Information vs. Robustness in Rank Aggregation: Models, Algorithms and a Statistical Framework for Evaluation*†

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

Journal of Digital Information Management

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 generate synthetic data sets and 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, CombMNZ 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.*

Categories and Subject Descriptors
H.3.1 [Content Analysis and Indexing] **G.3.8** [**Statistics and Probability]**; Statistical Computing **I.2.7 [Natural Language Processing**]

General Terms
Ranking algorithms, Search engines, Statistical testing, Rank aggretation

Keywords: Rank aggregation algorithms, Kendall-tau rank aggregator, Rank aggregators

Received 7 August 2006; Revised 11 March 2007; Accepted 12 April 2007

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's 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). The impact of the input rankers on the rank aggregation methods can be evaluated given specific knowledge regarding the input rankers.

Problem Statement. In this paper, we present a study of rank aggregation methods as a function of their relationship to the ground truth. In contrast with previous work which concentrates on the performance of the aggregation methods with respect to a given ground truth, our evaluation concentrates on generating different scenarios for rankers and run comparisons. In particular, we study the effect of noise, misinformation, missing objects and other factors that represent how rankers may be inaccurate representations of the ground truth.

Methodology. We present a realistic statistical framework in which the dependencies between the ground truth ranker and the individual rankers can be modeled. Within the framework of our statistical model, the relationship between the individual rankers and the ground truth is specified quantitatively by a set of parameters. We study the performance of various aggregation techniques under different model assumptions through rigorous experimental evaluation using synthetic data sets generated from this statistical framework. We designed a closed-system in which

* This work was partially supported by the National Science Foundation under grants IIS-0324947, CNS-0323324, 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.

†A preliminary version of this work was published in the *Proceedings of the International Workshop on Challenges in Web Information Retrieval and Integration (WIRI)* 2006.

we generate the ground truth ranker and individual rankers that are used as input into the rank aggregation algorithms. Our closed-system provides a structure to test the performance of each aggregator with respect to the ground truth ranker. 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 rank aggregation algorithm, which we call an *aggregator*, depends considerably on these factors and the error measure that is being optimized.

1.1 Our Contributions

We introduce a statistical framework for studying rank aggregation algorithms.We give a rigorous experimental study of several rank aggregators within this framework, including two novel rank aggregation methods which we have developed. We study different ranker characteristics and provide recommendations for which aggregation method to use, depending on the ranker characteristics.

Statistical Framework. We introduce a factor-based statistical framework in order to study the aggregation problem. Within this framework, the ground truth ranker measures a set of factors. Based on these factors, the ground truth ranker determines a ranking by sorting according to a score, which is a weighted linear combination of the factors. The rankers estimate the same factors to obtain their ranking, but make some mistakes. The *aggregation scenario* is therefore specified by the probabilistic relationship between the ranker measured factors and the ranker weights in the weighted linear combination as compared to the ground truth factors and weights. We assume that the ranker measured factors are contaminated by *measurement noise* (errors); there can also be correlations between the errors for different objects and among the different rankers for the same object. Different aggregation scenarios are obtained by varying the magnitudes of the measurement errors and the correlations.

Within the statistical framework, it is then possible to repeatedly generate instances of the aggregation problem, and compare the performance of various aggregate ranking algorithms with the ground truth, averaged over instances. Since CPU cycles are cheap, the accuracy of such a study can be made arbitrarily precise.

Rank Aggregation Algorithms. We study rank aggregation algorithms with respect to the precision, TREC style precision and Kendall-tau error measures. Among the aggregation algorithms we study are some standard ones – the average, median and Kendall-tau optimizers. We also introduce and study some new aggregators: some optimized variants of Markov chain aggregators based on PageRank; a simple precision optimal aggregator which we have not found mentioned in the literature and a novel iterative best flip Kendall-tau aggregator. The Kendall-tau optimization method that we introduce has $O(n^3)$ running time and it is competitive with many other rank aggregation methods even when it starts from a random starting point. We compute each aggregation method against the rankers assuming the ground truth is not known and then evaluate their effectiveness with respect to the ground truth. We find that for a number of aggregation scenarios, these new algorithms perform comparable and sometimes better than the existing ones.

The Information vs. Robustness Tradeoff in Aggregator Performance. We give a detailed study of how the aggregation scenario (input ranker characteristics) affect the performance of the aggregation algorithms. In particular, the performance of a particular aggregation algorithm can vary depending on the aggregation scenario. In particular, we find that two important aspects of the aggregation scenario are the level of noise in the rankers and the level of misinformation they contain (misinformation arises when the rankers use different weights to combine the factors as does the ground truth ranker). Aggregation methods can also be categorized by how robust they are to misinformation and noise. Intuitively, an aggregator which ignores "irrelevant" parts of the ranker data are robust to noise in some sense, whereas an aggregator which pays attention to every detail in the ranker data can easily be mislead by noise and misinformation. Our experiments reveal that there is a tradeoff, in that the more information an aggregator uses, the less likely it is to perform well in adverse aggregation scenarios. The optimal aggregation method can vary significantly with the aggregation scenario (level of noise and misinformation).

Real Data. Our experimental results reveal which aggregation methods work well in which aggregation scenarios. In a practical setting, one does not know *apriori* which aggregation scenario one is in, hence it is not easy to determine which method to use. We report on results using the Text REtrieval Conference (TREC) datasets. Given the first K (K = {5, 10, 20, 50}) documents from a set of five input rankers, we show how aggregation methods perform for different values of K. We show that PageRank is the optimal aggregator for TREC 3,5 and 9. We see that as the noise increases from TREC-3 to TREC-5 and TREC-9, there is a greater need for robustness and PrOpt, CombMNZ appear in the top rankers. We also present results using the output of real search engine queries and based on human evaluation of the aggregator outputs. Based on these (small sample) results, we conclude that the Precision Optimal or PageRank aggregation is optimal with the context of our limited sample size. Based on a simplistic analysis of the ranker data, we conclude that the aggregation scenario is the high-noise, small misinformation case, where our statistical framework indicates that the optimal aggregator should be the Precision Optimal aggregator, which is in accordance with our results.

1.2 Related Work

Rank aggregation has been studied extensively in the literature with the objective of finding 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 scores obtained from the properties of the ranked objects to find a final ranking. Yuwono and Lee [20] develop methods to extract scores from the ranking of objects. Meng et. al. [15] develop a number of rank aggregation methods for meta-searching that extract and use feature vectors from the short summaries for each document returned by search engines and use these to improve the rankings. These methods integrate score information on top of the ranking of objects.

A great deal of algorithms concentrate on using ranks only without access to any additional training data. These are the class of algorithms we study in this paper. One of the most common rank aggregation algorithms in the literature is Borda's method [4], which corresponds to ordering with respect to the average rank. One of the most efficient algorithms, CombMNZ [10, 14], orders objects with respect to each object's average rank and hit count (number of input rankers that contain that particular object). Dwork et al. [7] introduce the notion of an aggregate ranking that minimizes the total Kendall-tau distance to the input rankers, which models a consensus ranking based on an analogy with voting. Since finding such an aggregation is NPhard [2], 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 in meta-search. Spam indicates that a document has an undeservingly high rank which is a real problem in the case of web search. Resistance to spam may be used to indicate that the rank aggregation is resistant to noise in the rankers. Montague et al. [16] propose another meta-search algorithm, Condorcet-fuse which represents the ranked information in a graph where each weighted directed edge orders two documents based on whether the majority of the rankers rank the objects in this order. The algorithm uses quicksort to construct a total ordering of objects where each partition is consistent with the majority of the edges for the pivot node. We study all these methods in this paper together with the new IBF Kendall-tau optimization, our optimized Markov chain aggregator called PageRank and the PrOpt aggregator.

Another set of rank aggregation algorithms consider the availability of training data. Aslam et al. [1] introduces the Bayesian inference, Borda-fuse and Bayes-fuse aggregation methods which use the Bayes' rule for assessing relevance of objects. Similarly, Lilis et. al. [6] computes the probability that a document is relevant to a query based on possibly incomplete relevance judgements. Both of these algorithms consider ranked lists as training data with true/false relevance judgements. In contrast, the study in [12] tries to learn the ordering of relevant objects by using the clickthrough data recorded by search engines themselves. The order in which users click the data is used to infer the relevance ordering of objects. This data is fed to a learning module that learns the significance of various features such as object features and their ranking in search engines in obtaining a final ranking. These papers show the availability of training data may greatly improve the performance of rank aggregation methods. However, the type of training data required for this type of work may be too detailed and/or hard to get in some cases. For this reason, we concentrate on rank aggregation methods that only consider rank values.

The area of research most relevant to this is the study of the performance of different ranking algorithms. Most of the prior work in this area [16, 17, 19] uses the TREC data or special subsets of this data. In some cases, search engine results together with informal relevance judgements [7] have been introduced. While the TREC data provides a common base for testing, it provides data for a single scenario concerning noise and misinformation settings. Hence, it is not possible to generate data for different scenarios. The exact relationship of this data to the ground truth is not known as the ground truth is a relevance judgement, not a ranking. Finally, this data set is limited in size, making it hard to produce tests with high statistical significance. We introduce the statistical framework to solve this problem. The framework allows us to generate synthetic data to test the rank aggregation algorithms with respect to specific noise and misinformation scenarios. We show that the best algorithm changes based on these different scenarios, signaling a trade-off between informativeness and robustness of these methods. While most of the studies concentrate on the relative performance of different algorithms, Vogt [19] measures the improvement of performance when more and more rankings are aggregated. Shows that generally more rankers provide better aggregations. Our results show that while this is true when rankers are noisy versions of the ground truth, in cases where there is misinformation in the rankers, it might be better to disregard some of the rankers to achieve a better ranking.

2. Basics

In this section, we describe the terminology used and the rank aggregation methods that are studied in this paper. We use the terms *ranker* or *ranked list* interchangeably. A ranked list **R** contains a list of objects in sorted order where **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 implement a variety of performance 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 **R**. Let $DB(\mathbf{R})$ be the set of objects that appear in an ranker **R** and $|DB(\mathbf{R})|$ be the number of objects. We define aggregate ranker R^A as the sorted list of objects returned by an aggregation method.

The *Precision* $pr_K([R]^1_K, [R]^2_K)$ gives the number of common objects in the top K of both lists. A frequently used variant of precision is the *TRECstyle average precision (TSAP*), which is given by $tsapK([\mathbf{R}]^1_K, [\mathbf{R}]^2_K) = (\sum rel_i)/\mathbf{R}_K$ 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 in the lists.

Another performance measure is the Kendall-tau measure, $T_K([\mathbf{R}]^1_K, [\mathbf{R}]^2_K)$, 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. Kendall-tau measures the sortedness of one ranked list with respect to the other. Specifically, let $\varepsilon_{ij} = 1$ if $(r_{R^1}(o_i) - r_{R^1}(o_j)) * ((r_{R^2}(o_i) - r_{R^2}(o_j)) < 0$ and zero otherwise. Then $T_K([\mathbf{R}]^1_K, [\mathbf{R}]^2_K) = \sum_{i<j} \varepsilon_{ij}$. Fagin et. al. [9] introduce penalty measures when the two rankers are not identical. In our statistical framework, we assume rankers rank the same database of objects. As a result, we assume missing objects will appear below objects in the observed top K and use the default ranking $K+1$ for these objects. In the case when o_i, o_j both appear in one list and they are both absent in the other, we assume there is no disagreement. Note that this introduces an implicit bias toward existing rankers since it assumes their ranking is correct.

2.1 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.

2.1.1 CombMNZ

The prior work of Fox and Shaw [10] and Lee [14] present the CombMNZ algorithm, initially using scores [10] and later using ranks [14]. This algorithm first computes the Borda rank of each object and then multiplies this value by the number of times the object appears in different input rankers. The objects are then ordered in decreasing order of their modified Borda scores. More formally, we implement CombMNZ as follows. We now denote $DB(\mathbf{R}^A) = DB(\mathbf{R}^1) \cup \ldots \cup DB(\mathbf{R}^s)$ for s rankers. Given rankers $\{\mathbf{R}^1, \ldots, \mathbf{R}^s\}$, we perform Borda rank normalization (*brn*) for each object $o \in DB(R^A)$, as presented in Renda et al. [17].

$$\text{if } o \in R^i, \text{ then } brn^i = \frac{r_{R^i}(0) - 1}{|DR(R^A)|}, \text{ otherwise } brn^i = 0. \quad (1)$$

Now we can compute the aggregate score $sc(o)$ for each object o. We denote $h(\{\mathbf{R}^1, \ldots, \mathbf{R}^s\}, o)$ as the number of times object o appears in the rankers (in range [1,s]). The set of aggregate scores are then sorted in decreasing order.

$$CombMNZ: sc(o) = h(\{\mathbf{R}^1, \ldots, \mathrm{R}^s\}, o) * \sum_{i=1}^{s} brn^i(o) \quad (2)$$

2.1.2 Precision Optimal Aggregation (PrOpt)

The precision optimal aggregation 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. This is a simplified version of the **CombMNZ** algorithm discussed above.To our knowledge, this simple aggregation method has not been introduced in the literature.

2.1.3 PageRank (Pg)

We implement a Markov chain aggregator similar to MC_4 in Dwork et al. [7] with some modifications. In general, our model is a weighted version of MC_4. We found our setting of weights outperforms all other weighted and unweighted versions of this algorithm that we tested. In our model, if the current state is o_i, then the next state is chosen at random from any object that is ranked higher by some other engine. The transition probability from o_i to o_j is proportional to the number of rankers that rank o_i higher than o_j and the differences in the ranks of these two objects for these rankers. If two rankers rank o_i higher than o_j with ranks (1, 5) and (2, 3) respectively, then the probability of transition is proportional to (5 - 1) + (3 + 2) after normalization. While we do not explicitly transition only when majority of the rankers rank the object higher as in MC_4, we achieve this implicitly by the weights of the edges as described in detail below.

In this formulation, it is possible that a strongly connected component (SCC) of the graph may have no outgoing edges to the remainder of the graph, resulting in a sink component. Dwork et al. [7] solve this problem by first obtaining a partial ranking for this SCC and then by removing it to repeat the process. We note that the same problem is tackled in the PageRank algorithm [5] algorithm that also has an equivalent Markov chain formulation by the introduction of random jump probabilities. We adopt the PageRank solution to this problem and introduce outgoing edges (called random jumps) with a small probability from each node to every other node, given by (1- α)/N where N is the number of objects. We call this method PageRank (**Pg**). We observe almost no variation in the results for different settings of the -parameter (from 0.5 to 0.99) which we set to 0.85 in our tests. In the high values for α,the random jumps have almost no effect in the ranking. As a result, we do not expect a big difference between our results and the method described in [7].

We solve the system of linear equations satisfied by the pageranks exactly using algebraic methods (as opposed to iteratively). Note that Dwork et. al. [7] report on results that approximately compute the steady state probability. Formally, given the input rankers, we construct a graph $G = (V,E)$ where each object is a node. For each ranking where o_i is ranked higher than o_j , we introduce a directed edge (o_j , o_i) with weight equal to the difference in ranks. We then 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

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

where *outdeg* is the outdegree of a node. The probability of randomly jumping to a site is proportional to the in-degree (p_i) of that node where $p_i = \frac{indeg(o_i)}{\sum_{o_i \in V} indeg(o_j)}$

The in-degree measure approximates the ranking produced by the average rank aggregator for the random jump event.

2.1.4 Condorcet-fuse (CFuse)

The Condorcet-fuse algorithm [16] constructs a graph $G = (V,E)$ where V = DB(RA) and $e(o_i, o_j) \in E$ is an unweighted directed edge where $o_i \rightarrow o_j$ (or $o_j \rightarrow o_i$) indicates that o_i (o_j) dominated o_j (o_i) in the majority of rankers. Thus, for every pair of objects, there is at most one edge. The objects are then sorted in such a way to remain consistent with the directed edges. Therefore, the objects are listed such that all edges are forward edges. In the case of ties, the objects are randomly ordered.

2.1.5 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 improve the performance of $\mathbf{R}^A$ with respect to $\mathbf{R}^1, \ldots, \mathbf{R}^S$ where we define the performance as $\mathcal{E}_{av} = \frac{1}{s} \sum_{i=1}^{s} \mathcal{E}(\mathbf{R}^i, \mathbf{R}^A)$ wher $\mathcal{E}(\mathbf{R}^i, \mathbf{R}^A)$ could be any one of the performance measures discussed previously (such as precision, Kendall-tau or TSAP). If the input rankers give unbiased estimates of the ground truth ranker, then improving the performance with respect to the rankers should also improve the performance with respect to the ground truth ranker. In this paper, we consider the Kendall-tau error measure for optimization, for which the minimization problem is known to be NP-hard. We implement two heuristic optimization techniques. Note that the algorithms we give below are generic search algorithms and can make use of any performance measure.

Adjacent Pairs (ADJ) performs local optimization as proposed in Dwork et al. [7]. 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 rank aggregation algorithms 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 below. Given input rankers $\mathbf{R}^1, \ldots, \mathbf{R}^s$ and an initial aggregate ranker $\mathbf{R}^A$, this algorithm aims to improve the performance measure $\mathcal{E}_{av}$ 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** $\mathcal{E}_{av}$ improved then
5: permanently swap objects
6: repeat for-loop until no further reductions can be performed
7: return $\mathbf{R}^A$

Iterative Best Flip (IBF) is a novel optimization that we present. IBF is based on an algorithm to perform local combinatorial optimization originally introduced by Kernighan and Lin [13] in the specific context of graph partitioning. The general idea of the algorithm is to perform a sequence of greedy swaps between any pair of objects that eventually leads to a good local optimum of the given performance measure. A key feature is that we perform the greedy swap even if the performance gets worse 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 rank 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.

Given input rankers $\mathbf{R}^1, \ldots, \mathbf{R}^s$ and an initial aggregate ranker $\mathbf{R}^A$, this algorithm also aims to improve the performance measure εav 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 εav after the swap;

6: Perform the swap with the best εav in $\mathbf{R}^A$; { εav may get worse as a result}

7: Add $\mathbf{R}^A$ to *Config*

8: Let $\mathbf{R}^{new}$ be the ranking in *Config* with the optimum εav;

9: **if** $\mathbf{R}^{new}$ has better performance than $\mathbf{R}^{old}$ or $\mathbf{R}^{new}$ has the same performance 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 reduction in the performance. It is exactly this flexibility which has been found to help the algorithm escape from bad local minima/maxima. A straightforward implementation which computes the performance after each swap would have computational complexity $O(s \cdot f(n) \cdot n^2)$ where $f(n)$ is the cost of computing the performance measure. We are using the Kendall-tau error measure for measuring performance, 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 $O(n^3)$.

2.2 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, 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 aggregators 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 [18] 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 three orderings given are Kendall-

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

tau optimal with respect to the rankers $\mathbf{R}^1$,$\mathbf{R}^2$, while only one of them 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[7]. The two optimization methods we study here **ADJ** and **IBF** as well as the PageRank (**Pg**) and Condorcet Fuse (**CFuse**) are approximations of the Kendall-tau optimal ranking. As our tests show, the performance of the optimization 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 ptimizer 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. Similarly, CFuse starts

from a random starting point and uses only majority opinion, disregarding the value of the differences in the ranks. As a result, it is a worse approximation of the Kendall-tau optimal than **IBF** and **Pg**, but still a simple ranker as it disregards a great deal of information. 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.

3. Statistical Framework

In this section, we describe the statistical model that we use to generate synthetic data sets to evaluate the rank aggregation algorithms and their input rankers. Our statistical model is based on hypothetical search engines that rank objects based on multiple factors. We assume that each engine tries to find the best ranking by approximating a ground truth, but the approximation is imperfect for a number of reasons that are explained in detail below. In our model, we generate both a ground truth and then the imperfect rankers. In our evaluation, we assume the ground truth is not known during aggregation as it is the case in the real world. However, we are able to assess how well aggregators perform with respect to the ground truth after we compute the aggregation.

3.1 Ground Truth

Assume there are n objects $o_1, \ldots, o_n$. The ground truth denotes the ranking of objects with respect to some specific query. We denote $r(o_i)$ to be the ground truth rank of an 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 $f_l \in$ - [3, 3] for $l \in [1, F]$. 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. These factors have been used in the search engine literature in the past [5, 8] to compute both the relevance of a page as well as its quality. To simplify notation, we collect $f_1, \ldots, f_F$ into the vector f, and write $f(o_i)$ for the factors of object o_i. The ground truth score (or value) of object o_i, denoted V_i, is a weighted linear combination of the factors,

$$V_i = w^T \mathrm{f}(o_i) = \sum_{l=1}^{F} w_l \cdot f_l(o_i).$$

The weight vector **w** determines the relative importance of the factors. A negative weight vector indicates that the particular factor is detrimental to the value, or a low value for a factor is considered good. In our experiments we have set $\mathbf{w} > 0$ with $\sum w_l = 1$ which means a high value for a factor is always considered good. For simplicity, we will assume that no two objectsvhave the same score. The ground truth ranks $\{r(o_i)\}$ are obtained by ordering the objects in decreasing order of their scores. Thus, $r(o_i) = k$ if $V_j > V_i$ for k -1 objects and $V_j > V_i$ for n -k -1 objects. We denote by the ground truth ranking ***R*** the vector containing the objects in sorted order.

We collect all the ground truth scores for all objects in the vector V, and define the factor matrix **F** in which the rows of **F** are the object factor vectors **f** , i.e. $F_{il} = f_l(o_i)$. Then, $\mathbf{V} = \mathbf{Fw}$.

3.2 Rankers

We assume the existence of rankers $\mathbf{R}^1, \ldots, \mathbf{R}^s$ are each somehow related to the ground truth ranking **R** (we use superscript to refer to rankers). In fact, each ranker is trying to approximate the ground truth as best as it can. Our statistical framework provides a natural probabilistic approach to model the relationship between a ranking $\mathbf{R}^j$ and the ground truth **R**. Intuitively, each input ranker $\mathbf{R}^j$ is an approximation to **R** constructed as follows: the input ranker attempts to measure the same factors 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 $\mathbf{R}^j$ as $\mathbf{F}^j = \mathbf{F} + \mathrm{e}^j$. Ranker $\mathbf{R}^j$ may also not have the correct relative weights for the factors. Denoting ranker $\mathbf{R}^j$'s weights by $\mathbf{w}^j$, we have

$$\mathbf{V}^j = \mathbf{F}^j \mathbf{w}^j = (\mathbf{F} + \mathrm{e}^j)\mathbf{w}^j.$$

The ranking R^j is obtained by ordering objects according to scores in V^j . The top K lists $[\mathrm{R}^j]K$ are the inputs to the aggregation algorithm. The ground truth ranking, and the input to the aggregation algorithms are completely specified by $\mathrm{F}, \mathrm{e}^1, \ldots, \mathrm{e}^s, \mathbf{w}, \mathbf{w}^1, \ldots, \mathbf{w}^s$. The statistical model is therefore completely specified by the joint probability distribution

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

Such a general model can take into account various other types of correlations such as correlations among factor values (correlations in $\mathbf{F}$); correlations between factor values and ranker errors (correlations between $\mathbf{F}$ and e^j); correlations among ranker errors (correlations among the e^j); correlations between true weights **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. We discuss these in detail below.

We study two sets of weights for the factors given in the table below.

Weight vector	Values
w	$\frac{2}{F(F+1)}$ $[1, 2, \ldots, F]$
w^R	$\frac{2}{F(F+1)}$ $[F, \ldots, 2, 1]$

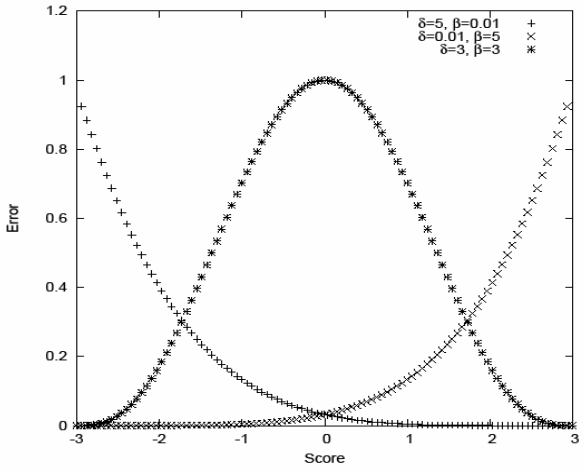


Figure 1. Possible shapes of the correlation between factors and the magnitude of errors

We note that while the actual rank formula used by different search engines, modeled here by rankers is not known, it is clear that the final rankings of search engines differ from each other. We associate this to two factors, the differences in the way the scores for factors are computed and the differences in the weights or the general combination formula. For example, the recent phenomenon called *Google Bombs* [11] show that at the time of these attacks Google attached a higher importance to two factors compared with other engines: recency of updates to a page and the number of times a keyword appears in links pointing to a page. This resulted in specific pages achieving a high rank in Google for specific keywords used in the anchor text pointing to this page. However, other search engines were not effected similarly by these effects. In the next section, we model some correlations that may exist between the errors and factors on top of this model. We give examples why such correlations may exist due to the often complex business relationships between search engines.

3.3 Correlations in the system

The first type of correlation we introduce is between the errors and the factors (defined by α^2, γ, δ, β), where e_{il}^j depends on F_{il}. In our model, we construct the error given to an object o_i's factor f_l in ranker $\mathbf{R}^j$ that would not outweigh the factor values assigned to the ground truth ranker **R**. More specifically, we set the variance $Var(e_{il}^j)$ to be a function of F_{il},

$$Var(e_{il}^j) = \sigma^2 \frac{(\gamma - F_{il})^\delta . (\gamma + F_{il})^\beta}{\max_{f \in [-3,3]} (\gamma - F_{il})^\delta . (\gamma + F_{il})^\beta}$$

The parameters γ, δ, β are shape parameters which determine how noise 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 β, γ, δ determine which information gets corrupted. We use this type of noise to model spam, i.e. pages receiving undeservedly high rank. In other words, while the ground truth score of a factor may be low, a ranker may give it a high score with some probability. A common example of spam is padding of text with many keywords. The ground truth in this case may correctly determine that the score of this page for this keyword should be low while the ranker may make a mistake and give it a high score. The distribution of noise shows us how this type of error occurs.

The functional dependence between the scores and the magnitude of the errors allows us to model different types of errors that the rankers may make. Figure 1 shows three possible shapes of this function for $\gamma = 3$ and $\sigma^2 = 1$. Along the x-axis, we have the possible values for an object's factors, in which higher values are good. Along the y-axis, we display the amount of error assigned to be added to the factors. When $\beta = \delta$ (denoted by the ' * ' symbol), then the errors are lowest for objects with the highest and lowest values. This setting allows objects appearing the the middle of the ranking to be affected by the additive errors but the ordering of the top objects remain consistent with the ground truth ranker. When $\beta << \delta$ (denoted by the 'x' symbol), the values are shifted upward which does not changes the ordering of the objects in the rankings. When $\delta >> \beta$ (denoted by the '+' symbol), the low valued objects are more likely to make large mistakes; hence increases their value and elevates them high into the rankings. This might be due to the adversarial techniques used by the engines to mislead the algorithms used by the rankers to estimate a specific factor. We use this last setting in our tests to model spam.

It is also possible to model missing objects by assuming with some probability p_m^i ranker i does not have a specific object in its database and does not rank this object at all.

Other correlations modeled in our system are as follows:

σ_f Correlation between two factors of the same object in the ground truth ranker.

This models whether two factors measure the same quantity or independent quantities. For example, the length of page and frequency of keywords would be correlated positively. Similarly, if we are counting the frequency of a keyword in title, text and the outgoing links (to measure a hub value for the page) than it is likely that there will be a correlation. However, this correlation can be negative as higher occurrence of a keyword in one field may imply a lower occurrence of another keyword for two texts of the same length.

σ_n Correlation between two objects for the same factor in the ground truth ranker.

This models the distribution of values for a single factor over all the objects. Hence, if the correlation is low, the objects take more or less random values. Otherwise, if there is a positive correlation, one object having low score may imply another having low score. So, if scores are very close to each other as a result, the problem becomes harder since the objects are indistinguishable from each other. For example, for very popular keywords, the distribution will be very dense meaning there will be lots of pages with the same frequency of keywords. However, for keywords that have political connotations, one can imagine a bipartisan situation. Certain objects from one party having a high occurrence may imply a low occurrence for the objects from the other party, leading to a negative correlation.

σ_{nf} Correlation between two different objects for different factors in the ground truth ranker.

This models when the value of an object for a factor may depend partially on the value of another object for another factor. It is possible to model this as $\sigma_n * \sigma_f$. It makes little sense to have a non-zero value for $sigma_{nf}$ when the other two are zero. For example, suppose we consider two objects, o_1 and o_2, where one of the factors of o_1 is the pagerank of o_1 which links to o_2 and one of the factors of o_2 is the frequency of keywords in the anchor links for o_2. Since a search engine may include the keywords of the link from page o_1 in the content of o_2, then the frequency depends on the number of number of incoming links to o_1. But, the page rank of o_1 depends on the number of its outgoing links and where the links lead to, which forms a cycle. So, there is a correlation between these two objects through different factors.

σ_f^* Correlation between the errors made by rankers for two factors of the same object (one set for each ranker).

The errors made by rankers for two different factors may depend on each other since rankers use similar algorithms for both. For example, frequency of keywords in anchor text and in regular text may be independent of each other. But, if the same algorithm for stemming and categorization is used then the algorithm makes similar errors in both. Another reason the errors of two factors may be correlated is when they depend on the underlying

.

method for normalizing the scores will make errors that have dependence on each other. Finally, time dependent factors will make errors that depend on the time of measurement and the actual time a specific change was made.

σ^*_n Correlation between the errors made by rankers for two objects for the same factor (one set for each ranker).

This simply means that the algorithm makes similar mistakes for two objects. If the value of the factors for two objects are correlated, then this is a reasonable assumption. For example, two rankers may use the same database of web pages to compute the pagerank independently. Even though the computation may differ, the pagerank values depend on the underlying graph. If the pagerank of a page is underestimated due to missing edges, then the pagerank of all the pages that are pointed to this page will also be underestimated resulting in a positive correlation.

σ^*_{nf} Correlation between the errors made by rankers for two different objects for different factors (one set for each ranker).

Suppose the ranker makes an error determining when a page was last updated and the last update time is a factor. Now suppose this page links to another page and the page rank depends on when the links are added (i.e. pages accumulating links very quickly are demoted). Then, this would effect the pagerank of pages that it links to and make the errors correlated.

σ^*_R Correlation among rankers.

Basically this models the case of a ranker making errors that correlate with the errors another ranker makes. An easy example of this is the case where a ranker may use the output of another ranker. This is an explicit relationship in the case of a meta-search engine. There are also other implicit relationships where a ranker may rely on another partially for different query types such as directory lookups or sponsored links.

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 σ^2 , γ , δ , β . The input to the aggregation algorithm are the top *K'* lists corresponding to each ranking. In our experiments we selected $K' = K$.

Let A generically refer to an aggregator, and let $\varepsilon([\mathbf{R}]_K, [\mathbf{R}^A]_K)$ be a performance measure, such as Kendall-tau, that measures the difference between the ground truth ranker **R** and the ranker $\mathbf{R}^A$ 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 best performance through simulation within this statistical setting. The statistical framework can embed qualitative features of the aggregation setting through the choice of $P(\mathbf{F}, \mathbf{e}^1, \ldots \mathbf{e}^s, \mathbf{w}, \mathbf{w}^1, \ldots, \mathbf{w}^s)$; rigorous simulation can then be used to obtain the appropriate aggregator for that particular aggregation setting.

4. Experimental Evaluation

In this paper, we study how the information and noise levels in the input rankers affect the performance of rank aggregation methods. First, we present tests using the synthetic data sets generated by the statistical model. Then, we validate our findings from the synthetic data sets against real data.

4.1 Synthetic Data

We use the statistical model described in Section 3 to generate data sets with a single ground truth using five factors and 100 objects each. We retrieve top 10 from each ranker and compute the precision and Kendall-tau for top 10 objects. We set the ground truth weights to $\mathbf{w} = (\frac{1}{15}, \frac{2}{15}, \frac{3}{15}, \frac{4}{15}, \frac{5}{15})$.

We generate five rankers with different relationship to the ground truth. All rankers have the same spam distribution ($Var(e^j_{il})$) obtained by setting δ = 5.0 and β = 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. For each ranker, we vary the variance parameter σ^2 between 0.1, 1, 5 and 7.5. Increasing the variance models more **noise**: higher values increase the likelihood of objects getting undeserved high scores. The noise level is the same for all factors for a specific test.

We also vary misinformation by changing the weights of a number (n_{MI}) of rankers to $\mathbf{w}^R = (\frac{5}{15}, \frac{4}{15}, \frac{3}{15}, \frac{2}{15}, \frac{1}{15})$. The remaining 5 -$n_{MI}$ rankers have the same weights as the ground truth 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 a value formula that differ from the ground truth. For example, a ranker may consider recency of updates to be a more important or reliable factor in ranking than the ground truth. 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
CombMNZ	CombMNZ
Me	median
Pg	PageRank
Rnd	Random
CFuse	Condorcet-fuse
PrOpt	precision optimal
x**ADJ**	adjacent pairs opt. after aggregator x
x**IBF**	iterative best flip opt. after aggregator x

Figure 2. Legend

Given these two settings, we perform tests with and without the adjacent and iterative best flip optimization for Av,Me,Pg resulting in three cent and iterative best flip optimization for **Av,Me,Pg** resulting in three different versions of each aggregator. The optimization is performed with respect to the Kendall-tau error

	n_{MI}=0	n_{MI}=1	n_{MI}=2	n_{MI}=3	n_{MI}=4
σ^2=7.5	PrOpt, CombMNZ Pg PgADJ	PrOpt, CombMNZ Pg PgADJ	PrOpt, CombMNZ Pg PgADJ	PrOpt, CombMNZ Pg PgADJ	PrOpt, CombMNZ Pg PgADJ
σ^2=5.0	PrOpt, CombMNZ Pg, Pg ADJ	PrOpt, CombMNZ Pg Pg ADJ	PrOpt, CombMNZ MeIBF Pg, RndIBF	PrOpt, CombMNZ Pg Av	PrOpt, CombMNZ Pg Av
σ^2=1.0	CombMNZ PrOpt, Pg, Av Pg ADJ, PgIBF	MeIBF Pg ADJ Cfuse	Me Me ADJ Cfuse	Av Pg CombMNZ	PrOpt CombMNZ Av
σ^2=0.10	PrOpt Av, Pg *ADJ, *IBF, Cfuse, CombMNZ Me	Pg ADJ, Cfuse CombMNZ MeIBF PrOpt, MeADJ	Me MeADJ Cfuse	Pg Av CombMNZ	PrOpt CombMNZ Av

(a) Summary of results for TSAP (with legend)

	less misinformation				high misinformation
high noise	PrOpt, Pg, CombMNZ MeIBF, RndIBF AvIBF	PrOpt, Pg, IBF CombMNZ Cfuse AvADJ	PrOpt, Pg, CombMNZ, IBF Cfuse AvADJ	PrOpt, Pg CombMNZ, Pg ADJ Pg, IBF MeIBF, RndIBF	Pg PrOpt, PgADJ CombMNZ Pg, IBF
	PrOpt, CombMNZ Pg, Pg ADJ Pg IBF	PrOpt, PgIBF MeIBF, RndIBF CombMNZ, Pg, PgADJ AvIBF Cfuse	MeIBF, RndIBF AvIBF. PgIBF PrOpt, CombMNZ	Av Pg AvADJ, PgADJ	Av AvADJ Pg
	PrOpt, Pg*, Av*, Me, CombMNZ MeADJ, Cfuse MeIBF	MeIBF, PgIBF, CombMNZ, Cfuse, PgIBF	Me MeADJ Cfuse	Av Pg AvADJ	Av AvADJ Me, Pg
low noise	Av PrOpt,*ADJ, *IBF, Cfuse, CombMNZ RndIBF	PgADJ, CombMNZ PrOpt, MeADJ, Cfuse, PgIBF, Me	Me MeADJ PrOpt	Av Pg AvADJ	Av AvADJ RndIBF

(b) Summary of results for precision (P@10)

	less misinformation				more misinformation
high noise	PrOpt, CombMNZ Pg PgADJ	PrOpt, CombMNZ Pg PgADJ	PrOpt, CombMNZ Pg, PgADJ, MeIBF RndIBF	Pg PrOpt,CombMNZ PgADJ	Pg PgADJ PrOpt, CombMNZ
	PrOpt, CombMNZ Pg, Pg ADJ	PgADJ Pg, PgIBF, MeIBF, RndIBF AvIBF, CombMNZ	MeIBF, PgIBF AvIBF. RndIBF PgADJ	Pg Av PrOpt, CombMNZ Av	Pg AvADJ
	Pg Av CombMNZ	MeIBF RndIBF PgADJ	MeADJ Me MeIBF	Av Pg PrOpt	Av PrOpt CombMNZ
low noise	PrOpt Av, Pg *ADJ, *IBF, Cfuse, CombMNZ Me	MeIBF MeADJ Cfuse	Me MeADJ MeIBF	Av Pg CombMNZ	PrOpt CombMNZ Av

(c) Summary of results for Kendall-tau

Figure 3. Summary of results for the baseline case

measure. We compute the aggregations using the rankers only, assuming the ground truth is unknown. We then evaluate the performance of the aggregation with respect to the ground truth to measure how well the aggregation would have done in a specific noise and misinformation scenario. We should note that the precision and TSAP are measures of accuracy to be maximized, whereas the Kendall-tau is an error to be minimized. Figure 2 lists the aggregation methods used in our tests. Note that we experimented with the PageRank algorithm using many different settings, including uniform distribution for random jump probabilities and no explicit weights for the edges. We found that the settings given in Section 2.1 provides the best results when compared with the ground truth.

Ordering Aggregators. We repeat the tests for a specific noise and misinformation setting (as well as other parameters studied later) for 40,000 datasets where each dataset contains its own ground truth ranker and five input rankers. We do a pairwise comparison among all pairs from the 11 aggregation methods. 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 performance measure values on each dataset. Based on the variance of these differences, we obtain a 99.9% confidence interval on the difference (3 *sigma*). If this confidence interval includes zero, then the two aggregators are incomparable (or equivalent). On the other hand, if we consider the case of precision error measure where the confidence interval is always positive (resp. negative), then A_i is better (resp. worse) than A_j , written $A_i > A_j$ (resp. $A_i < A_j$). These ordering relations are shown in the graphs of Figures 5, 6, 7. 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.

4.1.1 Noise vs. misinformation, baseline case

Figure 3(a) through (c) summarizes the findings for the TSAP (Trec style precision), precision at top 10 and the Kendall-tau error measure. Each cell in the box represents the summary of the tests on 40,000 different data sets. We use the notation **IBF* to denote starting from any initial aggregator (**Av, Me** and **Pg** and then performing the IBF optimization, and *x** to denote aggregator *x* with or without optimization. In each box, we list the best two or three aggregators together with the differences in the given performance measure between top 1 and 2, and top 2 and 3 aggregators.

When the misinformation is low n_{MI} = 0 and the noise is low σ^2 = 0.01, 0.1, almost all aggregation methods are equivalent with respect to all aggregators. In these cases, **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**, **CombMNZ Pg** and **IBF** optimized rankers become the winners. When the noise is low, and the asymmetry between rankers 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 (**Av**) 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**, **CombMNZ, Pg** and **IBF** optimization appears to be the best aggregation methods.

We observe that in high noise cases, **PrOpt, CombMNZ** 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**, **CombMNZ** perform be better than all others. However, for n_{MI} = 3, 4, **Pg** does better. For TSAP, **PrOpt, CombMNZ** do well even in high noise and high misinformation cases. This means that while **Pg** incorporates information about the ordering of the objects, it may occasionally miss some objects resulting in a lower precision value. **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 do 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. More information about the rankers needs to incorporated in these cases. Figures 5, 6 and 7 show the detailed results of the topological sort for various selected test cases.

In Sections 4.1.2- 4.1.3, we investigate the performance of these aggregatorsfor different settings of the statistical framework. We exclude CombMNZ and Cfuse in these tests as CombMNZ appears to be very similar to PrOpt and Cfuse does not appear to be a competitive method.

4.1.2 Missing Objects

We run experiments where each ranker is missing 10% and 50% of all the objects in the database and examine how the choice of aggregator may change. We show in Figures 8 and 9 the summary of the first and second best aggregators for both the precision and Kendall-tau error measures. With missing objects, there are fewer pairs of objects to compare for **IBF**and **ADJ** optimizations and the median values are based on fewer rankings. As a result, the effectiveness of these rankers is reduced. For example, for 10% missing objects, Me is no longer the best for the cases where there is asymmetry between rankers and *IBF becomes prominent in these cases. As for 50% missing objects, we see that **Av** and **AvADJ** become very prominent for almost all the cases as more and more information needs to be incorporated. **AvADJ** provides small amount of robustness over **Av** and becomes an important aggregator.

4.1.3 Correlation Experiments

We run experiments where we vary different correlations as discussed in Section 3. In each test, we vary only one of the correlations.

Correlation between two objects for the same factor for the ground truth (σn) In these tests, we introduce a correlation between objects for one factor. We test a positive correlation in Figure 10 and negative correlations in Figure 11. We do not see a significant change in the results for positive correlations. However, for negative correlations, **PrOpt** and **IBF** become more prominent which signals a need for robustness.

Correlation between two factors of the same object for the ground truth σf In the next set of tests, we introduce correlations between two factors for the ground truth and consider the case where the correlation is positive in Figure 12

and negative in Figure 13. Again, for positive correlation, we do not see a big difference. In case of negative correlations, we see that **PrOpt** becomes prominent for precision in low noise cases and **Pg** becomes prominent for Kendall-tau error measure. In high noise cases, **Av** and **AvADJ** become the best rankers as more and more information need to be incorporated from the input rankers.

Correlation between the errors made by rankers for two objects for the same factor (**one set for each ranker**) σ^*_n In these experiments, we assign the correlated errors of two of the five rankers to be fixed at 0.60. The results are shown in Figure 14. The Precision Optimal aggregator no longer dominates as the best aggregation for high noise cases as compared to the other experiments. **Me*** is robust with respect to noise in the bipartisan case when a majority of the rankers align with the ground truth (n_{MI} = 1, 2). We observe that our IBF optimization performs well in a large number of cases, including high noise and some misinformation.

Correlation amongst rankers σ^*_R We performed experiments for positive correlations amongst all the rankers in Figure 15. We observed that with our low setting of 0.10, there was no significant difference in performance for the precision and Kendall-tau error measures as compared to our experiments without any correlations.

4.2 TREC Data Collection

We use the rankings submitted by the participants of the Text REtreival Conference (TREC) as input rankers to the rank aggregation algorithms. We use three datasets (TREC-3, TREC-5 and TREC-9) each comprising of 50 queries, which was also used in [16]. Each participant devises a system, which retrieves 1000 documents, and returns a ranking of these documents for a particular query. In TREC, human evaluators are used to determine if a document is relevant or irrelevant. The relevant documents are then compared to these rankings of up to 1000 documents using different types of performance evaluators including the precision and TREC-style average precision as described previously.

For each query, we repeat the following 50 iterations: we select the top- K (K = {5, 10, 20, 50}) documents from a randomly chosen set of 5 input rankers as input to the rank aggregation methods. We compute the TRECstyle average precision comparing the aggregate ranker to the relevant documents determined by human evaluators. For each query, we find TSAP over the 50 iterations for each value of K and aggregator. We display themean average precision over all queries for each aggregator.

We show in Tables 1- 3, TSAP over the 50 queries in each TREC data collection. We use boldface to show the best two or three aggregators for each column. The last column sums the TSAP values from each top-K for each aggregator to determine which are the best aggregators overall.

We see throughout these results that **Pg** produces the best aggregators for the majority of values of K. The **CombMNZ**, **PrOpt** aggregators also perform well for most top-K values; however these methods are not one of the best overall. We

	top-5	top-10	top-20	top-50	Best Overall
Av	0.2769	0.1805	0.1109	0.0548	0.6231
Me	0.251	0.1697	0.1051	0.0525	0.5783
Pg	**0.2884**	**0.1882**	**0.1147**	**0.0562**	**0.64751**
AvIBF	0.2835	0.1857	0.1126	0.0548	0.6366
MeIBF	0.284	0.1863	0.1131	0.0549	0.6383
PgIBF	**0.2862**	0.1865	0.1132	0.055	**0.6409**
AvADJ	0.2755	0.1815	0.1111	0.0546	0.6227
MeADJ	0.2582	0.172	0.1058	0.0527	0.5887
PgADJ	0.2516	0.1876	0.1112	0.0559	0.6063
RndIBF	0.2724	0.1759	0.1094	0.0548	0.6125
PrOpt	0.2637	0.1806	0.1113	**0.0562**	0.6118
CFuse	0.284	0.1858	0.1106	0.0531	0.6335
CombMNZ	0.2697	**0.1877**	**0.1147**	**0.0562**	0.6283

Table 1. TREC-3 TSAP Results

	top-5	top-10	top-20	top-50	Best Overall
Av	0.2289	0.1428	0.0828	**0.0387**	0.4932
Me	0.2012	0.1274	0.0758	0.0357	0.4401
Pg	**0.2346**	**0.1463**	**0.0844**	**0.0388**	**0.5041**
AvIBF	**0.232**	0.1453	0.0835	0.0385	**0.4993**
MeIBF	0.2251	**0.1458**	0.0837	0.0375	0.4921
PgIBF	0.2135	0.1405	0.0836	0.0386	0.4762
AvADJ	0.218	0.1391	0.0828	0.0385	0.4784
MeADJ	0.2078	0.1302	0.0771	0.036	0.4511
PgADJ	0.196	0.1459	0.0841	**0.0388**	0.4648
RndIBF	0.2036	0.1454	0.0807	0.0384	0.4681
PrOpt	0.2139	**0.1458**	**0.0843**	**0.0387**	0.4827
CFuse	0.2216	0.1444	0.0828	0.0382	0.487
CombMNZ	0.2077	0.1408	0.0818	0.0376	0.4679

Table 2. TREC-5 TSAP Results

	top-5	top-10	top-20	top-50	Best Overall
Av	0.176	0.1033	0.0599	0.0272	0.3664
Me	0.1418	0.0888	0.0531	0.0247	0.3084
Pg	**0.1808**	**0.1065**	**0.0608**	0.0272	**0.3753**
AvIBF	0.1741	0.105	0.0606	**0.0276**	0.3673
MeIBF	0.1786	0.1057	0.0607	0.0274	0.3724
PgIBF	**0.1791**	0.1052	0.0607	0.0275	**0.3725**
AvADJ	0.1764	0.1025	0.0602	**0.0276**	0.3666
MeADJ	0.1503	0.0916	0.054	0.0251	0.321
PgADJ	0.1569	0.1062	**0.061**	**0.0277**	0.3518
RndIBF	0.1761	0.1053	0.0605	0.0263	0.3682
PrOpt	0.1781	0.1063	0.0606	0.027	0.372
CFuse	0.1723	0.1051	0.055	0.0265	0.3589
CombMNZ	0.1768	**0.1065**	0.0606	0.02591	0.3698

Table 3. TREC-9 TSAP Results

notice *x***IBF** and *x***ADJ** are among the aggregators that perform second best. We observe that certain aggregators perform best under different circumstances. We note that Me and its variants perform poorly indicating no misinformation. This is to be expected as the rankers are sampled uniformly from a larger set. We have also computed the variation in the ranks of each TREC data set to check if there was a difference in the noise level of each data set. We found that TREC-9 had slightly lower levels of noise in the data sets we generated. These data sets still constitute a small sample compared to our previous tests. However, we can see that **PrOpt** and **CombMNZ** do not appear in the top two for TREC-9, whereas these two aggregators are among the best in the other TREC data sets. As noise increases in TREC-3 and TREC-5, there is a greater need for robustness and hence these two aggregators become more prominent. This confirms our findings from the statistical framework.

4.3 Search Engine Data

We have tested five aggregation methods on a small sample (6 queries) using the following search engines for rankers:

queries	Computer viruses, death penalty, mining coal goal silver, photography, wireless communications
rankers	Altavista, Clusty, Dogpile, Excite, Google, Looksmart, Metacrawler, MSN, Search, Teoma,Yahoo
aggregators	Average, Median, Median with IBF optimization, PageRank, Precision Optimal

For each query and every pair of aggregation methods, we determined manually which aggregator was superior. To do this, we used the pair of objects with highest rank discrepancy in the two aggregate rankings, and then manually determined which object was more relevant to the query. The results between every pair of rankers were averaged over queries. The relative performance of these aggregators on this set of queries is summarized in the figure below.

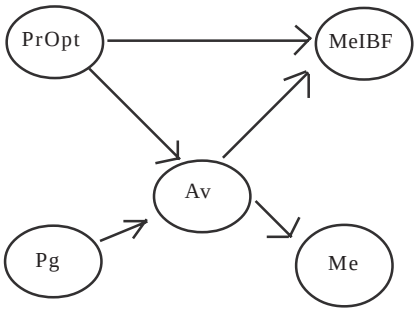


Figure 4. Small sample real dataset results

As can be observed in Figure 4, the optimal aggregator is Precision Optimal and PageRank aggregators. The average variance in the rank of an object over the rankers was high and a clustering of the rankings provided by the different rankers revealed only one significant cluster, indicating little asymmetry between the rankers. To find a cluster among the rankers, we constructed a graph where each ranker is a vertex and each edge has a weight that corresponds to the average Kendall-tau distance between the two rankers for the above studied queries. Assuming the rankers are not all using the incorrect weights, we can assume that there is little misinformation in this setting. As a result, we conclude that the we are in the high noise low misinformation aggregation scenario, and according to our results within the statistical framework, the optimal aggregator should be Precision Optimal and PageRank, which agrees with what was determined empirically. Our results indicate that through a somewhat qualitative analysis of the ranker results, the level of noise and misinformation can be roughly determined and leads to the correct choice of aggregator from within the statistical framework.

5. Conclusion

In this paper, we introduce a realistic statistical framework for creating synthetic data sets and we use model to study the rank 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, CombMNZ** 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** or **CombMNZ** are 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 some noise and asymmetry among the rankers. Further, IBF can be used in conjunction with any other aggregator as a starting point. Contrary to the conclusion in [7] we did not find any settings in which ADJ systematically outperforms the other aggregators. We also performed small sets of tests using real data from TREC Conferences and also from 11 commercial (meta)-search engines that supports our finding from the experimental evaluation. The real datasets had varying amount of noise but low misinformation.

The complexity robustness tradeoff is not unfamiliar in machine learning and appears in the bias/variance trade off and the VC-dimension/generalization tradeoff [3]. We see that this is an important tradeoff to keep in mind in the rank aggregation problem as well.

The real data results provide some evidence that our algorithms could be beneficial in a real world environment. In general, we cannot expect a single aggregation algorithm to do well in all scenarios. Search engines or any other set of input rankers may have varying levels of noise and misinformation depending on the specific query. The challenge is to develop methods to estimate the amount of noise and misinformation in the data with little user intervention and then use the most appropriate aggregator for a new scenario. However, our results also show that in presence of asymmetry between rankers, aggregating all the information may not always be the best choice. It may be desirable to either consider only the majority opinion or present

alternate rankings for each different opinion based on user preferences.

References

[1] Aslam, J. A., Montague, M (2001). Models of metasearch. *In* Proceedings of ACM SIGIR, p. 276–284.

[2] Bartholdi, J. J ., Tovey, C. A., Trick, M. A (1989). Voting chemes for which it can be difficult to tell who won the election. *Social Choice and Welfare, 6* (2) 157–165.

[3] Bishop, C. M (1995). Neural Networks for Pattern Recognition. Clarendon Press, Oxford.

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

[5] Brin, S., Page, L (1998). The anatomy of a large-scale hypertextual web search engine. *In:* Proceedings of ACM WWW, p. 107–117.

[6] Lillis, David R. C., Toolan, Fergus., Dunnion, J (2006). Probfuse: A probabilistic approach to data fusion. *In:* Proceedings of ACM SIGIR, p. 139–146.

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

[8] Eiron, N., McCurley, K.S (2003). Analysis of anchor text for web search. *In*: Proceedings of ACM SIGIR, p. 459–460.

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

[10] Fox, J., Shaw, E (1994). Combination of multiple sources: The trec-2 interactive track matrix experiment. *In:* Proceedings of ACM SIGIR.

[11] Hiler, J (2002). Google time bomb. *In:* www.microcontentnews.com/articles/googlebombs.htm.

[12] Joachims, T (2002). Optimizing search engines using clickthrough data. *In:* Proceedings of ACM SIGKDD, p. 133–142.

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

[14] J. H. Lee. Analyses of multiple evidence combination. In Proceedings of ACM SIGIR, pages 267–276, 1997.

[15] 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.

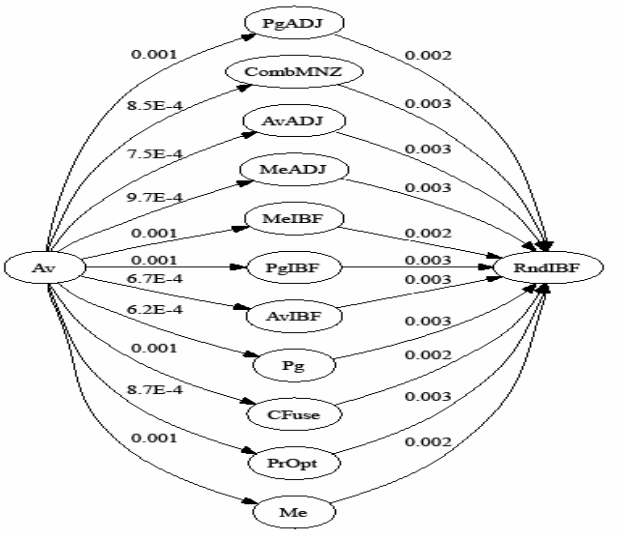
[16] Montague, M., Aslam, J.A (2002). Condorcet fusion for improved retrieval. *In*: Proceedings of ACM CIKM, p. 538–548.

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

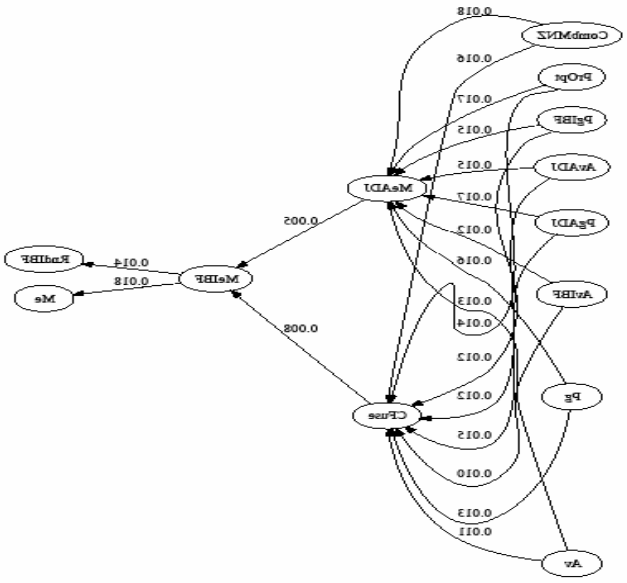
[18] Saari, D (1995). Basic Geometry of Voting. Springer-Verlag.

[19] Vogt, C.C (2000). How much more is better? characterizing the effects of adding more ir systems to a combination. *In:* Proceedings of RIAO Conference, 2000.

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



(a) $n_{MI} = 0$, σ^2=0.10



(b) $n_{MI} = 0$, σ^2=1.0

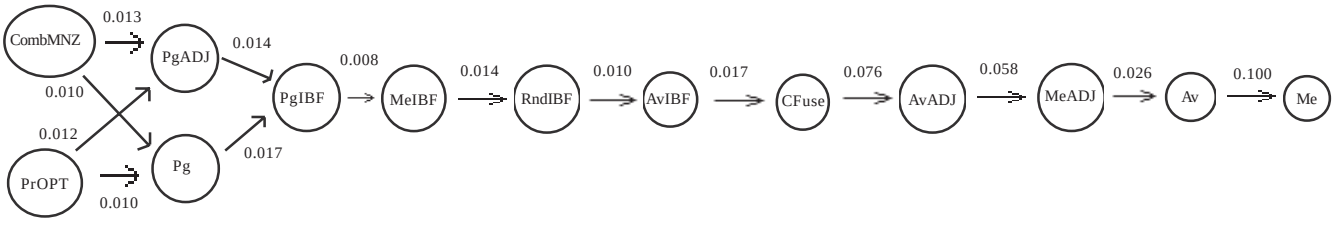
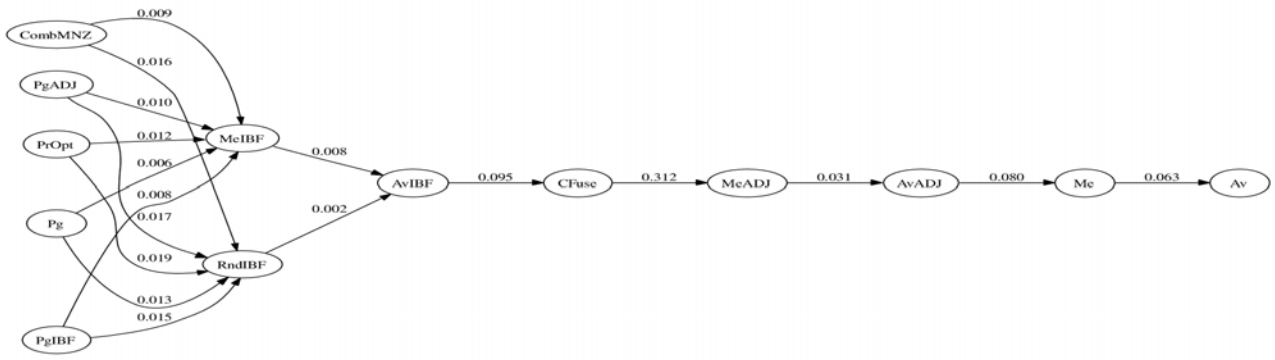


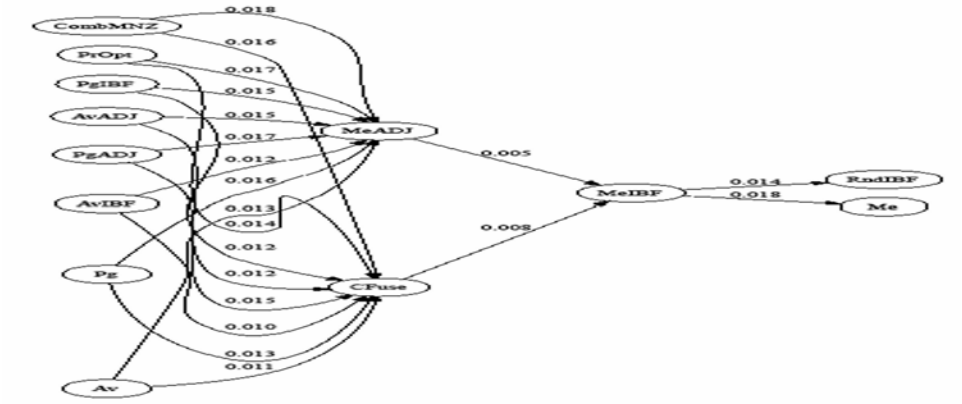
Figure 5. Precision (P@10) for $n_{MI} = 0$ with different levels of noise



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

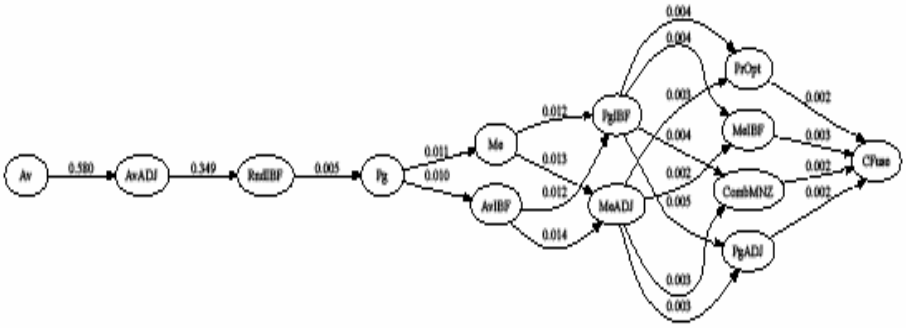
Figure 5. Precision (P@10) for $n_{MI} = 0$ with different levels of noise 40

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

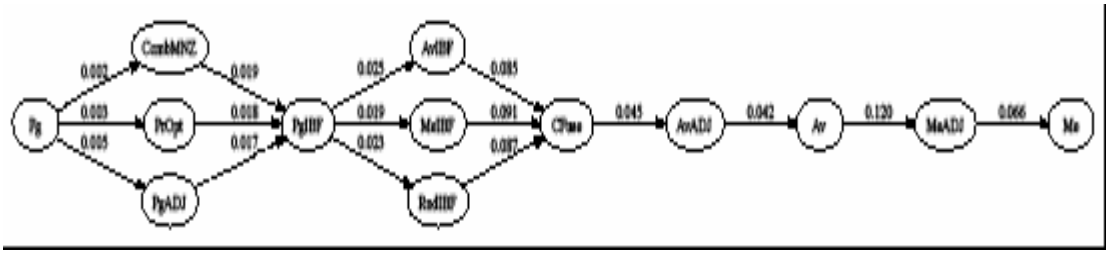


(b) $n_{MI} = 2$, $\sigma^2 = 7.5$

Figure 6. Precision (P@10) 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 7. Precision (P@10) for $n_{MI} = 4$ with different levels of noise

Pg PgADJ PrOpt	PgADJ PgIBF PrOpt	PgIBF MeIBF PrOpt	Pg PgADJ PrOpt	Pg PrOpt
PrOpt [Pg]	PrOpt [MeIBF]	RndIBF MeIBF	Av [Pg, AvADJ]	Av [AvADJ]
Av [AvADJ]	AvIBF [RndIBF,PgIBF]	RndIBF [PgIBF]	Av [AvADJ]	Av [AvADJ]
AvADJ [PgADJ]	AvIBF [PgIBF]	RndIBF [PgIBF]	Av [Pg]	Av [Pg]

high noise

low noise

less information

more information

(a) precision (P@10)

PrOpt Pg	PrOpt [Pg]	PrOpt [PgADJ]	PrOpt Pg	Pg [PrOpt, PgADJ]
PgADJ [PgIBF]	PgADJ [PgIBF]	PgIBF [AvIBF]	Av [Pg, PrOpt]	Av [AvADJ]
AvIBF [PgIBF]	RndBF [AvIBF]	RndIBF [AvIBF]	Av [Pg]	Av PrOpt
MeADJ [AvIBF]	AvIBF [AvADJ]	RndIBF [AvIBF]	Av [Pg]	Av PrOpt

less information

more information

(b) Kendall-tau error measure

Figure 8. 10% Missing Objects

PrOpt Pg	PrOpt [Pg, PgADJ]	PrOpt Pg PgADJ	PrOpt Pg PgADJ	Pg PrOpt
Av [AvADJ]	Av [AvADJ]	Av [AvADJ]	Av [AvADJ]	Av [AvADJ]
Av [AvADJ]	Av [AvADJ]	Av [AvADJ]	Av [AvADJ]	Av [AvADJ]
Av [AvADJ]	Av [AvADJ]	Av [AvADJ]	Av [AvADJ]	Av [AvADJ]

high noise

low noise

less information

more information

(a) precision (P@10)

PgIBF [PgADJ]	PgIBF [PgADJ, PrOpt]	PrOpt PgADJ PgIBF	PgADJ [Pg]	AvADJ [Av PgADJ, PgIBF]
Av [AvADJ]	AvADJ [Av]	Av [AvADJ]	Av [AvADJ]	Av [AvADJ]
Av AvADJ	AvADJ [Av]	Av [AvADJ]	Av [AvADJ]	Av [AvADJ]
AvADJ [Av]	AvADJ [Av]	Av [AvADJ]	Av [AvADJ]	Av [AvADJ]

less information

more information

(b) Kendall-tau error

Figure 9. 50% Missing Objects

PgADJ [PrOpt, Pg, PgIBF]	PrOpt PgADJ PgIBF	PrOpt [Pg*, *IBF]	PrOpt Pg*	PrOpt Pg
PrOpt Pg PgADJ	PrOpt [Pg, PgADJ, MeIBF]	PrOpt *IBF	Pg PgADJ, PrOpt]	Pg [PgADJ, PrOpt]
Pg [PrOpt, PgADJ,PgIBF]	PgIBF MeIBF	MeADJ [Me]	Av [Pg]	Av [AvADJ]
Av [*ADJ,	PrOpt MeIBF	Me [MeADJ, PrOpt]	Av [Pg]	Av [AvADJ]

high noise

low noise

less information

more information

(a) precision (P@10)

PrOpt Pg	PrOpt MeIBF	MeIBF [RndIBF, PrOpt]	PrOpt [Pg, MeIBF]	PrOpt Pg
PrOpt [Pg]	PrOpt [Pg]	PrOpt PgADJ	PrOpt Pg	PrOpt Pg
Pg [Av, PrOpt]	RndIBF [MeIBF]	MeIBF [RndIBF]	Av [Pg, PrOpt]	PrOpt Av
PrOpt Av Pg	MeIBF [MeADJ]	Me [MeADJ]	Av [Pg, PrOpt]	PrOpt [Av]

less information

more information

(b) Kendall-tau error measure

Figure 10. Objects are positively correlated for one factor of the ground truth with $\sigma n = 0.60$

PrOpt Pg* *IBF	PgADJ PgIBF	PrOpt AvIBF Pg*	PrOpt Pg	PrOpt Pg* *IBF
PrOpt Pg* *IBF	PrOpt Pg* *IBF	PrOpt Pg* *IBF	PgIBF [PgADJ, MeIBF, RndIBF]	PrOpt [Pg, Av, Me]
PrOpt [Pg]	PrOpt [Pg, PgADJ, AvIBF]	AvIBF [MeIBF,	Pg RndIBF]PgADJ PrOpt	Me [MeADJ]
Pg [PgADJ]	PgADJ [PrOpt, Pg, PgIBF]	RndIBF [PgIBF, MeIBF]	Av [AvADJ]	Av [AvADJ]

high noise

low noise

less information

more information

(a) precision (P@10)

PrOpt Pg* *IBF	PrOpt IBF, Pg]	PrOpt RndIBF	Pg [PgADJ, MeIBF, AvIBF]	Pg [PgADJ, AvIBF]
MeIBF RndIBF AvIBF]	MeIBF [RndIBF, PrOpt]	MeIBF [PrOpt,RndIBF]	Pg [PgADJ, MeIBF,	Pg [PgADJ, Me]
PrOpt [Pg]	PrOpt [Pg]	PrOpt [AvIBF]	Pg [PgADJ, PrOpt]	Me [Av]
Pg [Av]	PgADJ [PgIBF, RndIBF]	RndIBF [MeIBF, AvIBF]	Av [AvADJ]	Av [AvADJ, PrOpt]

less information

more information

(b) Kendall-tau error measure

Figure 11. Objects are negatively correlated for one factor of the ground truth with σ n = -0.60

PrOpt [PgADJ]	PrOpt [PgADJ, MeIBF]	PrOpt MeIBF MeADJ	Pg [PrOpt, Av, PgADJ, AvADJ]	PrOpt Av, Pg PgADJ
PrOpt [Pg*, AvADJ]	PrOpt [PgADJ, MeIBF]	MeADJ [MeIBF]	Av [Pg]	Av [Pg]
Av PrOpt, [AvADJ, AvIBF, PgADJ,	MeIBF PgADJ, MeADJ PgIBF]	Me MeADJ	Av [Pg]	Av [AvIBF]
PrOpt Av*,Me*, Pg*	PrOpt, MeIBF PgADJ, MeADJ	Me MeADJ	Av [Pg]	Av [AvIBF]

high noise

low noise

less information

more information

(a) precision (P@10)

PrOpt [PgADJ]	MeIBF PgADJ	MeIBF [RndIBF]	PrOpt [Pg]	PrOpt [Pg]
Pg [PrOpt, Av]	MeIBF PgADJ	MeADJ [MeIBF]	Av [Pg, PrOpt]	PrOpt [Av]
PrOpt Av Pg	MeADJ [MeIBF]	Me MeADJ	Av [Pg, PrOpt]	PrOpt [Av]
PrOpt *IBF,*ADJ Av*,Me*, Pg*	MeADJ [MeIBF]	Me MeADJ	Av [Pg, PrOpt]	PrOpt [Av]

less information

more information

(b) Kendall-tau error measure

Figure 12. Two factors are positively correlated for the ground truth with σ f = 0.60

Av [AvADJ]	Av [AvADJ]	Av [AvADJ]	Av [AvADJ]	Av [AvADJ]
Av [AvADJ]	Av [AvADJ]	Av [AvADJ]	Av [AvADJ]	Av [AvADJ]
PrOpt Pg	PrOpt [Pg]	PrOpt [Pg, PgADJ]	PrOpt [Pg]	PrOpt [Pg]
PrOpt, Av PgADJ,AvIBF	PrOpt PgADJ Av*	PrOpt [Av*, Pg*]	PrOpt Av*, Pg*	PrOpt Av*, Pg*

high noise

low noise

less information

more information

(a) precision (P@10)

Av [AvADJ]	Av [AvADJ]	Av [AvADJ]	Av [AvADJ]	Av [AvADJ]
Av [AvADJ]	Av [AvADJ]	Av [AvADJ]	Av [AvADJ]	Av [AvADJ]
Pg [PgADJ]	Pg [PgADJ]	Pg PgADJ	Pg PgADJ	Pg [PgADJ]
Av Pg PrOpt	Av Pg	Pg [Av]	Av Pg	Pg [Av]

high noise

how noise

less information

more information

(b) Kendall-tau error measure

Figure 13. Two factors are negatively correlated for the ground truth with σf = -0.60

MeIBF [*IBF]	AvIBF MeIBF RndIBF	AvIBF MeIBF RndIBF	PrOpt [Pg, PgADJ, AvIBF]	Pg [PrOpt,PgADJ]	high noise
PrOpt Pg* MeIBF	MeIBF [AvIBF, RndIBF]	MeIBF [RndIBF]	Pg [PrOpt, Av, PgADJ]	Av [AvADJ]	
PrOpt Pg PgADJ	MeIBF [PrOpt, PgADJ]	Me [PrOpt, MeADJ]	Av [Pg]	Av [AvADJ]	
Av, AvIBF MeADJ MeIBF	PrOpt PgADJ MeIBF	Me [PrOpt, MeADJ]	Av [Pg]	Av [AvADJ]	low noise

low information — more information

PrOpt [MeIBF]	MeIBF [PrOpt, RndIBF]	MeIBF [RndIBF]	MeIBF, RndIBF [PrOpt, Pg]	Pg [PrOpt, PgADJ]
PrOpt [Pg] [Av]	MeIBF [RndIBF]	MeIBF [RndIBF]	PrOpt Pg	Av [Pg]
Pg [Av]	MeIBF [MeADJ]	MeADJ [Me]	Av [PrOpt, Pg]	PrOpt Av [Pg]
[Me, RndIBF]	MeIBF [MeADJ]	Me MeADJ [MeIBF]	Av [PrOpt, Pg]	PrOpt [Av]

low information — more information

(a) precision (P@10) (b) Kendall-tau error measure

Figure 14. Positive correlation between errors performed by rankers for two objects for the same factor with σ_f =< 0.60, 0.60, 0.0, 0.0, 0.0 >

PrOpt Pg* MeIBF	PrOpt PgIBF MeIBF	PrOpt MeIBF RndIBF	PrOpt Pg PgADJ	Pg [PrOpt,PgADJ]	high noise
PrOpt [Pg, PgADJ]	PrOpt MeIBF	MeIBF RndIBF	Av [AvADJ,Pg]	Av [AvADJ]	
PrOpt Pg*	PrOpt PgADJ	Me [MeADJ,PrOpt]	Av [Pg]	Av [AvADJ]	
Av* [Me,RndIBF]	MeIBF PrOpt PgADJ	Me [MeADJ,PrOpt]	Av [Pg]	Av [AvADJ]	low noise

low information — more information

PrOpt [Pg]	PrOpt [Pg]	PrOpt [MeIBF, Pg]	PrOpt Pg [PgADJ]	PrOpt Pg [PgADJ]
PrOpt [Pg] Pg	PgADJ [MeIBF] MeIBF	MeIBF [PgIBF, RndIBF] Me	Pg [PrOpt, Av] Av	Av [Pg] PrOpt
[Av, PrOpt] [MeIBF]	[RndIBF] [Pg]	MeADJ	[PrOpt, Pg]	Av
PrOpt [Av, Pg]	MeIBF MeADJ [Me]	Me Av PrOpt [MeADJ]	[PrOpt, Pg]	[Av]

low information — more information

(a) precision (P@10) (b) Kendall-tau error measure

Figure 15. Positive correlation amongst the rankers with σ_R = 0.10