Expert Systems With Applications 49 (2016) 86–98

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

Preference relations based unsupervised rank aggregation for metasearch

Maunendra Sankar Desarkar[1,*], Sudeshna Sarkar, Pabitra Mitra

Department of CSE, IIT Kharagpur, Kharagpur 721302, India

ARTICLE INFO

Keywords:
Rank aggregation
Metasearch
Information retrieval

ABSTRACT

Rank aggregation mechanisms have been used in solving problems from various domains such as bioinformatics, natural language processing, information retrieval, etc. Metasearch is one such application where a user gives a query to the metasearch engine, and the metasearch engine forwards the query to multiple individual search engines. Results or rankings returned by these individual search engines are combined using rank aggregation algorithms to produce the final result to be displayed to the user. We identify few aspects that should be kept in mind for designing any rank aggregation algorithm for metasearch. For example, generally equal importance is given to the input rankings while performing the aggregation. However, depending on the indexed set of web pages, features considered for ranking, ranking functions used etc. by the individual search engines, the individual rankings may be of different qualities. So, the aggregation algorithm should give more weight to the better rankings while giving less weight to others. Also, since the aggregation is performed when the user is waiting for response, the operations performed in the algorithm need to be light weight. Moreover, getting supervised data for rank aggregation problem is often difficult. In this paper, we present an unsupervised rank aggregation algorithm that is suitable for metasearch and addresses the aspects mentioned above.

We also perform detailed experimental evaluation of the proposed algorithm on four different benchmark datasets having ground truth information. Apart from the unsupervised Kendall-Tau distance measure, several supervised evaluation measures are used for performance comparison. Experimental results demonstrate the efficacy of the proposed algorithm over baseline methods in terms of supervised evaluation metrics. Through these experiments we also show that Kendall-Tau distance metric may not be suitable for evaluating rank aggregation algorithms for metasearch.

© 2015 Elsevier Ltd. All rights reserved.

1. Introduction and motivation

The input to the rank aggregation problem is a number of rankings obtained from different sources. The sources can be human judges or algorithms. The task is to combine these input rankings and produce an aggregate ranking. Rank aggregation techniques have been used to solve problems from different applications domains. We mention here a few expert and intelligent systems applications or application domains where rank aggregation techniques have been used to arrive at solutions to different problems.

- *Voting:* For *voting* applications, there is a fixed list of candidates. There are voting schemes where the voters are allowed to rank the candidates based on the order of their choices. For such schemes, it is often necessary to combine the rankings provided by the voters to produce an aggregate ranking of the candidates, to obtain a *consensus ordering of the candidates*. This aggregate ranking can be determined by rank aggregation algorithms (de Borda, 1781; Davenport & Kalagnanam, 2004; Elkind & Lipmaa, 2005).
- *Metasearch:* Metasearch engines (e.g. MetaCrawler (http://www.metacrawler.com/), Dogpile (http://www.dogpile.com), Entireweb (http://www.entireweb.com/), etc.) accept queries from users, and forward that query to several *second-level* search engines. Each of these second-level search engines returns a ranked list of items for the query. The metasearch engine then combines these ranked lists to produce an aggregate list. This aggregate list is displayed to the user as a response

* Corresponding author. Tel.: +91 40 2301 8466.
E-mail addresses: maunendra@iith.ac.in (M.S. Desarkar), sudeshna@cse.iitkgp.ernet.in (S. Sarkar), pabitra@cse.iitkgp.ernet.in (P. Mitra).
[1] Present address: IIT Hyderabad, Kandi, Medak District 502 285, Telangana, India.

http://dx.doi.org/10.1016/j.eswa.2015.12.005
0957-4174/© 2015 Elsevier Ltd. All rights reserved.

to his/her query (Aslam & Montague, 2001; Chen, Wang, Song, & Zhang, 2008; Jansen, Spink, & Koshman, 2007; Thomas & Hawking, 2009). Thus rank aggregation is a central task for successful working of any metasearch engines.

- *Multi-criteria decision making:* There are systems where the objects (documents/products/candidates) might be scored or ranked based on multiple criteria. However, a single ordering of the objects is required as the final ranking. Rank aggregation algorithms are often used in tasks such as selecting product or services for recommendation (Shao, Chen, & Huang, 2010), combining feature based rankings for producing a single ranking for web search queries (Farah & Vanderpooten, 2007), candidate screening for hiring process in a large organization (Mehta, Pimplikar, Singh, Varshney, & Visweswariah, 2013), diversifying search results (Ozdemiray & Altingovde, 2015), etc.
- *Recommender systems:* Recommender systems have traditionally recommended items to individual users. Recently there has been a proliferation of recommender systems that recommend items to *groups of users* (Jameson & Smyth, 2007). Examples of such scenarios include a group of users listening to music, watching a movie, going to a restaurant or a museum etc. For recommending items to a group of users, Baltrunas, Makcinskas, and Ricci (2010) present a methodology where the system first gets ranked recommendation list for each member of the group, and then aggregates the individual lists to produce the recommendation for the group. Sohail, Siddiqui, and Ali (2015) aggregate user feedbacks to provide evaluation in product recommender systems.
- *Natural language processing:* For the language translation task, algorithms are suggested that for a source sentence, consider the ranked list of translations returned by different translator algorithms, and combine them to produce a final ranking of the candidate translations (Rosti et al., 2007). Similar techniques of combining ranked lists of candidate solutions for finding the final output are used for approaching the problems of syntactic dependency parsing (Sagae & Lavie, 2006) and word sense disambiguation (Brody, Navigli, & Lapata, 2006).
- *Networking:* In the networking domain, a number of metrics have been proposed to quantify the inherent robustness of a network topology against failures. However, each single metric usually offers only a limited view of network vulnerability. When applied to certain network configurations, different metrics rank the network topologies in different orders, and no single metric fully characterizes network robustness against different modes of failure. To overcome this problem, Yazdani and Leonardo Duenas-Osorio (2013) propose a multi-metric approach where the ordering of the topologies given by different individual metrics are combined to get an overall ranking of robustness for the network topologies. In social networking domain, Tabourier, Libert, and Lambiotte (2014) use rank aggregation for link prediction. Ordering of a user's neighbors according to various network-based features (such as Adamic-Adar, Jaccard Index, Katz measure, etc.) are identified. These orderings are aggregated to suggest set of possible new connections for the user.
- *Healthcare:* In Fields, Okudan, and Ashour (2013), the authors present a system where in an emergency department of a hospital, nurses provide orderings of the patients in terms of the severity of the patients' conditions. Nurses put patients requiring medical attention more urgently than others near the top of the list. Such orderings provided by multiple nurses are aggregated to produce a single ranking of patients and the patients are attended in that order.
- *Bioinformatics:* Rank aggregation algorithms are used in the bioinformatics domain also, for cluster validation in microarray data analysis (Pihur, Datta, & Datta, 2007), identifying differentially expressed genes (Fang, Feng, & Ng, 2011), high throughput screening in nanotoxicology (Patel et al., 2012), multimodal biometric systems (Monwar & Gavrilova, 2013), feature selection (Sarkar, Cooley, & Srivastava, 2014), etc.

It is evident from the discussion above that rank aggregation algorithms are used to solve problems in different expert and intelligent systems. Most of the rank aggregation algorithms discussed in literature are unsupervised in nature. This is because unsupervised methods can be easily ported across different applications. Supervised approaches to rank aggregation need supervised ranked data, which is expensive to acquire (Klementiev, Roth, & Small, 2008; Wu, Greene, & Cunningham, 2010). Therefore, unsupervised rank aggregation is an important problem to be studied. We wish to develop an unsupervised rank aggregation algorithm. Our main motivation is to work on the metasearch problem. In metasearch, ranked responses from different search engines are combined and the aggregate ranking is displayed as the output. As the aggregation is performed in runtime when the user is waiting for the final result for his query, it is essential for the algorithm to be of low complexity, and also the steps involved in performing the aggregation should involve low cost operations. Also, the quality of the input rankings given by individual search engines are affected by various factors such as indexed set of web-pages, features used for ranking, ranking function used etc. Due to this fact, qualities of the input rankings may not be equal. So, in metasearch, there is a need to identify the qualities of the input rankings and use that quality information while performing the aggregation. We have not come across any work in literature that emphasizes on *these requirements* while developing unsupervised algorithms for metasearch. The unsupervised algorithm proposed in this paper is developed keeping these aspects in mind.

Several unsupervised rank aggregation algorithms and their analysis are presented in the works (Aslam & Montague, 2001; Betzler, Bredereck, & Niedermeier, 2014; de Borda, 1781; Dwork, Kumar, Naor, & Sivakumar, 2001; Schalekamp & van Zuylen, 2009). These algorithms are widely used in metasearch engines and in metasearch literature. However, all these methods consider the input rankings to be equally good. Equal importance is given to the rankers for computing the aggregate ranking. Cohen, Schapire, and Singer (1998) learn quality weights of the rankers and uses that information for aggregation. The algorithm maintains a single query vector over the rankers at any time, which is irrespective of the query. However, relative qualities of the rankers can be different for different input queries. This is often true for metasearch, due to different sets of indexed pages maintained by different search engines. Also, the method learns the quality weights from user feedback data, which as mentioned earlier, may be difficult to get. Several other recent researches also use supervised approaches for rank aggregation (Liu, Liu, Qin, Ma, & Li, 2007; Pujari & Kanawati, 2012; Tabourier et al., 2014). The work in Rajkumar and Agarwal (2014) discusses desirable theoretical properties for rank aggregation algorithms and provides a supervised algorithm for rank aggregation.

The method proposed in this work considers each input ranking as a preference graph and aggregates the preference graphs to generate the aggregate ranking. We want to give different weights to different rankers (or the corresponding preference graphs) depending on their *qualities* or *goodness* on a given query. We want to assign the weights in an unsupervised manner. At the same time, we want the weight assignment and ranking aggregation algorithms to involve low cost operations, so that the algorithm is fast and suitable for real time processing. Although the algorithm is developed keeping in mind the metasearch problem, it can be used for all the applications/frameworks mentioned earlier in this section

(except for the voting application, where it is mandatory to give equal importance to all the voters).

We provide detailed experimental evaluation of the proposed algorithm. Different supervised and unsupervised evaluation metrics are used as evaluation measures. Generally unsupervised rank aggregation algorithms are evaluated using the unsupervised metric Kendall-Tau distance. This metric computes the number of *inversions* between to rankings. Lower number of inversions lead to lower values of the metric and are preferred by the algorithms. However, this metric is known to have some drawbacks (Carterette, 2009; Yilmaz, Aslam, & Robertson, 2008) as it does not distinguish between inversions at the top and bottom of the rankings. Recently, few benchmark datasets (Chapelle & Chang, 2010; Qin, Liu, Xu, & Li, 2009) with ground truth information have been released for the rank aggregation task. It is possible to use supervised evaluation measures for these datasets. We use these datasets for experimentation. The experimental results indicate the effectiveness of the proposed algorithm. From the results, we also see that algorithms that obtained good Kendall-Tau distance measure performed badly according to supervised evaluation measures and vice-versa. The ordering of the algorithms according to the Kendall-Tau distance measure was completely different from the ordering obtained using the supervised evaluation measures. Supervised measures use ground truth information for determining the quality of results. Hence the experimental evaluations indicate that if supervised labels are available, then it might be better to use supervised evaluation measures for evaluating the performances of the rank aggregation algorithms for metasearch.

The specific contributions of the work are three-fold:

- We propose an unsupervised rank aggregation algorithm that assigns weights to the rankers depending on their qualities. These quality weights are different for different queries and are assigned in an unsupervised manner.
- The algorithm is developed keeping in mind that the operations need to be fast for real time aggregation of the input rankings.
- We perform detailed evaluation of different unsupervised rank aggregation algorithms. Four different benchmark datasets are used for experimentation. Apart from the Kendall-Tau distance, we also use several supervised evaluation measures like Precision, NDCG, MAP, mean NDCG, ERR as evaluation metrics. By using both supervised and unsupervised evaluation measures for performance evaluation, we show that Kendall-Tau distance may not be suitable for evaluating Rank aggregation algorithms for metasearch.

The rest of the paper is organized as follows. We discuss the related work from literature in Section 2. A formal definition of the rank aggregation problem for metasearch is given in Section 3. The rationale behind using preference relations is described in Section 4. Section 5 describes a data structure that can be used to store the preference relations. The details of the proposed algorithm is presented in Section 6. Experimental results are presented and analyzed in Section 7. Section 8 concludes the discussion and also provides pointers for future research directions.

2. Related work

In this section, we describe some of the existing rank aggregation methods and discuss their merits and demerits.

2.1. Existing methods: unsupervised

We classify the existing unsupervised rank aggregation algorithms into different categories based on the ways the aggregation is performed and review each of these categories in detail.

2.1.1. Positional score based methods

Borda Count (de Borda, 1781) is the most widely used rank aggregation algorithm that is based on positional scores. In Borda Count, scores are assigned to the items or candidates based on their absolute ranks in the different input lists. For each ranking, the lowest item is assigned a score of x. The candidate at the second lowest position in that list is given a score $x+1$, the third lowest item is given a score of $x+2$ and so on. The value of x is often set to 1. For each item, the scores that it gets for the different lists are added up, and the items are sorted in decreasing order based on their total scores. This sorted list is output as the aggregate list. Amin and Emrouznejad (2011) present a positional score based algorithm that assigns weights to each position dynamically, by looking at the data. However, if there are N items in the set and the length of the longest list is L, then the method needs to solve N linear programming problems, each with $O(N+L)$ constraints.

2.1.2. Markov chain based methods

Several Markov chain based methods were used for rank aggregation in Dwork et al. (2001). The items in the ranked lists were considered as states. For each pair of items i_1 and i_2, the corresponding state transition probability depends on what fraction of the input rankers have ranked both the items, and also how many lists have i_1 before i_2. Four such methods were proposed, namely, MC1, MC2, MC3, and MC4. The methods are different from each other in the ways the transition probabilities are calculated.

2.1.3. Kemeny optimal aggregation

Methods belonging to this category try to optimize the average Kendall-Tau distance or the Kemeny distance between the aggregate list and the input lists. Kemeny distance between two ranked lists is defined as the number of item pairs (i, j) for which the two lists disagree on their relative ordering, i.e. one list places i above j, and the other has j above i.

Several researchers have posed the task of optimizing average Kemeny distance as the minimum feedback arc set problem and proposed algorithms for the task. There are weighted versions of the problem as well, and people have tried to find efficient approximate algorithms for these versions (Ailon, Charikar, & Newman, 2008; Coppersmith, Fleischer, & Rurda, 2010; Schalekamp & van Zuylen, 2009; van Zuylen & Williamson, 2007).

2.1.4. Condorcet procedure

This is a pairwise method and comes from the literature of voting theory. Condorcet winner for an election is the candidate who wins over or ties with every other candidate in the election. Repetitive application of the Condorcet procedure produces the aggregate ranking of the candidates (Condorcet, 1785).

For rank aggregation, the input rankings are viewed as the preference ordering of candidates as given by different voters. The aggregation is then performed in a number of steps, by selecting one Condorcet winner (which corresponds to an item from the input ranked lists) in each step. However, it is not necessary for an election to always have a Condorcet winner. Variations of the basic Condorcet procedure are suggested to address such situations. Montague and Aslam (2002) view the input lists as a preference graph over the candidates. It then partitions the graph into strongly connected components (SCC). Items in a single SCC are equally preferable. Edges between two SCCs denote the preference of one SCC over the other. There is another rule called Black rule (Jean, 1961) that combines Borda and Condorcet policies for rank aggregation. If there is a Condorcet winner, then Black rule selects that item. Otherwise Borda Count is used to determine the winner.

2.1.5. Linear aggregation

This scheme is often used when scores are available from the input rankers. Several *data fusion* approaches using linear aggregation of scores are suggested in literature for the *Learning to Rank* task. Weighted sum or its variations are used as aggregation operators (Farah & Vanderpooten, 2007; Lee, 1997; Shaw & Fox, 1994; Wu, Li, Zeng, & Bi, 2014). Shaw and Fox (1994) introduced CombSUM, CombMIN, CombMAX, CombANZ, CombMNZ strategies for data fusion. CombSUM ranks items based on sum of scores they have obtained from the rankers. The next two strategies consider for ranking the maximum and minimum scores respectively that an item has obtained. Scores are often normalized to a bounded range to make them comparable across different rankers. Different variants of these strategies are suggested in Farah and Vanderpooten (2007) and Wu et al. (2014).

2.1.6. Probabilistic models

Several research work in recent literature (Clémençon & Jakubowicz, 2010; Klementiev et al., 2008; Klementiev, Roth, Small, & Titov, 2009) suggest the use of probabilistic models such as the Mallows model on permutations to solve the problem of unsupervised rank aggregation. Algorithms that use Mallows model perform well, but have a complexity of $O(n!)$. Reduction in complexity is possible using sampling techniques (Klementiev et al., 2008). However, this affects the effectiveness of the model. Mallows model has also been used to aggregate "typed rankings" coming from domain-specific experts – where the type or domain of the candidate items are known (Klementiev et al., 2009).

Clémençon and Jakubowicz (2010) apply Luce model for performing rank aggregation, using generalized Kantorovich distances between rankings. The problem of measuring disagreement between rankings is cast as discrete mass transportation problem, by embedding the set of permutations in a convex set of doubly stochastic permutation matrices. Let the set of all doubly stochastic matrices of size $n \times n$ be denoted as K_n. Each input ranking σ_i can be represented as an element $A_i \in K_n$. If there are m such input rankings, then any $A^* \in K_n$ having minimum average distance from A_1, A_2, ..., A_m is called as the median ranking. The algorithm first finds one such A^*. It then builds the aggregate ranking sequentially, by including one new item in each step. This part of building the list sequentially is governed by the Luce model, where the probability of an item being included as the next item is computed based on the entries in the median matrix A^* already chosen. Complexity of the method is high. The step of distance computation between rankings takes $O(n^6)$ time in general. The complexity can be reduced by using certain specific cost functions.

2.2. Existing methods: supervised

The majority of the algorithms for the rank aggregation problem are unsupervised in nature. However, recently, due to the availability of few rank aggregation datasets with supervised information (Qin et al., 2009), people are discussing supervised algorithms also for the task. Here we briefly review some of the supervised methods for the rank aggregation task.

2.2.1. Probabilistic models

A supervised Probabilistic Fusion method for rank aggregation is presented in Lillis et al. (2006). The approach is based on the assumption that a ranker that does well on past queries, will do well for future queries also. Scores are treated as probabilities of relevance. The rank positions are divided into k segments. Given a ranker and a segment, the probability that the ranker has placed a relevant item in that segment is calculated from the training data. Score for a test item is calculated based on this probability as well as the scores given to the test item by the different rankers.

A probabilistic model for supervised rank aggregation by using a combination of Mallows and Luce models is suggested in Qin, Geng, and Liu (2010a). A coset-permutation distance based stagewise (CPS) model is used for this purpose. Specifically, the final permutation π of n objects is generated in n sequential stages. At the kth stage, the task is to select the kth object in the permutation π. The probability of this selection is defined using the coset-permutation distance between the right coset $S_{n-k}\pi$ and the input rankings σ, where S_n is the symmetric group of order n. The aggregated list can be computed in $O(mn^4)$ time if there are m input rankings. However, the authors mention that the runtime complexity can be brought down to $O(mn^2)$ by using efficient implementation techniques.

2.2.2. Machine learning based methods

An online learning based method for combining ranking "experts" is given in Cohen et al. (1998), which gives weights to each ranker. For each query in the training set, weights of the rankers that performed poorly for that query were reduced. When the next query comes, the algorithm starts with the current value of weights, and the process goes on. Clicklog data or document relevance information is used to measure the performance of a ranker for a query. In Liu et al. (2007), supervised versions of Markov chain based approaches (Dwork et al., 2001) were proposed. Wang et al. (2013) take the differences among queries into consideration and proposes a query similarity based supervised rank aggregation framework. First, the framework sets up a number of base rankers for each query. Next, the base rankers are aggregated to get the final ranked lists. A supervised approach is used to tune the weights of these base rankers. Semisupervised rank aggregation is proposed in Chen et al. (2008).

2.3. Learning to Rank

The Learning to Rank task has as input scores given to different query–document pairs by different ranking features (Chapelle & Chang, 2011; Li, 2011). Given a collection of such scores and also the relevance labels for different query–document pairs, ranking functions are learned. Given a new query, feature scores for documents are first determined. Then the learned ranking functions are applied on these feature scores to determine the final ranking. Different methods have been proposed for the task. The methods differ from each other in the (a) family of functions considered for ranking like feature-weighted linear aggregation (Cao et al., 2006), boosting trees (Chen, Bai, & Zheng, 2011), (b) family of loss functions used (Acharyya, Koyejo, & Ghosh, 2012; Burges, Ragno, & Le, 2006; Weston, Yee, & Weiss, 2013), (c) instances considered for training the ranking function – pointwise (Li, Burges, & Wu, 2008), pairwise (Sculley, 2009), listwise (Cao, Qin, Liu, Tsai, & Li, 2007), etc., (d) comparing different ways of supervision information (Chen & Hofmann, 2015), etc. Also, there are works that use the feature scores to generate additional intermediate features and use them for a two-stage learning of the ranking function (Keyhanipour, Moshiri, & Rahgozar, 2015). Semi-supervised approaches for the task by using both labeled and unlabeled data for training are also proposed (Pan, Luo, Qi, & Tang, 2011). It should be noted that, for the rank aggregation problem, the feature scores are not available as input. Only the relative orderings of documents are available. Still, one can assume the different features as rankers, induce rankings based on the feature scores, and combine these rankings using rank aggregation techniques.

As described in this section, both supervised and unsupervised algorithms exist for rank aggregation. In this paper, our focus is to develop unsupervised rank aggregation algorithm that is suitable for a real time application like metasearch. As input, we have only rankings obtained from different sources and not the scores

given to the items by the different rankers. Hence, aggregation algorithms that combine scores for the different items in the rankings cannot be used for the task at hand. Also, majority of the rank aggregation algorithms discussed in literature consider each input ranking to be equally good. Due to various reasons as mentioned in Section 1, the input rankings for metasearch may not be equally good. There are rank aggregation algorithms that rely on supervised data for determining the weights of the input rankers. However, our focus is to estimate the goodness of the input rankings without using any supervised data and use that information for determining the aggregate ranking. We now give a formal definition of the rank aggregation problem and explain our solution for approaching the problem.

3. Rank aggregation: problem formulation

Before describing our proposed algorithm for solving the problem, we formally introduce the task of rank aggregation.

Rank aggregation: Let I be a set of items. A ranked list or ranking R^l w.r.t. I is a permutation of the items from a set $S \subseteq I$. In other words, $R^l = \langle x_1, x_2, \cdots, x_d \rangle$, where each $x_i \in S$. If x_i appears before x_j in R^l, it is said that x_i is ranked above x_j in R^l. Given N such ranked lists $R^l_1, R^l_2, \cdots R^l_N$, the *rank aggregation* problem attempts to find an aggregate ranking R^l_A over the same item set I.

4. The rationale behind using preference relations

We wish to consider the *relative ordering* between items appearing in the input rankings for solving the rank aggregation task. If item i appears above item j in a ranking, we consider that i is more preferable to j in the ranking. This is intuitive, as several applications put better items at the top of the list. We use this *relative ordering* or *preference relations* between items for producing the aggregate ranking.

The inputs from human experts or algorithms about a set of items may come in different forms: (a) scores for all the items, (b) rankings of items and (c) pairwise preference relations. Inputs represented as scores or rankings can be easily converted to preference relations. Also, there can be cases where giving opinions in the form of preference relations might be easier. For example, if there is a set of movies to be ranked by a user, giving a set of preference relations might be less confusing than giving absolute scores to the movies or ranking the movies in order of user satisfaction. Different algorithms also may find it easier to provide relative ordering between items (possible translations for machine translators, pairs of matching texts for plagiarism detection, etc.). Hence, it is imperative to look at preference relations based algorithms for such aggregation tasks.

5. A data structure for storing the preference relations

We use a data structure called preference graph for storing the preference relations. Our algorithm views each ranked list as a collection of preference relations, and hence as a preference graph over the items present in that list.

Representing a ranking as a preference graph: A preference graph is a data structure where there is one directed edge between each pair of nodes. The edge denotes the preference relation between the items it connects. In our representation, the nodes in the preference graph represent the items present in the ranked list. If the list contains item i above item j, then the graph contains a preference edge (a directed edge) from node j to node i. The weight of this edge is set to one.

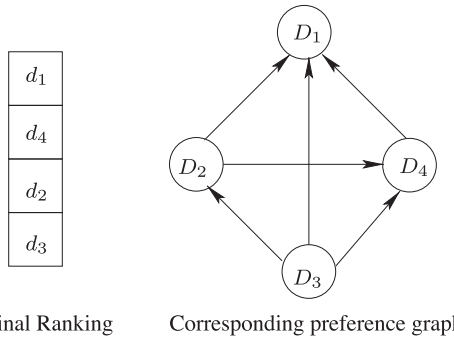


Fig. 1. Converting a ranking to a preference graph. $d_1, \ldots d_4$ are the documents. $D_1, \ldots D_4$ are the corresponding nodes in the preference graph. Edges denote preference relations.

As the input is a ranking where there is no tie, there can be exactly one edge between every node pair in this preference graph. Fig. 1 gives an example of representing a ranking as a preference graph.

6. A preference relations based unsupervised rank aggregation algorithm

We now describe the proposed rank aggregation algorithm. The algorithm works in two phases. The first phase views the input rankings as preference graphs. It then assigns weights to the input rankings (or the corresponding preference graphs) based on their *qualities*. A weighted combination of these preference graphs is performed to construct an aggregate graph. The second phase induces a ranking from the aggregate graph. This ranking is the output of the rank aggregation algorithm. These steps are described in detail in the next subsections.

6.1. Creating an aggregate graph from the input rankings

The first phase of the proposed algorithm creates an aggregate graph. For this, first the input rankings are viewed as preference graphs. All the input rankings (equivalently, the input preference graphs) may not be equally good in terms of representing the actual (ground truth) preference relations between items. If we can identify the rankings that are *good* and the ones that are *bad*, we may give higher importance to the *better rankings* during the aggregation process. We keep a weight vector on the input rankings. The weight of a ranking is indicative of the confidence that the algorithm has on that ranking. So, we expect the better rankings to have higher weights. This phase first computes the quality weights for the rankings, and uses these weights to produce the aggregate preference graph.

Identification of *better rankings* can be easily done if supervised information is available in the form of document relevance scores or clicklogs. In absence of supervised information, one can consider the *majority opinion* to be indicative of the supervised label for the relative importance between a pair of items. i.e., if more than half of the rankings say that i is better than j, then i is considered better than j in the ground truth setting. This is common for voting applications.

However, this may not be a good choice for metasearch. If there are 20 rankers and 9 say that i is better than j, it is difficult to say whether i is actually better than j. Again, if 5 rankers say that i is better and the remaining say otherwise, then the metasearch engine may assume that i is indeed better than j. In other words, if a webpage is preferred over another by *many* ranked lists, chances are high that the user will have *more* satisfaction for that page. To model this scenario, we consider the opinion of a ranker to be

incorrect if it fails to agree with a fraction α of rankers that rank both the items. In this context, we first define what we mean by *disagreement with α-majority*.

Definition: disagreement of a ranking withα-majority: Suppose there are N rankings. α and β are two constants such that $0 \leq \alpha \leq 0.5$ and $0 \leq \beta \leq 1$. For each item pair (i, j), the rankings may give their opinions. The set of possible opinions is $X = \{0, 1\}$. Also, there can be rankings that do not provide any opinion about the item pair. For the pair (i, j), let the number of rankings which give the opinion $x \in X$ is denoted as n_x.

Suppose the opinion by a ranking k is $x(k) \in X$. We say that ranking k has disagreed with the α-majority if and only if both the following conditions are satisfied.

$$n_0 + n_1 \geq \lceil \beta N \rceil \tag{1}$$

$$n_{x(k)} < \alpha(n_0 + n_1). \tag{2}$$

Inequality (1) signifies that if a minimum number of opinions is not available for the item pair, no ranking is marked as incorrect. Inequality (2) indicates that a ranker failing to agree with at least α fraction of available opinions for the same pair is marked as incorrect (in disagreement with α-majority).

*An example:*We now explain the above definition using examples. Suppose the number of rankings is $N = 20$. Also assume $\alpha = 0.3$ and $\beta = 0.5$.

- *Case 1:* 12 rankings say that i is better than j (consider this as opinion 0). 5 rankings say that j is better than i (consider this as opinion 1). So, $n_0 = 12$ and $n_1 = 5$.
  $n_0 + n_1 = 17$, $\lceil \beta N \rceil = \lceil 0.5 \times 20 \rceil = 10$.
  So, Eq. (1) is satisfied.
  Now, $\alpha(n_0 + n_1) = 0.3 \times (12 + 5) = 5.1$
  As $n_1 = 5 < \alpha(n_0 + n_1)$, Eq. (2) is satisfied for $x = 1$.
  Hence, according to Definition, all the rankings which say j is better than i are in disagreement with the α-majority for the pair (i, j).
- *Case 2:* 5 rankings say that i is better than j. 10 rankings say that j is better than i. So, $n_0 = 5$ and $n_1 = 10$.
  $n_0 + n_1 = 15$, $\lceil \beta N \rceil = \lceil 0.5 \times 20 \rceil = 10$.
  So, Eq. (1) is satisfied.
  Now, $\alpha(n_0 + n_1) = 0.3 \times (5 + 10) = 4.5$
  As $n_0 = 5 > \alpha(n_0 + n_1)$ and $n_1 = 10 > \alpha(n_0 + n_1)$, Eq. (2) is satisfied for neither $x = 0$ nor $x = 1$.
  Hence, according to the definition none of the rankings are in disagreement with the α-majority.

Based on this definition, we now design a weight assignment rule for the input rankings.

6.1.1. Assigning quality scores to input rankings

Let R_1 to R_N be the input rankings. Also, let d_{ijl} denote the preference relation between items i and j as obtained from the ranking R_l. As discussed in Section 5, preference relation between two items is determined by considering the relative positions of the items in the ranking. For each input ranking R_l, we also consider that all items that are present in R_l are preferred over all other items that are not in R_l. However, if both i and j are not present in R_l, we cannot comment on the opinion of R_l on the preference relation involving this item pair. We define the disagreement of R_l with input rankings (R) as the number of item pairs for which R_l differs from α-majority opinion on the relative ordering of the items. We define the *disagreement score* for ranking R_l as:

$$\Delta_l = \sum_{\substack{(i,j) \\ \in S \times S}} \delta_{ijl}. \tag{3}$$

where

$$\delta_{ijl} = \begin{cases} 0, & \text{if } R_l \text{ does not disagree with } \alpha\text{-majority for } (i, j) \\ 1, & \text{if } R_l \text{ disagrees with } \alpha\text{-majority for } (i, j) \\ 0.5, & \text{if both } i \text{ and } j \text{ are not ranked by } R_l. \end{cases} \tag{4}$$

Here S is the set of distinct items that appear in the input rankings. Disagreements with α-majority are determined using the conditions described in the previous section. α and β are parameters of the algorithm. Their values remain same for all input cases. Weight of the ranking R_l is then set as

$$w_l = 1 - \frac{\Delta_l}{\binom{|S|}{2}}. \tag{5}$$

w_l denotes the fraction of item pairs for which R_l agrees with α-majority. Rankings which agree with α-majority for more number of item pairs get high weights according to this scheme.

6.1.2. Weighted aggregation of input preference graphs

Once the weights of the input rankings are determined, an aggregate graph $(\mathcal{G}_\mathcal{A})$ can be constructed by taking a weighted combination of the preference relations obtained from the rankings. $\mathcal{G}_\mathcal{A}$ contains as nodes all the items in S, i.e. the items appearing in at least one of the input rankings. Weight of the edge from i to j in $\mathcal{G}_\mathcal{A}$, denoted as $e_{ij\mathcal{A}}$, is computed as:

$$e_{ij\mathcal{A}} = \sum_{e_{ijk} \text{ is in } G_k} \left(w_k e_{ijk}\right). \tag{6}$$

In Eq. (6), the summation runs over the individual rankings that provide preference relations for the pair (i, j). w_k is the weight of the ranking R_k, and e_{ijk} denotes the weight of the preference edge from i to j in the preference graph corresponding to the input ranking R_k. Similarly, weight for the directed edge from j to i in $\mathcal{G}_\mathcal{A}$ is set to $e_{ji\mathcal{A}} = \sum_{e_{jik} \text{ is in } G_k} \left(w_k e_{jik}\right)$.

It may be recalled from Section 5 that the weight of the edge from i to j is set to 1 in a preference graph if and only if the corresponding ranking prefers j over i. Hence, from Eq. (6) it can be said that: the weight of directed edge (i, j) in $\mathcal{G}_\mathcal{A}$ represents the *total weighted votes* in favor of the preference relation *j is better than i* as given by the input rankings. This weight is high if many of the *good* rankings place j above i in their orderings.

6.2. Inducing linear order

The aggregate graph $\mathcal{G}_\mathcal{A}$ may not represent a total order. However, the output of rank aggregation has to be a total order. So, we have to induce a total order from the preference relations stored in $\mathcal{G}_\mathcal{A}$. This can be achieved in several ways. The different algorithms suggested for the *minimum feedback arc set* problem try to pose this as an optimization problem. Extended Markov chain procedure explained in Dwork et al. (2001) also can be used for this purpose.

However, methods that use the extended Markov chain or minimize feedback arc set are of higher complexity. On the other hand, in metasearch, the aggregation is performed in runtime, when the user is waiting for the results. Hence, we wanted a fast algorithm that can be used for finding the total order. We use a heuristic based method for this purpose. It first computes the weighted indegree of each node of $\mathcal{G}_\mathcal{A}$. The weighted indegree of a node j is given by $\sigma_j = \sum_{i \in S} e_{ij\mathcal{A}}$. From the way the values $e_{ij\mathcal{A}}$ are computed, it can be seen σ_j is high if *many good rankers consider item j to be better than many other items*. The heuristic sorts the nodes of the aggregate graph in decreasing order of their weighted indegrees. This sorted order of the nodes gives the final output of the algorithm. The unweighted version of this algorithm, where all rankers are given equal weights, was proposed in Copeland (1951).

6.3. Complexity of the algorithm

We now analyze the complexity of the proposed rank aggregation algorithm.

- *Aggregating preference graphs:* In this phase, the algorithm finds the α-majority opinion for each *each item pair*. The time complexity for finding this information for an item pair is $O(N)$ where N is the number of input rankings. Once the majority opinion is obtained, updating the *disagreement count* Δ_I of the N rankings requires $O(N)$ time. If the total number of distinct items is m, then there are $O(m^2)$ item pairs, and finding disagreement counts of all rankings requires $O(Nm^2)$ time.
  Once disagreement counts of all the rankings are computed, computing the *quality weights* of the N rankings need $O(N)$ time. Hence the total complexity of this phase is $O(Nm^2+N) = O(Nm^2)$. It can be noted that any pairwise method for rank aggregation must have a complexity of $\Omega(Nm^2)$, as it has to look at all pairs in all the input rankings.
- *Inducing linear order:* This phase sorts the nodes in the graph by their weighted indegrees. As there are m nodes in the graph, computing the weighted indegrees takes $O(m^2)$ time. Sorting can be done in $O(m\log m)$ time. Hence, the complexity of this phase is $O(m^2+m\log m) = O(m^2)$.
  Hence the complexity of the entire algorithm is $O(Nm^2)$. It is worth mentioning that almost all of these operations involve comparisons and additions. Moreover, careful examination of the algorithm will indicate that the constants involved in the order notations are also very small. As a result, the method is very fast and suitable for real-time processing.

7. Experimental results

In this section, we compare the performance of the proposed algorithm against existing unsupervised rank aggregation algorithms.

7.1. Datasets used

We used Yahoo Learning to Rank Challenge (LTRC) dataset, Microsoft Learning to Rank (MSLR-WEB10K) dataset and LETOR 4.0 rank aggregation datasets (MQ2007-agg, MQ2008-agg) for our experiments. The LETOR datasets are specifically for rank aggregation and we use them directly.

The other two datasets viz. Yahoo-LTRC and MSLR-WEB10K datasets contain scores given to different documents based on different features. We considered each feature as a ranker. For each feature, we obtained the ranked list for a query by sorting the documents in decreasing order of the scores. For MSLR-WEB10K dataset, we considered the top 30 features according to the number of distinct scores assigned by the feature in the dataset. This was to reduce the number of ties in the input rankings. Typically, rankings received by metasearch engines do not have any ties. Upon cross-referencing with the feature list given in Qin, Liu, Ding, Xu, and Li (2010b), we found that these top 30 identified features are (in order): (1) LMDIR.ABS for whole document, (2) BM25 score for the whole document, (3) LMIR.DIR score for the whole document, (4) LMDIR.JM score for the whole document, (5) LMIR.ABS for the document body, (6) BM25 score for the document body, (7) LMIR.DIR score for the document body, (8) LMIR.JM score for the document body, (9) mean of tf*idf for the whole document, (10) sum of tf*idf for the whole document, (11) mean of tf*idf for the document body, (12) sum of tf*idf for the document body, (13) variance of tf*idf for the whole document, (14) variance of tf*idf for the document body, (15) vector space model score for the whole document, (16) vector space model score for the document body, (17) PageRank score, (18) SiteRank score, (19) sum of stream length normalized term frequency for the whole document, (20) max of tf*idf for the whole document, (21) LMIR.ABS score for the document title, (22) sum of stream length normalized term frequency for the document title, (23) max of tf*idf for the document body, (24) mean of stream length normalized term frequency for the whole document, (25) LMDIR.DIR score for the document title, (26) mean of stream length normalized term frequency for the document body, (27) BM25 score of the title, (28) max of stream length normalized term frequency for the whole document, (29) LMIR.JM score for the document title and (30) max of stream length normalized term frequency for the document body. It can be noted that, intuitively, all the features mentioned above are good indicators of a document's relevance for a given query. Hence we decided to use these features as rankers.

We also changed each relevance label from *rel* to $\lfloor \frac{rel}{2} \rfloor$. This was done for two reasons: (a) as there are 5 relevance labels (0–4), having a document with relevance label 1 in the original data would increase the precision, whereas there are other documents that are much more relevant than this. We wanted to have those in the top results. (b) If a document with relevance label 4 appears at the top, then it influences the NDCG a lot. Changes in ordering at remaining places may get unnoticed as a result. We performed experiments with original values of the relevance labels also. Ordering of the algorithms in terms of performance metrics were the same as the ordering that we obtained using the modified labels. Here we report the results obtained using the modified relevance labels.

The Yahoo-LTRC dataset also was used in a similar way. We considered top 45 features according to the number of distinct scores. However, names of the exact features are not available for this dataset.

7.2. Algorithms used for comparison

We compare the proposed algorithm with several other unsupervised rank aggregation algorithms, namely, BORDA (uses positional score) (de Borda, 1781), MC4 (uses Markov chain based method) (Dwork et al., 2001), QSORT (optimizes Kendall-Tau score), (Schalekamp & van Zuylen, 2009) LUCE-R (uses probabilistic model) (Clémençon & Jakubowicz, 2010). We refer to our algorithm as WT-INDEG. We implemented another algorithm, EQ-INDEG, as a baseline for our algorithm. EQ-INDEG gives equal weight to each ranking. EQ-INDEG was used as a baseline in Schalekamp and van Zuylen (2009) and is a variant of the Copeland method (Copeland, 1951). We did not implement LUCE-R due to complexity issues. For LUCE-R, we only compare the NDCG values reported in Clémençon and Jakubowicz (2010).

We also include for comparison performances of two supervised rank aggregation algorithms. The first algorithm is based on Coset Permutation Distance (Qin et al., 2010a). We denote this method as *CPS(sup)*. For CPS(sup), the values for the supervised evaluation measures on the MQ2007-agg and MQ2008-agg datasets are available at Qin et al. (2009). We use these values in our comparisons. The other supervised algorithm that we compare with is a tree adaptation based supervised algorithm (Chen et al., 2011). We denote the algorithm by *TRADA(sup)*. For training, TRADA(sup) uses feature scores for the query–document pairs. Such scores are not available in the LETOR datasets. Hence we do not have metric values of TRADA(sup) for MQ2007-agg and MQ2008-agg datasets. We report the results of this algorithm for the Yahoo dataset only.

For CPS(sup) and TRADA(sup), the word "sup" indicates that the algorithms are supervised. Comparisons with supervised algorithms should only be used as reference. Supervised algorithms are

Table 1
Effect of parameter β on the MQ2007-agg dataset.

β	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
Mean NDCG	0.359	0.359	0.359	0.360	0.360	0.359	0.359	0.357	0.356	0.355	0.314
MAP	0.350	0.351	0.351	0.351	0.351	0.351	0.351	0.349	0.348	0.346	0.309
KT	0.408	0.409	0.409	0.409	0.409	0.409	0.409	0.408	0.407	0.407	0.509
ERR	0.204	0.204	0.204	0.204	0.205	0.204	0.204	0.203	0.202	0.201	0.162

Table 2
Effect of parameter β on the MQ2008-agg dataset.

β	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
Mean NDCG	0.450	0.450	0.448	0.445	0.438	0.435	0.418	0.401	0.307	0.297	0.307
MAP	0.437	0.437	0.434	0.430	0.425	0.423	0.407	0.391	0.310	0.295	0.301
KT	0.403	0.403	0.405	0.405	0.404	0.403	0.398	0.393	0.385	0.291	0.396
ERR	0.266	0.266	0.266	0.264	0.261	0.261	0.253	0.243	0.175	0.168	0.175

Table 3
Effect of parameter β on the Yahoo dataset.

β	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
Mean NDCG	0.486	0.487	0.487	0.489	0.489	0.489	0.490	0.493	0.452	0.406	0.355
MAP	0.459	0.459	0.459	0.461	0.461	0.461	0.462	0.463	0.434	0.400	0.364
KT	0.345	0.345	0.346	0.346	0.346	0.346	0.347	0.347	0.362	0.391	0.435
ERR	0.280	0.280	0.280	0.282	0.282	0.282	0.282	0.286	0.259	0.226	0.191

Table 4
Effect of parameter β on the MSLR-WEB10K dataset.

β	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
mNDCG	0.389	0.389	0.389	0.389	0.389	0.389	0.389	0.389	0.390	0.382	0.251
MAP	0.277	0.276	0.276	0.276	0.276	0.277	0.277	0.276	0.276	0.267	0.170
KT	0.172	0.171	0.171	0.171	0.171	0.172	0.172	0.171	0.172	0.171	0.084
ERR	0.235	0.237	0.237	0.237	0.237	0.237	0.238	0.239	0.243	0.272	0.449

expected to perform better than unsupervised algorithms as they look at additional ground truth data for training.

7.3. Evaluation metrics

We have used one unsupervised evaluation metric, namely, average Kendall-Tau (KT) distance and several supervised evaluation metrics such as Precision, NDCG, MAP, mean NDCG and ERR (Chapelle, Metlzer, Zhang, & Grinspan, 2009) for determining the qualities of the aggregate rankings.

7.4. Parameter values for the proposed algorithm

As explained in Section 6.1, the proposed algorithm uses two parameters α and β for determining the quality weights of the rankers. We set α to 0.5 for all the datasets to indicate that we want the opinions given by the rankers to agree with majority (50% of the rankers). We used a small set of validation data to determine the value of β, that indicates the number of rankers that must have provided opinion for a document pair.

Ideally, high value of β indicates we want many rankers to provide relative orderings for each document pair. If β is set to 0, that means, we are not bothered about how many rankers provide relative rankings for the document pairs. Among the ones that provide opinion for the pair, we compute the majority opinion and adjust disagreement count accordingly. On the other hand, if β is set to the maximum value 1.0, then it means that we want all the rankers to provide opinions about the document pairs. In metasearch, it is unlikely that all the rankers provide relative ranking opinion for the document pairs. As a result, in this case, the disagreement count for all the rankings would remain as 0. Hence, all rankings would have equal weight, which, as explained in Section 6.1, may not be desired. It appears that β should be set to some *moderate* value. To determine that value, we used a subset of the data as validation set. In our experiments with validation data, we varied β from 0.0 to 1.0 in steps of 0.1. Based on the results of this experiment, we selected $\beta = 0.3$ for MQ2008-agg and $\beta = 0.5$ for the other datasets. Effect of different β on *complete* MQ2007-agg, MQ2008-agg, Yahoo and MSLR-WEB10K datasets are mentioned in Tables 1, 2, 3 and 4, respectively.

It can be seen that, for MQ2007-agg and Yahoo as β increases, the metric values do not change initially. However, after some time, ($\beta = 0.9$ for MQ2007-agg and $\beta = 0.7$ for Yahoo), the values start degrading. This behavior is expected, as high value of β results in less number of document pairs for which disagreement values can be updated. As a result, weights assigned to different rankers become almost similar, and the performance of the algorithm degrades. For MSLR-WEB10K data, the values were affected only when β was set to 1.0.

The behavior is quite different for MQ2008-agg. The degradation in performance starts much earlier. After $\beta = 0.5$, the metric values start degrading quickly. We tried to analyze the reason behind this difference in behavior. To do so, we calculated the average number of opinions for each document pair in the three datasets. This value for MQ2007-agg, Yahoo and MSLR-WEB10K datasets were 0.5, 0.7 and 0.9 respectively. Whereas, for MQ2008-agg, the value was only 0.18. As a result, even when β was set to a moderate value of 0.5, it was difficult to get *sufficient number of* opinions for the document pairs. Based on the observations, we recommend the following heuristic for setting the value of β. *If there are N rankers and the average number of opinions available for each document pair is more than 50% of N, then set the value of beta as 0.5. Otherwise set the value of β to 0.3.*

In the following section, we compare the performance of the proposed algorithm WT-INDEG with other algorithms mentioned in Section 7.2. In the rest of the paper, metric values reported for

Table 5
Comparing average KT distances.

Dataset →	Yahoo	MQ2007-agg	MQ2008-agg	MSLR-WEB10K
WT-INDEG	0.346	0.409	0.405	0.237
EQ-INDEG	0.311	0.379	0.357	0.216
BORDA	0.307	0.330	0.275	0.218
MC4	0.306	0.345	0.287	0.217
QSORT	**0.151**	**0.307**	**0.244**	**0.214**
TRADA(sup)	–	–	0.349	–

our algorithms correspond to $\alpha = 0.5$ and $\beta = 0.5$ for MQ2007-agg, Yahoo and MSLR-WEB10K datasets, and $\alpha = 0.5$ and $\beta = 0.3$ for MQ2008-agg. However, it can be noted that, for any dataset and any evaluation metric (except KT distance), the proposed method produces better result than the competitor algorithms for *moderate* values (e.g. $0 \leq \beta \leq 0.5$) of the parameter β.

7.5. The results

7.5.1. Comparing average Kendall-Tau (KT) distance

The KT value for LUCE-R is not mentioned in the corresponding paper. The KT scores obtained by the other unsupervised algorithms are shown in Table 5. Kendall-Tau distance between two rankings is computed by counting the number of item pairs (i, j) such that i is placed above j in one ranking and below j in another. Average KT distance for the aggregate ranking is the average of the Kendall-Tau distances of the aggregate ranking with all the input rankings. For this measure, lesser value indicates better performance. It is clear from the table that QSORT is the best algorithm for this evaluation metric. In fact QSORT is designed to optimize the average KT distance of the aggregate ranking from the input rankings. On the other hand, our algorithm WT-INDEG performs poorly according to this metric. It obtains very high KT scores for all the datasets.

This can be attributed to the following facts: (a) the proposed method is a modification of EQ-INDEG which itself does not perform well according to the measure, and (b) we do not give equal importance to all the rankers during the aggregation process. We give lesser importance weights to the *poor* rankers to keep the aggregate ranking far from them. This increases the average distance of the aggregate ranking from the input rankings, which increases the average Kendall-Tau distance. It has been pointed out in Yilmaz et al. (2008), Carterette (2009) and Desarkar, Joshi, and Sarkar (2011) that Kendall-Tau distance may not be suitable for evaluating of rank aggregation algorithms. Moreover, as the datasets used for experimentation contain ground truth information in the form of document relevance, supervised evaluation metrics can be used to measure the performances of the algorithms. So we did not try to modify our algorithm to obtain better KT score, but wanted to see how it works according to the supervised evaluation metrics.

7.5.2. Comparing NDCG and Precision

We now compare the performances of the algorithms based on the supervised evaluation metrics. We first consider NDCG and Precision. For both these measures, higher values indicate better performance. A comparison of NDCG values of the different algorithms for the MQ2007-agg, MQ2008-agg, Yahoo and MSLR-WEB10K datasets are shown in Tables 6, 7, 8 and 9, respectively. Compared to the other unsupervised algorithms, our method WT-INDEG obtains better NDCG scores for all four datasets.

Performances of CPS(sup) and TRADA(sup), wherever available, are provided for reference. Supervised algorithms are expected to perform better than unsupervised algorithms. However, it is interesting to note that WT-INDEG performs better than CPS(sup) for MQ2008-agg. Trada(sup) performs better than all other algorithms for the Yahoo dataset.

Table 6
NDCG comparison for MQ2007-agg dataset. Performance of CPS(sup) is used as a reference.

Algorithm	Rank positions			
Name	**2**	**4**	**6**	**8**
WT-INDEG	**0.234**	**0.250**	**0.265**	**0.279**
EQ-INDEG	0.210	0.225	0.237	0.250
BORDA	0.201	0.213	0.225	0.238
MC4	0.179	0.195	0.206	0.218
QSORT	0.122	0.145	0.159	0.172
LUCE-R	0.233	0.245	0.258	0.268
CPS(sup)	0.332	0.341	0.352	0.362

Table 7
NDCG comparison for MQ2008-agg dataset. Performance of CPS(sup) is used as a reference.

Algorithm	Rank positions			
Name	**2**	**4**	**6**	**8**
WT-INDEG	**0.346**	**0.398**	**0.438**	**0.464**
EQ-INDEG	0.308	0.370	0.416	0.441
BORDA	0.280	0.343	0.389	0.372
MC4	0.241	0.310	0.363	0.389
QSORT	0.155	0.228	0.283	0.325
LUCE-R	0.273	0.328	0.369	0.358
CPS(sup)	0.314	0.376	0.419	0.398

Table 8
NDCG comparison for Yahoo dataset. Performance of TRADA(sup) is used as a reference. Results of LUCE-R and CPS(sup) are not available for this dataset.

Algorithm	Rank positions			
Name	**2**	**4**	**6**	**8**
WT-INDEG	**0.416**	**0.425**	**0.438**	**0.451**
EQ-INDEG	0.347	0.373	0.389	0.404
BORDA	0.350	0.370	0.386	0.400
MC4	0.333	0.357	0.374	0.392
QSORT	0.341	0.361	0.376	0.394
TRADA(sup)	0.477	0.479	0.487	0.499

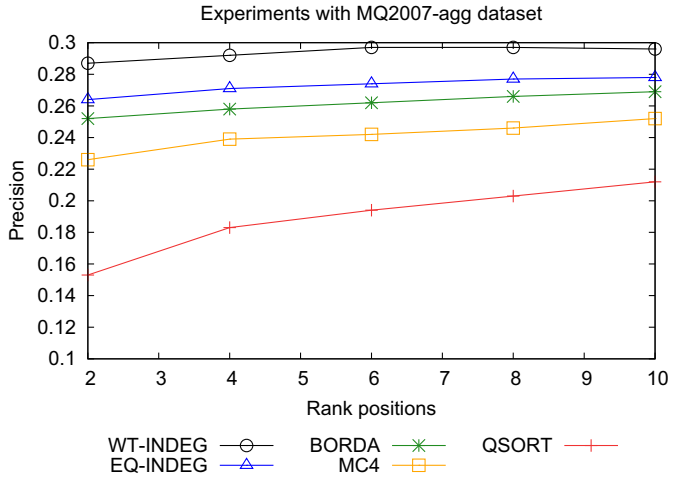
Table 9
NDCG comparison for MSLR-WEB10K dataset. Results of LUCE-R and CPS(sup) are not available for this dataset.

Algorithm	Rank positions			
Name	**2**	**4**	**6**	**8**
WT-INDEG	**0.249**	**0.251**	**0.260**	**0.265**
EQ-INDEG	0.236	0.240	0.247	0.252
BORDA	0.233	0.239	0.244	0.249
MC4	0.213	0.223	0.231	0.238
QSORT	0.211	0.218	0.222	0.231

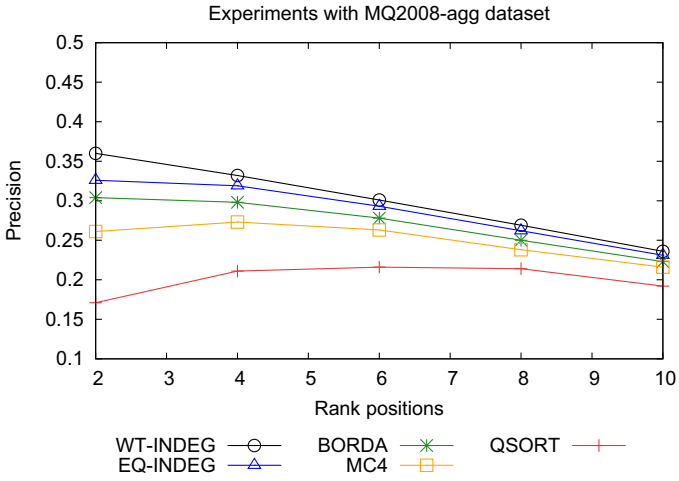
The Precision values of the algorithms are compared in Fig. 2. Comparison with LUCE-R is not shown as Precision values are not reported in the corresponding paper. From the figures it is clear that WT-INDEG is the best unsupervised algorithm according to Precision for all the four datasets.

7.5.3. Comparing MAP and mean NDCG

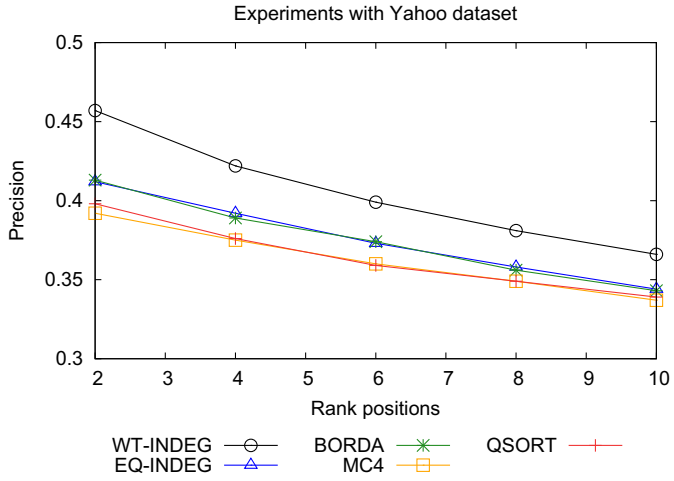
For MAP and meanNDCG measures, higher value indicates better performance. The comparison of MAP values is shown in Table 10. WT-INDEG obtains the best MAP value among the unsupervised algorithms for all the datasets. For all these datasets,



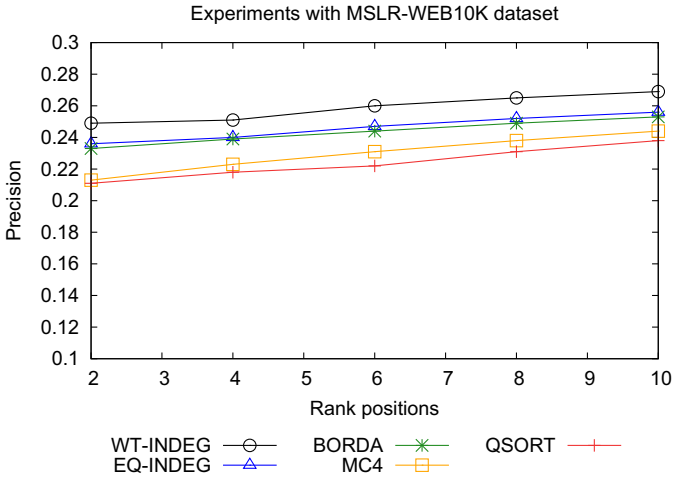
(a) Precision@k for MQ2007-agg dataset



(b) Precision@k for MQ2008-agg dataset



(c) Precision@k for Yahoo dataset



(d) Precision@k for MSLR-WEB10K dataset

Fig. 2. Comparison of Precision values of different unsupervised algorithms.

Table 10
Comparing MAP values. The results for CPS(sup) and TRADA(sup), wherever available, are shown as a reference.

	Dataset			
	MQ2007-agg	MQ2008-agg	Yahoo	MSLR-WEB10K
WT-INDEG	**0.351**	**0.430**	**0.461**	**0.277**
EQ-INDEG	0.335	0.419	0.438	0.271
BORDA	0.325	0.394	0.436	0.269
MC4	0.316	0.369	0.430	0.263
QSORT	0.276	0.301	0.433	0.259
CPS(sup)	0.407	0.410	–	–
TRADA(sup)	–	–	0.486	–

Table 11
Comparing mean NDCG. Results for CPS(sup) and TRADA(sup), wherever available, are shown as reference.

	Dataset			
	MQ2007-agg	MQ2008-agg	Yahoo	MSLR-WEB10K
WT-INDEG	**0.360**	**0.445**	**0.489**	**0.389**
EQ-INDEG	0.335	0.425	0.444	0.381
BORDA	0.322	0.390	0.442	0.378
MC4	0.309	0.372	0.430	0.372
QSORT	0.256	0.294	0.433	0.367
CPS(sup)	*0.433*	*0.413*	–	–
TRADA(sup)	–	–	0.526	–

EQ-INDEG appears as the second best unsupervised algorithm according to MAP. For MQ2007-agg, WT-INDEG achieves a 4.8% improvement over the closest baseline. For MQ2008-agg and Yahoo datasets, performance improvement by the proposed method are 2.6% and 5.2%, respectively. This improvement is 2.2% for the MSLR-WEB10K dataset.

The comparison of mean NDCG values is shown in Table 11. The results indicate that WT-INDEG performs the best among the unsupervised algorithms used for comparison according to this evaluation metric also. For all the four datasets, EQ-INDEG emerged as the second best unsupervised algorithm. For MQ2007-agg, MQ2008-agg and Yahoo datasets, the improvements by the proposed methods can be computed to be 7.5%, 4.7% and 10.1% respectively. The improvement is small (2.1%) for the MSLR-WEB10K dataset.

Table 12
Comparing ERR values. Value of TRADA(sup) for the Yahoo dataset is shown as a reference.

	Dataset			
	MQ2007-agg	MQ2008-agg	Yahoo	MSLR-WEB10K
WT-INDEG	**0.204**	**0.264**	**0.282**	**0.172**
EQ-INDEG	0.184	0.249	0.246	0.159
BORDA	0.166	0.198	0.244	0.158
MC4	0.163	0.203	0.232	0.153
QSORT	0.123	0.161	0.227	0.147
TRADA(sup)	–	–	0.326	–

7.5.4. Comparing expected reciprocal rank (ERR)

For ERR, higher value indicates better performance. To obtain high ERR value, an algorithm has to put the first relevant document near the top of the list. The comparisons are shown in Table 12. It can be seen from the table that the proposed method WT-INDEG obtains highest ERR score among the methods we compared. For MQ2007-agg, MQ2008-agg, Yahoo and MSLR-WEB10K datasets, the improvements in ERR by WT-INDEG can be computed to be 11%, 6%, 14.6% and 8.2%, respectively. High ERR score obtained by the proposed method suggests that in most of the cases, the method is able to put a relevant document near the top of the aggregate ranking.

7.6. Discussions

Unsupervised rank aggregation algorithms are generally compared against Kendall-Tau distance metric. However, as discussed in Section 7.5.1, it is better to evaluate rank aggregation algorithms against supervised evaluation measures. The proposed method WT-INDEG performed poorly according to Kendall-Tau and the explanation is given in Section 7.5.1. However, detailed evaluations using various supervised measures such as Precision, NDCG, MAP, mean NDCG and ERR indicate the efficacy of the proposed method. WT-INDEG consistently outperformed all other unsupervised algorithms for all the supervised evaluation metrics across all the datasets that we used for experimentation. Difference between the proposed method and other baselines were quite large for the MQ2007-agg, MQ2008-agg and Yahoo datasets. For MSLR-WEB10K dataset, although WT-INDEG was the best performer, all algorithms performed reasonably well. This is because the features used as rankers all are individually good and intuitively, they individually are strong signals for the relevance of a document for a query. As a consequence, the rankings induced from the individual features were *quite similar*. In other words, there were not too much disagreement between the rankings. Hence the performance of WT-INDEG was close to the performance of EQ-INDEG, which was the closest competitor for this dataset as well.

The experimental results provide a concrete evidence that Kendall-Tau distance may not be the best metric for metasearch. Average Kendall-Tau distance measures the distance of the aggregate ranking from the input rankings in terms of pairwise disagreements. On the other hand, Precision, NDCG, ERR measure the quality of the ordering in the complete aggregated ranking (including the position information), which is more important for metasearch application. The supervised metrics take into consideration the relevance labels of the document and hence aim to measure the user satisfaction for the aggregate ranking. A list that is good according to the Kendall-Tau measure may not be able to provide high user satisfaction. QSORT is the best algorithm according to Kendall-Tau distance for all the datasets, however it obtains poor scores for the supervised measures. In fact, QSORT is designed to optimize the Kendall-Tau measure. On the other hand, WT-INDEG gets the worst score according to Kendall-Tau distance. However, it appears as the best algorithm across all the datasets according to the supervised metrics. All other unsupervised methods used for experimentation achieve moderate Kendall-Tau scores but perform much better than QSORT on the supervised measures.

8. Conclusions

In this work, we propose a fast, simple, easy to implement and efficient algorithm for unsupervised rank aggregation. The algorithm has been designed keeping in mind the specific requirements for metasearch application. It assigns varying weights to the input graphs to reduce the influence of the *bad rankers* on the aggregation process. Detailed experimental comparisons against existing unsupervised rank aggregation algorithms were performed on several benchmark datasets related to web search. The proposed method consistently performed better than the other unsupervised methods used for experimentation. From complexity point of view, the proposed method runs in $O(Nm^2)$ time, where N is the number of input rankings and m is the total number of distinct items appearing in the lists. The algorithm mostly involves low cost operations such as comparison and addition. This makes the algorithm fast and suitable for applications such as metasearch which require low response time. Involvement of low cost operations in the process allows the algorithm to recompute the quality weights afresh for each query. Poor quality results produced by a ranker for a query does not affect the quality weight assigned to the ranker for other queries. This may be desirable in metasearch, since quality of results given by a single search engine may vary with queries.

Also, our experiments provide concrete evidence that Kendall-Tau distance metric is not a suitable metric for evaluating metasearch algorithms. If supervised information like relevance labels of the documents for different queries is available, then it is better to use supervised metrics for evaluating the algorithms. The proposed algorithm can also be used in other expert and intelligent systems where input rankings need to be combined, but it is not necessary to give equal importance to the input rankers. Examples of such applications are multi-criteria document selection, feature identification for data mining tasks, group recommendation, etc. However, it is necessary to evaluate the algorithm on benchmark data for these applications to understand the algorithm's performance on these tasks.

Though WT-INDEG shows good performance on real world benchmark datasets, few limitations of the work should be pointed out to complete the discussion. WT-INDEG algorithm uses two parameters to determine the qualities of the input rankings. Determining the values of these parameters in an unsupervised manner may be a challenge. It may be better to use properties or compositions of the input rankings to determine the qualities of the rankings, without needing to set up the parameter values explicitly.

As future work, we plan to improve the quality weight assignment method, with special attention to outlier detection. We would like to investigate what happens if we can detect the input rankings which are *outliers*, and whether ignoring the outlier rankers improves the quality of the aggregate ranking. Another interesting approach will be to have the pairwise preference relations labeled with the strength of the relation. For example, one can assign higher weights to a preference edge if the corresponding nodes are placed far apart in the ranking. In the paper, we demonstrated the efficacy of the proposed algorithm using detailed empirical evaluation on multiple benchmark datasets. It might be interesting to see whether the algorithm has any theoretical properties, using which it is also possible to theoretically establish the superiority of the algorithm. Since Kendall-Tau distance was shown to be not perfect for evaluating metasearch algorithm, designing of

alternative unsupervised evaluation metrics for metasearch problem would be another interesting research direction.

Acknowledgment

Work of the first author is supported by a Ph.D. Fellowship from Microsoft Research, India.

References

Acharyya, S., Koyejo, O., & Ghosh, J. (2012). Learning to rank with Bregman divergences and monotone retargeting. In *Proceedings of the twenty-eighth conference on uncertainty in artificial intelligence, Catalina Island, CA, USA, August 14–18, 2012* (pp. 15–25).

Ailon, N., Charikar, M., & Newman, A. (2008). Aggregating inconsistent information: ranking and clustering. *Journal of the ACM, 55*(5), 1–27.

Alireza, Y., & Leonardo Duenas-Osorio, Q. L. (2013). A scoring mechanism for the rank aggregation of network robustness. *Communications in Nonlinear Science and Numerical Simulation, 18*(10), 2722–2732.

Amin, G. R., & Emrouznejad, A. (2011). Optimizing search engines results using linear programming. *Expert Systems With Applications, 38*, 11534–11537.

Aslam, J. A., & Montague, M. (2001). Models for metasearch. In *Proceedings of the 24th annual international ACM SIGIR conference on research and development in information retrieval, SIGIR'01* (pp. 276–284). New York, NY, USA: ACM.

Baltrunas, L., Makcinskas, T., & Ricci, F. (2010). Group recommendations with rank aggregation and collaborative filtering. In *Proceedings of conference on recommender systems, RecSys* (pp. 119–126).

Betzler, N., Bredereck, R., & Niedermeier, R. (2014). Theoretical and empirical evaluation of data reduction for exact Kemeny rank aggregation. *Autonomous Agents and Multi-Agent Systems, 28*(5), 721–748.

de Borda, J. C. (1781). *MÃ©moire sur les Ã©lections au scrutin*. Histoire de l'AcadÃ©mie Royale des Sciences.

Brody, S., Navigli, R., & Lapata, M. (2006). Ensemble methods for unsupervised WSD. In *Proceedings of the 21st international conference on computational linguistics and the 44th annual meeting of the association for computational linguistics, ACL-44*.

Burges, C. J. C., Ragno, R., & Le, Q. V. (2006). Learning to rank with nonsmooth cost functions. In *Advances in neural information processing systems 19: proceedings of the twentieth annual conference on neural information processing systems, Vancouver, British Columbia, Canada, December 4–7, 2006* (pp. 193–200).

Cao, Y., Xu, J., Liu, T. Y., Li, H., Huang, Y., & Hon, H. W. (2006). Adapting ranking SVM to document retrieval. *Proceedings of the 29th annual international ACM SIGIR conference on research and development in information retrieval, SIGIR'06* (pp. 186–193). New York, NY, USA: ACM.

Cao, Z., Qin, T., Liu, T. Y., Tsai, M. F., & Li, H. (2007). Learning to rank: from pairwise approach to listwise approach. *Proceedings of the 24th international conference on machine learning, ICML'07* (pp. 129–136). New York, NY, USA: ACM.

Carterette, B. (2009). On rank correlation and the distance between rankings. In *Proceedings of the 32nd international ACM SIGIR conference on research and development in information retrieval, SIGIR'09* (pp. 436–443).

Chapelle, O. Chang, Y. (2010). Yahoo learning to rank challenge data. http://learningtorankchallenge.yahoo.com/datasets.php. Accessed: 27.08.15.

Chapelle, O., & Chang, Y. (2011). Yahoo! learning to rank challenge overview. In *Proceedings of the Yahoo! learning to rank challenge, held at ICML 2010, Haifa, Israel, June 25, 2010* (pp. 1–24).

Chapelle, O., Metlzer, D., Zhang, Y., & Grinspan, P. (2009). Expected reciprocal rank for graded relevance. In *Proceeding of the 18th ACM conference on information and knowledge management, CIKM'09* (pp. 621–630).

Chen, K., Bai, J., & Zheng, Z. (2011). Ranking function adaptation with boosting trees. *ACM Transactions on Information Systems, 29*(4), 18:1–18:31.

Chen, S., Wang, F., Song, Y., & Zhang, C. (2008). Semi-supervised ranking aggregation. In *Proceeding of the 17th ACM conference on information and knowledge management, Cikm'08* (pp. 1427–1428).

Chen, Y., & Hofmann, K. (2015). Online learning to rank: absolute vs. relative. *Proceedings of the 24th international conference on world wide web, WWW'15 Companion* (pp. 19–20). Republic and Canton of Geneva, Switzerland: International World Wide Web Conferences Steering Committee.

Clémençon, S., & Jakubowicz, J. (2010). Kantorovich distances between rankings with applications to rank aggregation. In *Proceedings of the 2010 european conference on machine learning and knowledge discovery in databases: part I, ECML PKDD'10* (pp. 248–263).

Cohen, W. W., Schapire, R. E., & Singer, Y. (1998). Learning to order things. In *Proceedings of the 1997 conference on advances in neural information processing systems 10, Nips'97* (pp. 451–457).

Condorcet, J. A. N. d. C. (1785). *Essai sur l'application de l'analyse a la probabilite des decisions rendues a la pluralite des voix*. Paris: Imprimerie Royale.

Copeland, A. (1951). A 'reasonable' social welfare function. In *Proceedings of seminar on applications of mathematics to social sciences*.

Coppersmith, D., Fleischer, L. K., & Rurda, A. (2010). Ordering by weighted number of wins gives a good ranking for weighted tournaments. *ACM Transactions on Algorithms, 6*, 55:1–55:13.

Davenport, A., & Kalagnanam, J. (2004). A computational study of the Kemeny rule for preference aggregation. *Proceedings of the 19th national conference on artifical intelligence, AAAI'04* (pp. 697–702). AAAI Press.

Desarkar, M. S., Joshi, R., & Sarkar, S. (2011). Displacement based unsupervised metric for evaluating rank aggregation. *Proceedings of the 4th international conference on pattern recognition and machine intelligence, PReMI'11* (pp. 268–273). Berlin, Heidelberg: Springer-Verlag.

Dwork, C., Kumar, R., Naor, M., & Sivakumar, D. (2001). Rank aggregation methods for the web. In *Proceedings of the 10th international conference on world wide web, WWW'01* (pp. 613–622).

Elkind, E., & Lipmaa, H. (2005). Hybrid voting protocols and hardness of manipulation. In *Proceedings of the 16th international symposium on algorithms and computation* (pp. 206–215). Springer-Verlag.

Fang, Q., Feng, J., & Ng, W. (2011). Identifying differentially expressed genes via weighted rank aggregation. *Proceedings of the 11th international conference on data mining, ICDM'11* (pp. 1038–1043). IEEE Computer Society.

Farah, M., & Vanderpooten, D. (2007). An outranking approach for rank aggregation in information retrieval. *Proceedings of the 30th annual international ACM SIGIR conference on research and development in information retrieval, SIGIR'07* (pp. 591–598). New York, NY, USA: ACM.

Fields, E. B., Okudan, G. E., & Ashour, O. M. (2013). Rank aggregation methods comparison: a case for triage prioritization. *Expert Systems with Applications, 40*(4), 1305–1311.

Jameson, A., & Smyth, B. (2007). *The adaptive web* (pp. 596–627). Berlin, Heidelberg: Springer-Verlag.

Jansen, B. J., Spink, A., & Koshman, S. (2007). Web searcher interaction with the dogpile.com metasearch engine. *Journal of the American Society for Information Science and Technology, 58*(5), 744–755.

Jean, M. (1961). Black (duncan) – the theory of committees and elections. *Revue Economique, 12*(4), 668.

Keyhanipour, A. H., Moshiri, B., & Rahgozar, M. (2015). Cf-rank: learning to rank by classifier fusion on click-through data. *Expert Systems with Applications, 42*(22), 8597–8608.

Klementiev, A., Roth, D., & Small, K. (2008). Unsupervised rank aggregation with distance-based models. In *Proceedings of the 25th international conference on machine learning, ICML'08* (pp. 472–479). New York, NY, USA: ACM.

Klementiev, A., Roth, D., Small, K., & Titov, I. (2009). Unsupervised rank aggregation with domain-specific expertise. In *Proceedings of IJCAI* (pp. 1101–1106).

Lee, J. H. (1997). Analyses of multiple evidence combination. In *Proceedings of the 20th annual international ACM SIGIR conference on research and development in information retrieval, SIGIR '97* (pp. 267–276).

Li, H. (2011). A short introduction to learning to rank. *IEICE Transactions, 94-D*(10), 1854–1862.

Li, P., Burges, C., & Wu, Q. (2008). Learning to rank using classification and gradient boosting. In *Proceedings of international conference on advances in neural information processing systems 20, MSR-TR-2007-74*. Cambridge, MA: MIT Press.

Lillis, D., Toolan, F., Mur, A., Peng, L., Collier, R., & Dunnion, J. (2006). Probability-based fusion of information retrieval result sets. *Artificial Intelligence Review, 25*(1–2), 179–191.

Liu, Y. T., Liu, T. Y., Qin, T., Ma, Z. M., & Li, H. (2007). Supervised rank aggregation. In *Proceedings of the 16th international conference on world wide web, WWW'07* (pp. 481–490).

Mehta, S., Pimplikar, R., Singh, A., Varshney, L. R., & Visweswariah, K. (2013). Efficient multifaceted screening of job applicants. *Proceedings of the 16th international conference on extending database technology, EDBT'13* (pp. 661–671). New York, NY, USA: ACM.

Montague, M., & Aslam, J. A. (2002). Condorcet fusion for improved retrieval. In *Proceedings of the eleventh international conference on information and knowledge management, CIKM'02* (pp. 538–548).

Monwar, M. M., & Gavrilova, M. L. (2013). Markov chain model for multimodal biometric rank fusion. *Signal, Image and Video Processing, 7*(1), 137–149.

Ozdemiray, A. M., & Altingovde, I. S. (2015). Explicit search result diversification using score and rank aggregation methods. *Journal of the Association for Information Science and Technology, 66*(6), 1212–1228.

Pan, Y., Luo, H., Qi, H., & Tang, Y. (2011). Transductive learning to rank using association rules. *Expert Systems with Applications, 38*(10), 12839–12844.

Patel, T., Telesca, D., Rallo, R., George, S., Tian, X., & Andre, N. (2013). Hierarchical rank aggregation with applications to nanotoxicology. *Journal of Agricultural, Biological, and Environmental Statistics, 18*(2), 159–177.

Pihur, V., Datta, S., & Datta, S. (2007). Weighted rank aggregation of cluster validation measures: a monte carlo cross-entropy approach. *Bioinformatics, 23*(13), 1607–1615.

Pujari, M., & Kanawati, R. (2012). Link prediction in complex networks by supervised rank aggregation. In *Proceedings of 2012 IEEE 24th international conference onTools with artificial intelligence, ICTAI: 1* (pp. 782–789).

Qin, T., Geng, X., & Liu, T. Y. (2010a). A new probabilistic model for rank aggregation. In *Proceedings of international conference on advances in neural information processing systems* (pp. 1948–1956).

Qin, T., Liu, T. Y., Ding, W., Xu, J. Li, H. (2010b). MSLR-WEB10K feature list. http://research.microsoft.com/en-us/projects/mslr/feature.aspx. Accessed: 27.08.15.

Qin, T., Liu, T. Y., Xu, J. Li, H. (2009). Letor dataset. http://research.microsoft.com/en-us/um/beijing/projects/letor/letor4dataset.aspx. Accessed: 27.08.15.

Rajkumar, A., & Agarwal, S. (2014). A statistical convergence perspective of algorithms for rank aggregation from pairwise data. In *Proceedings of the 31st international conference on machine learning, ICML-14*. In *JMLR Workshop and Conference Proceedings* (pp. 118–126).

Rosti, A. I., Ayan, N. F., Xiang, B., Matsoukas, S., Schwartz, R., & Dorr, B. J. (2007). Combining outputs from multiple machine translation systems. In *Proceedings of the North American chapter of the association for computational linguistics human language technologies* (pp. 228–235).

Sagae, K., & Lavie, A. (2006). Parser combination by reparsing. In *Proceedings of HLT/NAACL* (pp. 129–132).
Sarkar, C., Cooley, S., & Srivastava, J. (2014). Robust feature selection technique using rank aggregation. *Applied Artificial Intelligence, 28*(3), 243–257.
Schalekamp, F., & van Zuylen, A. (2009). Rank aggregation: together we're strong. In *Proceedings of ALENEX* (pp. 38–51).
Sculley, D. (2009). Large scale learning to rank. In *Proceedings of NIPS workshop on advances in ranking* (pp. 1–6).
Shao, Z., Chen, Z., & Huang, X. (2010). A mobile service recommendation system using multi-criteria ratings. *International Journal of Interdisciplinary Telecommunications and Networking, 2*(4), 30–40.
Shaw, J. A., & Fox, E. A. (1994). Combination of multiple searches. In *Proceedings of the second text retrieval conference, TREC-2* (pp. 243–252).
Sohail, S., Siddiqui, J., & Ali, R. (2015). User feedback based evaluation of a product recommendation system using rank aggregation method, *Advances in intelligent systems and computing: Vol. 320. Advances in intelligent informatics* (pp. 349–358). Springer International Publishing.
Tabourier, L., Libert, A. S., & Lambiotte, R. (2014). Rankmerging: learning to rank in large-scale social networks. In *Proceedings of dynamic networks and knowledge discovery workshop in ECML-PKDD 2014, DyNaK-II: Vol. 320*.
Thomas, P., & Hawking, D. (2009). Server selection methods in personal metasearch: a comparative empirical study. *Information Retrieval, 12*(5), 581–604.
Wang, Y., Huang, Y., Pang, X., Lu, M., Xie, M., & Liu, J. (2013). Supervised rank aggregation based on query similarity for document retrieval. *Soft Computing, 17*(3), 421–429.
Weston, J., Yee, H., & Weiss, R. J. (2013). Learning to rank recommendations with the k-order statistic loss. *Proceedings of the 7th ACM conference on recommender systems, RecSys'13* (pp. 245–248). New York, NY, USA: ACM.
Wu, G., Greene, D., & Cunningham, P. (2010). Merging multiple criteria to identify suspicious reviews. In *Proceedings of the fourth ACM conference on recommender systems, RecSys'10* (pp. 241–244).
Wu, S., Li, J., Zeng, X., & Bi, Y. (2014). Adaptive data fusion methods in information retrieval. *Journal of the Association for Information Science and Technology, 65*(10), 2048–2061.
Yilmaz, E., Aslam, J. A., & Robertson, S. (2008). A new rank correlation coefficient for information retrieval. In *Proceedings of the 31st annual international ACM SIGIR conference on research and development in information retrieval, SIGIR'08* (pp. 587–594).
van Zuylen, A., & Williamson, D. P. (2007). Deterministic algorithms for rank aggregation and other ranking and clustering problems. In *Proceeding of WAOA* (pp. 260–273).