The Impact of Ranker Quality on Rank Aggregation Algorithms

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 statistical framework for generating the ground truth ranker and the individual rankers. The individual rankers, which are the inputs to the ranker aggregation algorithm, are statistical perturbations of the ground truth ranker. Within this statistical framework, we give a rigorous experimental evaluation of how different characteristics of the input rankers affect the aggregate ranker, including: asymmetry among the rankers; type of noise and noise level of the rankers; correlation among the rankers; outliers among the rankers, including the effect of "bi-partisan" (or multi-partisan) rankers. We introduce and study a novel local optimization rank aggregator, which we compare to some well known rank aggregation algorithms (*average*, *median* and Markov chain aggregators (including PageRank)). We determine conditions under which it is better to use one aggregator over another.

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 such as *PageRank* [5], *hubs and authorities* [15] and information retrieval methods. Unfortunately, query results are subjective since different rankers use different ranking criteria. Further, query results are dynamic with new and changing content depending on the ranker databases. From a user perspective, the problem of accessing the ground truth ranking function appropriate 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 individual rankers relate to the ground truth ranker in forming a consensus ranking.

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 does it make more sense to 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 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 changing the input rankers on the rank aggregation 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. We then study the performance of various aggregation techniques under different model assumptions. We observe that different model hypotheses can significantly impact the performance of an aggregation algorithm, and we give guidelines for which aggregator is appropriate given the model assumptions. Two important characteristics of any aggregator are that it be computed efficiently and that

it gives an accurate consensus ranking. We introduce and study a novel local optimization aggregator which may be used as an aggregator itself, or as an engine for improving any other aggregator. The specific major 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 detailed study of how input ranker characteristics affect the performance of the aggregation algorithms, including: asymmetry among the rankers; type of noise and noise level of the rankers; correlation among the rankers; outliers among the rankers, including the effect of "bi-partisan" (or multi-partisan) rankers;
- a novel local optimization aggregation algorithm to produce a better consensus ranking starting from an initial ranking, which could be randomly chosen or the output of some another aggregator.

2. RELATED LITERATURE

In the case of web searching, ranking of web sites is a combination of multiple factors from link analysis to number of times a keyword occurs in the various places in text and anchor text [5]. In most cases, rank aggregation algorithms have been compared to Borda's method [4]. Dwork et al. [8] introduce the notion of an aggregate rank which minimizes the total Kendall-tau distance models the consensus ranking based on an analogy with voting. Since such an aggregation is NP complete, 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. [17] perform a more extensive study of these Markov chain methods that also compares rank-based to score-based aggregation methods.

Other rank aggregation methods have been proposed such as [3, 6, 11, 16] to improve a particular measure or execute more efficiently. Beg et al. [3] propose a rank aggregation algorithm which optimizes the footrule measure. Their genetic algorithm uses reproductions, crossover and mutation approaches observed from biology to construct the aggregate ranker. Chin et al. [6] present a heuristic rank aggregation algorithm using weighted version of the extended Cordorcet criteria and optimizes the final ranking with respect to the Kendall-tau measure. The medrank [11] algorithm was designed for similarity search and classification applications and approximates the selection of the median rank of a set of input rankers where median can be retrieved once at least 50% of the rankings have seen that particular object.

Machine learning approach is another way to solve the aggregation problem. But, it requires access to a set of good training data and/or ground truth. It does not generally give insight as to how the search engine results should generally be combined based on ranker quality which our emphasis here. The study in [13] 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 ample information in distinguishing the more relevant objects from those less importance objects. The learning of the retrieval and ranking functions can provide better results than established search engines. The offline experiment compares the rank of an object in other search engines, the query match and popularity-attribute correlation. The on-line experiment compares the learned framework for the retrieval and ranking function with selected search engines in the number of observed clicks. The authors of [1] set up a user study comprising of several experts for a set of popular search queries. The objective of the study is to assess how the rankings match human opinion using in-degree, out-degree, PageRank and the authority and hub score from Kleinberg. The experts ordered the objects with some degree of consensus but unable to obtain complete agreement for all objects. The conclusions indicate that the in-degree measure performs just as well as the more complex PageRank and Kleinberg approaches.

These methods take as input several rankers of a set of objects that are ordered with respect to some criteria. When all objects are ordered with monotonically increasing ranks, this is referred to as a *full list*; otherwise we have a *partial list*. A theoretical framework that compares different distance measures such as Kendall-tau, footrule, and others on ordered lists of objects are presented by Fagin et al. [10, 9]. They show the relationships between these measures for partial and full lists. In the event of real-world applications, Jansen et al [12] perform tests on user queries on the web discovering that a user query is on average 2 or 3 search terms which is far lower that other traditional information retrieval systems. The study also examined the components of successive queries by the same user as well as term frequency. Other studies of search engine performance have been performed, which address performance over time [2] and uses human subjects to evaluate performance balancing the precision and recall measures [18].

Prior research tries to rank for a community of users using either query-independent or query-dependent collection of relevant objects. Certain assumptions are also made about the quality of the input rankers from the Web whereas we investigate the information quality provided by the input rankers. We formulate a statistical model for constructing input rankers so we can analyze them with respect to the well-established aggregation algorithms. The statistical model for aggregation offers us a distinction between manipulated (bad) input rankers from (bipartisan) rankers that provide a different perspective of the same information. We propose a local optimization aggregation algorithm based on the principles found in [14]. This algorithm can be used as a stand-alone method or in conjunction with the current methods that produces a better ranking than the traditional aggregation methods.

3. ERROR MEASURES

In order to assess the degree of closeness or similarity between two rankers, r_1 and r_2, we use an error measure $\mathcal{E}(r_1, r_2)$. We select four error measures to serve as comparison tools for our model. These measures have been used

in prior research (individually or as a subset) but not as a collection. These error measures provide a more complete perspective of the rankers that have not been previously been studied. Suppose A_k is the ground truth ranking of the top-k objects and B_k is the result of a specific aggregation algorithm for the top-k objects. The error measures under consideration are:

1. Precision ($pr_k = A_k \cap B_k$) which gives the common objects are in the top-k.
2. Recall ($rcl_k = n(k)$) where $n(k)$ is the lowest j such that all the objects in A_k are seen in B_j.
3. Kendall-tau ($\tau_k = Z$) where Z is the total number of pairwise disagreements between rankers A_k and B_i, $Z = \sum_{i,j \in A_k, B_k} \overline{K}_{i,j}^{(p)}(A_k, B_k)$.
4. Footrule ($F = \sum |A(o) - B(o)|$) where $A(o)$ is the rank of object o in ranker A (and similarly for $B(o)$).

Each of these error measures have strengths and drawbacks. Precision and recall require little computation. Precision measure fails to provide the amount of sortedness between the two rankers. Recall is a subjective measure with many interpretations since the universe of relevant documents is unknown typically defined as the proportion of relevant documents that are retrieved. If any object in the top-k of A is ranked in the bottom of B, then the error measure must iterate almost the full ranker B resulting in a very high recall value.

The Kendall-tau and footrule error measures requires more calculations as the number of objects observed increase. Both measures take the rank of the objects under consideration. For objects that are not in the top-k for a particular ranker, both measures assign the rank to be $k+1$ meaning that the object's rank is at least greater than k. Kendall-tau is sensitive to sortedness of the objects requiring every pair of distinct objects to be compared to every other pair of objects in order to exhaust all possible combinations, which results in the bubble sort algorithm. We can consider versions of Kendall-tau using different penalties in the cases that objects exists in one ranker but not in the other as outlined in [10]. We review the four cases for objects i and j and display them in the table below.

In *Case* 1, i and j appear in both rankers. *Case* 2 states that i and j appear in one ranking (A_k), and only one of i or j appears in the other ranker (B_k). *Case* 3 suggests that i, but not j, appears in a ranker (A_k), and j but not i, appears in the other ranker (B_k). Lastly in *Case* 4, i and j appear in one ranker (A_k), but neither appear in the other ranker (B_k). For each case, there are two possible interpretations. The order of i and j are preserved in both rankers or the order of i and j constitutes a pairwise disagreement. In addition, the penalty p associated with a pairwise disagreement (in *Case* 4) can take an optimistic ($p = 0$) or neutral ($p = 1/2$) approach. If we select $p = 0$, then the result will be the minimizing of Kendall-tau distance. If we select $p = 1/2$, then the result will be the averaging of Kendall-tau distance. We chose the optimistic approach by assigning $p = 0$.

	Order Preserved	Pairwise Disagreement
Case 1	$\overline{K}_{i,j}^{(p)}(A_k, B_k) = 0$	$\overline{K}_{i,j}^{(p)}(A_k, B_k) = 1$
Case 2	$\overline{K}_{i,j}^{(p)}(A_k, B_k) = 0$	$\overline{K}_{i,j}^{(p)}(A_k, B_k) = 1$
Case 3	N/A	$\overline{K}_{i,j}^{(p)}(A_k, B_k) = 1$
Case 4	N/A	$\overline{K}_{i,j}^{(p)}(A_k, B_k) = p$

The footrule error measure provides some degree of sortedness. A "disagreement" in footrule is a measure that considers and object's position and not its relationship with with other objects (as seen with Kendall-tau). Footrule amounts to an addition of 1 to k ($\sum |A(o) - B(o)|$) and for this reason may make it hard to quantify the impact of input rankers. The maximum value of footrule error measure is achieved by the summation of differences in the rank positions. In the worse case, both rankers contain a distinct set of objects, which reduces to $2 * \sum_k k = 2 * 1/2 * (k^2 + k)$. As proven in [7], for any full rankers, the footrule distance is within 2 factors of Kendall-tau distance.

4. RANK AGGREGATION METHODS

We review the rank aggregation methods of interest for our study and present an aggregation optimization technique. We select these methods based on their popularity in prior research and quality of their aggregate ranker. Rank aggregation methods can be performed on full lists of objects if each ranker contains the same objects. The top-k objects can be returned by the rank aggregation method when partial lists of k objects are only available. We compare and analyze the Borda's method (we refer also to it as average), median rank and PageRank aggregation methods. The first two methods have proved to provide a meaningful representation of the input rankers while maintaining a low computational cost. The PageRank is slightly more costly with the creation of the graph of relevant objects. With any rank aggregation method, ties amongst the objects may be formed, which we have chosen to break arbitrarily. The order of the objects, as returned from the specific rank aggregation method, contains the distinct objects from all input rankers. We refer to this ordering of distinct objects as the aggregate ranker. In the case of top-k queries, we only return the first k objects in the order of appearance.

4.1 Average

The popularity of Borda's method lies in its usage of each input ranker to provide an unbiased representation producing an aggregate ranker. The difference in the average and Borda's method is its values associated with a particular position. In Borda's method, we assume full knowledge of the number of objects amongst the rankers where the highly ranked objects receive the largest value. We impose average, which is the reversal of the values given to Borda's method, since we can not make such claims on the number of objects.

4.2 Median

The median rank is slightly different from average since it relies on the middle value to determine the order of the objects. By not considering each object's rank in every input ranker, it is not sensitive to outlying values. The likelihood of ties may be more predominant with this model since the median can come from any one of the input rankers. As shown in [11], the median rank can approximate the footrule which in turn approximates the Kendall-tau when we have full access to all input rankers in order to accurately compute the median value. The benefit of this aggregation method can be shown through the bipartisan input rankers.

4.3 PageRank

The Markov chain aggregator (MC_4) in Dwork et al. [8] constructs the aggregate ranker in which the transition of an object in MC_4 occurs if the next object is ranked higher for the majority of the input rankers. The PageRank algorithm [5] can be directly used to construct the aggregate ranker in a similar fashion. For given input rankers, the algorithm proceeds as follows. Each distinct object (from the input rankers) represents a node in the graph. A link is made between two objects, we refer to them as in-degree and out-degree objects, respectively, when the out-degree object (lower rank) points to the in-degree object (higher rank). The pagerank algorithm of Google uses a similar model to compute the probability of being at a specific node in a graph through navigation. However, it assumes that at each node, the user will navigate with probability α and randomly jump to a page with probability $1-\alpha$. This model is very similar to the Markov model except it does not allow any sink nodes to exist. We compute the probability of each object o of being the next transition state through the other objects it links to, we call them neighbors $N(o)$, where d is the number of links between o and i and n is the number of objects in the graph: $pagerank(o) = \alpha * \sum_{i=N(o)} \frac{d}{out-degree(i)} + (1-\alpha) * \frac{1}{n}$. The α parameter measures the likelihood of transitioning to a random object or one of the neighbors. Hence, the aggregate ranker can depend solely or in part on the graph structure of the input rankers. The starting probabilities of the objects are uniformly distributed ($\frac{1}{n}$) and the out-degree links are each assigned the value of 1 making the endorsement of an object's final position the same for each pairwise comparison. We alter the PageRank in the following two manners: (1) starting probabilities to be the number of in-degree links of each object and (2) the weight of each link for every rank j to be $\sum_{i=1}^{j} \frac{(j-i)}{\binom{j}{2}} = 1$. Hence, a link is created for each pair of objects at positions i and j. In this case, the major proportion of the weight is given to the higher ranked objects. We normalized the weight for the out-degree links by the number of input rankers.

4.4 Local Optimization for Aggregation

We present a novel aggregation algorithm based on an algorithm to perform local combinatorial optimization which was introduced by Kernighan and Lin [14] in the specific context of graph partitioning. Our goal here, however, is to reduce the average error between the aggregate ranker r_A and the input rankers $r_1, \ldots, r_s$, where we define the average error to be $\mathcal{E}_{av} = \frac{1}{s}\sum_{i=1}^{s} \mathcal{E}(r_i, r_A)$. If the input rankers give unbiased estimates of the ground truth ranking, then minimizing the average error should also reduce the expected error w.r.t. the ground truth ranking. This minimization problem is usually NP-complete (for example if $\mathcal{E}$ is the Kendall-τ error measure), hence we resort to local optimization. The general idea behind our algorithm is to perform a sequence of greedy swaps that eventually leads to a good local minimum of the average error. We begin from some initial ranking, which could, for example, be the output of an initial aggregation algorithm (eg. average, median or PageRank). The algorithm proceeds as follows.

1: **for** each object $[n]$ **do**
2: **for** every possible swap $[n-1]$ **do**
3: Compute $\mathcal{E}_{av}$ after the swap;
4: Keep the swap with minimum $\mathcal{E}_{av}$; {$\mathcal{E}_{av}$ may increases}
5: $n+1$ rankings are visited; {including the initial ranking}
6: **return** the one with minimum $\mathcal{E}_{av}$;

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. The algorithm above is repeatedly executed, each time starting from its own output until no further progress is made (i.e., the ranking no longer changes). 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. rankings, we output the best which may be the initial ranking. For the remainder of this paper, we use the Kendall-τ error measure, for which $f(n) = O(n^2)$. However, we perform a pre-processing step which allows us to update the average error, instead of recomputing it from scratch, each time a swap is made. The resulting algorithm has an improved computational complexity of $O(s \cdot n^3)$.

5. STATISTICAL MODEL FOR AGGREGATION

In this section, we describe a statistical model that we use to evaluate the rank aggregation algorithms and their input rankers. In our framework, we suppose we can pose a web query to some search engine. The result returns a set of objects, we denote $p_1, \ldots, p_q$. This search engine uses a *rank function* in order to determine the order of these objects. The rank function comprises of a set of properties or $factors$, which we term $f_1, \ldots, f_g$ where $f_i = [0, 1]$. Typically all factors are used in the rank function but every factor may have different significance to the user. Thus each factor should be weighted according to user specifications which is represented by $weights = w_1, \ldots, w_g$ where $w_i \geq 0$ and $\sum_i^g w_i = 1$, the number of factors. Now we can see that the rank of an object is based upon a combination

	ground truth r	ranker r_1	ranker r_2	...	ranker r_s	aggregator r_A
p_1	$V(p_1) = \sum_i^g w_i * f_i$	$V^{r_1}(p_1) = \sum_i^g w_i * f_i^{r_1}$	$V^{r_2}(p_1)$	...	$V^{r_s}(p_1)$	$V^{r_A}(p_1)$
p_2	$V(p_2) = \sum_i^g w_i * f_i$	$V^{r_1}(p_2) = \sum_i^g w_i * f_i^{r_1}$	$V^{r_2}(p_2)$	...	$V^{r_s}(p_2)$	$V^{r_A}(p_2)$
...	...	...	...	...	...	...
p_q	$V(p_q) = \sum_i^g w_i * f_i$	$V^{r_1}(p_q) = \sum_i^g w_i * f_i^{r_1}$	$V^{r_2}(p_q)$	...	$V^{r_s}(p_q)$	$V^{r_A}(p_q)$

Figure 1: Matrix of objects and their values in our framework

of the factors and its weights that produces a score ranging $[0, 1]$. The scores are then sorted in descending order and assigned ranks between 1 to q.

By varying the rank function, we can construct many orderings of these objects. We represent each rank function using the term $rankers = r, r_1, \ldots, r_s$ where r is the ground truth ranker and $r_1, \ldots, r_s$ are the input rankers that are based on r with some degree of error. We impose this ground truth as a baseline for comparison and useful for error analysis. Let us assume the factors for the ground truth ranker, $f_1, \ldots, f_g$, and each weighted factor $w_i = \frac{1}{g}$. We suppose that the value of the factor of the ground truth ranker(f_i) and every other ranker($f_i^{r_j}$) are related as follows: $f_i^{r_j} = f_i + \varepsilon_i^{r_j}$. Thus, we introduce an error measure (ε) per factor for every ranker. We can therefore obtain for each ranker r_j the value of a object is based on the weighted factors to be $V^{r_j}(p) = \sum_i^g w_i^{r_j} * f_i^{r_j}$ and the results are ranked with respect to their value. These rankers, ground truth and input, combine the factors, weights and errors in a linear combination formula.

We can generate a set of object containing a set of factors such that for each object $p_i = (f_1, \ldots, f_g)$. We generate the error measures($\vec{\varepsilon_1}, \ldots, \vec{\varepsilon_g}$) as well where $\vec{\varepsilon_i}$ is independent of $\vec{\varepsilon_j}$ for all i and j but the components of $\vec{\varepsilon_i}$ may be dependent for each ranker. Hence, the errors for each factor are independent of each other. But, different rankers for the same factor may make errors that are correlated to each other. A simple, analytically tractable and common distribution assumption for $\vec{\varepsilon}$ that has these properties is to assume that the errors are jointly normally distributed with the specified covariance matrix. Let $\mathbf{Y}$ be a multivariate normal random variable with mean $\mathbf{0}$ identity covariance matrix. Each component of $\mathbf{Y}$ is either y_1 or y_2, which are two independent normal random variables each with mean zero and unit variance. Note that it is not known how to generate *one* Gaussian random variable, but it is known how to generate *two*. If you want one, you can always generate two and throw one away. Thus we can generate each component of $\mathbf{Y}$.

Given any matrix, $\mathbf{A}$, let $\mathbf{X} = \mathbf{AY}$. Then $\mathbf{X}$ is a joint normal random vector, with mean $\mathbf{0}$ and covariance matrix $\Sigma = \mathbf{AA}^T$. This is one way to get a correlated random vector $\mathbf{X}$ from $\mathbf{Y}$, which is independent. The resulting $\mathbf{X}$ matrix represents the correlated error vector for every object for each factor component. The error measures are dependent on the factor values. Thus,for some fixed factor i,

$$\begin{bmatrix} \epsilon_{i1} \\ \epsilon_{i2} \\ \vdots \\ \epsilon_{iq} \end{bmatrix} \sim (\mathbf{0}, \Sigma),$$

where Σ is the covariant matrix. We have thus fully specified the probabilistic model, which assigns a probability distribution to the ground truth ranking via the distribution of the $V(p_i)$'s, and determines how the rankers obtain their rankings. Thus the error measure for the factor f lies within range $[f, 1-f]$. We want to ensure that the values assigned to the input rankers also lie in range $[0, 1]$. We devise the covariance matrix with the diagonal values being 1 and multiply it by a coefficient, σ. We arbitrarily select σ to be $\frac{\bar{f}}{g}$ where $\bar{f} = min\{f, 1-f\}$ and g is the number of factors. The coefficient serves as a method to create the dependency of error measures to factors. Hence, it is assumed if the ground truth score of an object for a factor is very high or very low, the errors for that factor are likely to be low, and rankers will get this factor mostly correct. Suppose we have two factors, $f_1 = 0.1$ and $f_2 = 0.9$, the maximum error possible for either factor is 0.1. The influence of the errors associated to the factors is varied in which a smaller ground truth factor value can have a large proportionate error while a larger ground truth factor value has a small error. For example, if we consider two factors *pagerank*, and *keywordMatch*. If two different rankers use the same database of objects for their ranking, then the resulting factors may have similar errors. In this case, the same amount of links are missing in the computation of PageRank. But, the errors in page rank and keyword match are not correlated since they depend on different attributes of data.

Figure 1 displays the values created for each object for the ground truth, input and aggregate rankers. Now we can produce the ground truth ranker, e.g. $V(p_1), \ldots, V(p_q)$. Then we can compute a set of rankers, $V^{r_1}, \ldots, V^{r_s}$, adjusting the weights. The aggregator uses the rankers as input. We can investigate the effects of the error measure, factors and weights on the aggregator. We assume that each search engine has its own ranking. Today's search engines use weights that are an estimate of the ground truth weights for a population, but for different queries and user groups, these weights might differ greatly from the optimal weights.

Let $\mathcal{A}$ generically refer to an aggregator, and let $\mathcal{E}(\mathbf{r}, \mathbf{r_A})$ be a measure that measures the difference between the ground truth ranker $\mathbf{r}$ and the ranker $\mathbf{r}_\mathcal{A}$ obtained by the aggregator. One possible measure is pairwise disagreements. Among the available aggregators, we would like to select the one with the smallest average error. We can thus estimate the average error of any aggregator (or set of aggregators) within our model as follows.

1: // **Evaluation of Aggregators**
2: **while** Not Done **do**
3: Generate $\{f_i\}_{i=1}^g$ from a normal distribution $N(0, 1)$.
4: Construct weights vector such that $\sum_i^g w_i = 1$.

5: **for** all objects q **do**
6: Generate $V(p_i)_{i=1}^{q} = \sum_{k=1}^{g} w_k * f_k$
7: Sort $\{V(p_i)\}$ to obtain the ground truth ranker r.
8: **for** each ranker r_j, generate $V^{r_j}(p_i)$ as follows **do**
9: Generate $\epsilon_{i1}, \ldots, \epsilon_{ig}$ by drawing from the gaussian normal distribution $N(\mathbf{0}, \Sigma)$.
10: Generate $V^{r_j}(p_i) = \sum_{k=1}^{g} w_k * f_k^{r_j}$.
11: For each ranker r_j, generate its rankings by sorting $V^{r_j}(p_i)$.
12: Denote the resulting rankers by the vectors $\mathbf{r}(1), \ldots, \mathbf{r}(s)$.
13: **for** each aggregator $\mathcal{A}$ **do**
14: Compute the aggregate ranking $r_{\mathcal{A}} = \mathcal{A}(\mathbf{r}(1), \ldots, \mathbf{r}(s))$.
15: Compute the error $\mathcal{E}(\mathbf{r}, \mathbf{r_A})$and update the average error of aggregator $\mathbf{A}$.

6. EXPERIMENTAL EVALUATION

For our evaluation of the rankers and aggregation methods, we fix several of the input parameters providing a basis for comparison. Our statistical framework is designed using MatLab making the matrix computations more efficient. We perform tests on a test bed of 100 objects which is computationally easy for MatLab. The number of factors and number of rankers is fixed to 5. We also make the weights assigned to each factor equal ($w_i = 0.2$) except in the case of the bi-partisan approach, which we will discuss later. As part of the aggregation procedure, the input rankers consist of the top-k objects from each ranker. When we retrieve top-k objects from different ranked lists, we estimate all the missing ranks of objects to be $k + 1$. These are objects that are returned by some ranked lists but not all. For the PageRank algorithm, we fixed the α parameter at 0.85 since we did not observe any significant dependence of α in our experiments. We perform tests without and with the local optimization for aggregation. We execute 1000 datasets (where each dataset contains its own ground truth ranker and five input rankers) in which we compute the average to display the steady state of our model. When $k < 100$, we did not observe a dataset comprising of all the same objects in the ground truth and input rankers, nor a case of all distinct objects for all rankers. For each error measure, we record the aggregation algorithms and the counts based on the measure of comparison. We should note that the precision error measure is a maximization function whereas the other error measures are minimization functions.

In Figure 2, we display the results of the aggregation methods of average, median and the PageRank algorithm for the partial lists of top-10, top-25, top-50 and full list of top-100. The results shown in the table serve as a baseline of these rank aggregation methods. We notice that average and PageRank give similar counts with respect to each error measure. The median, on the other hand, gives worse performance, but remains close to the counts obtained with the other algorithms. In the case of top-100, precision and recall have values of 100 since we have full access to all objects under consideration in our model. We see that as we increase the value of k, the difference between average/PageRank and median gets larger for the Kendall-tau and footrule error measures. The estimation of ranks not appearing in all rankers becomes more predominant due to the nature of the measure. For instance with $k = 10$, the amount of error incurred will be lower with the maximum rank of 11 than when k is larger making the maximum rank larger. We found that the recall measure is a difficult measure to optimize since more objects in the top-k increases the possibility of error (the value of $n(k)$ thereby making improvements difficult to achieve). We also observed that the footrule error measure gives similar trends to that seen in Kendall-tau so we elected to only report the results of the Kendall-tau for the remaining tests.

Since users are mostly interested in the top few objects, for the remainder of the tests we only show the top-10 and top-15 objects. In Figure 3, we display the results of the aggregation methods of average, median and the PageRank algorithm and their respective results under local optimization for the partial lists. In the case of the top-10, the counts returned by the first three rank aggregation methods are the same as the ones shown in Figure 2. The local optimization method increases the precision slightly in all three methods. The counts for the top-10 and top-15 are also very close to each other indicating that the best ranking is located regardless of the initial start ranking of the objects. The Kendall-tau error measure reveals a larger discrepancy amongst the counts of the three aggregation methods. In general, the median has the worse performance in which the local optimization is able to discover a better ranking and improve its count by several perturbations.

We implement bipartisan input rankers such that we modified the weights assigned to the factors. Three rankers have roughly the same interpretations whereas two rankers have a different perspective but are dissimilar from the other three rankers. Three of the rankers make weights for each factor to be $\frac{1}{15}, \frac{2}{15}, \frac{3}{15}, \frac{4}{15}, \frac{5}{15}$ and the last two rankers have weights $\frac{5}{15}, \frac{4}{15}, \frac{3}{15}, \frac{2}{15}, \frac{1}{15}$. The numerator is the range from 1 to g, the number of factors, so that each factor has a different significance. The denominator is the number of possible perturbations or bubble sort of 5 factors. In Figure 4, we display the results of the aggregation methods in which the input ranker represent different opinions of the same objects. The opinions of both the left and right of the bipartisan are accurate but the combination these opinions make the average and PageRank more noisy. The precision associated with median is clearly superior to the other methods. Even with the local optimization, the median achieves the best ranking while the other two methods have an initial start ranking that can not reach the local minima of median. We observe this trend as well for the Kendall-tau error measure. The local optimization improves the ranking for all three aggregation methods by about 1 count unit.

As a special case of bipartisan, there is one opinion which the majority of input rankers have a consensus and the other input rankers are manipulated that produce a very different order of the objects. In Figure 5, we display the results of this case. It simulates the situation where three rankers are mostly correct in their rankings while the other two rankers are mostly incorrect. The meta search considers information from all input rankers in order to determine the best ordering. We add additional noise to the factors of two rankers

Aggregation Method	top-10	top-25	top-50	top-100
Average	8.359	22.233	46.652	100.000
Median	8.158	21.986	46.451	100.000
PageRank	8.388	22.29	46.748	100.000

(a)

Aggregation Method	top-10	top-25	top-50	top-100
Average	16.355	35.060	61.200	100.000
Median	16.530	35.065	61.190	100.000
PageRank	16.375	35.025	61.155	100.000

(b)

Aggregation Method	top-10	top-25	top-50	top-100
Average	14.846	61.669	160.649	285.086
Median	17.529	71.966	187.480	343.806
PageRank	14.893	62.117	160.286	284.830

(c)

Aggregation Method	top-10	top-25	top-50	top-100
Average	20.006	82.872	224.880	433.180
Median	22.806	94.944	260.188	515.758
PageRank	20.396	83.948	225.148	432.766

(d)

Figure 2: Baseline Comparison of Aggregation Methods under (a) precision error measure, (b) recall error measure, (c) Kendall-tau error measure, (d) footrule error measure

Aggregation Method	top-10	top-15
Average	8.359	12.918
Median	8.158	12.625
PageRank	8.300	12.956
Average-opt	8.407	12.977
Median-opt	8.416	12.930
PageRank-opt	8.419	12.967

(a)

Aggregation Method	top-10	top-15
Average	14.846	28.072
Median	17.592	33.576
PageRank	14.893	28.338
Average-opt	14.336	27.453
Median-opt	14.492	28.482
PageRank-opt	14.170	27.386

(b)

Figure 3: Baseline Comparison with Local Optimization under (a) precision error measure, (b) Kendall-tau error measure

while maintaining equal weights. The manipulated rankers produce more outlying ranks for all objects. Hence, these rankers provide more noise to the aggregate ranker making the order of objects not as valuable to the user. We notice for the precision and Kendall-tau the effectiveness of the median rank aggregation method is higher than the other two methods. Once again we see similar improvement of the counts using the local optimization.

We have conducted other tests by varying a number of other conditions. In general, we found that whenever there is no specific difference among the rankers, average seems to outperform median. This is due to the fact that median throws away some of the actual rank information that is uniformly useful in obtaining the aggregate ranking. When the errors are not correlated, average outperforms median with a larger margin since the amount of information provided by each ranker increases. When each ranker is missing the same percentage of objects in their ranks, average still outperforms median. However, as one ranker has significantly more missing objects, then the ranks of objects for that ranker becomes more noisy and the performance of median rank starts to improve. The relationship between average and median rank remains the same even if the scores are distributed according to a Zipf distribution. This only changes the number of objects that have high or low scores, but the resulting effect remains the same. The local optimization allows the aggregate ranker to reconsider the impact of the input rankers through several iterations. Hence, the aggregate ranker is improved by exploiting the relevant information contained in the input rankers.

7. CONCLUSION

In this paper, we introduce a realistic statistical model that captures the interdependencies between the ground truth and input rankers. The ground truth ranker comprises of full and accurate knowledge of the factors and weights used in the linear combination formula. The input rankers are variations of the ground truth ranker with some additional noise to produce a different ordering of objects. We perform a detailed study of how input ranker characteristics affect the performance of the rank aggregation methods including *average*, *median* and a Markovian chain method (PageRank). We also present a novel local optimization aggregation algorithm to produce a better aggregation starting from an initial ranking, which could be randomly chosen or it could be the output of some other aggregator. We discover under certain contexts when one aggregation method gives a better ordering of objects than the other methods. We implement two cases in which the median rank outperforms the average rank. With bipartisan rankers, the majority of the input rankers are designed with one opinion while the remaining input rankers represent another opinion. In a special case of bipartisan, only one correct opinion exists observed in a majority of input rankers while the remaining rankers are manipulated and inconsistent.

As part of future work, we plan on examining different correlation of errors. We are investigating how to determine members of the left and right bipartisan points of view. We also would like to consider a statistical model that does not use factors, errors and weights to determine the input rankers since intimate knowledge of all these components may not be available.

8. REFERENCES

Aggregation Method	top-10	top-15
Average	7.123	11.124
Median	7.687	12.088
PageRank	7.225	11.308
Average-opt	7.162	11.160
Median-opt	7.789	12.165
PageRank-opt	7.223	11.306

(a)

Aggregation Method	top-10	top-15
Average	29.272	59.103
Median	22.770	42.549
PageRank	28.936	57.794
Average-opt	28.534	58.153
Median-opt	21.564	41.408
PageRank-opt	27.773	55.761

(b)

Figure 4: Bi-partisan Local Optimization under (a) precision error measure, (b) Kendall-tau error measure

Aggregation Method	top-10	top-15
Average	7.356	11.401
Median	7.718	12.023
PageRank	7.654	11.799
Average-opt	7.369	11.433
Median-opt	7.863	12.152
PageRank-opt	7.641	11.776

(a)

Aggregation Method	top-10	top-15
Average	25.519	51.811
Median	22.383	42.784
PageRank	24.682	50.298
Average-opt	25.211	50.985
Median-opt	20.644	40.675
PageRank-opt	23.389	47.649

(b)

Figure 5: Local Optimization with 2 manipulated rankers under (a) precision error measure, (b) Kendall-tau error measure

[1] B. Amento, L. Terveen, and W. Hill. Does authority mean quality? predicting expert quality ratings of web documents. In *Proc. ACM SIGIR*, 2000.

[2] J. Bar-Ilan. Methods for measuring search engine performance over time. *Journal of the American Society for Information Science and Technology*, 53(4):308–319, 2002.

[3] 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.

[4] J. C. Borda. Mémoire sur les élections au scrutin. 1781.

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

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

[7] P. Diaconis and R. Graham. Spearman's footrule as a measure of disarray. *Journal of the Royal Statistical Society. Series B*, 39(2):262–268, 1977.

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

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

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

[11] R. Fagin, R. Kumar, and D. Sivakumar. Efficient similarity search and classification via rank aggregation. In *Proc. ACM SIGMOD*, 2003.

[12] B. Jansen, A. Spink, J. Bateman, and T. Saracevic. Real life information retrieval: A study of user queries on the web. *ACM SIGIR Forum*, 32(1):5–17, 1998.

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

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

[15] R. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632, 1999.

[16] A. Lotem, R. Fagin, and M. Naor. Optimal aggregation algorithms for middleware. *Journal of Computer and System Sciences*, 66:614–656, 2003.

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

[18] L. Vaughan. New measurements for search engine evaluation proposed and tested. *Information Processing and Management*, 40(4):677–691, 2004.