The Impact of Ranker Quality on Rank Aggregation Algorithms: Information vs. Robustness*

Sibel Adalı, Brandeis Hill, Malik Magdon-Ismail
110 8th Street
Rensselaer Polytechnic Institute
Troy, New York 12180
{sibel, hillb, magdon}@cs.rpi.edu

Abstract

The rank aggregation problem has been studied extensively in recent years with a focus on how to combine several different rankers to obtain a consensus aggregate ranker. We study the rank aggregation problem from a different perspective: how the individual input rankers impact the performance of the aggregate ranker. We develop a general statistical framework based on a model of how the individual rankers depend on the ground truth ranker. Within this framework, one can study the performance of different aggregation methods. The individual rankers, which are the inputs to the rank aggregation algorithm, are statistical perturbations of the ground truth ranker. With rigorous experimental evaluation, we study how noise level and the misinformation of the rankers affect the performance of the aggregate ranker. We introduce and study a novel Kendall-tau rank aggregator and a simple aggregator called PrOpt, which we compare to some other well known rank aggregation algorithms such as average, median and Markov chain aggregators. Our results show that the relative performance of aggregators varies considerably depending on how the input rankers relate to the ground truth.

1 Introduction

The rank aggregation problem supposes that a set of objects are ordered by several judges. Typically, the goal is to best represent, according to some measure, the input rankers, independent of the accuracy or correctness of the individual rankers. Such an approach tends to overlook the ultimate goal, which is to obtain a ranking that is "closer" to some ground truth ranking. For Web information retrieval, data in the form of individual rankers is abundant, for example *Google, Yahoo, MSN, ...*, which are generally based upon ranking algorithms that incorporate information retrieval methods, link based algorithms and other algorithms used to compute the relevance of web pages to a given query. Unfortunately, query results of different rankers differ from each other due to the differences in ranking criteria and the specific algorithms and databases employed by specific rankers. Given this wide variety of differences, what is the best method to aggregate rankers? From a user perspective, the problem of accessing the appropriate ground truth ranking function for that user (or a group of users) is no longer equivalent to the problem of providing an overall aggregate representation of all the rankers. Rather, one must take into account how the aggregate ranker relates to the ground truth ranker.

To illustrate, imagine two sets of *bi-partisan* rankers, one representing the *left* and the other the *right* points of view. Given these two sets of rankers, is it appropriate to output a consensus ranking that represents all the rankers, in some sense rendering a non-opinion, or should one output a consensus ranking from one of these sets of rankers according to what is more appropriate for a particular user? The answer to this question is dependent on the objective of the consensus ranking: is it to somehow give a summary ranking for the population of rankers (for general queries) or is it to give a ranking that is most useful for the specific user to make actionable choices (with the consideration of user preferences). We take a step along the latter direction by investigating how the specific relationship between the individual rankers and the ground truth ranker affects the performance of the aggregate ranker *with respect to the ground truth ranker*. Thus, given qualitative criteria governing the properties of the input rankers and the preferences of the user, one is better positioned to select an appropriate consensus ranking that is most useful to the user.

The impact of the input rankers on the rank aggregation

*This work was partially supported by the National Science Foundation under grants EIA-0091505 and IIS-9876932. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

methods can be evaluated given specific knowledge regarding the input rankers. In this paper, we present a realistic statistical framework in which the dependencies between the ground truth ranker and the individual rankers can be modeled. Previous evaluations of these methods do not provide insight into how the aggregation methods perform as a function of the input rankers and how they relate to the ground truth. Especially when using real data, one of the challenges is to sample a statistically significant number of data sets to arrive at a definite conclusion for a specific scenario. We study the performance of various aggregation techniques under different model assumptions through rigorous experimental evaluation. We report on the effect of two distinct aspects of the input rankers: the amount of correct information about the ground truth rank they contain, and the amount of noise present in the data. We show that the best aggregator depends considerably on these factors and the error measure that is being optimized.

The specific contributions of this paper are:

- a statistical model for evaluating aggregation algorithms with respect to the properties of the input rankers and ground truth ranker;
- a study of rank aggregation algorithms with respect to the precision, TREC style precision and Kendall-tau error measures;
- a detailed study of how input ranker characteristics affect the performance of the aggregation algorithms, including: noise, misinformation and aymmetry among the rankers. better consensus ranking starting from an initial ranking, which could be randomly chosen or the output of some another aggregator.

2 Related Literature

Ranker aggregation has been studied extensively in the literature, to find a final ranking that has the best performance as measured by a metric such as precision or recall with respect to some ground truth or substitute relevance judgment. Some of the proposed rank aggregation methods use the ranks together with the properties of the Web pages themselves to find a final ranking. Yuwono and Lee [15] develop methods to extract scores from the ranking of objects. Meng et. al. [12] develop a number of rank aggregation methods that extract and use feature vectors from the short summaries returned by search engines for each document and use these to improve the rankings. These methods integrate information on top of the ranking of objects. As a result, we view them complementary to the problem we study here which concentrates on improving the aggregations that make use of only the rankings of objects.

One of the most common rank aggregation algorithms in the literature is Borda's method [3], which corresponds to ordering with respect to the average rank. Dwork et al. [6] introduce the notion of an aggregate ranking that minimizes the total Kendall-tau distance, which models a consensus ranking based on an analogy with voting. Since finding such an aggregation is NP-hard, they introduce a number of Markov chain models to approximate it. These rank aggregation methods are compared with well-known methods in hopes of decreasing the appearance of spam documents. Renda et al. [13] perform a more extensive study of these Markov chain methods that also compare rank-based and score-based aggregation methods. This study is performed using TREC data. Our method relies on a statistical framework that allows us to vary how rankers relate to a ground truth and test different aggregation methods with respect to a large number of data sets. These types of tests are not possible with real data sets. The insights provided by this study are therefore complementary to those found in [13] and similar studies. The Kendall-tau measure has been studied extensively [7, 8, 9] and extended to partial lists. Fagin [9] introduces algorithms to compute the median rank which is known to be robust to outliers. Chin et al. [5] present a heuristic rank aggregation algorithm that is 2-approximation to the Kendall-tau aggregator. Beg et al. [1] propose a rank aggregation algorithm which aims to optimize the footrule measure which is known to approximate the Kendall-tau measure within a factor of 2. Their genetic algorithm uses reproductions, crossover and mutation approaches observed from biology to construct the aggregate ranker. In our paper, we study the Markov chain models of approximation and also present an iterative optimization method for the Kendall-tau error measure that is competitive with many other rank aggregation methods even when it starts from a random starting point.

Supervised machine learning is another approach to the rank aggregation problem. However, this approach requires access to good training data, in particular the ground truth for the data. It does not generally give insight as to how the search engine results should be combined based on ranker quality which is our emphasis here. The study in [10] tries to learn the relevant objects of a search query by using the clickthrough data recorded by search engines themselves. The objects that users select provide information that is used to distinguish the more relevant objects from the others. The learning of the best retrieval and ranking functions for a specific user can provide better results than established search engines that are tuned for the general public as shown in this work. Our approach differs from this as we would like to learn the best aggregation methods based on the specific characteristics of search engines.

3 Basics

We use the terms *ranker* or *ranked list* interchangeably. A ranked list $\mathbf{R}$ contains a list of objects in sorted order where $\mathbf{R}(1)$ is the object with highest score (rank 1), and in general if $\mathbf{R}(m) = o_i$, then we say that object o_i has rank m, denoted by $r_{\mathbf{R}}(o_i) = m$. We will denote by $[\mathbf{R}]_K$ the partial list consisting of the top-K ranked objects (in order).

An aggregation method takes as input a number of (partial) ranked lists and produces as output another ranked list. Normally, if only the top K objects are being aggregated, then the aggregator will output the top K objects in the aggregated list. We introduce a variety of error measures to assess the degree of closeness or similarity between two rankers, $\mathbf{R}^1$ and $\mathbf{R}^2$. In the following, we use $o_i \in \mathbf{R}$ to denote that object o_i appears in list $\mathbf{R}$.

The *Precision* $pr_K([\mathbf{R}]^1_K, [\mathbf{R}]^2_K)$ gives the number of common objects in the top K of both lists. A frequently used variant of precision is the *TREC-style average precision (TSAP)*, which is given by $tsap_K([\mathbf{R}]^1_K, [\mathbf{R}]^2_K) = (\sum_i rel_i)/K$ where $rel_i = 1/i$ if the ith object in $[\mathbf{R}]^1_K$ is in $[\mathbf{R}]^2_K$ and $rel_i = 0$ otherwise. The TSAP measure takes into account not only the number of relevant objects, but also where they appear on the lists.

Another error measure is the Kendall-tau measure, $\tau_K([\mathbf{R}]^1_K, [\mathbf{R}]^2_K)$, which is the total number disagreements in the lists $[\mathbf{R}]^1_K$ and $[\mathbf{R}]^2_K$ over all pairs of objects (o_i, o_j) appearing in the union of the lists. Specifically, let $\epsilon_{ij} = 1$ if $(r_{\mathbf{R}^1}(o_i) - r_{\mathbf{R}^1}(o_j) \cdot (r_{\mathbf{R}^2}(o_i) - r_{\mathbf{R}^2}(o_j)) < 0$ and zero otherwise. Then $\tau_K([\mathbf{R}]^1_K, [\mathbf{R}]^2_K) = \sum_{i<j} \epsilon_{ij}$. Fagin et. al. [8] introduce penalty measures when there are missing objects between the lists. Since we assume each ranker ranks all the objects, we assume missing objects will appear below objects in the observed top-K. In the case when o_i, o_j both appear in one list and they are both absent in the other, we use a penalty of zero. Note that this introduces an implicit bias toward existing rankers since it assumes their ranking is correct.

3.1 Rank Aggregation Methods

In our tests, we study a number of rank aggregation methods. In all our rank aggregation methods, we process the top K objects obtained from all ranked lists. If an object is missing a rank value because it is not in the top-K, we assign a default implied rank of $K+1$. The **median (Me)** and **average (Av)** rank aggregation methods are defined in the usual way and make use of the implied ranks as well as the observed ranks.

Precision Optimal Aggregation (PrOpt) method ranks objects by the number of times they appear in the input rankers top-K list disregarding any implied ranks. We choose the top-K objects from this list. If there are ties for the K^{th} object, then we break ties with respect to the order imposed by the average aggregation method (Av). We break any further ties randomly. To our knowledge, this simple aggregation method has not been studied in the literature.

We implement a **PageRank (Pg)** aggregator which approximates the Markov chain aggregator MC_4 in Dwork et al. [6]. This chain is constructed such that the transition from an object i to j occurs if object j is ranked lower than object i for the majority of the input rankers. Even though the authors do not describe in [6] how they handle sink nodes, we believe the PageRank algorithm [4] provides a fairly good approximation for MC_4 as there is very little fluctuation in our results for different settings of the α-parameter in the PageRank algorithm. The details of our implementation are provided in the appendix. We solve the system of linear equations satisfied by the pageranks algebraically (as opposed to iteratively).

3.2 Kendall-tau Optimal Aggregators

Given a set of input rankers $\mathbf{R}^1, \ldots, \mathbf{R}^s$, and an aggregate ranker $\mathbf{R}^A$, our objective is to reduce the average error between $\mathbf{R}^A$ and $\mathbf{R}^1, \ldots, \mathbf{R}^s$ where we define the average error to be $\mathcal{E}_{av} = \frac{1}{s} \sum_{i=1}^{s} \mathcal{E}(\mathbf{R}^i, \mathbf{R}^A)$ where $\mathcal{E}(\mathbf{R}^i, \mathbf{R}^A)$ could be any one of the error measures discussed in Section 3. If the input rankers give unbiased estimates of the ground truth ranker, then minimizing the average error should also reduce the expected error with respect to the ground truth ranker. In this paper, we consider the Kendall-tau error measure, for which the minimization problem is known to be NP-hard. We implement two heuristic optimization techniques.

Adjacent Pairs (ADJ) performs local optimization as proposed in Dwork et al. [6]. The general approach is to take a ranked list and swap adjacent objects until no further improvement on the Kendall-tau measure is possible. This local optimization algorithm can be initiated from the output of any of the ranking aggregators discussed above. For example, **AvADJ** refers to the result of taking the average aggregation method and then applying the ADJ local optimization to it. For completeness, we include a description of adjacent pairs optimization in the appendix.

Iterative Best Flip (IBF) is a novel optimization that we introduce in this paper. IBF is based on an algorithm to perform local combinatorial optimization originally introduced by Kernighan and Lin [11] in the specific context of graph partitioning. The general idea behind the algorithm is to perform a sequence of greedy swaps between *any* pair of objects that eventually leads to a good local minimum of the average error. A key feature is that we perform the greedy swap even if the error increases temporarily. In this way, the algorithm has a limited amount of look ahead. We begin with an initial ranking, which could be one of the elementary aggregation algorithms and continue until no improvements are possible. For example, **AvIBF** refers to the result of taking the average aggregation method and then applying the IBF optimization to it. The detailed description of the algorithm is provided in the appendix.

3.3 Information vs. robustness trade-off

Before introducing our tests, we would like to elaborate on the properties of the above rank aggregation methods. In general, an aggregate ranker is considered a "complex" ranker if it adapts its final ranking to the subtle nuances in the input rankers. Thus, necessarily, such a complex aggregator will easily be misled by noise in the data – it is too sensitive to small fluctuations (inconsistencies) in the data, and as a result its performance rapidly degrades as such inaccuracies appear in the data. Conversely, consider the other extreme, an aggregator which considers little or none of the information contained in the rankers – for example an aggregator which completely ignores the input rankers and outputs a constant ranking. Such an ignorant ranker will have a poor performance, however its performance will not degrade as inaccuracies appear in the input rankers. Such a ranker uses less information, however it is *robust*. We consider an aggregator that uses less of the information contained in the input rankers as a "simple" aggregator even though a simple aggregator may be hard (computationally complex) to construct.

One of the simplest aggregator we introduce is the *precision optimal aggregator (PrOpt)* that disregards all information regarding the actual ranks except for the number of times an object appears in the input rankers. However, when the input rankers contain almost the same objects, then the ranks produced by PrOpt are very similar to the output of the average aggregation method which is used for breaking the ties. The *median* aggregator disregards a specific type of information since it throws away all rank information for an object except for the middle one. Hence, it is not affected by changes in the actual rank values of outliers. The *average* aggregator is one of the most complex methods in our tests since it includes all the rank values in the computation. Note that neither median nor average explicitly take into account the number of times an object appears in the input rankers.

The question is then whether optimizing for Kendall-tau introduces more or less information about the input rankers. One of the best known voting paradoxes as discussed by Saari [14] show that when aggregating votes to find the optimal ranking of candidates, the winner of pairwise elections may not be the winner of the plurality vote. We examine what this means for the Kendall-tau optimal aggregator. In the table below, we show an example of two rankers, $\mathbf{R}^1$ and $\mathbf{R}^2$, for three objects. As we can see in the table, all

$\mathbf{R}^1$	$\mathbf{R}^2$	average	Kendall-tau optimal		
o_1	o_3	o_1	o_3	o_1	o_1
o_2	o_1	o_3	o_1	o_2	o_3
o_3	o_2	o_2	o_2	o_3	o_2

three orderings given are Kendall-tau optimal with respect to the rankers $\mathbf{R}^1, \mathbf{R}^2$, while only one of them is corresponds to the unique average aggregator. So, the Kendall-tau optimal ranking that uses only pairwise comparisons seems to *ignore* some information about the rankers which the average ranker uses. To see why, note that for $\mathbf{R}^1$, when we compare o_3 to o_2 and o_1, we do not take into account the fact that o_2 is ranked below o_1 and hence o_3 is ranked third. Thus in the Kendall-tau, a flip is a flip, no matter how far apart the flipped objects are. Hence, it is insensitive to the location of the flips. On the other hand, the average takes into consideration the distance between flipped objects.

Given that Kendall-tau optimal ranking ignores some information, it leads to an aggregation method that is robust to noise. This was the main motivation behind the introduction of this optimization method in the literature [6]. The two optimization methods we study here *ADJ* and *IBF* as well as the PageRank (*Pg*) methods are approximations of the Kendall-tau optimal ranking. As our tests show, the performance of these methods depends on the initial starting ranking from which the optimization proceeds. In the case of *Pg*, we use the average rank as the starting point. We evaluate all other methods using different starting rankings. Our results show that *IBF* is a far superior optimizer than *ADJ*, hence it produces a good approximation to the true Kendall-tau optimal aggregator (even if our optimizer starts from a random ranking). Therefore the *IBF* optimized version of any ranker depends less on the input rankers than the *ADJ* optimized version. Generally, we expect that both *Pg* and the *AvIBF* contain less information about the location of flips than the average aggregator. However, any Kendall-tau optimization possibly leads to an aggregator that contains more information than *PrOpt* especially in noisy scenarios. If we compare *Me* with *MeIBF*, we can argue that *MeIBF* possibly contains less information about the middle ranker but more information about the other rankers than *Me*.

The aggregators we study in this paper incorporate different types of information to varying degrees. This provides us with a fairly extensive set of methods to test the information and robustness trade-off in aggregation methods and highlight when a specific aggregation method outperforms the others.

4 Statistical Framework

In this section, we describe the statistical model that we use to evaluate the rank aggregation algorithms and their input rankers. Assume there are n objects $o_1, \ldots, o_n$. Denote by $r(o_i)$ the ground truth rank of object o_i. In our framework, we assume that the ground truth rank of an object (a web page in the case of a search engine) is determined using a *score* computed from a set of factors $f_1, \ldots, f_F$, where $f_\ell \in [-3, 3]$ for $\ell \in [1, F]$. Each factor f_ℓ measures some property of the object; examples of factors are an object's PageRank, the number of occurrences of the query keywords in the object's text, the amount of time the page has been live, and the frequency of updates. To simplify notation, we collect $f_1, \ldots, f_F$ into the vector $\mathbf{f}$, and write $\mathbf{f}(o_i)$ for the factors of object o_i. The *score* (or value) of object o_i, denoted V_i, is a weighted linear combination of the factors,

$$V_i = \mathbf{w}^T \mathbf{f}(o_i) = \sum_{\ell=1}^{F} w_\ell \cdot f_\ell(o_i).$$

The weight vector $\mathbf{w}$ determines the relative importance of the factors. A negative weight vector indicates that that particular factor is detrimental to the value. In our experiments we have set $\mathbf{w} > 0$ with $\sum \mathbf{w}_\ell = 1$. Collect all the scores in the vector $\mathbf{V}$, and define the factor matrix $\mathbf{F}$ in which the rows of $\mathbf{F}$ are the object factor vectors $\mathbf{f}^T$, i.e. $F_{i\ell} = f_\ell(o_i)$. Then, $\mathbf{V} = \mathbf{F}\mathbf{w}$.

For simplicity, we will assume that no two objects have the same score. The ground truth ranks $\{r(o_i)\}$ are obtained by ordering the objects according to their scores. Thus, $r(o_i) = k$ if $V_j > V_i$ for $k-1$ objects o_j and $V_i > V_j$ for $n-k-1$ objects o_j. We denote by the ground truth ranking $\mathbf{R}$ the vector containing the objects in sorted order. The input to the error measures are partial lists of the form $[R]_K$.

The top-K ground truth ranking $[\mathbf{R}]_K$ is what we would like to estimate. The available data are the top-K' rankings of some other rankers ($K' \leq n$). Thus, suppose we have other rankings $\mathbf{R}^1, \ldots, \mathbf{R}^s$ which are each somehow related to the ground truth ranking $\mathbf{R}$ (we use superscript to refer to rankers). Our statistical framework provides a natural probabilistic approach to model the relationship between ranking $\mathbf{R}^j$ and the ground truth $\mathbf{R}$. Intuitively, each *input ranker* $\mathbf{R}^j$ is an approximation to $\mathbf{R}$ constructed as follows: the input ranker attempts to measure the same factors $\mathbf{f}$ which are relevant to the ground truth ranking. However, its measurements may incur some errors, so we will write the factor matrix obtained by ranker j as $\mathbf{F}^j = \mathbf{F} + \boldsymbol{\epsilon}^j$. Ranker j may also not have the correct relative weights for the factors. Denoting ranker j's weights by $\mathbf{w}^j$, we have

$$\mathbf{V}^j = \mathbf{F}^j \mathbf{w}^j = (\mathbf{F} + \boldsymbol{\epsilon}^j)\mathbf{w}^j.$$

The ranking $\mathbf{R}^j$ is obtained by ordering objects according $\mathbf{V}^j$. The top-K' lists $[\mathbf{R}^j]_{K'}$ are the inputs to the aggregation algorithm. In this paper, we assume that all rankers rank all objects, however it is easy to generalize our framework to the case where this is not so. The ground truth ranking, and the input to the aggregation algorithms are completely specified by $\mathbf{F}, \boldsymbol{\epsilon}^1, \ldots, \boldsymbol{\epsilon}^s, \mathbf{w}, \mathbf{w}^1, \ldots, \mathbf{w}^s$. The statistical model is therefore completely specified by the joint probability distribution

$$P(\mathbf{F}, \boldsymbol{\epsilon}^1, \ldots, \boldsymbol{\epsilon}^s, \mathbf{w}, \mathbf{w}^1, \ldots, \mathbf{w}^s).$$

Such a general model can take into account: correlations among factor values (correlations in $\mathbf{F}$); correlations between factor values and ranker errors (correlations between $\mathbf{F}$ and $\boldsymbol{\epsilon}^j$); correlations among ranker errors (correlations among the $\boldsymbol{\epsilon}^j$); correlations between true weights $\mathbf{w}$ and ranker weights $\mathbf{w}^j$ (the degree of similarity between rankers and the truth); correlations between ranker weights and the errors of different rankers. A complete investigation along all these dimensions is beyond the scope of this present paper, so we make some simplifying assumptions. First, we set the true weights $\mathbf{w}$ to all equal $\frac{2}{F(F+1)}[1, 2, \ldots, F]$. We only considered two possibilities for the $\mathbf{w}^j$: $\mathbf{w}^j = \mathbf{w}$ and $\mathbf{w}^j = \mathbf{w}^R = \frac{2}{F(F+1)}[F, \ldots, 2, 1]$, which represents a biased treatment of the factors with respect to the true weights. We only introduced correlations between the errors and the factors, so $\epsilon^j_{i\ell}$ depends only on $F_{i\ell}$. More specifically, we set the variance $Var(\epsilon^j_{i\ell})$ to be a function of $F_{i\ell}$,

$$Var(\epsilon^j_{i\ell}) = \sigma^2 \frac{(\gamma - F_{i\ell})^\delta \cdot (\gamma + F_{i\ell})^\beta}{\max_{f \in [-3,3]} (\gamma - f)^\delta \cdot (\gamma + f)^\beta}.$$

This functional dependence allows us to model spam by setting the variance in the error for negative valued factors to be large, which means they experience large errors that may propel them high into the rankings. The parameters γ, δ, β are shape parameters which determine how spam enters the rankings, and σ^2 is a parameter governing the maximum possible variance – how noisy the ranker factors are. The noise parameter σ^2 measures how much the true information is getting corrupted. The shape parameters γ, δ, β determine which information gets corrupted.

To complete the model description, the factors for each object are chosen independently and identically from a uniform distribution with variance 1, and hence lie approximately in the range $[-3, 3]$. The errors for each factor and each ranker are chosen independently from a uniform distribution with mean zero and variance given by the formula above for some choice of $\sigma^2, \gamma, \delta, \beta$. The input to the aggregation algorithm are the top-K' lists corresponding to each ranking. In our experiments we selected $K' = K$.

Let $\mathcal{A}$ generically refer to an aggregator, and let $\mathcal{E}([\mathbf{R}]_K, [\mathbf{R}_\mathcal{A}]_K)$ be an error measure, such as Kendall-tau, that measures the difference between the ground truth ranker $\mathbf{R}$ and the ranker $\mathbf{r}_\mathcal{A}$ obtained by the aggregator. In a realistic setting, $[\mathbf{R}]_K$ is not known, however in our setting, $[\mathbf{R}]_K$ is known. Thus, among the available aggregators, we can select the aggregator with the smallest average error through simulation within this statistical setting. The statistical framework can embed qualitative features of the aggregation setting through the choice of $P(\mathbf{F}, \boldsymbol{\epsilon}^1, \ldots, \boldsymbol{\epsilon}^s, \mathbf{w}, \mathbf{w}^1, \ldots, \mathbf{w}^s)$; rigorous simulation can then be used to obtain the appropriate aggregator for that particular aggregation setting.

5 Experimental Evaluation

In this paper, we study how the information and noise levels in the input rankers affects the performance of rank aggregation methods. We use five rankers, five factors and 100 objects. For the PageRank algorithm, we fixed the α parameter at 0.85 since we did not observe any significant dependence on α (when $\alpha > 0$) in our experiments. We set the ground truth weights to $\mathbf{w} = \langle \frac{1}{15}, \frac{2}{15}, \frac{3}{15}, \frac{4}{15}, \frac{5}{15} \rangle$. We model spam in our model ($Var(\epsilon^j_{i\ell})$) by setting $\delta = 5.0$ and $\beta = 0.01$. This results in smaller errors in factors with high scores and low rank, and larger errors in factors with very low scores. Hence, while good objects will have high scores, bad objects may also get high scores occasionally. We vary the variance parameter σ^2 between $0.1, 1, 5$ and 7.5 for all factors. Increasing the variance models more **noise**: higher values increase the likelihood of objects getting undeserved high scores.

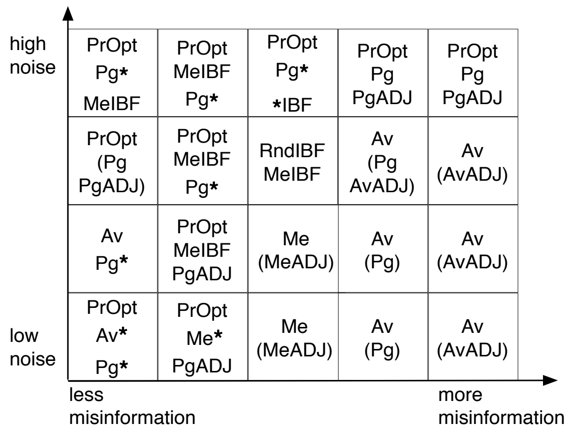
We also vary **misinformation** by changing the weights used by n_{MI} of the rankers to $\mathbf{w}' = \langle \frac{5}{15}, \frac{4}{15}, \frac{3}{15}, \frac{2}{15}, \frac{1}{15} \rangle$. The remaining $5 - n_{MI}$ rankers have the same weights as the true ranker. When $n_{MI} = 0$, there is no misinformation, all rankers have the same weights as the ground truth. As n_{MI} increases, the information about the input factors being transmitted by the rankers decreases. We call this an increase in misinformation. To see why this is different from noise, consider the case when we have infinite number of rankers. It is then possible that by averaging these rankers we are able to average out all the noise. However, information lost by the rankers that use incorrect weights can never be recovered in this case. Misinformation models the case when the rankers use weights that differ from the user's preferences. For example, the user may not care about recency of updates to a page in determining the final ranking, but the rankers may. The noise models the case where rankers incorrectly estimate the score of a factor; this is the case in many text based spam methods which result in inflated scores for specific keywords. Other examples of noise are errors made in the pagerank computation due to the incompleteness of the underlying web graph and errors in time based factors due to the frequency of crawls to a site.

Method	Description
Av	average
Me	median
Pg	PageRank
Rnd	Random
PrOpt	precision optimal
x**ADJ**	adjacent pairs opt. after aggregator x
x**IBF**	iterative best flip opt. after aggregator x

Figure 1. Legend

Given these two settings, we perform tests with and without the adjacent and iterative best flip optimization resulting in three different versions of each aggregator. We repeat each tests for 40,000 datasets where each dataset contains its own ground truth ranker and five input rankers. For each error measure, we compute the performance of the aggregation algorithms. Figure 1 lists the aggregation methods used in our tests. We should note that the precision and TSAP errors are both to be maximized, whereas the Kendall-tau error is to be minimized.

We do a pairwise comparison among all pairs from the 11 aggregation methods (see Figure 1 for notation abbreviations). We use the notation xADJ to denote adjacent pairs optimization starting from aggregator x (and similarly for xIBF). For every pair of aggregation methods A_i, A_j, we calculate the difference $(A_i - A_j)$ of the error measure values on each dataset. Based on the variance of these differences, we obtain a 99.9% confidence interval on the difference. If this confidence interval includes zero, then the two aggregators are incomparable (or equivalent). On the other hand, if the confidence interval is always positive (resp. negative), then A_j is better (resp. worse) than A_i, written $A_j > A_i$ (resp. $A_j < A_i$). These ordering relations are shown in the graphs of Figures 3, 4, 5. In each graph, an edge from aggregator A_i to A_j exists if A_i is a better aggregator than A_j for that error measure. To reduce the complexity of the graph, we remove all edges that would be implied by transitivity.


high noise
low noise
less misinformation
more misinformation

high noise	PrOpt Pg⋆ MeIBF	PrOpt MeIBF Pg⋆	PrOpt Pg⋆ ⋆IBF	PrOpt Pg PgADJ	PrOpt Pg PgADJ
	PrOpt (Pg PgADJ)	PrOpt MeIBF Pg⋆	RndIBF MeIBF	Av (Pg AvADJ)	Av (AvADJ)
	Av Pg⋆	PrOpt MeIBF PgADJ	Me (MeADJ)	Av (Pg)	Av (AvADJ)
low noise	PrOpt Av⋆ Pg⋆	PrOpt Me⋆ PgADJ	Me (MeADJ)	Av (Pg)	Av (AvADJ)

Figure 2. Summary of results for precision

Figure 2 summarizes the findings for the precision error. We use the notation $*$IBF to denote IBF starting from any initial aggregator, and $x*$ to denote aggregator x with or without optimization. In each box, we list the best two or three aggregators. If an aggregators is not equivalent to the best aggregator, then it is placed in parentheses. However, in most cases, the difference between the performance of the listed aggregators is very small. When the misinformation is low $n_{MI} = 0$ and the noise is low $\sigma^2 = 0.01, 0.1$, almost all aggregation methods are equivalent. PrOpt reduces to Av due its tie breaking methodology. When misinformation is low, as the noise increases, there is a greater need for robustness. In this case PrOpt, Pg and IBF optimized rankers become the winners. When the noise is low, as misinformation increases ($n_{MI} = 1, 2$), median becomes the dominant aggregation method as it is not effected by the outliers. This is a "bi-partisan" case where the majority of the rankers are correct, but there are one or two outliers. In these cases, as noise increases, there is a greater need for robustness. In this case, MeIBF is the clear winner. When noise is low but misinformation is high ($n_{MI} = 3, 4$), there is a greater need to incorporate as much information as possible from the input rankers. Hence, average becomes the best ranker again. This remains true even in the presence of moderate levels of noise. We note that when the noise is high, PrOpt, Pg and IBF optimization appears to be best aggregation methods.

These result for precision remain unchanged for the TSAP error measure except for high levels of noise. In the high noise cases, PrOpt and Pg appear to be winners but IBF optimization appears to loose its competitiveness. This is a surprising result as optimizing for positional information, in fact, results in a loss of information that hurts performance for a measure that relies on positional information. The results for Kendall-tau error are similar to precision as well. Note that this error measures if the objects are in relatively correct order. The first difference we note is that for high noise, PrOpt does not always do as well since it does not directly optimize for positional information. For the highest noise value and $n_{MI} = 0, 1, 2$, PrOpt performs be better than all others. However, for $n_{MI} = 4, 5$, Pg does better. For precision, both of these aggregation methods have equivalent performance. Since Pg incorporates information about the objects with missing ranks implicitly, this allows Pg to incorporate more useful information about the rankers. MeIBF appears to do very well (first or second place) in almost all noise cases for $n_{MI} = 1, 2$. Another interesting thing that we notice is that for high noise cases, IBF optimizers do not as well as PrOpt and Pg. Note that, IBF optimizer reduces the error with respect to the input rankers but ends up with worse performance with respect to the ground ranker for the high noise cases. Similar to TSAP, more information about the rankers needs to incorporated in these cases. Figures 3, 4 and 5 show the detailed results of the topological sort for various selected test cases.

6 Conclusion

In this paper, we introduce a realistic statistical framework for modeling the ranker aggregation problem. Within this framework, rankers are constructed as perturbations of the ground truth model. The ground truth ranker (not *a priori* known) has the benefit of complete and accurate knowledge of the factors and weights in the linear combination formula. The perturbations of the other rankers can be chosen to represent a number of different, realistic features that may arise in a realistic ranker aggregation problem: spam, correlation among rankers, hard vs. easy aggregation problems, outlier rankers (eg. the bi-partisan setting). The main feature of the statistical framework is that, given some estimates of the characteristic properties of the ranker ensemble (such as symmetry and level of noise), one can quantitatively investigate the performance of different aggregation methods. This gives a more principled approach to selecting an aggregation method for the particular application setting. We performed an experimental evaluation of several aggregators using our statistical framework and provided initial guidelines for choosing an aggregator based on the information used and robustness of the rankers, and the aggregation setting. To summarize our conclusions, if computational complexity is a concern, then one of Av, Me and PrOpt will usually perform well, depending on the ranker properties: when there is little noise or asymmetry among the rankers, Av is good; when there is significant asymmetry among the rankers, then Me is good; and, when there is significant noise, then PrOpt is good. Pg and Kendall-tau optimized aggregators are difficult to compute, however they appear to perform well generally. In particular, IBF appears most robust and performs well when there is noise and some asymmetry among the rankers. Further, IBF can be used in conjunction with any other aggregator as a starting point. Contrary to the conclusion in [6] we did not find any settings in which ADJ systematically outperforms the other aggregators.

The complexity robustness tradeoff is not unfamiliar in machine learning and appears in the bias/variance trade off and the VC-dimension/generalization tradeoff [2]. We see that this is an important tradeoff to keep in mind in the rank aggregation problem as well.

References

[1] M. M. S. Beg and N. Ahmad. Soft computing techniques for rank aggregation on the world wide web. *World Wide Web: Internet and Web information Systems*, 6(1):5–22, 2003.

[2] C. M. Bishop. *Neural Networks for Pattern Recognition*. Clarendon Press, Oxford, 1995.

[3] J. C. Borda. Mémoire sur les élections au scrutin. In *Histoire de l'Académie Royale des Sciences*, 1781.

[4] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. In *Proceedings of ACM WWW*, pages 107–117, 1998.

[5] F. Y. L. Chin, X. Deng, Q. Fang, and S. Zhu. Approximate and dynamic rank aggregation. *Theoretical Computer Science*, 325(3):409–424, 2004.

[6] C. Dwork, R. Kumar, M. Naor, and D. Sivakumar. Rank aggregation methods for the web. In *Proceedings of ACM WWW*, pages 613–622, 2001.

[7] R. Fagin, R. Kumar, M. Mahdian, D. Sivakumar, and E. Vee. Comparing and aggregating rankings with ties. In *Proceedings of ACM PODS*, pages 47–58, 2004.

[8] R. Fagin, R. Kumar, and D. Sivakumar. Comparing top k lists. *SIAM J. Discrete Mathematics*, 17(1):134–160, 2003.

[9] R. Fagin, R. Kumar, and D. Sivakumar. Efficient similarity search and classification via rank aggregation. In *Proceedings of ACM SIGMOD*, pages 301–312, 2003.

[10] T. Joachims. Optimizing search engines using clickthrough data. In *Proceedings of ACM SIGKDD*, pages 133–142, 2002.

[11] B. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *Bell Systems Technical Journal*, 49(1):291–307, 1970.

[12] Y. Lu, W. Meng, L. Shu, C. Yu, and K.-L. Liu. Evaluation of result merging strategies for metasearch engines. In *Proceedings of the 6th international Conference on Web Information Systems Engineering (WISE)*, 2005.

[13] M. E. Renda and U. Straccia. Web metasearch: Rank vs. score based rank aggregation methods. In *Proceedings of ACM SAC*, pages 841–846, 2003.

[14] D. Saari. *Basic Geometry of Voting*. Springer-Verlag, 1995.

[15] B. Yuwono and D. L. Lee. Server ranking for distributed text retrieval systems on the internet. In *Proceedings of the Fifth International Conference on Database Systems for Advanced Applications (DASFAA)*, pages 41–50, 1997.

APPENDIX

PageRank (Pg) We use the original pagerank algorithm [4] with specific parameters. Each distinct object (from the input rankers) represents a node in the graph $G = (V, E)$. A directed edge from object o_i to object o_j is introduced in E for each ranker which ranks o_i above o_j including the implied ranks. The link is given weight $w(o_i, o_j)$ that is proportional to the difference of ranks it represents. We normalize the weights so that outgoing edges have total weight of 1 for each node. The pagerank $Pg(o_i)$ of an object o_i is given by

$$Pg(o_i) = (1-\alpha)p_i + \alpha * \sum_{(o_j,o_i)\in E} \frac{Pg(o_j) * w(o_i, o_j)}{outdeg(o_j)}$$

where $outdeg$ is the outdegree of a node. The probability of randomly jumping to a site is proportional to the indegree of that node where $p_i = \frac{indeg(o_i)}{\sum_{o_j \in V} indeg(o_j)}$. This measure approximates the ranking produced by the average rank aggregator.

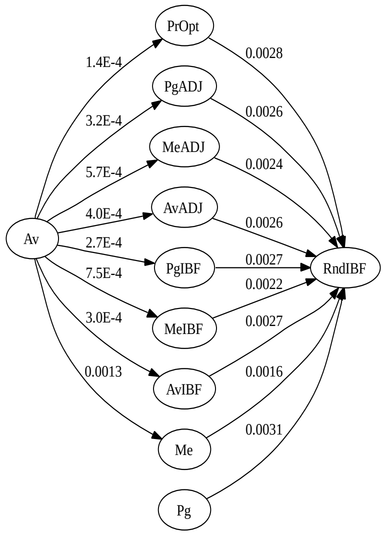
Adjacent Pairs (ADJ) Algorithm Given input rankers $\mathbf{R}^1, \ldots, \mathbf{R}^s$ and an initial aggregate ranker $\mathbf{R}^A$, this algorithm aims to reduce the average Kendall-tau error $\mathcal{E}_{av}$ between $\mathbf{R}^A$ and $\mathbf{R}^1, \ldots, \mathbf{R}^s$.

```
1: for each object $[o_i]$ in $\mathbf{R}^A$ do
2:     swap $o_i$ with $o_{i+1}$ in $\mathbf{R}^A$
3:     compute $\mathcal{E}_{av}$ after the swap;
4:     if $\mathcal{E}_{av}$ reduced then
5:         permenantly swap objects
6: repeat for-loop until no further reductions can be performed
7: return $\mathbf{R}^A$
```

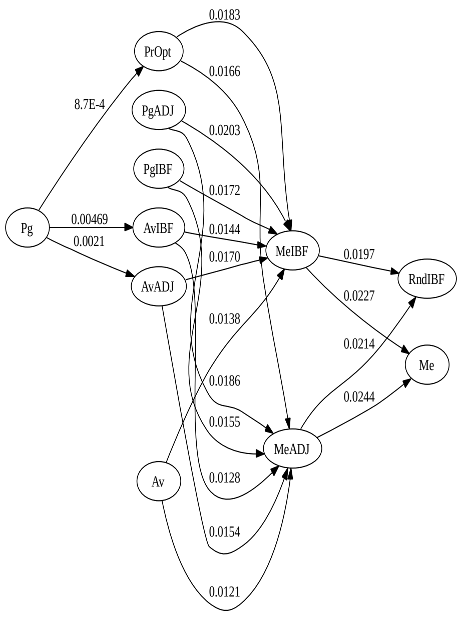
Iterative Best Flip (IBF) Algorithm Given input rankers $\mathbf{R}^1, \ldots, \mathbf{R}^s$ and an initial aggregate ranker $\mathbf{R}^A$, this algorithm also aims to reduce the average Kendall-tau error $\mathcal{E}_{av}$ between $\mathbf{R}^A$ and $\mathbf{R}^1, \ldots, \mathbf{R}^s$.

```
1: repeat
2:     $\mathbf{R}^{old} = \mathbf{R}^A$, $Config = \langle \mathbf{R}^A \rangle$, $finished = false$;
3:     for each object $[o_i]$ in $\mathbf{R}^A$ do
4:         for every possible swap $[o_j]$ in $\mathbf{R}^A$ do
5:             Compute $\mathcal{E}_{av}$ after the swap;
6:         Perform the swap with minimum $\mathcal{E}_{av}$ in $\mathbf{R}^A$; {$\mathcal{E}_{av}$ may increase as a result}
7:         Add $\mathbf{R}^A$ to $Config$
8:     Let $\mathbf{R}^{new}$ be the ranking in $Config$ with the minimum $\mathcal{E}_{av}$;
9:     if $\mathbf{R}^{new}$ has smaller error than $\mathbf{R}^{old}$ or $\mathbf{R}^{new}$ has the same error as $\mathbf{R}^{old}$ but is a new configuration then
10:        $\mathbf{R}^A = \mathbf{R}^{new}$
11:    else
12:        $finished = true$
13: until finished
14: return $\mathbf{R}^A$
```

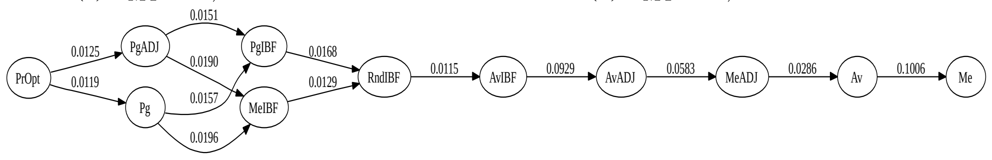
Note that the algorithm is *forced* to make a swap when considering each object sequentially (according to some arbitrary ordering). The best swap is made even if this leads to a temporary increase in the average error. It is exactly this flexibility which has been found to help the algorithm escape from bad local minima. A straightforward implementation which computes the average error after each swap would have computational complexity $O(s \cdot f(n) \cdot n^2)$ where $f(n)$ is the cost of computing the average error for an aggregate ranking. We are using the Kendall-tau error measure, for which $f(n) = O(n^2)$. By performing a pre-processing step which allows us to update the average error, instead of recomputing it from scratch, each time a swap is made, we improve the computational complexity to $O(n^3)$. We postpone the details of the implementation to a full version of the paper.



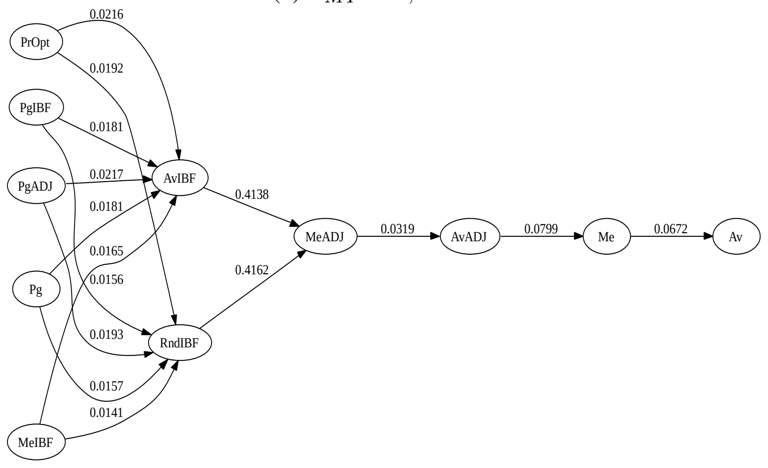
(a) $n_{MI} = 0, \sigma^2$=0.10



(b) $n_{MI} = 0, \sigma^2$=1.0

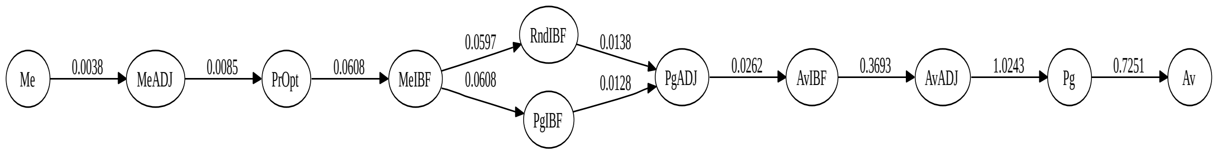


(c) $n_{MI} = 0, \sigma^2$=5.0

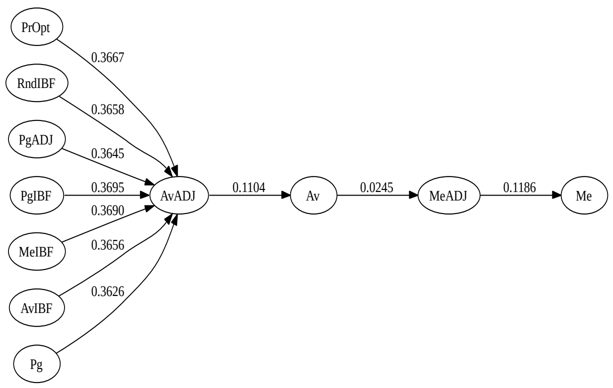


(d) $n_{MI} = 0, \sigma^2$=7.5

Figure 3. Precision error for $n_{MI} = 0$ with different levels of noise

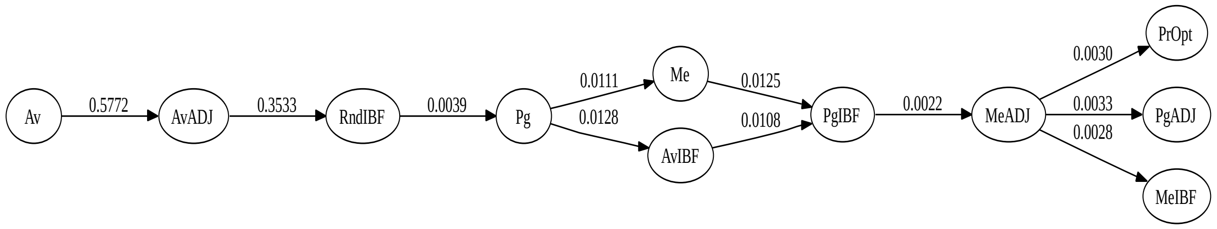


(a) $n_{MI} = 2, \sigma^2 = 0.10$

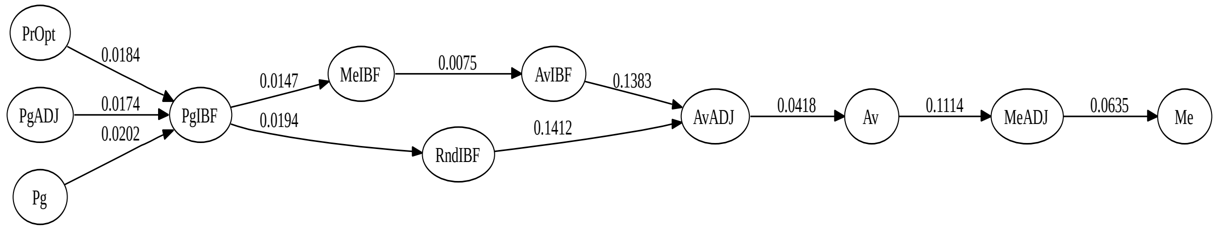


(b) $n_{MI} = 2, \sigma^2 = 7.5$

Figure 4. Precision error for $n_{MI} = 2$ with different levels of noise



(a) $n_{MI} = 4, \sigma^2 = 0.10$



(b) $n_{MI} = 4, \sigma^2 = 7.5$

Figure 5. Precision error for $n_{MI} = 4$ with different levels of noise