Rank Aggregation Methods for the Web

Cynthia Dwork* Ravi Kumar† Moni Naor‡ D. Sivakumar§

ABSTRACT

We consider the problem of combining ranking results from various sources. In the context of the Web, the main applications include building meta-search engines, combining ranking functions, selecting documents based on multiple criteria, and improving search precision through word associations. We develop a set of techniques for the rank aggregation problem and compare their performance to that of well-known methods. A primary goal of our work is to design rank aggregation techniques that can effectively combat "spam," a serious problem in Web searches. Experiments show that our methods are simple, efficient, and effective.

Keywords: rank aggregation, ranking functions, meta-search, multi-word queries, spam

1. INTRODUCTION

The task of ranking a list of several alternatives based on one or more criteria is encountered in many situations. One of the underlying goals of this endeavor is to identify the best alternatives, either to simply declare them to be the best (e.g., in sports) or to employ them for some purpose. When there is just a *single* criterion (or "judge") for ranking, the task is relatively easy, and is simply a reflection of the judge's opinions and biases. (If simplicity were the only desideratum, dictatorship would prevail over democracy.) In contrast, this paper addresses the problem of computing a "consensus" ranking of the alternatives, given the individual ranking preferences of *several* judges. We call this the *rank aggregation problem*. Specifically, we study the rank aggregation problem in the context of the Web, where it is complicated by a plethora of issues. We begin by underscoring the importance of rank aggregation for Web applications and clarifying the various characteristics of this problem in the context of the Web. We provide the theoretical underpinnings for stating criteria for "good" rank aggregation techniques and evaluating specific proposals, and we offer novel algorithmic solutions. Our experiments provide initial evidence for the success of our methods, which we believe will significantly improve a variety of search applications on the Web.

1.1 Motivation

As of February 2001, there were at least 24 general-purpose search engines (see Search Engine Watch [1]), as well as numerous special-purpose search engines. The very fact that there are so many choices is an indication that no single search engine has proven to be satisfactory for all Web users. There are a number of good reasons why this is the case, even if we restrict attention to search engines that are meant to be "general purpose." Two fairly obvious reasons are that no one ranking algorithm can be considered broadly acceptable and no one search engine is sufficiently comprehensive in its coverage of the Web. The issues, however, are somewhat deeper.

Firstly, there is the question of "spam" — devious manipulation by authors of Web pages in an attempt to achieve undeservedly high rank. No single *ranking function* can be trusted to perform well for all queries. A few years ago, query term frequency was the single main heuristic in ranking Web pages; since the influential work of Kleinberg [16] and Brin and Page [7], link analysis has come to be identified as a very powerful technique in ranking Web pages and other hyperlinked documents. Several other heuristics have been added, including anchor-text analysis [8], page structure (headers, etc.) analysis, the use of keyword listings and the url text itself, etc. These well-motivated heuristics exploit a wealth of information, but are often prone to manipulation by devious parties.

Secondly, in a world governed by (frequently changing) commercial interests and alliances, it is not clear that users have any form of protection against the biases/interests of individual search engines. As a case in point, note that "paid placement" and "paid inclusion" (see [2]) appear to be gaining popularity among search engines.

In some cases, individual ranking functions are inadequate

*Compaq Systems Research Center, 130 Lytton Ave., Palo Alto, CA 94301. dwork@pa.dec.com

†IBM Almaden Research Center, 650 Harry Road, San Jose, CA 95120. ravi@almaden.ibm.com

‡Department of Computer Science and Applied Mathematics, Weizmann Institute of Science, Rehovot 76100, Israel. This work was done while the author was visiting the IBM Almaden Research Center and Stanford University. naor@wisdom.weizmann.ac.il

§IBM Almaden Research Center, 650 Harry Road, San Jose, CA 95120. siva@almaden.ibm.com

WWW10, May 1-5, 2001, Hong Kong.
ACM 1-58113-348-0/01/0005.

for a more fundamental reason: the data being ranked are simply not amenable to simple ranking functions. This is the case with querying about multimedia documents, e.g. "find a document that has information about Greek islands with pictures of beautiful blue beaches." This is a problem conventionally studied in database middleware (see [15]). Several novel approaches have been invented for this purpose, but this problem cannot be considered well-solved by any measure. Naturally, these problems fall under the realm of rank aggregation.

Thus, our first motivation for studying rank aggregation in the context of the Web is to provide users a certain degree of *robustness* of search, in the face of various shortcomings and biases — malicious or otherwise — of individual search engines. That is, to find robust techniques for *meta-search*.

There is a second, very broad, set of scenarios where rank aggregation is called for. Roughly described, these are the cases where the user preference includes a variety of criteria, and the logic of classifying a document as acceptable or unacceptable is too complicated or too nebulous to encode in any simple query form. As prototypical examples, we list some cases that Web users experience frequently. Broadly, these can be classified as *multi-criteria selection* and *word association queries*. Examples of multi-criteria selection arise when trying to choose a product from a database of products, such as restaurants or travel plans. Examples of word association queries arise when a user wishes to search for a good document on a topic; the user knows a list of keywords that collectively describe the topic, but isn't sure that the best document on the topic necessarily contains all of them. (See Section 5 for specific examples of both categories.) This is a very familiar dilemma for Web search users: when we supply a list of keywords to a search engine, do we ask for documents that contain *all* the keywords, or do we ask for documents that contain *any* of the keywords? Notice that the former may produce no useful document, or too few of them, while the latter may produce an enormous list of documents where it is not clear which one to choose as the best. We propose the following natural approach to this problem:

> **Associations Ranking**: Rank the database with respect to several small subsets of the queries, and aggregate these rankings.

1.2 Challenges

The ideal scenario for rank aggregation is when each judge (search engine in the case of meta-search, individual criterion for multi-criteria selection, and subsets of queries in the case of word association queries) gives a complete ordering of all the alternatives in the universe of alternatives. This, however, is far too unrealistic for two main reasons.

The first reason is a particularly acute problem in doing meta-search: the coverage of various search engines is different; it is unlikely that all search engines will (eventually) be capable of ranking the entire collection of pages on the Web, which is growing at a very high rate. Secondly, search engines routinely limit access to about the first few hundreds of pages in their rank-ordering. This is done both to ensure the confidentiality of their ranking algorithm, and in the interest of efficiency. The issue of efficiency is also a serious bottleneck in performing rank aggregation for multi-criteria selection and word association queries.

Therefore, any method for rank aggregation for Web applications must be capable of dealing with the fact that only the top few hundred entries of each ranking are available. Of course, if there is absolutely no overlap among these entries, there isn't much *any* algorithm can do; the challenge is to design rank aggregation algorithms that work when there is limited but non-trivial overlap among the top few hundreds or thousands of entries in each ranking. Finally, in light of the amount of data, it is implicit that any rank aggregation method has to be computationally efficient.

1.3 Our results

We provide a mathematical setting in which to study the rank aggregation problem, and propose several algorithms. By drawing on the literature from social choice theory, statistics, and combinatorial optimization, we formulate precisely what it means to compute a good consensus ordering of the alternatives, given several (partial) rankings of the alternatives. Specifically, we identify the method of Kemeny, originally proposed in the context of social choice theory, as an especially desirable approach, since it minimizes the total disagreement (formalized below) between the several input rankings and their aggregation. Unfortunately, we show that computing optimal solutions based on Kemeny's approach is NP-hard, even when the number of rankings to be aggregated is only 4. Therefore, we provide several heuristic algorithms for rank aggregation and evaluate them in the context of Web applications. Besides the heuristics, we identify a crucial property of Kemeny optimal solutions that is particularly useful in combatting spam, and provide an efficient algorithm for minimally modifying *any* initial aggregation so as to enjoy this property. This property is called the "extended Condorcet criterion," and we call the efficient process that is guaranteed to achieve it "local Kemenization."

Our algorithms for initial aggregation are based on two broad principles. The first principle is to achieve optimality not with respect to the Kemeny guidelines, but with respect to a different, closely related, measure, for which it is possible to find an efficient solution. The second principle is through the use of Markov chains as a means of combining partial comparison information — derived from the individual rankings — into a total ordering. While there is no guarantee on the quality of the output, the latter methods are extremely efficient, and usually match or outperform the first method.

We report experiments and quantitative measures of quality for the meta-search problem, and give several illustrations of our methods applied for the problems of spam resistance and word association queries.

1.4 Organization

We describe our framework, including the notions of ranking, distance measures, and optimal aggregation in Section 2. This section also contains a brief description of concepts from graph theory and Markov chains we need for this paper. Section 3 discusses spam, the extended Condorcet principle, and local Kemenization. Section 4 describes various rank aggregation methods, including the well-known Borda method and several other new methods. Section 5 presents five major applications of our methods and Section 6 presents an experimental study of some of them. Finally, Section 7 concludes the paper with some remarks on future work.

2. PRELIMINARIES

2.1 Ranking

Given a universe U, an *ordered list* (or simply, a list) τ with respect to U is an ordering (aka ranking) of a subset $S \subseteq U$, i.e., $\tau = [x_1 \geq x_2 \geq \cdots \geq x_d]$, with each $x_i \in S$, and $\geq$ is some ordering relation on S. Also, if $i \in U$ is present in τ, let $\tau(i)$ denote the position or rank of i (a highly ranked or preferred element has a low-numbered position in the list). For a list τ, let $|\tau|$ denote the number of elements. By assigning a unique identifier to each element in U, we may assume without loss of generality that $U = \{1, 2, \ldots, |U|\}$.

Depending on the kind of information present in τ, three situations arise:

(1) If τ contains all the elements in U, then it is said to be a *full list*. Full lists are, in fact, total orderings (permutations) of U. For instance, if U is the set of all pages indexed by a search engine, it is easy to see that a full list emerges when we rank pages (say, with respect to a query) according to a fixed algorithm.

(2) There are situations where full lists are not convenient or even possible. For instance, let U denote the set of all Web pages in the world. Let τ denote the results of a search engine in response to some fixed query. Even though the query might induce a total ordering of the pages indexed by the search engine, since the index set of the search engine is almost surely only a subset of U, we have a strict inequality $|\tau| < |U|$. In other words, there are pages in the world which are unranked by this search engine with respect to the query. Such lists that rank only some of the elements in U are called *partial lists*.

(3) A special case of partial lists is the following. If S is the set of all the pages indexed by a particular search engine and if τ corresponds to the top 100 results of the search engine with respect to a query, clearly the pages that are not present in list τ can be assumed to be ranked below 100 by the search engine. Such lists that rank only a subset of S and where it is implicit that each ranked element is above all unranked elements, are called *top d lists*, where d is the size of the list.

A natural operation of *projection* will be useful. Given a list τ and a subset T of the universe U, the projection of τ with respect to T (denoted $\tau_{|T}$) will be a new list that contains only elements from T. Notice that if τ happens to contain all the elements in T, then $\tau_{|T}$ is a full list with respect to T.

2.1.1 Distance measures

How do we measure distance between two full lists with respect to a set S? Two popular distance measures are [12]:

(1) The *Spearman footrule distance* is the sum, over all elements $i \in S$, of the absolute difference between the rank of i according to the two lists. Formally, given two full lists σ and τ, the distance is given by $F(\sigma, \tau) = \sum_{i=1}^{|S|} |\sigma(i) - \tau(i)|$. After dividing this number by the maximum value $|S|^2/2$, one can obtain a normalized value of the footrule distance, which is always between 0 and 1. The footrule distance between two lists can be computed in linear time.

(2) The *Kendall tau distance* counts the number of pairwise disagreements between two lists; that is, the distance between two full lists σ and τ is $K(\sigma, \tau) = |\{(i, j) \mid i < j, \sigma(i) < \sigma(j), \text{ but } \tau(i) > \tau(j)\}|$. Dividing this number by the maximum possible value $\binom{|S|}{2}$ we obtain a normalized version of the Kendall distance. The Kendall distance for full lists is the 'bubble sort' distance, i.e., the number of pairwise adjacent transpositions needed to transform from one list to the other. The Kendall distance between two lists of length n can be computed in $n \log n$ time using simple data structures.

The above measures are metrics and extend in a natural way to several lists. Given several full lists $\sigma, \tau_1, \ldots, \tau_k$, for instance, the normalized Footrule distance of σ to $\tau_1, \ldots, \tau_k$ is given by $F(\sigma, \tau_1, \ldots, \tau_k) = (1/k) \sum_{i=1}^{k} F(\sigma, \tau_i)$.

One can define generalizations of these distance measures to partial lists. If $\tau_1, \ldots, \tau_k$ are partial lists, let U denote the union of elements in $\tau_1, \ldots, \tau_k$ and let σ be a full list with respect to U. Now, given σ, the idea is to consider the distance between τ_i and the projection of σ with respect to τ_i. Then, for instance, we have the *induced footrule distance* $F(\sigma, \tau_1, \ldots, \tau_k) = \sum_{i=1}^{k} F(\sigma_{|\tau_i}, \tau_i)/k$. In a similar manner, induced Kendall tau distance can be defined. Finally, we define a third notion of distance that measures the distance between a full list and a partial list on the same universe:

(3) Given one full list and a partial list, the *scaled footrule distance* weights contributions of elements based on the size of the lists they are present in. More formally, if σ is a full list and τ is a partial list, $F'(\sigma, \tau) = \sum_{i \in \tau} |\sigma(i)/|\sigma| - \tau(i)/|\tau||$. We will normalize F' by dividing by $|\tau|/2$.

Note that these distances are not necessarily metrics.

To a large extent, our interpretations of experimental results will be in terms of these distance measures. While these distance measures seem natural, why these measures are good is moot. We do not delve into such discussions here; the interested reader can find such arguments in the books by Diaconis [12], Critchlow [11], or Marden [17].

2.1.2 Optimal rank aggregation

In the generic context of rank aggregation, the notion of 'better' depends on what distance measure we strive to optimize. Suppose we wish to optimize Kendall distance, the question then is: given (full or partial) lists $\tau_1, \ldots, \tau_k$, find a σ such that σ is a full list with respect to the union of the elements of $\tau_1, \ldots, \tau_k$ and σ minimizes $K(\sigma, \tau_1, \ldots, \tau_k)$. The aggregation obtained by optimizing Kendall distance is called *Kemeny optimal aggregation* and in a precise sense, corresponds to the geometric median of the inputs. We show that computing the Kemeny optimal aggregation is NP-Hard even when $k = 4$ (see the Appendix). (Note that in contrast to the social choice scenario where there are many voters and relatively few candidates, in the web aggregation scenario we have many candidates (pages) and relatively few voters (the search engines).)

Kemeny optimal aggregations have a maximum likelihood interpretation. Suppose there is an underlying "correct" ordering σ of S, and each order $\tau_1, \ldots, \tau_k$ is obtained from σ by swapping two elements with some probability less than 1/2. Thus, the τ's are "noisy" versions of σ. A Kemeny optimal aggregation of $\tau_1, \ldots, \tau_k$ is one that is maximally likely to have produced the τ's (it need not be unique) [24]. Viewed differently, Kemeny optimal aggregation has the property of eliminating noise from various different ranking schemes. Furthermore, Kemeny optimal aggregations are essentially the only ones that simultaneously satisfy natural and important properties of rank aggregation functions, called neutrality and consistency in the social choice literature, and the so-called Condorcet property [25]. Indeed, Kemeny optimal

aggregations satisfy the *extended Condorcet criterion*. In Section 3 we establish a strong connection between satisfaction of the extended Condorcet criterion and fighting search engine "spam."

Given that Kemeny optimal aggregation is useful, but computationally hard, how do we compute it? The following relation shows that Kendall distance can be approximated very well via the Spearman footrule distance.

PROPOSITION 1. *[13] For any two full lists σ, τ, $K(\sigma, \tau) \leq F(\sigma, \tau) \leq 2K(\sigma, \tau)$.*

This leads us to the problem of *footrule optimal aggregation*. This is the same as before, except that the optimizing criterion is the footrule distance. In Section 4 we exhibit a polynomial time algorithm to compute optimal footrule aggregation (scaled footrule aggregation for partial lists). Therefore we have:

PROPOSITION 2. *If σ is the Kemeny optimal aggregation of full lists $\tau_1, \ldots, \tau_k$ and σ' optimizes the footrule aggregation, then $K(\sigma', \tau_1, \ldots, \tau_k) \leq 2K(\sigma, \tau_1, \ldots, \tau_k)$.*

Later, in Section 4, we develop rank aggregation methods that do not optimize any obvious criteria, but turn out to be very effective in practice.

2.2 Basic notions

Readers familiar with the notions in graph theory and Markov chains can skip this section.

2.2.1 Some concepts from graph theory

A *graph* $G = (V, E)$ consists of a set of *nodes* V and a set of *edges* E. Each element $e \in E$ is an unordered pair (u, v) of incident nodes, representing a connection between nodes u and v. A graph is *connected* if the node set cannot be partitioned into components such that there are no edges whose incident nodes occur in different components.

A *bipartite graph* $G = (V_1, V_2, E)$ consists of two disjoint sets of nodes V_1, V_2 such that each edge $e \in E$ has one node from V_1 and the other node from V_2. A bipartite graph is *complete* if each node in V_1 is connected to every node in V_2. A *matching* is a subset of edges such that for each edge in the matching, there is no other edge that shares a node with it. A *maximum matching* is a matching of largest cardinality. A *weighted graph* is a graph with a (non-negative) weight w_e for every edge e. Given a weighted graph, the *minimum weight maximum matching* is the maximum matching with minimum weight. The minimum weight maximum matching problem for bipartite graphs can be solved in time $O(n^{2.5})$ where n is the number of nodes.

A *directed graph* consists of nodes and edges, but this time an edge is an ordered pair of nodes (u, v), representing a connection from u to v. A *directed path* is said to exist from u to v if there is a sequence of nodes $u = w_0, \ldots, w_k = v$ such that (w_i, w_{i+1}) is an edge, for all $i = 0, \ldots, k-1$. A *directed cycle* is a non-trivial directed path from a node to itself. A *strongly connected component* of a graph is a set of nodes such that for every pair of nodes in the component, there is a directed path from one to the other. A *directed acyclic graph* (DAG) is a directed graph with no directed cycles. In a DAG, a *sink* node is one with no directed path to any other node.

2.2.2 Markov chains

A (homogeneous) *Markov chain* for a system is specified by a set of states $S = \{1, 2, \ldots, n\}$ and an $n \times n$ non-negative, stochastic (i.e., the sum of each row is 1) matrix M. The system begins in some start state in S and at each step moves from one state to another state. This transition is guided by M: at each step, if the system is in state i, it moves to state j with probability M_{ij}. If the current state is given as a probability distribution, the probability distribution of the next state is given by the product of the vector representing the current state distribution and M. In general, the start state of the system is chosen according to some distribution x (usually, the uniform distribution) on S. After t steps, the state of the system is distributed according to xM^t. Under some niceness conditions on the Markov chain (whose details we will not discuss), irrespective of the start distribution x, the system eventually reaches a unique fixed point where the state distribution does not change. This distribution is called the *stationary distribution*. It can be shown that the stationary distribution is given by the principal left eigenvector y of M, i.e., $yM = \lambda y$. In practice, a simple power-iteration algorithm can quickly obtain a reasonable approximation to y.

An important observation here is that the entries in y define a natural ordering on S. We call such an ordering the *Markov chain ordering of M*. A technical point to note while using Markov chains for ranking is the following. A Markov chain M defines a weighted graph with n nodes such that the weight on edge (u, v) is given by M_{uv}. The strongly connected components of this graph form a DAG. If this DAG has a sink node, then the stationary distribution of the chain will be entirely concentrated in the strongly connected component corresponding to the sink node. In this case, we only obtain an ordering of the alternatives present in this component; if this happens, the natural extended procedure is to remove these states from the chain and repeat the process to rank the remaining nodes. Of course, if this component has sufficiently many alternatives, one may stop the aggregation process and output a partial list containing some of the best alternatives. If the DAG of connected components is (weakly) connected and has more than one sink node, then we will obtain two or more clusters of alternatives, which we could sort by the total probability mass of the components. If the DAG has several weakly connected components, we will obtain incomparable clusters of alternatives. Thus, when we refer to a Markov chain ordering, we refer to the ordering obtained by this extended procedure.

3. SPAM RESISTANCE AND CONDORCET CRITERIA

In 1785 Marie J. A. N. Caritat, Marquis de Condorcet, proposed that if there is some element of S, now known as the *Condorcet alternative*, that defeats every other in pairwise simple majority voting, then that this element should be ranked first [9]. A natural extension, due to Truchon [22] (see also [21]), mandates that if there is a partition $(C, \bar{C})$ of S such that for any $x \in C$ and $y \in \bar{C}$ the majority prefers x to y, then x must be ranked above y. This is called the *extended Condorcet criterion* (ECC). We will show that not only can the ECC be achieved efficiently, but it also has excellent "spam-fighting" properties when used in the context of meta-search.

Intuitively, a search engine has been spammed by a page in its index, on a given query, if it ranks the page "too highly" with respect to other pages in the index, in the view of a "typical" user. Indeed, in accord with this intuition, search engines are both rated [18] and trained by human *evaluators*. This approach to defining spam: (1) permits an author to raise the rank of her page by improving the content; (2) puts *ground truth* about the relative value of pages into the purview of the users — in other words, the definition does not assume the existence of an absolute ordering that yields the "true" relative value of a pair of pages on a query; (3) does not assume unanimity of users' opinions or consistency among the opinions of a single user; and (4) suggests some natural ways to automate training of engines to incorporate useful biases, such as geographic bias.

We believe that reliance on evaluators in defining spam is unavoidable. (If the evaluators are human, the typical scenario during the design and training of search engines, then the eventual product will incorporate the biases of the training evaluators.) We *model* the evaluators by the search engine ranking functions. That is, we make the simplifying assumption that for any pair of pages, the relative ordering by the majority of the search engines comparing them is the same as the relative ordering by the majority of the evaluators. Our intuition is that if a page spams all or even most search engines for a particular query, then no combination of these search engines can defeat the spam. This is reasonable: Fix a query; if for some pair of pages a majority of the engines is spammed, then the aggregation function is working with overly bad data — garbage in, garbage out. On the other hand, if a page spams strictly fewer than half the search engines, then a majority of the search engines will prefer a "good" page to a spam page. In other words, under this definition of spam, the spam pages are the Condorcet losers, and will occupy the bottom partition of any aggregated ranking that satisfies the extended Condorcet criterion. Similarly, assuming that good pages are preferred by the majority to mediocre ones, these will be the Condorcet winners, and will therefore be ranked highly.

Many of the existing aggregation methods (see Section 4) do not ensure the election of the Condorcet winner, should one exist. Our aim is to obtain a simple method of modifying any initial aggregation of input lists so that the Condorcet losers (spam) will be pushed to the bottom of the ranking during this process. This procedure is called *local Kemenization* and is described next.

3.1 Local Kemenization

We introduce the notion of a locally Kemeny optimal aggregation, a relaxation of Kemeny optimality, that ensures satisfaction of the extended Condorcet principle and yet remains computationally tractable. As the name implies, local Kemeny optimal is a 'local' notion that possesses some of the properties of a Kemeny optimal aggregation.

A full list π is a *locally Kemeny optimal* aggregation of partial lists $\tau_1, \tau_2, \ldots, \tau_k$ if there is no full list π' that can be obtained from π by performing a single transposition of an adjacent pair of elements and for which $K(\pi', \tau_1, \tau_2, \ldots, \tau_k) < K(\pi, \tau_1, \tau_2, \ldots, \tau_k)$. In other words, it is impossible to reduce the total distance to the π's by flipping an adjacent pair.

Every Kemeny optimal aggregation is also locally Kemeny optimal, but the converse is false. Nevertheless, we show that a locally Kemeny optimal aggregation satisfies the extended Condorcet property and can be computed (see the Appendix) in time $O(kn \log n)$.

We have discussed the value of the extended Condorcet criterion in increasing resistance to search engine spam and in ensuring that elements in the top partitions remain highly ranked. However, specific aggregation techniques may add considerable value beyond simple satisfaction of this criterion; in particular, they may produce good rankings of alternatives within a given partition (as noted above, the extended Condorcet criterion gives no guidance within a partition). We now show how, using any initial aggregation μ of partial lists $\tau_1, \ldots, \tau_k$ — *one that is not necessarily Condorcet* — we can efficiently construct a locally Kemeny optimal aggregation of the τ's that is in a well-defined sense maximally consistent with μ. For example, if the τ's are full lists then μ could be the Borda ordering on the alternatives (see Section 4.1 for Borda's method). Even if a Condorcet winner exists, the Borda ordering may not rank it first. However, by applying our "local Kemenization" procedure (described below), we can obtain a ranking that is maximally consistent with the Borda ordering but in which the Condorcet winners are at the top of the list.

A *local Kemenization* (LK) of a full list μ with respect to $\tau_1, \ldots, \tau_k$ is a procedure that computes a locally Kemeny optimal aggregation of $\tau_1, \ldots, \tau_k$ that is (in a precise sense) maximally consistent with μ. Intuitively, this approach also preserves the strengths of the initial aggregation μ. Thus:

(1) the Condorcet losers receive low rank, while the Condorcet winners receive high rank (this follows from local Kemeny optimality)
(2) the result disagrees with μ on the order of any given pair (i, j) of elements only if a majority of those τ's expressing opinions disagrees with μ on (i, j)
(3) for every $1 \leq d \leq |\mu|$, the length d prefix of the output is a local Kemenization of the top d elements in μ.

Thus, if μ is an initial meta-search result, and we have some faith that the top, say, 100 elements of μ contain enough good pages, then we can build a locally Kemeny optimal aggregation of the projections of the τ's onto the top 100 elements in μ.

The local Kemenization procedure is a simple inductive construction. Without loss of generality, let $\mu = (1, \ldots, |\mu|)$. Assume inductively for that we have constructed π, a local Kemenization of the projection of the τ's onto the elements $1, \ldots, \ell-1$. Insert element ℓ into the lowest-ranked "permissible" position in π: just below the lowest-ranked element y in π such that (a) no majority among the (original) τ's prefers x to y and (b) for all successors z of y in π there is a majority that prefers x to z. In other words, we try to insert x at the end (bottom) of the list π; we bubble it up toward the top of the list as long as a majority of the τ's insists that we do.

A rigorous treatment of local Kemeny optimality and local Kemenization is given in the Appendix, where we also show that the local Kemenization of an aggregation is unique. On the strength of these results we suggest the following general approach to rank aggregation:

> Given $\tau_1, \ldots, \tau_k$, use your favorite aggregation method to obtain a full list μ. Output the (unique) local Kemenization of μ with respect to $\tau_1, \ldots, \tau_k$.

4. RANK AGGREGATION METHODS

4.1 Borda's method

Borda's method [6] is a "positional" method, in that it assigns a score corresponding to the positions in which a candidate appears within each voter's ranked list of preferences, and the candidates are sorted by their total score. A primary advantage of positional methods is that they are computationally very easy: they can be implemented in linear time. They also enjoy the properties called anonymity, neutrality, and consistency in the social choice literature [23]. However, they cannot satisfy the Condorcet criterion. In fact, it is possible to show that no method that assigns a weights to each position and then sorts the results by applying a function to the weights associated with each candidate satisfies the Condorcet criterion (see the Appendix and [23]).

Full lists. Given full lists $\tau_1, \tau_2, \ldots, \tau_k$, then for each candidate $c \in S$ and list τ_i, Borda's method first assigns a score $B_i(c) =$ the number of candidates ranked below c in τ_i, and then the total Borda score $B(c)$ is defined as $\sum_{i=1}^{k} B_i(c)$. The candidates are then sorted in decreasing order of total Borda score.

We remark that Borda's method can be thought of as assigning a k-element position *vector* to each candidate (the positions of the candidate in the k lists), and sorting the candidates by the L_1 norm of these vectors. Of course, there are plenty of other possibilities with such position vectors: sorting by L_p norms for $p > 1$, sorting by the median of the k values, sorting by the geometric mean of the k values, etc. This intuition leads us to several Markov chain based approaches, described in Section 4.3.

Partial lists. It has been proposed (e.g., in a recent article that appeared in *The Economist* [19]) that the right way to extend Borda to partial lists is by apportioning all the excess scores equally among all unranked candidates. This idea stems from the goal of being *unbiased*; however, it is easy to show that for any method of assigning scores to unranked candidates, there are partial information cases in which undesirable outcomes occur.

4.2 Footrule and scaled footrule

Since the footrule optimal aggregation is a good approximation of Kemeny optimal aggregation, it merits investigation.

Full lists. Footrule optimal aggregation is related to the median of the values in a position vector:

PROPOSITION 3. *Given full lists $\tau_1, \ldots, \tau_k$, if the median positions of the candidates in the lists form a permutation, then this permutation is a footrule optimal aggregation.*

Now, we obtain an algorithm for footrule optimal aggregation via the following proposition:

PROPOSITION 4. *Footrule optimal aggregation of full lists can be computed in polynomial time, specifically, the time to find a minimum cost perfect matching in a bipartite graph.*

PROOF. (Sketch): Let the union of $\tau_1, \ldots, \tau_k$ be S with n elements. Now, we define a a weighted complete bipartite graph (C, P, W) as follows. The first set of nodes $C = \{1, \ldots, n\}$ denotes the set of elements to be ranked (pages). The second set of nodes $P = \{1, \ldots, n\}$ denotes the n available positions. The weight $W(c, p)$ is the total footrule distance (from the τ_i's) of a ranking that places element c at position p, given by $W(c, p) = \sum_{i=1}^{k} |\tau_i(c) - p|$. It can be shown that a permutation minimizing the total footrule distance to the τ_i's is given by a minimum cost perfect matching in the bipartite graph. □

Partial lists. The computation of a footrule-optimal aggregation for partial lists is more problematic. In fact, it can be shown to be equivalent to the NP-hard problem of computing the minimum number of edges to delete to convert a directed graph into a DAG.

Keeping in mind that footrule optimal aggregation for full lists can be recast as a minimum cost bipartite matching problem, we now describe a method that retains the computational advantages of the full list case, and is reasonably close to it in spirit. We define the bipartite graph as before, except that the weights are defined differently. The weight $W(c, p)$ is the *scaled* footrule distance (from the τ_i's) of a ranking that places element c at position p, given by $W(c, p) = \sum_{i=1}^{k} |\tau_i(c)/|\tau_i| - p/n|$. As before, we can solve the minimum cost maximum matching problem on this bipartite graph to obtain the footrule aggregation algorithm for partial lists. We called this method the *scaled footrule aggregation* (SFO).

4.3 Markov chain methods

We propose a general method for obtaining an initial aggregation of partial lists, using Markov chains. The states of the chain correspond to the n candidates to be ranked, the transition probabilities depend in some particular way on the given (partial) lists, and the Markov chain ordering is the aggregated ordering. There are several motivations for using Markov chains:

(1) Handling partial lists and top d lists: Rather than require every pair of pages (candidates) i and j to be compared by every search engine (voter), we may now use the the *available* comparisons between i and j to determine the transition probability between i and j, and exploit the connectivity of the chain to (transitively) "infer" comparison outcomes between pairs that were not explicitly ranked by any of the search engines. The intuition is that Markov chains provide a more holistic viewpoint of comparing all n candidates against each other — significantly more meaningful than ad hoc and local inferences like "if a majority prefer A to B and a majority prefer B to C, then A should be better than C."

(2) Handling uneven comparisons: If a Web page P appears in the bottom half of about 70% of the lists, and is ranked Number 1 by the other 30%, how important is the quality of the pages that appear on the latter 30% of the lists? If these pages all appear near the bottom on the first set of 70% of the lists and the winners in these lists were not known to the other 30% of the search engines that ranked P Number 1, then perhaps we shouldn't consider P too seriously. In other words, if we view each list as a tournament within a league, we should take into account the strength of the schedule of matches played by each player. The Markov chain solutions we discuss are similar in spirit to the approaches considered in the mathematical community for this problem (eigenvectors of linear maps, fixed points of nonlinear maps, etc.).

(3) Enhancements of other heuristics: Heuristics for combining rankings are motivated by some underlying principle. For example, Borda's method is based on the idea "more wins is better." This gives some figure of merit for each candidate. It is natural to extend this and say "more wins against *good* players is even better," and so on, and iteratively refine the ordering produced by a heuristic. In the context of Web searching, the HITS algorithm of Kleinberg [16] and the PageRank algorithm of Brin and Page [7] are motivated by similar considerations. As we will see, some of the chains we propose are natural extensions (in a precise sense) of Borda's method, sorting by geometric mean, and sorting by majority.

(4) Computational efficiency: In general, setting up one of these Markov chains and determining its stationary probability distribution takes about $\Theta(n^2k+n^3)$ time. However, in practice, if we explicitly compute the transition matrix in $O(n^2k)$ time, a few iterations of the power method will allow us to compute the stationary distribution. In fact, we suggest an even faster method for practical purposes. For all of the chains that we propose, with about $O(nk)$ (linear in input size) time for preprocessing, it is usually possible to simulate one step of the chain in $O(k)$ time; thus by simulating the Markov chain for about $O(n)$ steps, we should be able to sample from the stationary distribution pretty effectively. This is usually sufficient to identify the top few candidates in the stationary distribution in $O(nk)$ time, perhaps considerably faster in practice.

We now propose some specific Markov chains, denoted $\mathrm{MC}_1, \mathrm{MC}_2, \mathrm{MC}_3$ and MC_4. For each of these chains, we specify the transition matrix and give some intuition as to why such a definition is reasonable. In all cases, the state space is the union of the sets of pages ranked by various search engines.

MC$_1$: If the current state is page P, then the next state is chosen uniformly from the multiset of all pages that were ranked higher than (or equal to) P by some search engine that ranked P, that is, from the multiset $\bigcup_i \{Q \mid \tau_i(Q) \leq \tau_i(P)\}$. The main idea is that in each step, we move from the current page to a better page, allowing about $1/j$ probability of staying in the same page, where j is roughly the average rank of the current page.

MC$_2$: If the current state is page P, then the next state is chosen by first picking a ranking τ uniformly from all the partial lists $\tau_1, \ldots, \tau_k$ containing P, then picking a page Q uniformly from the set $\{Q \mid \tau(Q) \leq \tau(P)\}$.

This chain takes into account the fact that we have several *lists* of rankings, not just a collection of pairwise comparisons among the pages. As a consequence, MC_2 is arguably the most representative of minority viewpoints of sufficient statistical significance; it also protects specialist views. In fact, MC_2 generalizes the geometric mean analogue of Borda's method. For full lists, if the initial state is chosen uniformly at random, after one step of MC_2, the distribution induced on its states produces a ranking of the pages such that P is ranked higher than (preferred to) Q iff the *geometric mean* of the ranks of P is lower than the geometric mean of the ranks of Q.

MC$_3$: If the current state is page P, then the next state is chosen as follows: first pick a ranking τ uniformly from all the partial lists $\tau_1, \ldots, \tau_k$ containing P, then uniformly pick a page Q that was ranked by τ. If $\tau(Q) < \tau(P)$ then go to Q, else stay in P.

This chain is a generalization of Borda method. For full lists, if the initial state is chosen uniformly at random, after one step of MC_3, the distribution induced on its states produces a ranking of the pages such that P is ranked higher than Q iff the Borda score of P is higher than the Borda score of Q. This is natural, considering that in any state P, the probability of staying in P is roughly the fraction of pairwise contests (with all other pages) that P won — a very Borda-like measure.

MC$_4$: If the current state is page P, then the next state is chosen as follows: first pick a page Q uniformly from the union of all pages ranked by the search engines. If $\tau(Q) < \tau(P)$ for a *majority* of the lists τ that ranked both P and Q, then go to Q, else stay in P.

This chain generalizes Copeland's suggestion of sorting the candidates by the number of pairwise majority contests they have won [10].

There are examples that differentiate the behavior of these chains. One can also show that the Markov ordering implied by these chains need not satisfy the extended Condorcet principle.

5. APPLICATIONS

We envisage several applications of our rank aggregation methods in the context of searching and retrieval in general, and the Web in particular. We present five major applications of our techniques in the following sections.

5.1 Meta-search

Meta-search is the problem of constructing a meta-search engine, which uses the results of several search engines to produce a collated answer. Several meta-search engines exist (e.g., metacrawler [3]) and many Web users build their own meta-search engines. As we observed earlier, the problem of constructing a good meta-search engine is tantamount to obtaining a good rank aggregation function for partial and top d lists. Given the different crawling strategies, indexing policies, and ranking functions employed by different search engines, meta-search engines are useful in many situations.

The actual success of a meta-search engine directly depends on the aggregation technique underlying it. Since the techniques proposed in Section 4 work on partial lists and top d lists, they can be applied to build a meta-search engine. The idea is simple: given a query, obtain the top (say) 100 results from many search engines, apply the rank aggregation function with the universe being the union of pages returned by the search engines, and return the top (say) 100 results of the aggregation. We illustrate this scheme in Section *6.2.1* and examine the performance of our methods.

5.2 Aggregating ranking functions

Given a collection of documents, the problem of indexing is: store the documents in such a manner that given a search term, those most relevant to the search term can be retrieved easily. This is a classic information retrieval problem and reasonably well-understood for static documents (see [20]). When the documents are hypertext documents, however, indexing algorithms could exploit the latent relationship between documents implied by the hyperlinks. On the Web, such an approach has already proved tremendously successful [16, 8, 7].

One common technique for indexing is to construct a *ranking function*. With respect to a query, a ranking function

can operate in two ways: (i) it can give an absolute score to a document indicating the relevance of the document to the query (score-based) or (ii) it can take two documents and rank order them with respect to the query (comparison-based). Based on the underlying methodology used, many competing ranking functions can be obtained. For instance, term-counting yields a simple ranking function. Another ranking function might be the consequence of applying the vector-space model and an appropriate distance measure to the document collection. Yet other ranking functions might be the ones implied by PageRank [7] and Clever [16, 8]. It is important to note that if the ranking function is score-based, the ordering implied by the scores makes more sense than the actual scores themselves, which are often either meaningless or inaccurate. And, for a particular ranking function and a query, it is often easier to return the top few documents relevant to the query than to rank the entire document base.

Given many ranking functions for a single document base, we have the case of top d lists, where d is the number of documents returned by each of the ranking functions. Our techniques can be applied to obtain a good aggregation of these ranking functions. Notice that we give equal weight to all the ranking functions, but this could be easily modified if necessary.

Such rank aggregation may be useful in other domains as well: many airline reservation systems suffer from lack of ability to express preferences. If the system is flexible enough to let the user specify various preference criteria (travel dates/times, window/aisle seating, number of stops, frequent-flier preferences, refundable/non-refundable nature of ticket purchase, and of course, price), it can rank the available flight plans based on each of the criteria, and apply rank aggregation methods to give better quality results to the user. Similarly, in the choice of restaurants from a restaurant database, users might rank restaurants based on several different criteria (cuisine, driving distance, ambiance, star-rating, dollar-rating, etc.). In both examples, users might be willing to compromise one or more of these criteria, provided there is a clear benefit with respect to the others. In fact, very often there is not even a clear order of importance among the criteria. A good aggregation function is a very effective way to make a selection in such cases.

5.3 Spam reduction

As we discussed earlier, the extended Condorcet principle is a reasonable cure for spam. Using the technique of local Kemenization, it is easy to take any rank aggregation method and tweak its output to make it satisfy the extended Condorcet principle. In fact, we suggest this as a general technique to reduce spam in search engines or meta-search engines: apply a favorite rank aggregation to obtain an initial ranking and then apply local Kemenization. This extra step is inexpensive in terms of computation cost, but has the benefit of reducing spam by ranking Condorcet losers below Condorcet winners. Again, we illustrate this application in Section *6.2.2* by examples.

5.4 Word association techniques

Different search engines and portals have different (default) semantics of handling a multi-word query. For instance, Altavista seems to use the OR semantics (it is enough for a document to contain one of the given query terms to be considered) while Google seems to use the AND semantics (it is mandatory for all the query words to appear in a document for it to be considered). As discussed in Section 1.1, both these scenarios are inconvenient in many situations.

Many of these tasks can be accomplished by a complicated Boolean query (via advanced query), but we feel that it is unreasonable to expect an average Web user to subscribe to this. Note also that simply asking for documents that contain as many of the keywords as possible is not necessarily a good solution: the best document on the topic might have only three of the keywords, while a spam document might well have four keywords. As a specific motivating example, consider searching for the job of a software engineer from an on-line job database. The user lists a number of skills and a number of potential keywords in the job description, for example, `"Silicon Valley C++ Java CORBA TCPIP algorithms start-up pre-IPO stock options"`. It is clear that the "AND" rule might produce no document, and the "OR" rule is equally disastrous.

We propose a *word association* scheme to handle these situations. Given a set of query words $w_1, \ldots, w_\ell$, we propose to construct several (say, k) sub-queries which are subsets of the original query words. We query the search engine with these k sub-queries (using the AND semantics) and obtain k top d (say, $d = 100$) results for each of the sub-queries. AND 2. locally Kemenize Then, we can use the methods in Sections 3 and 4 to obtain a locally Kemenized aggregation of the top d lists and output this as the final answer corresponding to the multi-word query. By examples, we illustrate this application in Section *6.2.3*.

Where do the words come from? One way to obtain such a set of query words is to prompt the user to associate as many terms as possible with the desired response. This might be too taxing on a typical user. A less demanding way is to let the user highlight some words in a current document; the search term are then extracted from the "anchor text," i.e., the words around the selected words.

5.5 Search engine comparison

Our methods also imply a natural way to compare the performance of various search engines. The main idea is that a search engine can be called good when it behaves like a least noisy expert for a query. In other words, a good search engine is one that is close to the aggregated ranking. This agrees with our earlier notion of what an expert is and how to deal with noisy experts. Thus, the procedure to rank the search engines themselves (with respect to a query) is as follows: obtain a rank aggregation of the results from various search engines and rank the search engines based on their (Kendall or footrule) distance to the aggregated ranking.

6. EXPERIMENTS AND RESULTS

6.1 Infrastructure

We conducted three types of experiments. The first experiment is to build a meta-search engine using different aggregation methods (Section 4) and compare their performances. The second experiment is to illustrate the effect of our techniques in combating spam. The third experiment is to illustrate the technique of word association for multi-word queries. While we provide numerical values for the first experiment, we provide actual examples for the second and third experiments.

We use the following seven search engines: Altavista (AV), Alltheweb (AW), Excite (EX), Google (GG), Hotbot (HB), Lycos (LY), and Northernlight (NL). For each of the search engines, we focused only on the top 100 queries. Our distance measurements are with respect to union of the top 100 results from these search engines.

For measuring the performance of our methods (first experiment), we selected the following 38 general queries (these queries are a superset of the 28 queries used in several earlier papers [4, 8]). For the second experiment, we pick some queries that were spammed in popular search engines. For the third experiment, we pick multi-word queries that perform poorly with existing search engines. Our notion of two urls being identical is purely syntactic (up to some canonical form); we do not use the content of page to determine if two urls are identical.

6.2 Results

6.2.1 Meta-search

The queries we used for our experiment are the following: "affirmative action", alcoholism, "amusement parks", architecture, bicycling, blues, cheese, "citrus groves", "classical guitar", "computer vision", cruises, "Death Valley", "field hockey", gardening, "graphic design", "Gulf war", HIV, java, Lipari, "lyme disease", "mutual funds", "National parks", "parallel architecture", "Penelope Fitzgerald", "recycling cans", "rock climbing", "San Francisco", Shakespeare, "stamp collecting", sushi, "table tennis", telecommuting, "Thailand tourism", "vintage cars", volcano, "zen buddhism", and Zener. The average intersection in the top 100 for any pair of search engines is given in Table 1, which shows the number of pages as a function of number of search engines in which they are present. For instance, the fourth column in the table means that 27.231 pages (on average) were present in exactly three of the search engine results. The second column indicates that around 284 pages were present in only one search engine while the last column indicates that less than 2 pages were present in all the search engines.

# engines	1	2	3	4	5	6	7
# pages	284.5	84.0	27.2	12.9	8.1	4.7	1.8

Table 1: Overlap among 7 search engine results.

The results of our first experiment are presented in Table 2. The performance is calculated in terms of the three distance measures described in Section 2.1. Each row corresponds to a method presented in Section 4. Local Kemenization (LK) was applied to the result of each of these methods.

6.2.2 Spam reduction

In the following we present anecdotal evidence of spam reduction by our methods. We use the following queries: `Feng Shui, organic vegetables, gardening`. For each of these queries, we look at the (top) pages that we consider spam. Notice that our definition of spam does not mean evil! — *it is just that in our opinion, these pages obtained an undeservedly high rank from one or more search engines.* It is easy to find urls that spammed a single search engine.

	K		IF		SF	
	− LK	+ LK	− LK	+ LK	− LK	+ LK
Borda	0.221	0.214	0.353	0.345	0.440	0.438
SFO	0.112	0.111	0.168	0.167	0.137	0.137
MC_1	0.133	0.130	0.216	0.213	0.292	0.291
MC_2	0.131	0.128	0.213	0.210	0.287	0.286
MC_3	0.116	0.114	0.186	0.183	0.239	0.239
MC_4	0.105	0.104	0.151	0.149	0.181	0.181

Table 2: Performance of various rank aggregation methods for meta-search. "K" is Kendall distance, "IF" is induced footrule distance, and "SF" is scaled footrule distance. "− LK" and "+ LK", respectively, denote without and with Local Kemenization.

On the other hand, we were interested in urls that spammed at least two search engines — given that the overlap among search engines was not very high, this proved to be a challenging task. Table 3 presents our examples: the entries are the rank within individual search engines' lists. A blank entry in the table indicates that the url was not returned as one of the top 100 by the search engine. Based on results from Section 6.2.1, we restrict our attention to SFO and MC_4 with local Kemenization.

6.2.3 Word associations

We use Google to perform our experiments on word associations. As noted earlier, Google uses AND semantics and hence for many interesting multi-word queries, the number or the quality of the pages returned is not very high. On the other hand, the fact that it uses the AND semantics is convenient to work with, when we supply small subsets of a multi-word query, in accordance to the word association rule described earlier. The queries, the top 5 results from Google and some of the top results from SFO and MC_4 (after local Kemenization) appear in the Appendix. We chose every pair of terms in the multi-word query to construct several lists and the apply rank aggregation (SFO and MC_4) to these lists.

6.3 Discussion

Of all the methods, MC_4 outperforms all others. In fact, it beats Borda by a huge margin. This is very interesting since Borda's method is the usual choice of aggregation, and perhaps the most natural. Scaled footrule and MC_3 (a generalization of Borda) seem to be on par. Recall that the footrule procedure for partial lists was only a heuristic modification of the footrule procedure for full lists. The above experimental evidence suggests that this heuristic is very good. MC_1 and MC_2 are always worse than the other Markov chains, but they are strictly better than Borda.

In general, local Kemenization seems to improve around 1–3% in terms of the distance measures. It can be shown formally that local Kemenization never does worse in the sense that the Kendall distance never deteriorates after local Kemenization. Interestingly, this seems to be true even for footrule and scaled footrule distances (although we don't know if this true always). We conclude that local Kemenization procedure is always worth applying: either the improvement is large and if not, then the time spent is small.

Examining the results in Section 6.2.2, we see that SFO and MC_4 are quite effective in combating spam. While we

url	AV	AW	GG	HB	LY	NL	SFO	MC_4
www.lucky-bamboo.com	4	43			41		144	63
www.cambriumcrystals.com		9	51		5		31	59
www.luckycat.com	11	14	26		13		49	36
www.davesorganics.com	84	19	1		17		77	93
www.frozen.ch		9		63	11		49	121
www.eonseed.com		18		6	16		23	66
www.augusthome.com	26	16		27	12	16	57	54
www.taunton.com		25			21		78	67
www.egroups.com		34			29		108	101

Table 3: Ranks of "spam" pages for the queries: `Feng Shui`, `organic vegetables` and `gardening`.

do not claim that our methods completely eliminate spam, our study shows that they reduce spam in general.

The results in Section *6.2.3* shows that our technique of word association combined with rank aggregation methods can improve the quality of search results for multi-word queries. In each of the three examples presented, Google typically produced a total of only around 10–15 pages, and the top 5 results were often poor (a direct consequence of the AND semantics). In sharp contrast, the urls produced by the rank aggregation methods turned out to contain a wealth of information about the topic of the query.

7. CONCLUSIONS AND FURTHER WORK

We have developed the theoretical groundwork for describing and evaluating rank aggregation methods. We have proposed and tested several rank aggregation techniques. Our methods have the advantage of being applicable in a variety of contexts and try to use as much information as available. The methods are also simple to implement, do not have any computational overhead, and out-perform popular classical methods like Borda's method. We have established the value of the extended Condorcet criterion in the context of meta-search, and have described a simple process, local Kemenization, for ensuring satisfaction of this criterion.

Further work involves trying to obtain a qualitative understanding of why the Markov chain methods perform very well. Also, it will be interesting to measure the efficacy of our methods on a document base with several competing ranking functions. Finally, this work originated in conversations with Helen Nissenbaum on bias in searching. A formal treatment of bias seems difficult but alluring.

8. REFERENCES

[1] Search Engine Watch, `www.searchenginewatch.com`

[2] Search Engine Watch article, `www.searchenginewatch.com /sereport/00/11-inclusion.html`

[3] Metacrawler, `www.metacrawler.com`

[4] K. Bharat and M. Henzinger. Improved algorithms for topic distillation in a hyperlinked environment. *ACM SIGIR*, pages 104–111, 1998.

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

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

[7] S. Brin and L. Page. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks*, 30(1-7):107–117, 1998.

[8] S. Chakrabarti, B. Dom, D. Gibson, R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins. Experiments in topic distillation, *Proc. ACM SIGIR Workshop on Hypertext Information Retrieval on the Web*, 1998.

[9] M.-J. Condorcet. *Éssai sur l'application de l'analyse à la probabilité des décisions rendues à la pluralité des voix*, 1785.

[10] A. H. Copeland. A reasonable social welfare function. *Mimeo*, University of Michigan, 1951.

[11] D. E. Critchlow. *Metric Methods for Analyzing Partially Ranked Data*, LNS 34, Springer-Verlag, 1985.

[12] P. Diaconis. *Group Representation in Probability and Statistics*. IMS Lecture Series 11, IMS, 1988.

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

[14] G. Even, J. Naor, B. Schieber, and M. Sudan. Approximating minimum feedback sets and multicuts in directed graphs. *Algorithmica*, 20(2):151–174, 1998.

[15] R. Fagin. Combining Fuzzy information from multiple systems. *JCSS*, 58(1):83–99, 1999.

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

[17] J. I. Marden. *Analyzing and Modeling Rank Data*. Monographs on Statistics and Applied Probability, No 64, Chapman & Hall, 1995.

[18] Media Metrix search engine ratings. `www.searchenginewatch.com/reports/mediametrix.html`

[19] D. G. Saari. The mathematics of voting: Democratic symmetry. *Economist*, pp. 83, March 4, 2000.

[20] G. Salton. *Automatic Text Processing—the Transformation, Analysis, and Retrieval of Information by Computer*. Addison-Wesley, 1989.

[21] J. H. Smith. *Aggregation of Preferences with Variable Electorate. SIAM J. on Applied Math.*, 41:1027–1041, 1973.

[22] M. Truchon. An extension of the Condorcet criterion and Kemeny orders. *cahier 98-15 du Centre de Recherche en Économie et Finance Appliqueés*, 1998.

[23] H. P. Young. An axiomatization of Borda's rule. *J. Economic Theory*, 9:43–52, 1974.

[24] H. P. Young. Condorcet's theory of Voting. *Amer. Political Sci. Review*, 82:1231–1244, 1988.

[25] H. P. Young and A. Levenglick. A consistent extension of Condorcet's election principle. *SIAM J. on Applied Math.*, 35(2):285–300, 1978.

APPENDIX

The Appendix is available through the on-line version of these conference proceedings.