Int. J. Mach. Learn. & Cyber. (2016) 7:1131–1145
DOI 10.1007/s13042-015-0442-6

ORIGINAL ARTICLE

Integration of data fusion and reinforcement learning techniques for the rank-aggregation problem

Amir Hosein Keyhanipour[1] · Behzad Moshiri[1] · Masoud Rahgozar[1] · Farhad Oroumchian[2] · Ali Asghar Ansari[1]

Received: 28 January 2015 / Accepted: 8 October 2015 / Published online: 20 October 2015

Abstract Rank-aggregation or combining multiple ranked lists is the heart of meta-search engines in web information retrieval. In this paper, a novel rank-aggregation method is proposed, which utilizes both data fusion operators and reinforcement learning algorithms. Such integration enables us to use the compactness property of data fusion methods as well as the exploration and exploitation capabilities of reinforcement learning techniques. The proposed algorithm is a two-steps process. In the first step, ranked lists of local rankers are combined based on their mean average precisions with a variety of data fusion operators such as optimistic and pessimistic ordered weighted averaging (OWA) operators. This aggregation provides a compact representation of the utilized benchmark dataset. In the second step, a Markov decision process (MDP) model is defined for the aggregated data. This MDP enables us to apply reinforcement learning techniques such as Q-learning and SARSA for learning the best ranking. Experimentations on the LETOR4.0 benchmark dataset demonstrates that the proposed method outperforms baseline rank-aggregation methods such as Borda Count and the family of coset-permutation distance based stage-wise (CPS) rank-aggregation methods on P@n and NDCG@n evaluation criteria. The achieved improvement is especially more noticeable in the higher ranks in the final ranked list, which is usually more attractive to Web users.

Keywords Rank-aggregation · Data fusion · Reinforcement learning · Markov decision process

✉ Amir Hosein Keyhanipour
keyhanip@ut.ac.ir

Behzad Moshiri
moshiri@ut.ac.ir

Masoud Rahgozar
rahgozar@ut.ac.ir

Farhad Oroumchian
farhadoroumchian@uowdubai.ac.ae

Ali Asghar Ansari
ansari@ut.ac.ir

[1] Control & Intelligent Processing, Center of Excellence, School of ECE, University of Tehran, No. 5, Zolfaghary Alley, Shadmehr Street, SattrKhan Avenue, Tehran, Iran

[2] Faculty of Engineering & Information Sciences, University of Wollongong in Dubai, Dubai, UAE

1 Introduction

It is estimated that by the end of December 2014, about 60 billion Web pages are indexed by major Web search engines such as Google, Yahoo and Bing [37]. For the moment, Web retrieval systems, especially Web search engines are main access points for Web users to this vast ocean of information. Web information retrieval has become a crucial and challenging task. Based on the available statistics, almost all of Web users use Web search engines in order to locate their required data and services [14]. Also, most of the traffic of Web sites is originated from Web search engines [41]. On the other hand, most of the users of search engines, visit only the top items of the results pages of their queries [11, 25]. In such a circumstance, one may consider ranking as the heart of search engines. Although significant research works have been accomplished on the ranking problem, there is still considerable room left for improvements.

This paper is devoted to a special kind of ranking method which is called rank-aggregation. Rank-aggregation is extensively used in Web meta-search engines as well as in many other decision making tasks. The aim of the rank-aggregation method is to combine a set of ranked lists provided by local rankers based on their own ranking strategies and provide a more precise ranking of the underlying data items. These ranked lists might be noisy, incomplete or even disjoint. Obviously, it is required to aggregate such locally ranked lists in order to achieve more precise view of the appropriate data. By having such a general definition of the rank-aggregation problem, the task of ranking could also be considered as an aggregation of some local rankings. Especially, such a definition is useful in the context of learning to rank which is a promising solution for the ranking problem [20, 21].

In the rank-aggregation problem, we encounter with the dynamic and sophisticated information needs of Web users. On the other hand, as the quality of Web information resources are constantly changing, local rankers need to use complicated ranking mechanisms. Moreover, due to the commercial competitions of Web search engines, their ranking algorithms are unknown to rank-aggregation systems. This dynamic and complex environment has motivated us to look at the problem as a learning problem, where the exhaustive exploration–exploitation capability of methods such as reinforcement learning could be useful in finding near-optimal solutions. This paper proposes a novel rank-aggregation method with two sequential steps. In the first step, for each query, ranked lists of local rankers are aggregated based on their mean average precisions (MAP) values by a variety of data fusion operators. This step produces a compact representation for the benchmark dataset. Then, in the second step, a Markov decision process (MDP) model is learned by a reinforcement learning (RL) method. This model assigns the best relevance label to each query-document pair based on similarly viewed items.

Major contributions of this research could be listed as:

- Proposition of a novel rank-aggregation method by combining information fusion theory and reinforcement learning techniques.
- Combination of decisions of local rankers based on their MAPs by a variety of data fusion operators especially the family of exponential ordered weighted averaging (OWA) operators.
- Designation of a reinforcement learning model for the rank-aggregation problem and application of temporal-difference (TD) learning methods as the solution.

The rest of this paper is organized as follows: Sect. 2 provides a brief overview of rank-aggregation methods. In Sect. 3, fundamental ideas of the proposed rank-aggregation method and a review of utilized techniques will be described. Then in Sect. 4, the evaluation of the proposed method as well as the analytical discussion about the results will be presented; and finally, Sect. 5 concludes the research work and gives some further research directions.

1.1 Notation

For the abbreviation, in this paper we will use the following acronyms; ‘WA’ for weighted averaging, ‘WMV’ for weighted majority voting, ‘OWA_Opt’ for optimistic version of ordered weighted averaging and ‘OWA_Pess’ for the pessimistic ordered weighted averaging. These data fusion techniques are described in Sect. 3.

2 Survey on related works

The rank-aggregation problem broadly refers to a variety of techniques in which different ranking lists received from multiple ranking systems are combined in order to obtain a better ranking list. The problem of rank-aggregation has its roots in the social fair election demand in the eighteenth century [13]. However, in recent decades, rank-aggregation techniques have been applied in various applications including: Bioinformatics [18, 30], Meta-search engines [6, 9, 16, 19, 28], and Web spam detection [3, 32].

In the context of Web information retrieval, rank-aggregation could be considered from two points of view: score-based and order-based. Algorithms of the former category, take as input, scores assigned to data items by local rankers, while in the latter category, orders of data items are used as input. Although score-based methods have shown better performance, but lack of score within local lists in real-world situation, has made these methods ineffective in practice [28].

A number of score-based rank-aggregation techniques have been proposed such as [4, 35] and coset-permutation distance based stagewise (CPS) [27]. CPS is a probabilistic model over permutations, which is defined with a coset-permutation distance and models the generation of a permutation as a stage-wise process.

The order-based methods exploit only relative positions of data items in the ranked lists to perform the aggregation, thus they are also called positional methods. Borda Count [29] is a notable example of such algorithms. The final rank of a given data item in Borda Count is related to the number of items, which locate below that item in all locally ranked lists [7]. Another example of order-based method is the Median rank-aggregation, in which the final position of each item is calculated as the median of occurrence positions of that item in all the local lists [8]. Another approach is taken by assuming the existence of a Markov chain on items. In this model the Markov transition model learns and predicts the order of data items in the ranked lists [6].

Dwork et al. have experimented with a number of methods such as MC1, MC2, MC3 and MC4 for computing the transition matrix for the assumed Markov chain [6].

Most of the above mentioned algorithms implicitly assume an equal weight for all of local rankers. Obviously, such assumption may not be realistic. In order to distinguish between locally sorted lists, authors in [2] have introduced a Borda Fuse approach distinguishes local rankers by their MAP values. Their evaluations showed that Borda Fuse outperforms Borda Count. Authors of [36] have also used OWA weights in order to aggregate preference rankings. Their approach allows the weights associated with different ranking places to be determined in terms of a decision maker's optimism level characterized by an Orness degree. In [40], OWA operators are used in the multi-criteria decision making problem with multiple priorities, in which priority weights associated with the lower-priority criteria are related to the satisfactions of the higher-priority criteria. Another recent related work is [17], in which, both Optimistic and Pessimistic OWA operators are used for the aggregation of ranking of the preferences lists. A recent related research work is [12], which proposes an ensemble strategy for the local information-based link prediction algorithms using the OWA operator.

In recent years, it is observed that in order to improve the accuracy of rank aggregation, it is better to employ a supervised learning approach [22], in which an order-based aggregation function is learned within an optimization framework using labeled data. Examples of these methods are [5, 22]. Another noticeable supervised rank-aggregation method is QuadRank, which takes into account the metadata accompanying each item such as title, snippet and URL, in addition to their order in local lists [1].

Our proposed technique differs from the above mentioned approaches because it combines the data fusion and reinforcement learning in the context of rank-aggregation. Considering the dynamic nature of Web information resources as well as the constantly changing users' information needs, the behavior and quality of the local rankers could be completely dynamic over time. The proposed rank-aggregation method, takes advantage of a merit-based aggregation of the decision of the local rankers based on their MAP values, which indicates their average precision in local ranked lists. The proposed method introduces a Markov decision process model for the rank-aggregation problem and applies reinforcement learning methods in order to learn the above mentioned dynamism of the Web environment. Experimental results show that the proposed approach outperforms baseline algorithms, especially in the high ranks, which are the most attractive part of the ranked lists for Web users.

3 Fundamentals of the proposed approach

In this section, after providing a formal definition of the rank-aggregation problem, an outline of the proposed method will be presented along with its information fusion operators and the reinforcement learning framework. Afterwards the two steps of the proposed rank-aggregation method will be described in detail.

3.1 Formal definition of the rank-aggregation

As mentioned earlier, ranking is the heart of the web information retrieval systems. Recently, a new area called learning to rank has emerged in the intersection of machine learning, information retrieval, and natural language processing. We may consider learning to rank as the application of machine learning technologies in the ranking problem. In the learning to rank, the input space X consists of lists of feature vectors and the output space Y contains a list of grades (relevance labels). Assuming x is a member of X which represents a list of feature vectors and y is an element of Y, which represents a list of grades, $F(.)$ is a ranking function that maps a list of feature vectors x to a list of scores y. Having m training instances such as: $(\mathrm{x}_1, \mathrm{y}_1), (\mathrm{x}_2, \mathrm{y}_2), \ldots, (\mathrm{x}_m, \mathrm{y}_m)$; the goal of learning to rank is to automatically learn an acceptable approximation $G(.)$ of the ranking function $F(.)$ [20].

In a general view, learning to rank methods could be categorized into two different categories: rank-aggregation and rank-creation algorithms. Rank-aggregation is actually a process of combining multiple ranking lists into a single ranking list, which is better than any of the original ranking lists. The rank-creation algorithm generates ranking based on features of queries and documents [20].

This research deals with the rank-aggregation problem. The rank-aggregation problem can be defined as following: Given a set of entities S, let: $V \subseteq S$ and assume that there is a total order among the entities in V. τ is called a *ranking list* with respect to S, if τ is a list of the entities in V maintaining the same total order relation, i.e., $\tau = [d_1, \cdots, d_m]$, if $d_1 \geq \cdots \geq d_m$, $d_i \in$, $i = 1, \cdots, m$, where $\geq$ denotes the ordering relation and $m = |V|$ (here, we draw heavily on the notation of [22]). If V equals S, τ is called a *full list*, otherwise, it is called a *partial list*. Let $\tau_1, \cdots, \tau_l$ denote the ranking lists with respect to S and n denotes the number of entities in S. The *aggregation function* is defined as Ψ: $\tau_1, \cdots, \tau_l \mapsto x$, where x denotes the final score vector of all entities. That is, if $x = (\tau_1, \cdots, \tau_l)$, then all the entities are ranked by the scores in x. In this context, finding a suitable aggregation function is what the rank-aggregation algorithms deal with.

3.2 Outline of the proposed rank-aggregation method

As mentioned before, the proposed method consists of two consecutive steps. In the first step, information fusion techniques are used in order to provide a combination of local rankers' decisions based on their MAP values. Then, in the second step, a MDP model including States, Actions and Rewards will be built over the output of the previous phase. Figure 1 demonstrates what happens inside the proposed rank-aggregation method. Details of this process will be described in Sect. 3.5.

At the end of the learning process, the optimal action for each state is learned. In other words, based on the learned Markov model, one can identify the appropriate relevance label for any input query-document pair. In this regard, each input which is a query-document pair, is mapped into a state of the proposed MDP model. Thereafter, based on the learning process of the proposed algorithm, which is summarized in the Action-Value table, the most appropriate relevance label will be assigned to the input query-document pair. This process is illustrated in Fig. 2. In this figure, each input query-document pair is represented by a feature vector ⟨Relevance, qID, v_1, v_2, …, v_N, docID⟩, in which vi's are features extracted from the qID-docID pair.

3.3 The ordered weighted averaging

The ordered weighted operators (OWA) operators were introduced by Yager [38]. These aggregation operators are based on the arithmetic average. Due to their ability in modeling different aggregation scenarios, the family of OWA operators have been widely used in different computational intelligence applications. Formally, an OWA operator of dimension n is a mapping $f: R^n \rightarrow R$, that has an association with weighting vector $W = [w_1\, w_2 \ldots w_n]^T$ such that:

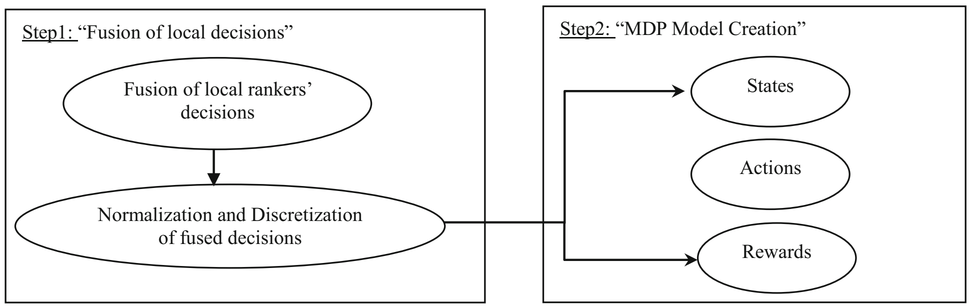
$$\sum_i w_i = 1;\ w_i \in [0,\ 1],\ \text{and}: f(a_1, \ldots, a_n) = \sum_{j=1}^{n} w_j b_j$$

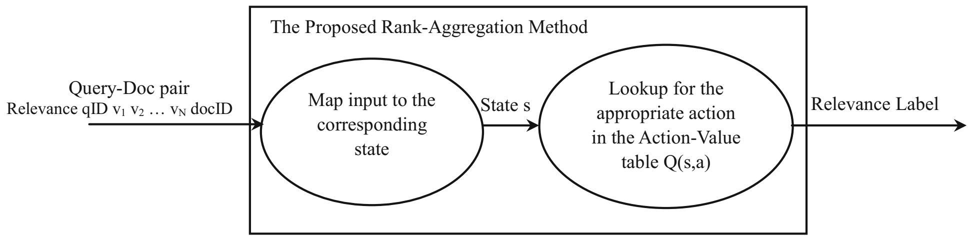
In the above, b_j is the jth largest element of the collection of n aggregate objects a_1, a_2, …, a_n. The function value $f(a_1, \ldots, a_n)$ determines the aggregated value of arguments, a_1, a_2, …, a_n. Having n weights w_1, w_2, …, w_n corresponding to decadently sorted objects b_1, b_2, …, b_n, the *Orness* is defined as follow [10]:

$$Orness(W) = \frac{1}{n-1}\sum_{i=1}^{n}(n-i)w_i$$

In fact, the *Orness* characterizes the degree to which the aggregation is like the *OR* (*Max*) operation.

One important issue in the definition of OWA aggregation operators, is how to determine the associated weights. In fact, a number of approaches have been suggested for obtaining OWA weights [39]. One of the earliest methods was introduced by O'Hagan [26]. O'Hagan's approach calculates the vector of the OWA weights for a predefined level of *Orness*. However, the computational cost of this method is high; because it involves the solution of a constrained nonlinear programming problem. Another approach for finding OWA weights is the exponential OWA operators.


Step1: "Fusion of local decisions"
Fusion of local rankers' decisions
Normalization and Discretization of fused decisions
Step2: "MDP Model Creation"
States
Actions
Rewards
Fig. 1 The process view of the proposed rank-aggregation method


The Proposed Rank-Aggregation Method
Query-Doc pair
Relevance qID v1 v2 … vN docID
Map input to the corresponding state
State s
Lookup for the appropriate action in the Action-Value table Q(s,a)
Relevance Label
Fig. 2 A black-box view of the proposed rank-aggregation method

Assuming we have n items to be aggregated, Optimistic Exponential OWA weights are computed as follow [10]:

$$w_1 = \alpha, w_2 = \alpha(1-\alpha), w_3 = \alpha(1-\alpha)^2, \ldots, w_{n-1} = \alpha(1-\alpha)^{n-2}, w_n = (1-\alpha)^{n-1} \text{ (Optimistic OWA)}$$

In the above formulae, α belongs to the unit interval, $0 \le \alpha \le 1$. It is proven that in the Optimistic Exponential OWA, the *Orness* is a monotonically increasing function of α [10]. Furthermore, they have shown that for $n = 2$, the *Orness* value of this operator is always equal to α, and for $n > 2$, it is higher than α. For this reason, the OWA operator associated with these weights is called as Optimistic Exponential OWA operator.

An alternative related OWA operator can be derived by considering the following OWA weights [10]:

$$w_1 = \alpha^{n-1}, w_2 = (1-\alpha)\alpha^{n-2}, w_3 = (1-\alpha)\alpha^{n-3}, \ldots, w_{n-1} = \alpha(1-\alpha), w_n = (1-\alpha) \text{ (Pessimistic OWA)}$$

Contrary to the previous case, for a fixed α, the *Orness* of this aggregation decreases as n increases. Furthermore, it can be shown that for $n = 2$, the *Orness* value of this aggregation is always equal to α and for any value of $n > 2$, it is lower than the value of the parameter α. Due to this situation, the OWA operator associated with these weights is named as Pessimistic Exponential OWA operator.

In this research, OWA weights are calculated based on both optimistic and pessimistic exponential OWA operators.

3.4 Reinforcement learning framework

Reinforcement learning (RL) is referred to a category of learning approaches in which the model of the environment is gradually built based on the interactions of the learning agent with the surrounding environment. During the learning process, in each state of the environment, the RL agent tries to maximize its cumulative reward by selecting the most appropriate action according to its current estimation of values of possible actions [33]. The cumulative reward is usually a numerical signal received based on the configuration of the environment. The environments which are explored by reinforcement learning methods are usually defined by a set of states, a number of actions and a reward function which maps actions that are taken in a given state into a numerical reward. Such environments need to fulfill the constraints of MDPs. An MDP environment is a discrete time stochastic process. Being in state s at a moment, the learning agent is able to select action $a \in A$, where A is the set of all possible actions. Actually, the agent selects the action which maximizes its expected reward. The MDP environment responds to agent's action at the next time step by randomly moving into a new state s' based on the state transition function $P_a(s,s')$ and gives the learning agent a corresponding reward $R_a(s,s')$. Therefore, the next state s' depends on the current state s and the selected action a, but conditionally independent of all previous states and actions [34].

SARSA and Q-learning are two most powerful reinforcement learning algorithms. Both algorithms are from the temporal-difference (TD) learning family, but they differ in the updating mechanism of their estimations on the value of possible actions in different states of the problem's environment. Briefly, Q-learning is an off-policy TD, while SARSA is an on-policy TD [33]. Assuming $Q(s,a)$ represents the estimated value of taking action a in the state s under a given policy, and state s' as the next state that the reinforcement-learning agent will change into after the state s and action a, $Q(s', a')$ is the estimation of the value of performing action a' in state s'. Based on the goodness of the value of $Q(s', a')$, the learning agent needs to update its estimation of the goodness of performing the action a in the state s *or* $Q(s', a')$. The SARSA algorithm dose this updating by:

$$Q(s,a) \leftarrow Q(s,a) + \beta\,[r + \gamma\, Q(s',a') - Q(s,a)]$$

Updating mechanism for the Q-learning algorithm is done through the following function:

$$Q(s,a) \leftarrow Q(s,a) + \beta\left[r + \gamma \max_{a'} Q(s',a') - Q(s,a)\right]$$

In the above formula, r is the reward obtained after transition from state s to s' or $R_a(s,s')$; a' is the accomplished action in state s'; β is a constant step-size parameter and γ is discount rate which is $0 \le \gamma \le 1$. Based on the above updating mechanisms, Q-learning is called an off-policy learning method, while SARSA is an on-policy one. In off-policy methods, *behavior* policy differs from the *estimation* policy, but they are same in on-policy learning methods. The policy used to generate the behavior of the RL agent, called the *behavior* policy, may in fact be unrelated to the policy that is evaluated and improved, called the *estimation* policy.

In this investigation, in order to provide the maximum of exploration, $Q(s,a)$ values are initialized according to the "Optimistic Initial Values" approach, in which initial values of $Q(s,a)$ are set to a very large positive number [33]. Whichever actions are initially selected, the reward is less than the starting estimates; thus the learner switches to other actions, being disappointed with the rewards it is receiving. Although this exploration may reduce the performance of the RL agent in the beginning of its learning process, however all actions are tried several times before the convergence of the estimated action-values [33].

3.5 Details of the proposed method

The proposed method uses the exploration and exploitation capabilities of reinforcement learning methods besides the compressing ability of data fusion models to provide a novel approach in dealing with the rank-aggregation problem. Generally, in its first step, the proposed method provides a combination of local rankers by utilizing data fusion operators such as min, max, average, weighted average, and optimistic/pessimistic exponential ordered weighted averaging. In the second step, our algorithm builds a MDP representation of the rank-aggregation task. This model is constructed by defining the set of states, possible actions and the reward function. By having such a model, the proposed method learns to assign an appropriate relevance label to each query-document pair.

Formally, by having N different local rankers such as $v_1, v_2, \ldots, v_N$ and considering if M different relevance levels could be assigned to each query-document pair, the aggregation function F is defined as:

$$F\left(d_i, q_j\right) = F\left(v_1\left(d_i, q_j\right), v_2\left(d_i, q_j\right), \ldots, v_N\left(d_i, q_j\right)\right) \tag{1}$$

According to the above equation, for any document d_i and any query q_j, the aggregation is accomplished on the votes of N local rankers on this query-document pair. In our experimentation, different aggregation functions such as min, max, average, weighted average, and optimistic/pessimistic ordered weighted averaging are investigated. In this work, for aggregation of the local rankers is accomplished according to the mean average precision (MAP) values of their relevance judgments in comparison with ground truth relevance labels of investigated documents.

Then, the min–max normalization process [23] will be done on aggregated votes.

$$F'\left(d_i, q_j\right) = \frac{F\left(d_i, q_j\right) - \min_k\left\{F\left(d_k, q_j\right)\right\}}{\max_k\left\{F\left(d_k, q_j\right)\right\} - \min_k\left\{F\left(d_k, q_j\right)\right\}} \tag{2}$$

Normalized aggregated decisions of local rankers are passed through a discretization process which is formally described in Eq. 3. Here, for each query-document pair, the normalized and aggregated votes of local rankers will fall into one of M possible relevance levels.

$$G\left(d_i, q_j\right) = \begin{cases} 0 & if \quad F'\left(d_i, q_j\right) < 1/M \\ 1 & if \quad 1/M \le F'\left(d_i, q_j\right) < 2/M \\ \ldots & \ldots \\ M-1 & , else \end{cases} \tag{3}$$

The above mentioned process forms the first step of the proposed rank-aggregation algorithm. Using the result of the first step, within the second step of the proposed algorithm, a MDP model will be suggested for the rank-aggregation problem, which includes states, possible actions and expected rewards. In this MDP model, the set of all possible states are defined as:

$$S = \{(NNV,\ G(d, q))\} \tag{4}$$

In the above formulation, NNV is the number of local voters which their votes were not null for a given query-document pair (d,q). In this definition, each state consists of two integers: the first part indicates the number of local voters, which their votes were not null for a corresponding query-document pair, and the second part, is a discretized aggregation of the votes of local rankers for the mentioned query-document pair. In this way, there may be different pairs of queries-documents, which are mapped to the same state. The mapping of different pairs of queries-documents to a single state, is the generalization property of the proposed algorithms, which makes it possible to predict suitable relevance labels for unseen query-document pairs. In fact, the proposed definition for the set of states, is a compact and lossless representation of the data associated with each pair of query-document. The intuition behind mapping query-document pairs to states is that the RL agent encounters with different states (different pairs of queries-documents) and needs to select appropriate actions (relevance labels). As it would be described later in Sect. 4, a data item of the rank-aggregation benchmark dataset, includes the truth relevance of the corresponding query-document pair and votes of different voters. The voters output NULL if the specified document does not exist in their ranked results for the given query. Regarding the probable noise in the votes of the voters, the aggregation of their votes, is used in the representation of the corresponding state.

Also, the set of all possible actions in a given state, is defined as M different relevance degrees which could be assigned to any query-document pair, which are: 0 (irrelevant), 1 (a little relevant), … and M-1 (completely relevant). The learning agent is designed to select the right action (assign the right relevance label) to each query-document pair, therefore we have:

$$A = \{0, 1, \ldots, M-1\} \tag{5}$$

In the same way, the reward function is considered as:

$$r(d, q) = -|action(d, q) - G(d, q)| \tag{6}$$

In this formulation, $action(d,q)$ stands for the relevance label which is predicted by the proposed method for a query-document pair such as (d,q). The RL agent learns to estimate the relevance label of each document query pair based on current state. The above mentioned reward function motivates the learning agent to predict the relevance label of a given query-document pair based on the aggregated votes of the local rankers. That is because it is assumed the aggregation of the votes of local rankers could reduce the possible noises in the decisions of individual rankers.

By having such a RL definition of the rank-aggregation problem, each given a query-document pair, is mapped to the set of possible states. In each state, the learning agent can select different actions, which are possible relevance labels that could be assigned to the corresponding query-document pair. By selecting an action, 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 MDP model, we can perceive that the Markov property, which is about the independence of receiving a reward at a particular state from the previous states and actions, is established [33]. 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, s_t stands for the state of the agent at time t, a_t is the accomplished action at time t and r_t is the received reward at time t by doing the action a_t at the state s_t. Since data items are selected from the training set by a uniform distribution probability, therefore the Markov property holds in the proposed MDP model. In fact, for the proposed MDP model, the state-transition model is:

$$\forall s, s' \in S,\ \forall a \in A : P_a(s, s') = \frac{1}{|S|}$$

Availability of the Markov property in the proposed MDP model lets the learning agent to utilize temporal-difference learning methods such as Q-learning and SARSA during its interactions with this environment.

As an explanatory example, suppose there are only two documents corresponding to the query 'q_1', which are listed in Table 1. In each row, the first column is the relevance label of this query-document pair, which is either 0 (non-relevant) or 1 (relevant). The second column is query ID, and following columns are ranks of the document in local lists of four different local rankers, and the document ID is at the end of the row. Also suppose that the arithmetic mean is utilized as the aggregation operator during the fusion of local rankers' votes.

Table 1 An example rank-aggregation dataset

…
0 qid:q_1 voter#1:1 voter#2:3 voter#3:8 voter#4:NULL docid = d_1
1 qid:q_1 voter#1:2 voter#2:7 voter#3:4 voter#4:1 docid = d_2
…

In this way, based on Eqs. 1–3, we have:

$$F(d_1, q_1) = Mean(1, 3, 8, 0) = 4,$$
$$F(d_2, q_1) = Mean(2, 7, 4, 11) = 6;$$
$$F'(d_1, q_1) = 0,\ F'(d_2, q_1) = 1;$$
$$G(d_1, q_1) = 0,\ G(d_2, q_1) = 1;$$

Therefore, according to Eq. 4, the first query-document pair will correspond with state (3, 0), while the second query-document pair matches with state (4, 1). Based on Eqs. 4 and 5, in this example, there are two different relevance levels, so there will be two different possible actions in each state. Also, as there are four voters, so the number of non-null voters could be either: 0, 1, 2, 3 or 4. So:

$$|A| = 2,\ NVV_{\max} = 5,\ |S| = 10$$

Therefore, at the end of the learning process, the RL agent has its own estimations about values of possible actions in each state. In fact, the result of agent's learning could be represented in an action-value table like Table 2.

At the end of the learning process, while being in an arbitrary state, the RL agent selects the action with maximum action-value from the list of all possible actions. As it is presented in the Sect. 5, using this MDP model, the proposed rank-aggregation method outperforms well-known algorithms based on the evaluation criteria.

4 The evaluation framework

4.1 Benchmark dataset

For the evaluation of our method, we used a subset of the LETOR4.0 dataset [24]. This version of the LETOR dataset was released by the Microsoft Research Center at China, in 2009. The subset of LETOR 4.0 dataset that was used is called MQ2008, which is prepared for the "Rank-

Table 2 An example of the action-value table

		Actions	
		a_1	a_2
States	s_1	$e_{1,1}$	$e_{1,2}$
	s_2	$e_{2,1}$	$e_{2,2}$
		…	…
	s_{10}	$e_{10,1}$	$e_{10,2}$

Table 3 Sample of LETOR4.0 MQ2008 benchmark dataset

0 qid:10002 1:1 2:30 3:48 4:133 5:NULL… 25:NULL #docid = GX008 inc = 1 prob = 0.086622
0 qid:10002 1:NULL 2:NULL 4:NULL 5:NULL… 25:NULL #docid = GX037 inc = 0.00358 prob = 0.0852
2 qid:10032 1:6 2:96 3:88 4:NULL 5:NULL… 25:NULL #docid = GX029 inc = 0.01198 prob = 0.13984

aggregation" task. There are 800 queries in MQ2008 with labeled documents.

In this setting, every query is associated with a number of ranked lists. Each ranked list is assumed to be the output of a search engine. The task of rank-aggregation is to prepare a better final ranked list by aggregating the multiple input lists. A row in this dataset indicates a query-document pair. Some sample rows of the utilized benchmark dataset are presented in Table 3.

In each row, the first column is the relevance label of this query-document pair, which could be either as {0, 1, 2}: 'Relevant or 2', 'Moderate or 1' and 'Non-relevant or 0'. The second column is query ID, the following columns are ranks of the document in the input ranked lists, and finally at the end of the row, there are some comments about the pair, including ID of the document. In the above example, 2:30 in the first row means that the rank of the document is 30 in the second input list (the output of second search engine). Note that large ranks mean top positions in the input ranked list, and '*NULL*' means the document does not appear in a ranked list. The larger the relevance label, the more relevant the query-document pair [24]. There are 25 input lists from 25 search engines in MQ2008 dataset.

MQ2008 dataset, also includes the results of a few baseline rank-aggregation methods such as: Borda Count and three extensions of the CPS algorithm which are CPS-KendallTau, CPS-SpearmanFootrule and CPS-SpearmanRankCorrelation. Borda Count is a popular aggregation method which simply computes the ranking score of an object based on the number of objects that are ranked below it in all of the ranking lists [7]. Specifically, in Borda Count method, each candidate (or alternative) gets one point for each last place vote received, two points for each next-to-last point vote, etc., all the way up to N points for each first-place vote (where N is the number of candidates/alternatives). The candidate with the largest point total wins the election. For instance, in a four candidate election, each 4th place vote is worth one point, each 3rd place vote is worth two points, each 2nd place vote is worth three points, and each 1st place vote is worth four points.

The CPS is a well-known probabilistic model over permutations, which is defined with a coset-permutation distance and models the generation of a permutation as a stage-wise process. The CPS model has rich expressiveness as well as low complexity. This is due to the stage-wise decomposition of the permutation probability and the efficient computation of most coset-permutation distances [27]. There are many well-defined metrics to measure the distance between two permutations, such as Spearman's rank correlation, Spearman's Footrule and Kendall's tau, which lead to different extension of CPS model, which are CPS-KendallTau, CPS-SpearmanFootrule and CPS-SpearmanRankCorrelation, respectively [27].

In order to apply the proposed method on MQ2008 dataset, in the first step of the algorithm, by taking into account three relevance labels of {0,1,2} for each query-document pair, equations (Eqs. 1, 2, 3) have been performed in order to create an aggregation of the base data. Then, during the second step of the proposed rank-aggregation method, a MDP model was prepared for using equations (Eqs. 4, 5, 6). Specifically, considering the structure of the LETOR4.0 MQ2008 benchmark dataset which includes votes of 25 local ranker on each query-document pair and consideration of three relevance levels, we have:

$$NNV_{\max} = 26,\ |S| = 26 * 3 = 78$$

The action space for this MDP model is defined with the given three relevance levels, as:

$$A = \{0, 1, 2\} \Rightarrow |A| = 3$$

4.2 Evaluation metrics

There are many measures for evaluation of the performance of a search engine such as *Kendall-Tau* [15], *P@n*, *NDCG@n* and *Mean Average Precision (MAP)* [23]. In this paper, the following evaluation criteria are used:

- *Precision at position n (P@n)* indicates the ratio of the relevant documents in a list of *n* retrieved documents. The main aim of this metric is to calculate the precision of retrieval systems from users' perspective. As users visit only top documents from the list of results, the evaluation measures usually just consider *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 be relevant or irrelevant with a specified query. Then, *P@n* is defined as:
- $$P@n = \frac{\#relevant\, docs\, in\, top\, n\, results}{n}$$
- *Mean Average Precision* (*MAP*) For a single query *q*, *Average Precision* (*AP*) is defined as the average of the *P@n* values for all relevant documents.

$$AP(q) = \frac{\sum_{j=1}^{|D_q|} \left(r_j \times P@j\right)}{|R_q|}$$

- In this formulation, r_j is the relevance score assigned to a document d_j with respect to a given query q. r_j *is* 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, *MAP* would be the mean of average precisions of all queries of the benchmark dataset used in the experiment as:

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

Normalized Discount Cumulative Gain at position n (*NDCG@n*) Evaluation criteria such as *P@n* and *MAP* consider only binary degrees of relevance in the evaluation of query-document pairs. Therefore, their analysis may not be precise or satisfactory. 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.

Table 4 Configuration parameters used for Q-learning and SARSA

$\beta = 0.1$, $\gamma = 0.1$, $\varepsilon = 0.01$, Number of iterations = 10,000, Episode length = 100

5 Experimental results

We have utilized the LETOR's Eval-Tool [22] for calculating all the above mentioned measures in all of our experiments. All reported results are based on the usage of Fold1 of training set of LETOR4.0 MQ2008 benchmark dataset. In this investigation, we have experimented with Q-learning and SARSA reinforcement learning methods for the second step of our model. The best configuration parameters for these methods which are achieved in trials, are listed in Table 4.

As mentioned before, β is a constant step-size parameter and γ is the discount rate. In the implementation of Q-learning and SARSA techniques, the ε-greedy action selection policy is used [33]. Within this policy, in each state, the action with the highest $Q(s,a)$, is selected with the probability of 1-ε, but in order to preserve the exploration, the learning agent may select among the rest of possible actions with the probability of ε. The learning agent applies the SARSA and Q-learning methods on the proposed MDP model to learn the best action for each state of the MDP.

For the first step of our model, we have experimented with the following techniques as our rank-aggregation function (F):

1. Weighted majority voting (WMV), in which the weights associated with local rankers are their *MAP* values.
2. Optimistic OWA, with $\alpha = 0.9$. In this case, ordering of local rankers is based on their *MAP* values.

In this work, a variety of data fusion operators such as min, max, average, WMV and optimistic OWA are implemented and tested.

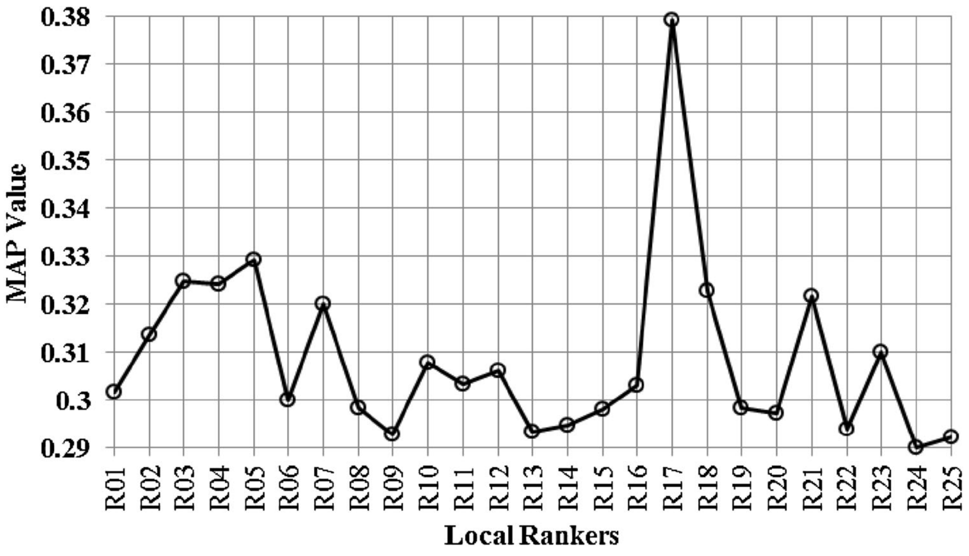


Fig. 3 Comparison of local rankers of LETOR4.0 MQ2008 benchmark dataset based on their *MAP* values

Figure 3 illustrates the quality of the local rankers based on the *MAP* measure.

Table 5 gives a comparison of different implementations of the proposed methods with the baseline rank-aggregation algorithms based on the *MAP* measure. For the ease of tracking, in the following tables, results of the best implementations of the proposed rank-aggregation method are set to bold. As it could be observed, the Q-learning implementation of the proposed method in combination with weighted majority voting (WMV) as the aggregation function, outperforms other configurations of the proposed approach by about 6.01 %. Based on the MAP evaluation criterion, this configuration of the proposed method, also has achieved an improvement of about 14.8 % in comparison with the best baseline rank-aggregation algorithms.

Table 6 provides a comparison of different implementations of the proposed algorithm with other rank-aggregation methods based on the *P@n* criterion. As it could be seen, the configuration based on Q-learning as the learning agent and the Optimistic OWA as aggregation function, has demonstrated a noticeably higher performance on *P@n* criteria. Since most users of Web search engines, often focus on top three items of the results set [31], our method which produces the better performance on top two items would have an edge over other approaches. For *P@1* criterion, this improvement is 15.71 % in comparison with the best baseline method. For *P21* criterion the improvement is about 7.97 % compared to the best conventional aggregation method.

Table 7 presents similar comparison based on the *NDCG@n* criterion. Again, one can observe that the combination of Q-learning and Optimistic OWA function has produced the best performance. The improvement for *NDCG@1* is about 13.93 % in comparison to the best baseline method and about 16.55 % in comparison with best conventional aggregation method. However, similar to baseline rank-aggregation algorithms, the proposed method, suffers from a sharp fall within *NDCG@9* and *NDCG@10* values in comparison with other values of *n* in *NDCG@n* criterion.

In order to gain more insight into the behavior of the Optimistic OWA and WMV aggregation functions, we have looked into accumulated average of achieved rewards during the learning process for Q-learning and SARSA methods. As illustrated in Figs. 4 and 5 the Optimistic OWA method, can achieve more accumulated average with a higher convergence rate of about 1.3 in comparison with the WMV as the aggregation function.

As it can be observed, in Fig. 4, the average reward falls down in the beginning of the reinforcement learning process. This is mainly due to the utilization of the “Optimistic Initial Values” method for initializing $Q(s,a)$ values. In fact, this method encourages the RL agent to do

Table 5 Comparison of different implementations of the proposed method with some conventional aggregation methods as well as baseline algorithms based on the *MAP* criterion

Rank aggregation method	Conventional aggregation methods				Different implementations of the proposed algorithm				Best baseline algorithms		
	MAX	WA	Borda Count	WMV	Q-learning_ WMV	SARSA_ WMV	Q-learning_ OWA-Opt	Q-learning_ BordaCount	CPS-KendallTau	CPS-Spearman footrule	CPS-Spearman rank correlation
MAP value	0.3794	0.33	0.341	0.3843	**0.4623**	0.4361	0.3866	0.3806	0.4049	0.385	0.4007

Table 6 Comparison of different implementations of the proposed method with other rank-aggregation algorithms based on the *P@n* criterion

Aggregation method	Conventional aggregation methods		Different implementations of the proposed algorithm				Best baseline algorithms		
	WMV	OWA_Opt	SARSA_ WMV	Q-learning_ WMV	SARSA_ OWA_Opt	Q-learning_ OWA_Opt	CPS-KendallTau	CPS-Spearman footrule	CPS-Spearman rank correlation
P@1	0.2803	0.3185	0.2696	0.3397	0.2484	**0.3439**	0.2972	0.2633	0.2972
P@2	0.2749	0.3153	0.2718	0.3227	0.2569	**0.3312**	0.3227	0.2919	0.3142
P@3	0.2583	0.3107	0.2661	0.305	0.2548	0.3114	0.3128	0.2895	0.3163
P@4	0.2564	0.301	0.2521	0.2978	0.2516	0.3052	0.311	0.2956	0.311
P@5	0.251	0.2913	0.2442	0.2866	0.2518	0.2951	0.2981	0.287	0.3074
P@6	0.2481	0.2746	0.242	0.2757	0.2459	0.2834	0.2866	0.2799	0.2958
P@7	0.2414	0.2645	0.2402	0.2648	0.2369	0.2748	0.2772	0.2696	0.2812
P@8	0.2282	0.2497	0.2341	0.2519	0.2282	0.2593	0.2604	0.2558	0.2641
P@9	0.2154	0.2317	0.2189	0.2354	0.2137	0.2432	0.246	0.2418	0.2463
P@10	0.2053	0.2191	0.2068	0.2223	0.2021	0.2291	0.2325	0.2295	0.2323

Table 7 Comparison of different implementations of the proposed method with other rank-aggregation algorithms based on the *NDCG@n* criterion

Aggregation method	Conventional aggregation methods		Different implementations of the proposed algorithm				Best baseline algorithms		
	WMV	OWA_Opt	SARSA_ WMV	Q-learning_ WMV	SARSA_ OWA_Opt	Q-learning_ OWA_Opt	CPS-KendallTau	CPS-Spearman footrule	CPS-Spearman rank correlation
NDCG@1	0.2123	0.2477	0.2201	0.2774	0.1918	**0.2887**	0.2463	0.2109	0.2534
NDCG@2	0.2496	0.287	0.2488	0.3015	0.2279	**0.3224**	0.3057	0.2689	0.308
NDCG@3	0.2691	0.3187	0.2713	0.323	0.2574	**0.3433**	0.329	0.3031	0.3376
NDCG@4	0.2903	0.3428	0.2863	0.3506	0.284	**0.3696**	0.3627	0.3398	0.3652
NDCG@5	0.3142	0.3677	0.3069	0.3692	0.31	0.3919	0.382	0.3622	0.3921
NDCG@6	0.336	0.3813	0.3291	0.3855	0.3281	0.4083	0.4009	0.3787	0.4071
NDCG@7	0.3515	0.3943	0.3431	0.3975	0.3399	0.421	0.4158	0.3926	0.4177
NDCG@8	0.3276	0.3674	0.3292	0.3785	0.3255	0.3991	0.3916	0.3728	0.3919
NDCG@9	0.1402	0.1726	0.1405	0.1734	0.1384	0.1894	0.1815	0.166	0.1851
NDCG@10	0.1472	0.1781	0.1462	0.1799	0.1439	0.1942	0.1872	0.173	0.1899

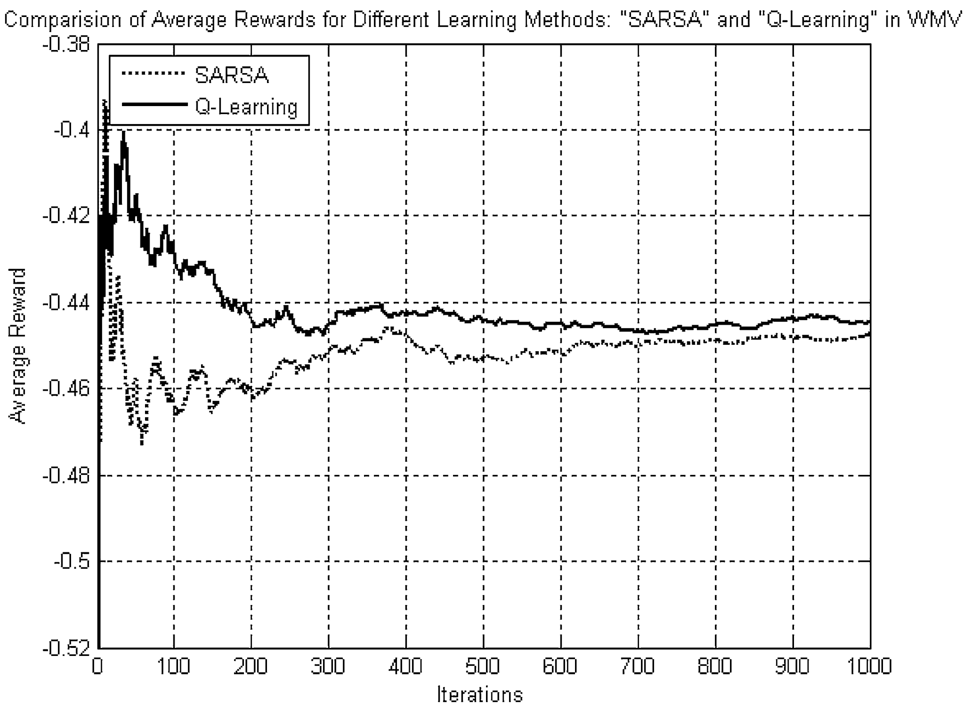


Fig. 4 Accumulated average reward for WMV method

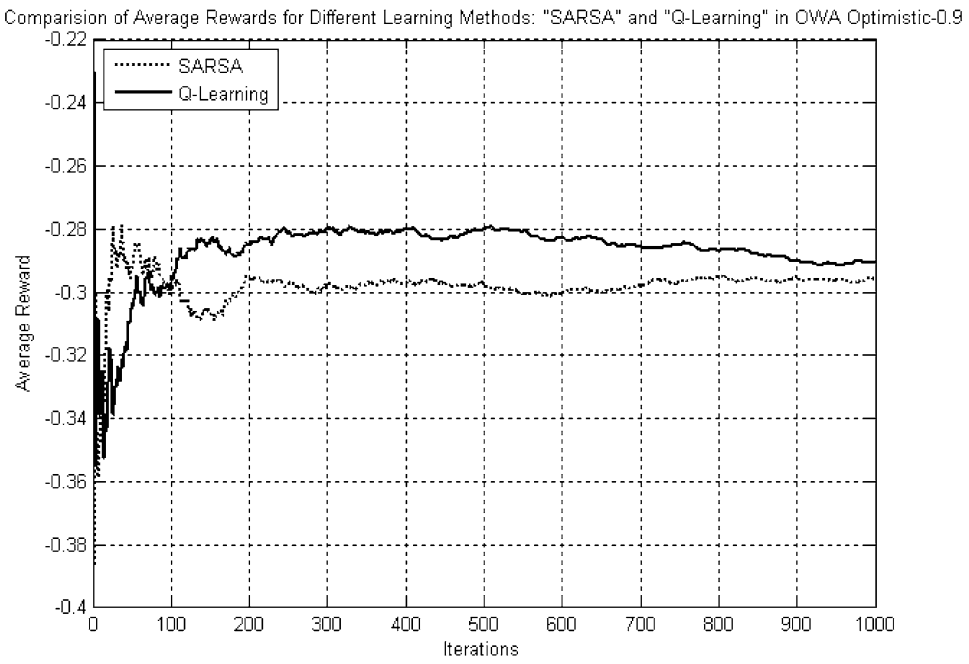


Fig. 5 Accumulated average reward for optimistic OWA algorithm

sufficient exploration in the action-space at the beginning of the learning process. Besides, this phenomenon is more observable in the learning curve of the Q-learning method. Actually, as Q-learning is an off-policy learning approach, sometimes its behavior leads to the selection of non-optimal actions. This situation will be more noticeable in the beginning of the learning process where the estimations of the values of actions in different states are not precise.

Also the “Percentage of selecting the Optimal Action” during the learning process, is an important factor which demonstrates the internal behavior of the learning method. This data which is shown in Figs. 6 and 7, indicates that by utilizing Optimistic OWA method as the aggregation function, the RL agent selects the optimal action about 10 % more often than while the Weighted Majority Voting method is employed. Also, it could be observed that based

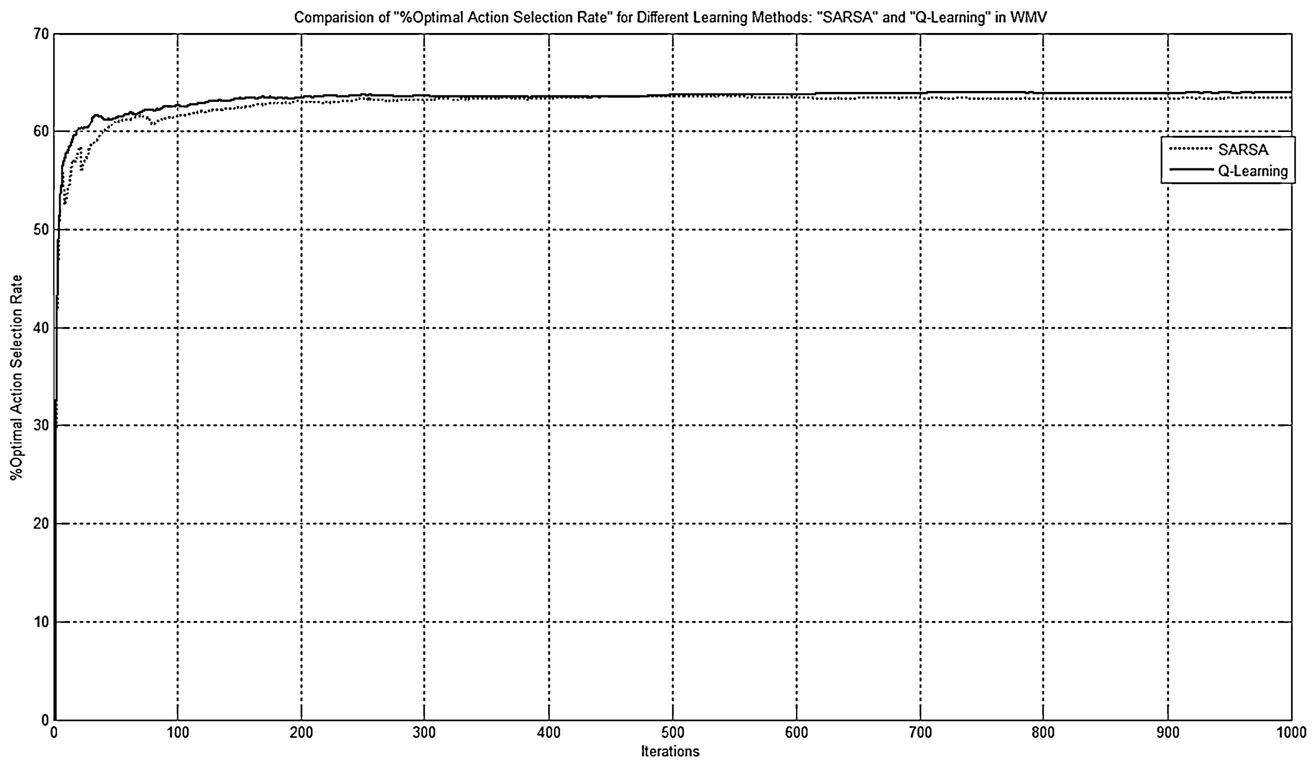


Fig. 6 Percentage of optimal action selection for WMV method

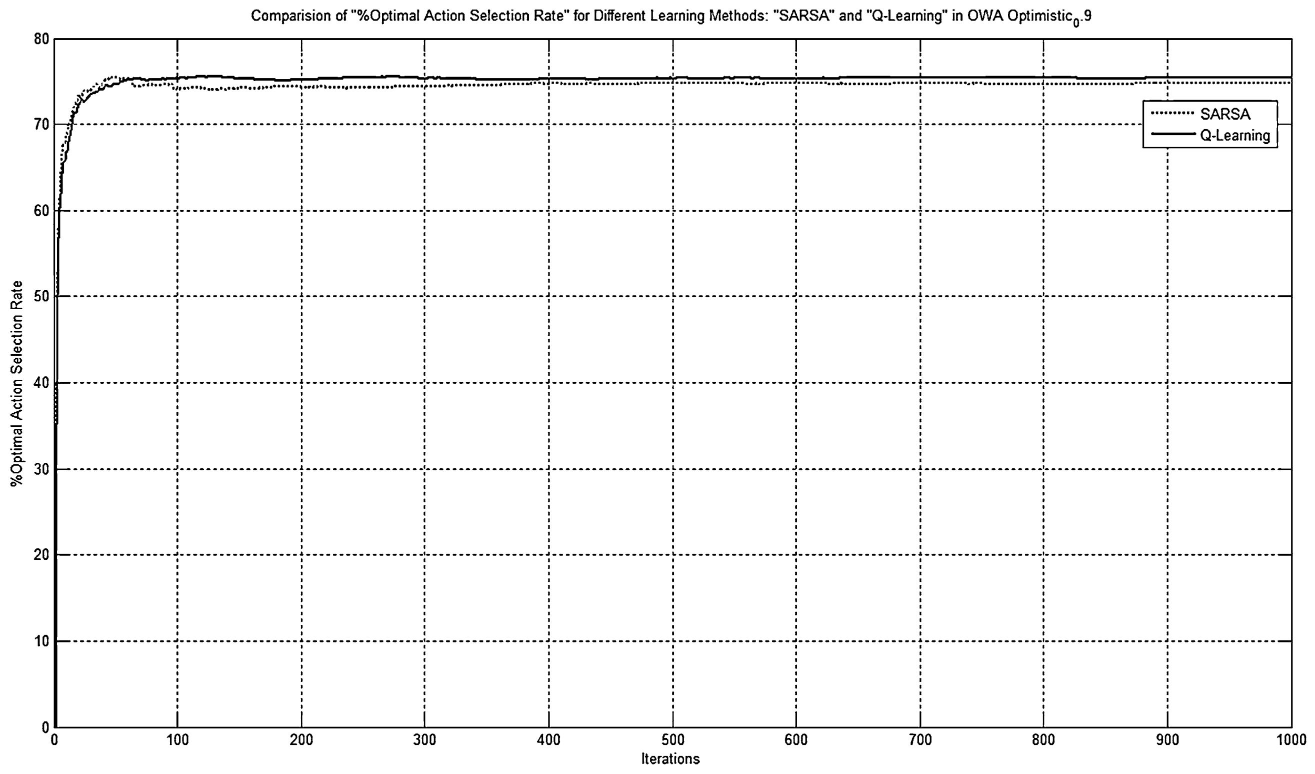


Fig. 7 Percentage of optimal action for optimistic OWA algorithm

Table 8 Convergence rate for various implementations of the proposed algorithm

Aggregation function	Elapsed time for Q-learning (s)	Elapsed time for SARSA (s)
Weighted majority voting	0.820574	0.565199
Optimistic OWA	0.824910	0.552102

on the percentage of optimal action selection, both Q-learning and SARSA methods, have relatively similar performance during the learning process.

All of the above mentioned experiments were done with a Pentium 5 PC containing a CPU of 2.62 GHz and 3 GB RAM. A comparison of the convergence rate of the proposed algorithm implemented with SARSA and Q-learning is presented in Table 8.

It must be noticed that by using the fixed low discount rate ($\gamma = 0.1$), the RL agent becomes too myopic. This limitation could be resolved by using other discount rates, especially those which are descending along the time. In this way, in the beginning of the reinforcement learning process, the agent pays more attention to the immediately received rewards, while at the end of the learning, it notices more to the accumulated rewards.

6 Discussion and further works

This paper presents a novel rank-aggregation method which integrates the capabilities of data fusion operators and reinforcement learning techniques. The main idea of the proposed method is to build a MDP model for the rank-aggregation problem and apply reinforcement learning techniques upon the aggregated list of the local rankers. We have experimented with different data fusion operators such as Min, Max and Optimistic and Pessimistic OWA for creating an aggregated list for each query from the ranked lists provided by local rankers. This aggregation process was guided by the precision of local rankers signified by their MAP values. Then, a MDP model is constructed upon the aggregated list of the local rankers. This model enables us to assign a suitable relevance label to a given query-document pair based on the average of rewards received from doing so, for similar items.

The comparison of the proposed method with well-known rank-aggregation techniques on LETOR benchmark dataset demonstrated that the proposed method outperforms baseline algorithms based on *P@n* and *NDCG@n* evaluation criteria, especially on the top ranks that are of particular interest to Web users. It was also noted that the Q-learning approach due to its off-policy nature which enables it to have a better generalization of the problem environment, has shown better performance than SARSA approach with its on-policy learning approach.

As the future work, we plan to investigate the usefulness of the proposed method in situations where more data are available besides local ranked lists such as URL of the ranked items, title of corresponding documents, and their textual snippets. Another direction for further work is to extend the proposed method for the learning to rank problem in which, different features of query-document pairs could be assumed as local rankers. In such a setting, the quality of the proposed method could be compared with the state-of-the-art learning to rank algorithms.

Acknowledgments This work is supported by the University of Tehran (Grant Number 8101004/1/02). The authors thank the Editor-in-Chief and four anonymous reviewers for their helpful comments and suggestions, which were very helpful in improving the paper. We also give special thanks to Ms. Maryam Piroozmand and Dr. Kambiz Badie for their helps and inspiring discussions.

References

1. Akritidis L, Katsaros D, Bozanis P (2011) Effective rank-aggregation for meta-searching. J Syst Softw 84(1):130–143
2. Aslam JA, Montague M (2001) Models for metasearch. In: 24th annual international ACM SIGIR conference research and development in information retrieval, pp 276–284
3. Becchetti L, Castillo C, Donato D, Leonardi S, Italia R (2008) Web spam detection: link-based and content-based techniques. In: Final workshop for European integrated project dynamically evolving, large scale information systems, pp 99–113
4. Beg M (2004) Parallel rank-aggregation for the World Wide Web. Worldw Web 6(1):5–22
5. Chen S, Wang F, Song Y, Zhang C (2011) Semi-supervised ranking aggregation. Inf Process Manag 47(3):415–425
6. Dwork C, Kumar R, Naor M, Sivakumar D (2001) Rank-aggregation methods for the Web. In: 10th international conference on World Wide Web, pp 613–622
7. Erp MV, Schomaker L (2000) Variants of the Borda Count method for combining ranked classifier hypotheses. In: 7th international workshop on frontiers in handwriting recognition, pp 443–452
8. Fagin R, Kumar R, Sivakumar D (2003) Efficient similarity search and classification via rank-aggregation. In: 2003 ACM SIGMOD international conference management of data, pp 301–312
9. Fang Q, Xiao H, Zhu S (2010) Top-d rank-aggregation in Web meta-search engine. In: Lee D-T, Chen DZ, Ying S (eds) Frontiers in algorithmics. Lecture Notes in Computer Science, vol 6213. Springer, Berlin, pp 35–44
10. Filev D, Yager RR (1998) On the issue of obtaining OWA operator weights. Fuzzy Set Syst 94:157–169
11. Granka LA, Joachims T, Gay G (2004) Eye-tracking analysis of user behavior in WWW search. In: 27th annual international ACM SIGIR conference on research and development in information retrieval, pp 478–479
12. He Y, Liu J, Hu Y, Wang X (2015) OWA operator based link prediction ensemble for social network. Expert Syst Appl 42(1):21–50
13. Hemaspaandra E, Hemaspaandra LA, Rothe J (1997) Exact analysis of Dodgson Elections: Lewis Carroll's 1876 voting

system is complete for parallel access to NP. J ACM (JACM) 44(6):214–224
14. Kehoe C, Pitkow J, Sutton K, Aggarwal G, Rogers JD (1999) Results of GVU's tenth World Wide Web user survey. Graphic, Visualization, & Usability Center. http://www.cc.gatech.edu/gvu/user_surveys/survey-1998-10/tenthreport.html. Accessed 15 Jan 2015
15. Kendall MG (1938) A new measure of rank correlation. Biometrika 30(1–2):81–89
16. Keyhanipour AH, Moshiri B, Kazemian M, Piroozmand M, Lucas C (2007) Aggregation of web search engines based on users' preferences in WebFusion. Knowl Based Syst 20(4):321–328
17. Khodabakhshi M, Aryavash K (2015) Aggregating preference rankings using an optimistic–pessimistic approach. Comput Ind Eng 85:13–16
18. Kolde R, Laur S, Adler P, Vilo J (2012) Robust rank-aggregation for gene list integration and meta-analysis. Bioinformatics 28(4):573–580
19. Lam KW, Leung CH (2004) Rank-aggregation for meta-search engines. In: 13th international conference on World Wide Web, pp 384–385
20. Li H (2011) Learning to rank for information retrieval and natural language processing. Morgan & Claypool Publishers, San Rafael
21. Liu TY (2011) Learning to rank for information retrieval. Springer, Berlin
22. Liu YT, Liu TY, Qin T, Ma ZM, Li H (2007) Supervised rank-aggregation. In: 16th international conference on World Wide Web, pp 481–490
23. Manning CD, Raghavan P, Schutze H (2008) Introduction to information retrieval. Cambridge University Press, New York
24. Microsoft Research Asia (2010) LETOR dataset. http://research.microsoft.com/en-us/um/beijing/projects/letor//default.aspx. Accessed 15 Jan 2015
25. Miller M (2012) 53% of organic search clicks go to first link. http://searchenginewatch.com/article/2215868/53-of-Organic-Search-Clicks-Go-to-First-Link-Study. Accessed 15 Jan 2015
26. O'Hagan M (1988) Aggregating template rule antecedents in real-time expert systems with fuzzy set logic. In: 22nd annual IEEE Asilomar conference on signals, systems and computers, pp 681–689
27. Qin T, Geng X, Liu TY (2010) A new probabilistic model for rank-aggregation. In: 24th annual conference neural information processing systems, pp 1948–1956
28. Randa ME, Straccia U (2003) Web metasearch: rank vs. score based rank-aggregation methods. In: 2003 ACM symposium on applied computing, pp 841–846
29. Saari DG (2000) Mathematical structure of voting paradoxes. Econ Theory 15(1):55–102
30. Sese J, Morishita S (2001) Rank-aggregation method for biological databases. Genome Inform 12:506–507
31. Slingshot SEO Inc (2011) A tale of two studies establishing google & bing click-through rates. http://www.slingshotseo.com/wp-content/uploads/2011/10/Google-vs-Bing-CTR-Study-2011.pdf. Accessed 15 Jan 2015
32. Spirin N, Han J (2011) Survey on Web spam detection: principles and algorithms. ACM SIGKDD Explor Newslett 13(2):50–64
33. Sutton RS, Barto AG (1998) Reinforcement learning: an introduction. MIT Press, Cambridge
34. Szepesvari C (2010) Algorithms for reinforcement learning. Morgan & Claypool Publishers, San Rafael
35. Vogt CC, Cottrell GW (1999) Fusion via a linear combination of scores. Inf Retr 1(3):151–173
36. Wang YM, Luo Y, Hua Z (2007) Aggregating preference rankings using OWA operator weights. Inf Sci 177:3356–3363
37. World-Wide-Web-Size (2015) The size of the World Wide Web (the internet). http://www.worldwideWebsize.com. Accessed 15 Jan 2015
38. Yager RR (1988) On ordered weighted averaging aggregation operators in multicriteria decision making. IEEE Trans Syst Man Cybern 18(1):183–190
39. Yager RR (1993) Families of OWA operators. Fuzzy Set Syst 55:255–271
40. Yan HB, Huynh VN, Nakamori Y, Murai T (2011) On prioritized weighted aggregation in multi-criteria decision making. Expert Syst Appl 38(1):812–823
41. Zeckman A (2015) Organic search accounts for up to 64 % of website traffic. Search engine watch. http://searchenginewatch.com/article/2355020/Organic-Search-Accounts-for-Up-to-64-of-Website-Traffic-STUDY. Accessed 15 Jan 2015