	0.406	0.3807	0.3652	0.3564	0.3471	0.3383	0.3279	0.313	0.2999	0.49767
QRC.L3	*0.4437*	*0.4188*	*0.4058*	*0.3885*	*0.3794*	*0.3659*	*0.3529*	*0.3424*	*0.3256*	*0.3122*	*0.52352*

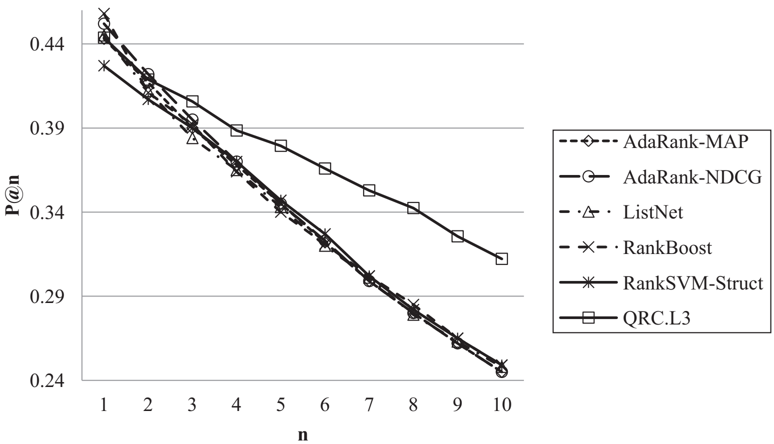

Figure 7. Comparison of the best configuration of the proposed algorithm against baseline methods on *P@n* criterion on the LETOR data set

Table XIII.
Performance of baseline methods based on the NDCG criterion on the LETOR4.0 data set

Baseline methods	NDCG @1	NDCG @2	NDCG @3	NDCG @4	NDCG @5	NDCG @6	NDCG @7	NDCG @8	NDCG @9	NDCG @10	Mean NDCG
AdaRank-MAP	0.375	0.414	*0.437*	0.461	0.479	0.492	0.497	0.461	0.225	0.229	0.492
AdaRank-NDCG	0.383	*0.421*	0.442	*0.465*	*0.482*	*0.495*	*0.499*	*0.464*	*0.227*	*0.231*	*0.495*
ListNet	0.375	0.411	0.432	0.457	0.475	0.489	0.498	0.463	*0.227*	0.230	0.491
RankBoost	*0.386*	0.399	0.429	0.448	0.467	0.482	0.490	0.457	0.221	0.226	0.485
RankSVM-Struct	0.363	0.398	0.429	0.451	0.470	0.485	0.491	0.456	0.224	0.228	0.483

QRC-Rank configuration	NDCG @1	NDCG @2	NDCG @3	NDCG @4	NDCG @5	NDCG @6	NDCG @7	NDCG @8	NDCG @9	NDCG @10	Mean NDCG
QRC.L1	0.3637	0.3888	0.4057	0.4256	0.4464	0.4641	0.4773	0.4497	0.2423	0.2464	0.4675
QRC.L2	0.3795	0.4027	0.4152	0.4337	0.4564	0.4741	0.4886	0.4601	0.2463	0.2507	0.4785
QRC.L3	*0.3816*	*0.4038*	*0.4311*	*0.4503*	*0.4742*	*0.4893*	*0.501*	*0.473*	*0.2628*	*0.2679*	*0.4917*

Table XIV.
Performance of different variants of the proposed method based on the NDCG evaluation criterion on the LETOR4.0 data set

The same analysis is repeated on the LETRO4.0 data set, and its results are depicted in Figures 12 and 13. In these figures, features of the QRC.L3 configuration are compared with F39 "LMIR.DIR of whole document" (Appendix 2) which is the best original contributing feature on the LETOR4.0 data set to ranking based on MAP criteria.

Based on the above statistics, some of the proposed click-through features are more informative than the original features. As seen in the figures, proposed click-through features related to the click-related category are more informative because they have higher MAP and MeanNDCG values. This phenomenon confirms that click-through data are useful in the learning to rank process (Macdonald and Ounis, 2009). To sum up, the results of this analysis clearly show that proposed click-through features together when combined in scenarios are more informative than the original features. These proposed click-through features are working well with the explorative and exploitative capabilities of the RL methods in finding the suitable rankings. This combination has

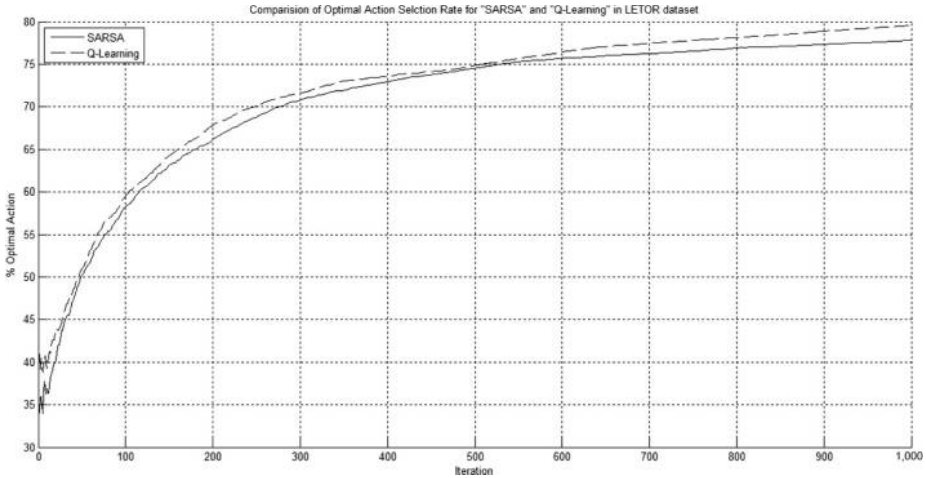

Figure 8.
Comparison of the performance of SARSA and Q-Learning methods based on the "Optimal action selection rate" on the LETOR4.0 data set

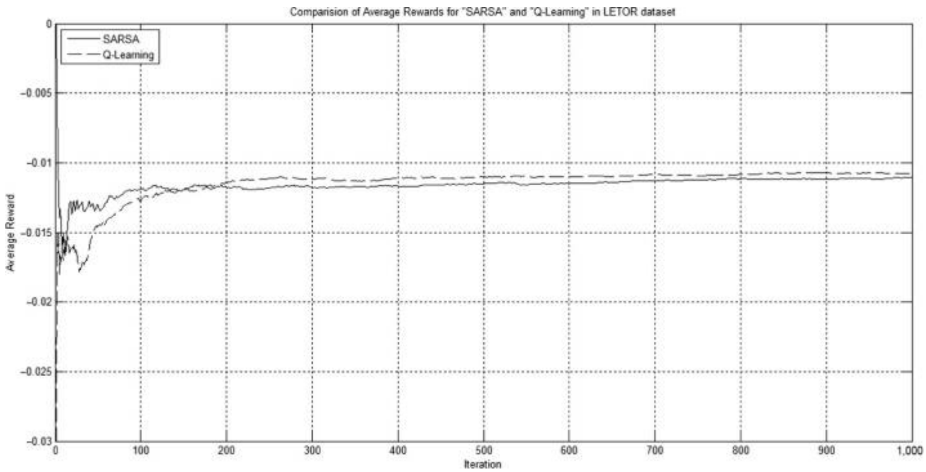

Figure 9.
Comparison of the performance of SARSA and Q-Learning methods based on the "Average received rewards" on the LETOR4.0 data set

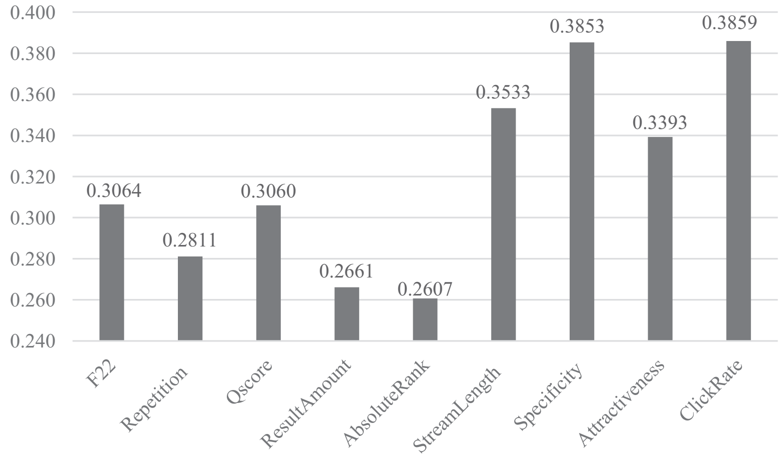

Figure 10. Comparison primitive and click-through features based on the MAP measure on WCL2R data set

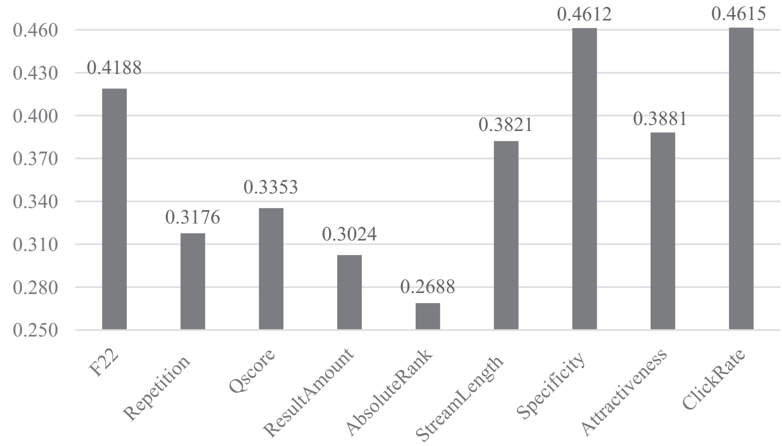

Figure 11. Comparison primitive and click-through features based on the MeanNDCG measure on WCL2R data set

resulted in the higher performance of the proposed QRC-Rank method in comparison to those of the baseline ranking methods.

As it is observed in Section 4, the set of the proposed click-through features is fixed for all benchmark data sets. Nevertheless, the calculation scenarios of these features may vary depending on the data provided by the utilized benchmark data set. Comparing the set of primitive features used in the calculation of click-through features on WCL2R and LETOR4.0 data sets shows that although there are some common features such as TF-IDF, Document Length, PageRank and BM25 between successful scenarios on these data sets, but there are also some major differences. For WCL2R, effective scenarios have used some click-through data besides HITS-related features. Conversely, successful scenarios on the LETOR4.0 data set have used Language Model features, as well as some structural features such as In-link number, number of slashes in a URL and length of the URL. Table XV provides a listing of the set of primitive

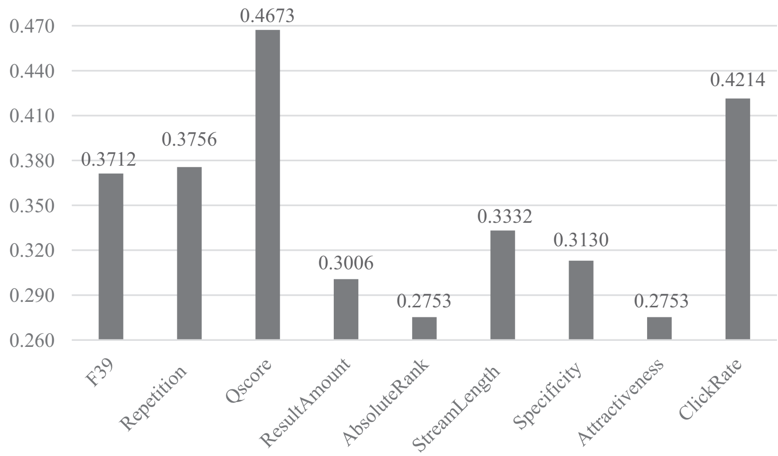

Figure 12. Comparison primitive and click-through features based on the MAP measure on LETOR4.0 data set

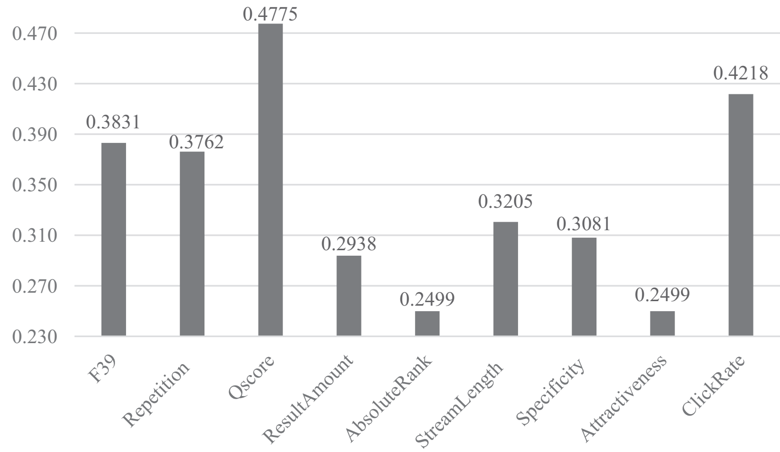

Figure 13. Comparison primitive and click-through features based on the MeanNDCG measure on LETOR4.0 data set

features that have been used in suitable scenarios for the calculation of click-through features on WCL2R and LETOR4.0 data sets.

Analysis of the proposed method on WCL2R and LETOR4.0 data sets indicates that suitable configurations of the proposed ranking method on these data sets are almost the same. Specifically, on the LETOR4.0 data set, by using optimistic initial mechanism for the initialization of the state-action values, [$Q(s,a)$], better results are achieved. This is mainly because of the availability of fewer relevant documents per any given query in the LETOR4.0 data set compared with the WCL2R data set. In this situation, by using the optimistic initial values mechanism on the LETOR4.0 data set, the RL agent has the chance to explore all of the possible actions in each state to identify the most appropriate one. It is also observed that for the WCL2R data set, usage of the Softmax technique as the action-selection policy is effective. In comparison, on the LETOR4.0 data set, exploration with the ε-greedy mechanism is more useful. This observation could also be interpreted using the nature of the investigated data sets. In the Softmax policy, the

WCL2R		LETOR4.0	
Feature ID	Description	Feature ID	Description
F03	TF-IDF (term frequency × inverse document frequency)	F15	TF (term frequency) × IDF (inverse document frequency) of whole document
F10	DL (document length)	F20	DL (document length) of whole document
F13	BM25	F37	BM25 of whole document
F14	PageRank	F38	LMIR.ABS of whole document
F15	HITS hub	F39	LMIR.DIR of whole document
F16	HITS authority	F40	LMIR.JM of whole document
F17	First of session	F41	PageRank
F23	Number of queries clicked	F42	In-link number
F24	Number of single clicks in distinct sessions	F44	Number of slash in URL
		F45	Length of URL

Table XV. Set of primitive features used in the calculation of click-through features on WCL2R and LETOR4.0 data sets

probability of selecting different possible actions is related to their estimated goodness, which is embedded in their $Q(s,a)$ values. On the other hand, ε-greedy provides no discrimination between non-optimal possible actions. In fact, while dealing with the LETOR4.0 data set, the RL agent examines all of the so far identified as actions for finding better ones during the learning process.

6. Concluding remarks and further works

Machine learning has been applied successfully to the field of IR. These learning to rank algorithms are exhaustively dependent of the benchmark data sets. However, there are some limitations with the available benchmark data sets. The main restriction is originated from the lack of click-through data, which is the implicit feedback of users about the retrieval performance of Web search engines. Besides, the high dimensionality of data items in the benchmark data sets adds to the complexity and probably the inefficiency. In this paper, a novel ranking algorithm named QRC-Rank is introduced. QRC-Rank works both data sets that contain click-through information and those that lack such information. QRC-Rank is a two phase retrieval system. In the first phase, it processes the data set and generates a new data set that contains additional more complex click-through information. The new click-through features reduce the high dimensionality of search space because there are only eight such features are calculated. Second, under scenarios these features are combined with each other to create a compact representation. In this way, the proposed method can build click-through features even when those informations are not explicitly present in the data set. The compactness of the new secondary data set reduces the complexity of developing ranking functions. Thereafter, the QRC-Rank algorithm builds a RL model based on these compact representations of features. In this model, the RL agent tries to find the best appropriate label for a given state, which corresponds to a visited query-document pair. Evaluation of the proposed method based on the *P@n*, *MAP* and *NDCG* criteria on WCL2R and

LETOR4.0 data sets demonstrate that QRC-Rank is able to significantly outperform well-known ranking algorithms if click-through data is available in the data set. The performance of the proposed algorithm is comparable with the baseline ranking methods even in absence of click-through data (i.e. LETOR4.0 data set).

This research could be extended by applying information fusion techniques such as ordered-weighted averaging in the calculation of scenarios based on the click-through features. It would also be helpful if it would be possible to find ways to deal with the inherit uncertainty and ambiguous of the relevance judgments provided by humans. Perhaps methods of handling the uncertainty such as Dempster-Shafer theory (Shafer, 1976) and fuzzy integral operators (Grabisch, 1995) may be useful. In the meantime, one can also look at generating other types of features or scenarios for the data set.

References

Agichtein, E., Brill, E. and Dumais, S. (2006a), "Improving web search ranking by incorporating user behavior information", *International ACM SIGIR Conference on Research & Development of Information Retrieval*, Washington, DC, pp. 19-26.

Agichtein, E., Brill, E., Dumais, S. and Ragno, R. (2006b), "Learning user interaction models for predicting web search result preferences", *29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Washington, DC, pp. 3-10.

Alcantara, O.D.A., Pereira, A.R. Jr, de Almeida, H.M., Goncalves, M.A., Middleton, C. and Baeza-Yates, R. (2010), "WCL2R: a benchmark collection for learning to rank research with click through data", *Journal of Information and Data Management*, Vol. 1 No. 3, pp. 551-566.

Almeida, H.M., Gonçalves, M.A., Cristo, M. and Calado, P. (2007), "A combined component approach for finding collection-adapted ranking functions based on genetic programming", *30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, New York, NY, pp. 399-406.

Bai, Y., Yang, K., Yu, W., Ma, W.Y. and Zhao, T. (2013), "Learning high-level image representation for image retrieval via multi-task dnn using clickthrough data", arXiv:1312.4740 [cs.CV].

Cao, Z., Qin, T., Liu, T.Y., Tsai, M.F. and Li, H. (2007), "Learning to rank: from pairwise approach to list wise approach", *24th International Conference on Machine Learning, Taipei*, pp. 129-136.

Derhami, V., Khodadadian, E., Ghasemzadeh, M. and Zareh Bidoki, A.M. (2013), "Applying reinforcement learning for web pages ranking algorithms", *Applied Soft Computing*, Vol. 13 No. 4, pp. 1686-1692.

Dou, Z., Song, R., Yuan, X. and Wen, J.R. (2008), "Are click-through data adequate for learning web search rankings?", *17th ACM Conference on Information and Knowledge Management*, pp. 73-82.

Dupret, G. and Liao, C. (2010), "A model to estimate intrinsic document relevance from the click-through logs of a web search engine", *Third ACM International Conference on Web Search and Data Mining*, pp. 181-190.

Freund, Y., Iyer, R., Schapire, R.E. and Singer, Y. (2003), "An efficient boosting algorithm for combining preferences", *Journal of Machine Learning Research*, Vol. 4 No. 1, pp. 933-969.

Gao, J., Nie, J.Y., Yuan, W., Li, X. and Deng, K. (2009), "Smoothing clickthrough data for web search ranking", *32nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 355-362.

Google (1998), "Google search engine", available at: www.google.com (accessed 20 December 2015).

Grabisch, M. (1995), "Fuzzy integral in multicriteria decision making", *Fuzzy Sets and Systems*, Vol. 69 No. 3, pp. 279-298.

Granka, L.A., Joachims, T. and Gay, G. (2004), "Eye-tracking analysis of user behavior in WWW search", *27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 478-479.

Herbrich, R., Graepel, T. and Obermayer, K. (2000), "Large margin rank boundaries for ordinal regression", in Smola, A.J., Bartlett, P., Schölkopf, B. and Schuurmans, D. (Eds), *Advances in Large Margin Classifiers*, MIT Press, Cambridge.

Hofmann, K., Schuth, A., Whiteson, S. and Rijke, M.D. (2013), "Reusing historical interaction data for faster online learning to rank for IR", *Sixth ACM International Conference on Web Search and Data Mining*, pp. 183-192.

Huang, P.S., He, X., Gao, J., Deng, L., Acero, A. and Heck, L. (2013), "Learning deep structured semantic models for web search using click-through data", *22nd ACM International Conference on Information and Knowledge Management*, pp. 2333-2338.

Jain, V. and Varma, M. (2011), "Learning to re-rank: query-dependent image re-ranking using click data", *20th international conference on World Wide Web*, pp. 277-286.

Ji, S., Xue, G.R., Zhou, K., Chapelle, O., Sun, G., Liao, C., Zheng, Z. and Zha, H. (2009), "Global ranking by exploiting user clicks", *32nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 35-42.

Joachims, T. (2002), "Optimizing search engines using clickthrough data", *8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 133-142.

Joachims, T. (2006), "Training linear SVMs in linear time", *12th International Conference on Knowledge Discovery and Data Mining*, pp. 217-226.

Kendall, M.G. (1948), *Rank Correlation Methods*, Oxford University Press, London.

Keyhanipour, A.H., Moshiri, B., Piroozmand, M. and Lucas, C. (2007), "Aggregation of multiple search engines based on users' preferences in WebFusion", *Knowledge-Based Systems*, Vol. 20 No. 4, pp. 321-328.

LETOR4.0 Datasets (2009), "LETOR4.0 datasets", available at: http://research.microsoft.com/en-us/um/beijing/projects/letor/letor4dataset.aspx (accessed 20 December 2015).

LETOR4.0's Features List (2009), "LETOR4.0's features list", available at: http://research.microsoft.com/en-us/um/beijing/projects/letor/LETOR4.0/Data/Features_in_LETOR4.pdf (accessed 20 December 2015).

Macdonald, C. and Ounis, I. (2009), "Usefulness of quality click-through data for training", 2009 Workshop on Web Search Click Data, pp. 75-79.

Macdonald, C., Santos, R.L.T. and Ounis, I. (2013), "The whens and hows of learning to rank for web search", *Information Retrieval*, Vol. 16 No. 5, pp. 584-628.

Ma, H., Yang, H., King, I. and Lyu, M.R. (2008), "Learning latent semantic relations from click-through data for query suggestion", *17th ACM Conference on Information and Knowledge Management*, pp. 709-718.

Manning, C.D., Raghavan, P. and Schütze, H. (2008), *An Introduction to Information Retrieval*, Cambridge University Press, Cambridge.

Miller, M. (2012), "53% of Organic search clicks go to first link", available at: http://searchenginewatch.com/article/2215868/53-of-Organic-Search-Clicks-Go-to-First-Link-Study (accessed 20 December 2015).

Qin, T., Liu, T.Y., Xu, J. and Li, H. (2007), "LETOR: benchmark dataset for research on learning to rank for information retrieval", *SIGIR 2007 Workshop on Learning to Rank for Information Retrieval*, pp. 3-10.

Radlinski, F. and Joachims, T. (2005), "Query chains: learning to rank from implicit feedback", *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 239-248.

Radlinski, F. and Joachims, T. (2007), "Active exploration for learning rankings from click-through data", *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 570-579.

Shafer, G. (1976), *A Mathematical Theory of Evidence*, Princeton University Press, Princeton.

Sutton, R.S. and Barto, A.G. (1998), *Reinforcement Learning: An Introduction*, MIT Press, Cambridge.

Szepesvari, C. (2010), *Algorithms for Reinforcement Learning*, Morgan & Claypool, San Rafael, CA.

TodoCL (2002), "TodoCL search engine website", available at: www.todocl.cl (accessed 20 December 2015).

Veloso, A.A., Almeida, H.M., Goçalves, M.A. and Meira, M.J. (2008), "Learning to rank at query-time using association rules", *31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 267-274.

Wang, H., Zhai, C., Liang, F., Dong, A. and Chang, Y. (2014), "User modeling in search logs via a nonparametric bayesian approach", *7th ACM International Conference on Web Search and Data Mining*, pp. 203-212.

WCL2R (2010), "WCL2R website", available at: www.latin.dcc.ufmg.br/collections/wcl2r (accessed 20 December 2015).

Xu, J., Chen, C., Xu, G., Li, H. and Torres Abib, E.R. (2010), "Improving quality of training data for learning to rank using click-through data", *Third ACM International Conference on Web Search and Data Mining*, pp. 171-180.

Xu, J. and Li, H. (2007), "Adarank: a boosting algorithm for information retrieval", *30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 391-398.

Xu, Q. (2010), "Learning to rank from clickthrough data", available at: http://isabel-drost.de/projects/tuberlin/imsem2010/learningtorank_paper_2010.pdf (accessed 20 December 2015).

Zareh Bidoki, A.M., Ghodsnia, P., Yazdani, N. and 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.

Zhai, C. and Lafferty, J. (2001), "A study of smoothing methods for language models applied to Ad Hoc information retrieval", *24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 334-342.

Corresponding author
Amir Hosein Keyhanipour can be contacted at: keyhanip@ut.ac.ir

Appendix 1

Feature ID	Feature name	Feature type
F1	TF	Standard features
F2	IDF	
F3	TF-IDF (term frequency × inverse document frequency)	
F4	TF (term frequency) of title	
F5	IDF (inverse document frequency) of title	
F6	TF-IDF (term frequency × inverse document frequency) of title	
F7	TF (term frequency) of URL	
F8	IDF (inverse document frequency) of URL	
F9	TF-IDF (term frequency × inverse document frequency) of URL	
F10	DL (document length)	
F11	DL (document length) of title	
F12	DL (document length) of URL	
F13	BM25	
F14	PageRank	
F15	HITS hub	
F16	HITS authority	
F17	First of session	Click-through features
F18	Last of session	
F19	Number of clicks in a document for a query	
F20	Number of sessions a document was clicked for a query	
F21	Number of clicks	
F22	Number of sessions clicked	
F23	Number of queries clicked	
F24	Number of single clicks in distinct sessions	
F25	Number of single clicks in distinct queries	
F26	Absolute number of single clicks in queries	
F27	Number of single clicks in queries grouped by session	
F28	Number of non-single click sessions	
F29	Number of non-single click queries	

Table AI. List of features in the WCL2R benchmark data set

Appendix 2

Feature ID	Feature name
F1	TF (term frequency) of body
F2	TF (term frequency) of anchor
F3	TF (term frequency) of title
F4	TF (term frequency) of URL
F5	TF (term frequency) of whole document
F6	IDF (inverse document frequency) of body
F7	IDF (Inverse document frequency) of anchor
F8	IDF (inverse document frequency) of title
F9	IDF (inverse document frequency) of URL
F10	IDF (inverse document frequency) of whole document
F11	TF (term frequency) × IDF (inverse document frequency) of body
F12	TF (term frequency) × IDF (inverse document frequency) of anchor
F13	TF (term frequency) × IDF (inverse document frequency) of title
F14	TF (term frequency) × IDF (inverse document frequency) of URL
F15	TF (term frequency) × IDF (inverse document frequency) of whole document
F16	DL (document length) of body
F17	DL (document length) of anchor
F18	DL (document length) of title
F19	DL (document length) of URL
F20	DL (document length) of whole document
F21	BM25 of body
F22	LMIR.ABS of body
F23	LMIR.DIR of body
F24	LMIR.JM of body
F25	BM25 of anchor
F26	LMIR.ABS of anchor
F27	LMIR.DIR of anchor
F28	LMIR.JM of anchor
F29	BM25 of title
F30	LMIR.ABS of title
F31	LMIR.DIR of title
F32	LMIR.JM of title
F33	BM25 of URL
F34	LMIR.ABS of URL
F35	LMIR.DIR of URL
F36	LMIR.JM of URL
F37	BM25 of whole document
F38	LMIR.ABS of whole document
F39	LMIR.DIR of whole document
F40	LMIR.JM of whole document
F41	PageRank
F42	In-link number
F43	Out-link number
F44	Number of slash in URL
F45	Length of URL
F46	Number of child page

Table AII. List of features in the LETOR4.0 benchmark data set