A Framework for Unsupervised Rank Aggregation

Alexandre Klementiev, Dan Roth, and Kevin Small
Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, IL 61801
{klementi,danr,ksmall}@uiuc.edu

ABSTRACT

The need to meaningfully combine sets of rankings often comes up when one deals with ranked data. Although a number of heuristic and supervised learning approaches to rank aggregation exist, they generally require either domain knowledge or supervised ranked data, both of which are expensive to acquire. To address these limitations, we propose[1] a mathematical and algorithmic framework for learning to aggregate (partial) rankings in an unsupervised setting, and instantiate it for the cases of combining permutations and combining top-k lists. Furthermore, we also derive an unsupervised learning algorithm for rank aggregation (ULARA), which approximates the behavior of this framework by directly optimizing the weighted Borda count. We experimentally demonstrate the effectiveness of both approaches on the data fusion task.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Retrieval models

General Terms

Algorithms, Theory

Keywords

Ranking, Rank Aggregation, Distance-Based Models

1. INTRODUCTION

Consider the scenario where each member of a panel of judges independently generates a (partial) ranking over a set of items while attempting to reproduce a true underlying ranking according to their level of expertise. This setting motivates a fundamental machine learning and information retrieval (IR) problem - the necessity to meaningfully aggregate preference rankings into a joint ranking. The IR community refers to this as *data fusion*, where a joint ranking is derived from the outputs of multiple retrieval systems, possibly from several heterogeneous sources. A canonical data fusion task is *meta-search* where the aim is to aggregate Web search query results from several engines into a more accurate ranking.

One impediment to solving *rank aggregation* tasks is the high cost associated with acquiring full or partial preference information, making supervised approaches (e.g. [22, 23]) of limited utility. For data fusion, efforts to overcome this difficulty include applying domain specific heuristics [26] or collecting such preference information indirectly (e.g. using clickthrough data [15]). In order to address this limitation, we consider the task of learning to aggregate (partial) rankings *without supervision*.

Analyzing ranked data is an extensively studied problem in statistics [25], economics [2], information retrieval [26], and machine learning literature [1]. Mallows [24] introduced a distance-based model for fully ranked data and investigated its use with Kendall's and Spearman's metrics. The model was later generalized to other distance functions and for use with partially ranked data [4]. [20] proposed a multi-parameter extension, where multiple modal rankings (e.g. expert opinions) are available and use their formalism for supervised ensemble learning; they also analyzed their model for partially ranked data [21].

The first key contribution of our work is the derivation of an EM-based algorithm for learning the parameters of the extended Mallows model without supervision. We instantiate the model with appropriate distance functions for two important scenarios: combining permutations and combining top-k lists. In the context of defining distances between rankings, various metrics have been proposed and analyzed [4, 9]. Distances over top-k lists, i.e. rankings over the k most preferable objects, receive particular attention in the IR community [10]. [12] show that a class of distance functions between full rankings, such as Kendall's and Cayley's metrics, decompose into a sum of independent components allowing for efficient parameter estimation of the standard Mallows model.

The second key contribution of our work is the derivation of a novel decomposable distance function for top-k lists. We show it to be a generalization of the Kendall metric and demonstrate that it can be decomposed, enabling us to estimate the parameters of the extended Mallows model efficiently.

[1]This paper unifies and extends work from [18] and [19]

SIGIR'08 LR4IR Workshop, July 24, 2008, Singapore.

The third contribution is the derivation of an unsupervised learning algorithm for rank aggregation (ULARA), which approximates the learning of the parameters of this model by directly learning the parameters of a weighted Borda count through an optimization procedure.

The remainder of the paper is organized as follows: section 2 formalizes distance-based ranking models and introduces relevant notation. Section 3 derives our EM-based algorithm for learning model parameters and specifies the requirements for efficient learning and inference. Section 4 instantiates the framework for two common scenarios: permutations (full rankings) and top-k lists. Section 5 describes our approximation method based on an optimization of the weighted Borda count. Section 6 experimentally demonstrates the model's effectiveness in both cases. Finally, section 7 concludes the work and gives ideas for future directions.

2. DISTANCE-BASED RANKING MODELS

2.1 Notation and Definitions

Let $\{x_1, \ldots, x_n\}$ be a set of objects to be ranked, i.e. assigned rank-positions $1, \ldots, n$, by a judge. We denote the resulting permutation $\pi = (\pi(1), \ldots, \pi(n))$, where $\pi(i)$ is the rank assigned to object x_i. Correspondingly, we use $\pi^{-1}(j)$ to denote the index of the object assigned to rank j.

Let $\mathcal{S}_n$ be the set of all $n!$ permutations over n items, and let $d : \mathcal{S}_n \times \mathcal{S}_n \rightarrow \mathbb{R}$ be a distance function between two permutations. We will require $d(\cdot, \cdot)$ to be a *right-invariant metric* [6]: in addition to the usual properties of a metric, we will also require that the value of $d(\cdot, \cdot)$ does not depend on how the set of objects is indexed. In other words, $d(\pi, \sigma) = d(\pi\tau, \sigma\tau)\ \forall \pi, \sigma, \tau \in \mathcal{S}_n$, where $\pi\tau$ is defined by $\pi\tau(i) = \pi(\tau(i))$.

In particular, note that the right-invariance property implies that $d(\pi, \sigma) = d(\pi\pi^{-1}, \sigma\pi^{-1}) = d(e, \sigma\pi^{-1})$, where $e = (1, \ldots, n)$ is the identity permutation. That is, the value of d does not change if we re-index the objects such that one of the permutations becomes e and the other $\nu = \sigma\pi^{-1}$. Borrowing the notation from [12] we abbreviate $d(e, \nu)$ as $D(\nu)$. In a later section, when we define ν as a random variable, we may treat $D(\nu) = D$ as a random variable as well: whether it is a distance function or a r.v. will be clear from the context.

2.2 Mallows Models

While a large body of work on ranking models exists in statistics literature, of particular interest to us are the distance based conditional models first introduced in [24]. Let us give a brief review of the formalism and elucidate some of the its properties relevant to our work. The model generates a judge's rankings according to:

$$p(\pi|\theta, \sigma) = \frac{1}{Z(\theta, \sigma)} \exp(\theta\ d(\pi, \sigma)) \tag{1}$$

where $Z(\theta, \sigma) = \sum_{\pi \in \mathcal{S}_n} \exp(\theta\ d(\pi, \sigma))$ is a normalizing constant. The parameters of the model are $\theta \in \mathbb{R}$, $\theta \leq 0$ and $\sigma \in \mathcal{S}_n$, referred to as the dispersion and the location parameters, respectively. The distribution's single mode is the modal ranking σ; the probability of ranking π decreases exponentially with distance from σ. When $\theta = 0$, the distribution is uniform, and it becomes more concentrated at σ as θ decreases.

One property of (1) is that the normalizing constant $Z(\theta, \sigma)$ does not depend on σ due to the right invariance of the distance function:

$$Z(\theta, \sigma) = Z(\theta) \tag{2}$$

Let us denote the moment generating function of D under (1) as $M_{D,\theta}(t)$, and as $M_{D,0}(t)$ under the uniform distribution ($\theta = 0$). Since (1) is an exponential family,

$$M_{D,\theta}(t) = \frac{M_{D,0}(t + \theta)}{M_{D,0}(\theta)}$$

Therefore,

$$\begin{aligned} E_\theta(D) &= \frac{1}{M_{D,0}(\theta)} \frac{dM_{D,0}(t+\theta)}{dt}\bigg|_{t=0} \\ &= \frac{d \ln(M_{D,0}(t))}{dt}\bigg|_{t=\theta} \end{aligned} \tag{3}$$

[12] note that if a distance function can be expressed as $D(\pi) = \sum_{i=1}^{m} V_i(\pi)$, where $V_i(\pi)$ are independent (with π uniformly distributed) with m.-g.f. $M_i(t)$, then $M_{D,0}(t) = \prod_{i=1}^{m} M_i(t)$. Consequently, (3) gives:

$$E_\theta(D) = \frac{d}{dt} \sum_{i=1}^{m} \ln M_i(t)\bigg|_{t=\theta} \tag{4}$$

We will call such distance functions *decomposable* and will later use (4) in section 4 in order to estimate θ efficiently.

2.3 Extended Mallows Models

[20] propose a natural generalization of the Mallows model to the following conditional model:

$$p(\pi|\boldsymbol{\theta}, \boldsymbol{\sigma}) = \frac{1}{Z(\boldsymbol{\theta}, \boldsymbol{\sigma})} p(\pi) \exp\left(\sum_{i=1}^{K} \theta_i\ d(\pi, \sigma_i)\right) \tag{5}$$

where $\boldsymbol{\sigma} = (\sigma_1, \ldots, \sigma_K) \in \mathcal{S}_n^K$, $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_K) \in \mathbb{R}^K$, $\boldsymbol{\theta} \leq \mathbf{0}$, $p(\pi)$ is a prior, and normalizing constant $Z(\boldsymbol{\theta}, \boldsymbol{\sigma}) = \sum_{\pi \in \mathcal{S}_n} p(\pi) \exp(\sum_{i=1}^{K} \theta_i\ d(\pi, \sigma_i))$.

The rankings σ_i may be thought of as votes of K individual judges, e.g. rankings returned by multiple search engines for a particular query in the meta-search setting. The free parameters θ_i represent the degree of expertise of the individual judges: the closer the value of θ_i to zero, the less the vote of the i-th judge affects the assignment of probability.

Under the right-invariance assumption on d, we can use property (2) to derive the following generative story underlying the extended Mallows model:

$$p(\pi, \boldsymbol{\sigma}|\boldsymbol{\theta}) = p(\pi) \prod_{i=1}^{K} p(\sigma_i|\theta_i, \pi) \tag{6}$$

That is, π is first drawn from prior $p(\pi)$. $\boldsymbol{\sigma}$ is then made up by drawing $\sigma_1 \ldots \sigma_K$ *independently* from K Mallows models $p(\sigma_i|\theta_i, \pi)$ with the *same* location parameter π.

It is straightforward to generalize both Mallows models [4], and the extended Mallows models to *partial rankings* by constructing appropriate distance functions. We will assume this more general setting in the following section.

3. LEARNING AND INFERENCE

In this section, we derive the general formulation of Expectation Maximization algorithm for parameter estimation of the extended Mallows models (5), and suggest a class of distance functions for which learning can be done efficiently. We then describe an inference procedure for the model.

3.1 EM Background and Notation

Let us start with a brief overview of Expectation Maximization (EM) [5], mostly to introduce some notation. EM is a general method of finding maximum likelihood estimate of parameters of models which depend on unobserved variables. The EM procedure iterates between:

E step: estimate the expected value of complete data log-likelihood with respect to unknown data $\mathcal{Y}$, observed data $\mathcal{X}$, and current parameter estimates θ':

$$T(\theta, \theta') \quad = \quad E[\log p(\mathcal{X}, \mathcal{Y}|\theta)|\mathcal{X}, \theta']$$

M step: choose parameters that maximize the expectation computed in the E step:

$$\theta' \leftarrow \operatorname*{argmax}_{\theta} T(\theta, \theta')$$

In our setting, the $K > 2$ experts generate votes $\boldsymbol{\sigma}$ corresponding to the unobserved true ranking π. We will see multiple instances of $\boldsymbol{\sigma}$ so the observed data we get are ranking vectors $\mathcal{X} = \{\boldsymbol{\sigma}^{(j)}\}_{j=1}^{Q}$ with the corresponding true (unobserved) rankings $\mathcal{Y} = \{\pi^{(j)}\}_{j=1}^{Q}$.

In the meta-search example, $\sigma_i^{(j)}$ is the ranking of the i-th (of the total of K) search engine for the j-th (of the total of Q) query. The (unknown) true ranking corresponding to the j-th query is denoted as $\pi^{(j)}$.

3.2 EM Derivation

We now use the generative story (6) to derive the following propositions (proofs omitted due to space constraints):

PROPOSITION 1. *The expected value of the complete data log-likelihood under* (5) *is:*

$$T(\boldsymbol{\theta}, \boldsymbol{\theta'}) = \sum_{(\pi^{(1)}, \ldots, \pi^{(Q)}) \in \mathcal{S}_n^Q} \mathcal{L}_{\boldsymbol{\theta}} \; \mathcal{U}_{\boldsymbol{\theta'}} \tag{7}$$

where the complete data log-likelihood $\mathcal{L}_{\boldsymbol{\theta}}$ *is:*

$$\mathcal{L}_{\boldsymbol{\theta}} = \sum_{j=1}^{Q} \log p(\pi^{(j)}) - Q \sum_{i=1}^{K} \log Z(\theta_i) + \sum_{j=1}^{Q} \sum_{i=1}^{K} \theta_i \; d(\pi^{(j)}, \sigma_i^{(j)})$$

and the marginal distribution of the unobserved data $\mathcal{U}_{\boldsymbol{\theta'}}$ *is:*

$$\mathcal{U}_{\boldsymbol{\theta'}} = \prod_{j=1}^{Q} p\left(\pi^{(j)} | \boldsymbol{\theta'}, \boldsymbol{\sigma}^{(j)}\right)$$

PROPOSITION 2. $T(\boldsymbol{\theta}, \boldsymbol{\theta'})$ *is maximized by* $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_K)$ *such that:*

$$E_{\theta_i}(D) = \sum_{\substack{(\pi^{(1)}, \ldots, \pi^{(Q)}) \\ \in \mathcal{S}_n^Q}} \left(\frac{1}{Q} \sum_{q=1}^{Q} \; d(\pi^{(q)}, \sigma_i^{(q)}) \right) \mathcal{U}_{\boldsymbol{\theta'}} \tag{8}$$

That is, on each iteration of EM, we need to evaluate the right-hand side (RHS) of (8) and solve the LHS for θ_i for each of the K components.

3.3 Model Learning and Inference

At first, both evaluating the RHS of (8) and solving the LHS for θ_i seem quite expensive ($> n!$). While true in general, we can make the learning tractable for a certain type of distance functions.

In particular, if a distance function can be decomposed into a sum of independent components under the uniform distribution of π (see section 2.2), property (4) may enable us to make the estimation of the LHS efficient. In Section 4, we show two examples of such distance functions (for permutations and top-k lists).

In order to estimate the RHS, we use the Metropolis algorithm [14] to sample from (5). The chain proceeds as follows: denoting the most recent value sampled as π_t, two indices $i, j \in \{1, \ldots, n\}$ are chosen at random and the objects $\pi_t^{-1}(i)$ and $\pi_t^{-1}(j)$ are transposed forming π'_t. If $a = p(\pi'_t|\boldsymbol{\theta}, \boldsymbol{\sigma})/p(\pi_t|\boldsymbol{\theta}, \boldsymbol{\sigma}) \geq 1$ the chain moves to π'_t. If $a < 1$, the chain moves to π'_t with probability a; otherwise, it stays at π_t. [7] demonstrates rapid convergence for Mallows model with Cayley's distance. While no convergence results are known for the extended Mallows model with arbitrary distance, we found experimentally that the MC chain converges rapidly with the two distance functions used in this work ($10n$ steps in experiments of Section 6). As the chain proceeds, we update the distance value with the incremental change due to a single transposition, instead of recomputing it from scratch, resulting in substantial savings in computation.

Alternatively, we also found (Section 6.1) that combining rankings σ_i with the Borda count weighted by $\exp(-\theta_i)$ provides a reasonable and quick estimate for evaluating the RHS.

Sampling or the suggested alternative RHS estimation used during training is also used for model inference.

4. MODEL APPLICATION

Overcoming the remaining hurdle (the LHS estimation) in learning the model efficiently depends on the definition of a distance function. We now consider two particular types of (partial) rankings: permutations, and top-k lists. The latter is the case when each judge specifies a ranking over k most preferable objects out of n. For instance, a top-10 list may be associated with the 10 items on the first page of results returned by a web search engine. For both permutations and top-k lists, we show distance functions which satisfy the decomposability property (Section 2.2), which, in turn, allows us to estimate the LHS of (8) efficiently.

4.1 Combining Permutations

Kendall's tau distance [16] between permutations π and σ is a right-invariant metric defined as the minimum number of pairwise adjacent transpositions needed to turn one permutation into the other. Assuming that one of the permutations, say σ, is the identity permutation e (we can always turn one of the permutations into e by re-indexing the objects without changing the value of the distance, see Section 2.1), it can be written as:

$$D_K(\pi) = \sum_{i=1}^{n-1} V_i(\pi)$$

where[2] $V_i(\pi) = \sum_{j>i} I(\pi^{-1}(i) - \pi^{-1}(j))$. V_i are independent and uniform over integers $[0, n-i]$ [11] with m.-g.f. $M_i(t) = \frac{1}{n-i+1}\sum_{k=0}^{n-i} e^{tk}$. Following [12], equation (4) gives:

$$E_\theta(D_K) = \frac{ne^\theta}{1-e^\theta} - \sum_{j=1}^{n} \frac{je^{\theta j}}{1-e^{\theta j}} \quad (9)$$

$E_\theta(D_K)$ is monotone decreasing, so line search for θ will converge quickly.

4.2 Combining Top-*k* Lists

We now propose an extension of the Kendall's tau distance to top-k lists, i.e. the case where π and σ indicate preferences over different (possibly, overlapping) subsets of $k \leq n$ objects.

Let us denote by F_π and F_σ the elements in π and σ respectively, noting that $|F_\pi| = |F_\sigma| = k$. We define $Z = F_\pi \cap F_\sigma$, $|Z| = z$, $P = F_\pi \setminus F_\sigma$, and $S = F_\sigma \setminus F_\pi$ (note that $|P| = |S| = k - z = r$). We treat π and σ as rankings, which in our case means that the smallest index will indicate the top, i.e. contain the most preferred object. For notational convenience, let us now define the *augmented ranking* $\tilde{\pi}$ as π augmented with the elements of S assigned the same index $(k+1)$, one past the bottom of the ranking as shown on Figure 1 ($\tilde{\sigma}$ is defined similarly). We will slightly abuse our notation and denote $\tilde{\pi}^{-1}(k+1)$ to be the set of elements in position $(k+1)$.

Kendall's tau distance D_K is naturally extended from permutations to augmented rankings.

DEFINITION 1. *Distance $\tilde{D}_K(\tilde{\pi}, \tilde{\sigma})$ between augmented rankings $\tilde{\pi}$ and $\tilde{\sigma}$ is the minimum number of adjacent transpositions needed to turn $\tilde{\pi}$ into $\tilde{\sigma}$.*

It can be shown that $\tilde{D}_K(\tilde{\pi}, \tilde{\sigma})$ is a right-invariant metric, thus we will again simplify the notation denoting it as $\tilde{D}_K(\tilde{\pi})$. This distance can be decomposed as:

$$\tilde{D}_K(\tilde{\pi}) = \sum_{\substack{i=1 \\ \tilde{\pi}^{-1}(i) \in Z}}^{k} \tilde{V}_i(\tilde{\pi}) + \sum_{\substack{i=1 \\ \tilde{\pi}^{-1}(i) \notin Z}}^{k} \tilde{U}_i(\tilde{\pi}) + \frac{r(r+1)}{2}$$

where

$$\tilde{V}_i(\tilde{\pi}) = \sum_{\substack{j=i \\ \tilde{\pi}^{-1}(j) \in Z}}^{k} I(\tilde{\pi}^{-1}(i) - \tilde{\pi}^{-1}(j)) + \sum_{j \in \tilde{\pi}^{-1}(k+1)} I(\tilde{\pi}^{-1}(i) - j)$$

$$\tilde{U}_i(\tilde{\pi}) = \sum_{\substack{j=i \\ \tilde{\pi}^{-1}(j) \in Z}}^{k} 1$$

[2] $I(x) = 1$ if the variable $x > 0$ or a predicate x is true, and 0 otherwise.

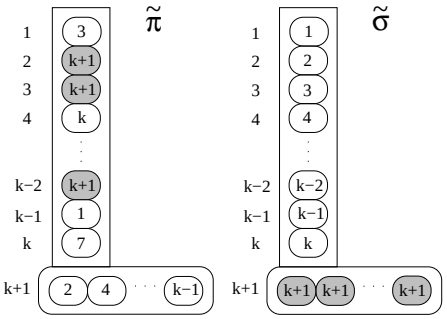

Figure 1: An example of augmented permutations $\tilde{\pi}$ (left) and identity augmented permutation $\tilde{\sigma}$ (right, in natural order). Grey boxes are objects in π but not in σ. $\tilde{D}_K(\tilde{\pi})$ is the minimum number of adjacent transpositions needed to turn $\tilde{\pi}$ into $\tilde{\sigma}$: namely, bring all grey boxes into the position $k+1$ and put the remaining k objects in their natural order.

Decomposing $\tilde{D}_K(\tilde{\pi})$, the second term is the minimum number of adjacent transpositions necessary to bring the r elements not in Z (grey boxes on Figure 1) to the bottom of the ranking. The third term is the minimum number of adjacent transpositions needed to switch them with the elements in $\tilde{\pi}^{-1}(k+1)$, which would then appear in the correct order in the bottom r positions. Finally, the first term is the adjacent transpositions necessary to put the k elements now in the list in the natural order.

It can be shown that the random variable summands comprising $\tilde{D}_K(\tilde{\pi})$ are independent when $\tilde{\pi}$ is uniformly distributed. Furthermore, $\tilde{V}_i$ and $\tilde{U}_j$ are uniform over integers $[0, k-i]$ and $[0, z]$, with moment generating functions $\frac{1}{k-i+1}\sum_{j=0}^{k-i} e^{tj}$ and $\frac{1}{z+1}\sum_{j=0}^{z} e^{tj}$, respectively. Assuming $z > 0$, and $r > 0$ equation (4) gives:

$$E_\theta(\tilde{D}_K) = \frac{ke^\theta}{1-e^\theta} - \sum_{j=r+1}^{k} \frac{je^{j\theta}}{1-e^{j\theta}} + \frac{r(r+1)}{2} - r(z+1)\frac{e^{\theta(z+1)}}{1-e^{\theta(z+1)}} \quad (10)$$

If $r = 0$ (i.e. the augmented rankings are over the same objects), both the distance and the expected value reduce to the Kendall distance results. Also, if $z = 0$ (i.e. the augmented rankings have no objects in common), $\tilde{D}_K = E_\theta(\tilde{D}_K) = k(k+1)/2$, which is the smallest number of adjacent transpositions needed to move the $r = k$ objects in $\tilde{\pi}^{-1}(k+1)$ into the top k positions.

$E_\theta(\tilde{D}_K)$ decreases monotonically, so we can again use line search to find the value of θ. Notice that the expected value depends on the value of z (the number of common elements between the two permutations). We will compute the average value of z as we estimate the RHS of (8) and use it to solve the LHS for θ.

5. DIRECTLY OPTIMIZING THE WEIGHTED BORDA COUNT

A common approach to aggregate a *single* set of votes is to find a permutation with the minimum average Kendall tau distance to those votes. Computing such a ranking,

Algorithm 1 Training

1: **Input:** $\{\boldsymbol{\sigma}^{(q)}\}_{q=1}^{Q}, \{\kappa_i\}_{i=1}^{K}, \lambda, \nu$
2: $\mathbf{w} \leftarrow \mathbf{0}$
3: $t \leftarrow 1$
4: **for** $q \leftarrow 1, \ldots, Q$ **do**
5: **for** $j \leftarrow 1, \ldots, n$ **do**
6: **if** $K_j^{(q)} \geq \nu$ **then**
7: $\mu^{(q)}(j) = \frac{\sum_{i \in \mathcal{K}_j^{(q)}} \sigma_i^{(q)}(j)}{K_j^{(q)}}$
8: **for** $i \leftarrow 1, \ldots, K$ **do**
9: **if** $\sigma_i^{(q)}(j) \leq \kappa_i$ **then**
10: $\nabla_i \leftarrow \left[\sigma_i^{(q)}(j) - \mu^{(q)}(j)\right]^2$
11: **else**
12: $\nabla_i \leftarrow \left[\kappa_i + 1 - \mu^{(q)}(j)\right]^2$
13: $w_i^t \leftarrow w_i^{t-1} + \lambda \cdot \nabla_i$
14: $t \leftarrow t + 1$
15: NORMALIZE(w)
16: **Output:** $\boldsymbol{w} \in [0,1]^K$

Figure 2: An unsupervised algorithm for rank aggregation: Training.

known as the Kemeny-optimal aggregation, is known to be NP-hard [8]. However, the well known Borda count method provides a good approximation [3] and is known to minimize the average Spearman's distance (11) to the constituent rankings.

$$d_S(\sigma, \pi) = \sum_{i=1}^{n} (\sigma(i) - \pi(i))^2 \tag{11}$$

Empirically, we found (see Section 6.1) the Borda count method *augmented* with weights representing relative ranker quality to be a good alternative in the inference step of Section 3.3. Here, we explore this idea further and propose a simple unsupervised algorithm (ULARA) to learn these weights directly by minimizing the empirical average (weighted) Spearman's distance between the votes of the constituent rankers and a surrogate true ranking. As before, we extend the algorithm to handle the top-k setting.

More formally, for an item x_j and a given query, let $\rho_B(j) = \sum_{i=1}^{K} \sigma_i(j); \forall j = 1, \ldots, n$ be the Borda count yielding a predicted true ranking $\hat{\pi}_B = \text{argsort}_{j=1,\ldots,n}\, \rho_B(j)$. Correspondingly, let $\rho_W(j) = \sum_{i=1}^{K} w_i \cdot \sigma_i(j); \forall j = 1, \ldots, n$ be the weighted Borda count yielding a predicted true ranking $\hat{\pi}_W = \text{argsort}_{j=1,\ldots,n}\, \rho_W(j)$. Furthermore, denote $\mathcal{K}_j$ as the set of rankers which placed the item x_j above their respective set threshold values κ_i, and $|\mathcal{K}_j| = K_j = \sum_{i=1}^{K} I(\sigma_i(j) \leq \kappa_i)$. Finally, let $\mu(j) = \frac{\sum_{i \in \mathcal{K}_j} \sigma_i(j)}{K_j}$ denote the mean ranking of x_j. The values $\mu = (\mu(1), \ldots, \mu(n))$ will be used in computing the distance (11) and will play the role of a surrogate true ranking. Slightly abusing the notation, we will use μ in place of a permutation when computing distance $d_S(\sigma, \mu)$.

We aim to learn the weights $\boldsymbol{w}$ minimizing the average weighted Spearman's distance with the additional restriction that they are positive and add up to one, i.e.

Algorithm 2 Evaluation

1: **Input:** $\boldsymbol{w}, \boldsymbol{\sigma}, \{\kappa_i\}_{i=1}^{K}$
2: **for** $j \leftarrow 1, \ldots, n$ **do**
3: $\rho_W(j) \leftarrow 0$
4: **for** $i \leftarrow 1, \ldots, K$ **do**
5: **if** $\sigma_i(j) \leq \kappa_i$ **then**
6: $\rho_W(j) \leftarrow \rho_W(j) + w_i \cdot \sigma_i(j)$
7: **else**
8: $\rho_W(j) \leftarrow \rho_W(j) + w_i \cdot (\kappa_i + 1)$
9: $\hat{\pi}_W \leftarrow \text{argsort}_{j=1,\ldots,n}\, \rho_W(j)$
10: **Output:** $\hat{\pi}_W$

Figure 3: An unsupervised algorithm for rank aggregation: Evaluation.

$$\underset{\boldsymbol{w}}{\text{argmin}} \quad \sum_{q=1}^{Q} \sum_{i=1}^{K} w_i \; d_S(\sigma_i^{(q)}, \mu^{(q)}) \tag{12}$$

$$\text{s.t.} \quad \sum_{i=1}^{K} w_i = 1; \forall i, w_i \geq 0. \tag{13}$$

Informally, the weights should be small for rankers which tend to disagree with the others, and vice versa.

5.1 The algorithm

As opposed to optimizing this problem directly, we use iterative gradient descent [17] to derive an online learning algorithm 1. It takes as input a set of rankings for each query $\{\boldsymbol{\sigma}^{(q)}\}_{q=1}^{Q}$ along with the associated ranking function threshold values $\{\kappa_i\}_{i=1}^{K}$, a learning rate λ, and a significance threshold value ν, all discussed in greater detail below. For each query q and item x_j, the rankings $(\sigma_1^{(q)}(j), \ldots, \sigma_K^{(q)}(j))$ are used to calculate the mean $\mu^{(q)}(j)$ (line 7), the gradient is determined (line 10), and the weight update is made (line 13). Once all of these updates are completed, the weight vector is normalized (line 15) to generate a probability vector for evaluation in algorithm 2. The remaining discussion entails algorithmic details for practical situations:

- Missing Rankings (κ_i) - For most settings, there are more items in the instance space than the individual ranking functions will return. In the top-k setting, for instance, systems return rankings over a subset of documents and most corpus documents remain unranked. We denote this threshold value as κ_i, noting that rankers may have different thresholds. If an item does not appear in a ranking, we substitute $\kappa_i + 1$ for update calculations (line 12), assuming unranked items are ranked just below the last ranked item.

- Variable Number of Rankers (ν) - Some items may only appear in the rankings of a subset of judges. If less than ν rankers, as defined by the user, rank an item, no updates are made for this item (line 6).

6. EXPERIMENTAL EVALUATION

We demonstrate the effectiveness of our approach for permutations and top-k lists considered in Section 4, as well the performance of the alternative algorithm proposed in Section 5.

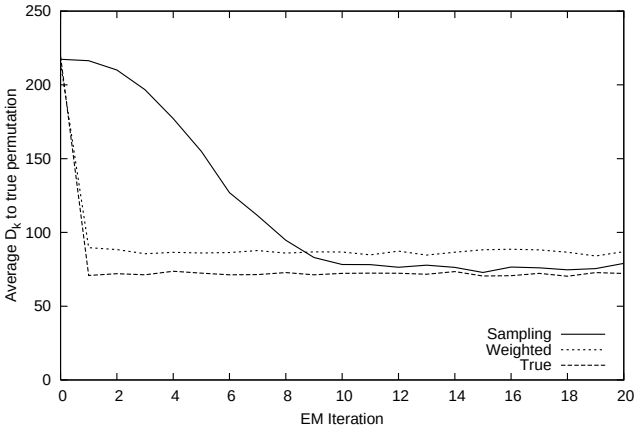

Figure 4: Permutations: learning performance of the model (averaged over 5 runs) when RHS is estimated using sampling (Sampling), the proposed weighted Borda count approximation (Weighted), or the true permutation (True). As expected, the model trained with the sampling method achieves better performance, but the approximation method performs quite well too and converges faster.

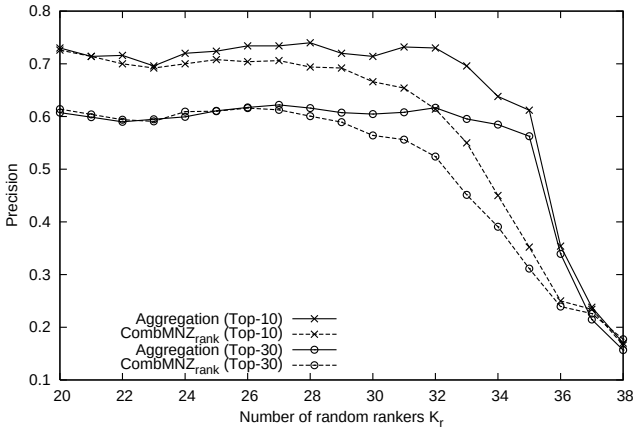

Figure 5: Top-k lists: precision of the aggregate ranker as a function of the number of random component rankers K_r in top 10 and top 30 documents. Our algorithm learns to discount the random components without supervision substantially improving over $CombMNZ_{rank}$.

6.1 Permutations

We first consider the scenario of aggregating permutations. For this set of experiments, the votes of $K = 10$ individual experts were produced by sampling standard Mallows models (1), with the same location parameter $\sigma^* = e$ (an identity permutation over $n = 30$ objects), and concentration parameters $\theta^*_{1,2} = -1.0$, $\theta^*_{3,\ldots,9} = -0.05$, and $\theta^*_{10} = 0$ (the latter generating all permutations uniformly randomly). The models were sampled 10 times, resulting in $Q = 10$ lists of permutations (one for each "query"), which constituted the training data.

In addition to the sampling procedure described in Section 3.3 to estimate the RHS of (8), we also tried the following approximation. For each "query" q, K constituent votes were combined into a single permutation $\hat{\sigma}_q$ with the weighted Borda method (defined in Section 5). The weights are computed using the current values of the model parameters as $\exp(-\theta_i)$. The rationale is that the smaller the absolute value of θ_i, the lower the relative quality of the ranker, and the less it should contribute to the aggregate vote. Finally, the RHS for the i-th component is computed as the distance from its vote to $\hat{\sigma}_q$ averaged over all Q queries.

We also tried using the true permutation σ^* in place of $\hat{\sigma}_q$ to see how well the learning procedure can do.

At the end of each EM iteration, we sampled the current model (5), and computed the Kendall's tau distance between the generated permutation to the true σ^*. Figure 4 shows the model performance when sampling and the proposed approximation are used to estimate the RHS. Although the convergence is much faster with the approximation, the model trained with the sampling method achieves better performance approaching the case when the true permutation is known.

6.2 Top-*k* lists

In order to estimate the model's performance in the top-k list combination scenario, we performed data fusion experiments using the data from the ad-hoc retrieval shared task of the TREC-3 conference [13]. Our goal here is to examine the behavior of our approach as we introduce poor judges into the constituent ranker pool. In this shared task, 40 participants submitted top-1000 ranking over a large document collection for each of the 50 queries. For our experiments, we used top-100 ($k = 100$) rankings from $K = 38$ of the participants (two of the participants generated shorter rankings for some of the queries and were not used) for all $Q = 50$ queries. We replaced a specific number $K_r \in [0, K]$ of the participants with random rankers (drawing permutations of k documents from the set of documents returned by all participants for a given query uniformly randomly). We then used our algorithm to combine top-k lists from K_r random rankers and $(K - K_r)$ participants chosen at random.

We measure performance using the precision in top-$\{10, 30\}$ documents as computed by *trec_eval*[3] from the TREC conference series. As a baseline, we use $CombMNZ_{rank}$, a variant of a commonly used $CombMNZ$ [26]. Given a query q for each document x_j in the collection it computes a score $\rho_{MNZ}(j) = K_j \cdot \sum_{i=1}^{K}(k+1-\sigma_i^{(q)}(j))$, where $\sigma_i^{(q)}(j) = (k+1)$ if the document doesn't appear in the ranking. K_j is the total number of participants which place x_j in their top-k rankings. The aggregate ranking is obtained by sorting documents according to their scores in descending order. Essentially, $CombMNZ_{rank}$ is a weighted Borda count method where the weighting is determined by the number of judges that rank the given document. Intuitively, the more judges rank a document highly, the higher it appears in the aggregate ranking.

Figure 5 shows that our algorithm learns to discount the random components *without supervision* substantially improving over the baseline as $K_r \to K$.

6.3 Model Dispersion Parameters

In order to demonstrate the relationship between the learned dispersion parameters of the model, $\boldsymbol{\theta}$, and the relative per-

[3]Available at http://trec.nist.gov/

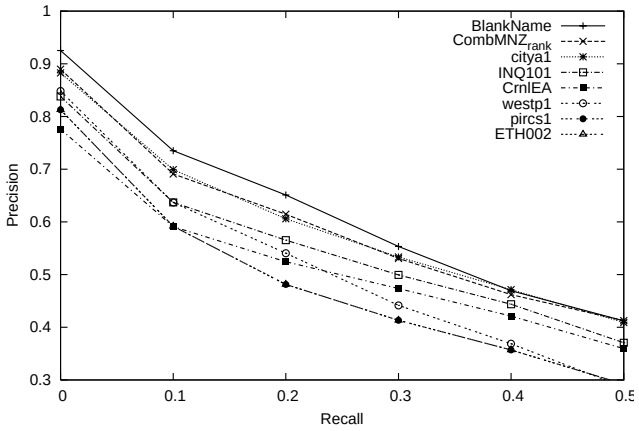

Figure 6: Experimental results for data fusion of the retrieval systems submitted to the TREC-3 shared task. While $CombMNZ_{rank}$ only negligibly outperforms the top system, ULARA performs significantly better than any component system at multiple recall levels.

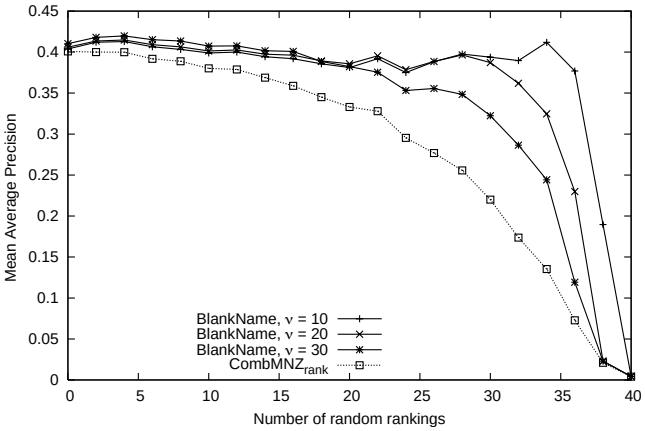

Figure 7: Experimental results where a number of TREC-3 systems are replaced with random rankings, demonstrating robustness of ULARA. While the performance of the $CombMNZ_{rank}$ algorithm deteriorates rapidly, ULARA performs well even when more than half of the systems are replaced.

Table 1: $MRPR$ of the four search engines and their corresponding model parameters; the results suggest a correlation between the magnitude of the dispersion parameters and the relative system performance.

	$S1$	$S2$	$S3$	$S4$
θ	-0.065	0.0	-0.066	-0.049
$MRPR$	0.86	0.43	0.82	0.78

formance of the constituent rankers, we also conducted a meta-search experiment. First, we generated $Q = 50$ queries which result in an unambiguous most relevant document and submitted them to $K = 4$ commercial search engines. For each engine, we kept the 100 highest ranked documents (10 pages of 10 documents each) after removing duplicates, and unified URL formatting differences between engines. We measure performance with Mean Reciprocal Page Rank ($MRPR$), which we define as mean reciprocal rank of the page number on which the correct document appears.

Table 1 shows $MRPR$ of the four search engines and their corresponding model parameters. As expected, the results suggest a correlation between the magnitude of the dispersion parameters and the relative system performance, implying that their values may also be used for unsupervised search engine evaluation. Finally, our model achieves $MRPR = 0.92$ beating all of the constituent rankers.

6.4 Top-*k* lists with ULARA

We also studied the ad-hoc retrieval shared task of the TREC-3 conference with the alternative algorithm we proposed in Sect. 5, demonstrating competitive behavior with significantly faster running times. In this experiment, we used the top-1000 rankings for each of the 50 queries from all $K = 40$ participants, setting $\kappa = 1000$. ULARA was used to combine the rankings of the individual research groups into an aggregate ranking $\hat{\pi}_W$. As previously, performance is quantified by the precision/recall curves and mean average precision metric as provided by the software (`trec_eval`) from the TREC conference series.

Figure 6 shows the results of the top individual submissions, $CombMNZ_{rank}$, and ULARA for the data fusion task. We observe that ULARA outperforms all component ranking functions as well as $CombMNZ_{rank}$. More significantly, while $CombMNZ_{rank}$ performs slightly better than the top system, ULARA achieves a relative increase in average precision of 4.0% at the top ranking, 6.4% at 0.1 recall, and 6.0% at 0.2 recall over $CombMNZ_{rank}$.

In the second experiment, as in Sec. 6.2, we demonstrate the robustness properties of ULARA by adding poor judges to constituent ranker pool. As before, we replaced a specified number of the $K = 40$ systems with a rankings drawn uniformly from all documents returned by all systems for a given query, denoted as *random rankings*. As figure 7 shows, the mean average precision of ULARA versus $CombMNZ_{rank}$ is consistently superior, becoming more pronounced as the number of random rankings is increased. To further explore this effect, we varied ν and observe that as more noise is added, ν must be lowered to accommodate the lack of agreement between rankers. Even under relatively extreme cicumstances, ULARA produces an aggregate ranking competitive with a noise free system; however, unlike the approach in Section 4, it does require manual setting of additional parameters.

7. CONCLUSIONS AND FUTURE WORK

We propose a formal mathematical and algorithmic framework for aggregating (partial) rankings without supervision. We derive an EM-based algorithm for the extended Mallows model and show that it can be made efficient for the right-invariant decomposable distance functions. We instantiate the framework and experimentally demonstrate its effectiveness for the important cases of combining permutations and combining top-k lists. In the latter case, we introduce the notion of augmented permutation and a novel decomposable distance function for efficient learning. In addition, we present an unsupervised algorithm for rank aggregation (ULARA) which approximates the mathematical framework

by directly optimizing a weighted Borda count.

A natural extension of the current work is to instantiate our framework for other types of partial rankings, as well as to cases where ranking data is not of the same type. The latter is of practical significance since often preference information available is expressed differently by different judges (e.g. top-k rankings of different lengths).

Another direction for future work is to extend the rank aggregation model to accommodate position dependence. In IR, more importance is generally given to results appearing higher in the rankings. Within our framework one may be able to design a distance function reflecting this requirement. Additionally, the quality of votes produced by individual components may depend on the rank, e.g. in the top-k scenario some rankers may be better at choosing few most relevant objects, while others may tend to have more relevant objects in the k selected but may not rank them well relative to one another. This case may be modeled by adding a dependency on rank to the dispersion parameters of the model.

Acknowledgments

We would like to thank Ming-Wei Chang, Sariel Har-Peled, Vivek Srikumar, and the anonymous reviewers for their valuable suggestions. This work is supported by NSF grant ITR IIS-0428472, DARPA funding under the Bootstrap Learning Program and by MIAS, a DHS-IDS Center for Multimodal Information Access and Synthesis at UIUC.

8. REFERENCES

[1] W. W. Cohen, R. E. Schapire, and Y. Singer. Learning to order things. *Journal of Artificial Intelligence Research*, 10:243–270, 1999.

[2] V. Conitzer. *Computational Aspects of Preference Aggregation*. PhD thesis, Carnegie Mellon University, 2006.

[3] D. Coppersmith, L. Fleischer, and A. Rudra. Ordering by weighted number of wins gives a good ranking for weighted tournaments. In *Proc of the Annual ACM Symposium on Discrete Algorithms*, pages 776–782, 2006.

[4] D. E. Critchlow. *Metric Methods for Analyzing Partially Ranked Data*, volume 34 of *Lecture Notes in Statistics*. Springer-Verlag, 1985.

[5] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, 39:1–38, 1977.

[6] P. Diaconis and R. L. Graham. Spearman's footrule as a measure of disarray. *Journal of the Royal Statistical Society*, 39:262–268, 1977.

[7] P. Diaconis and L. Saloff-Coste. What do we know about the Metropolis algorithm? *Journal of Computer and System Sciences*, 57:20–36, 1998.

[8] C. Dwork, R. Kumar, M. Naor, and D. Sivakumar. Rank aggregation methods for the web. In *Proc. of the International World Wide Web Conference (WWW)*, pages 613–622, 2001.

[9] V. Estivill-Castro, H. Mannila, and D. Wood. Right invariant metrics and measures of presortedness. *Discrete Applied Mathematics*, 42:1–16, 1993.

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

[11] W. Feller. *An Introduction to Probability Theory and Its Applications*, volume 1. John Wiley and Sons, Inc., 1968.

[12] M. A. Fligner and J. S. Verducci. Distance based ranking models. *Journal of the Royal Statistical Society*, 48:359–369, 1986.

[13] D. Harman. Overview of the third Text REtrieval Conference (TREC-3), 1994.

[14] W. K. Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97–109, April 1970.

[15] T. Joachims. Unbiased evaluation of retrieval quality using clickthrough data. In *SIGIR Workshop on Mathematical/Formal Methods in Information Retrieval*, 2002.

[16] M. G. Kendall. A new measure of rank correlation. *Biometrika*, 30(1/2):81–93, Jun. 1938.

[17] J. Kivinen and M. K. Warmuth. Additive versus exponentiated gradient updates for linear prediction. In *Proc, of the Annual ACM Symposium on Theory of Computing*, pages 209–218, 1995.

[18] A. Klementiev, D. Roth, , and K. Small. An unsupervised learning algorithm for rank aggregation. In *Proc. of the European Conference on Machine Learning (ECML)*, pages 616–623, 2007.

[19] A. Klementiev, D. Roth, , and K. Small. Unsupervised rank aggregation with distance-based models. In *Proc. of the International Conference on Machine Learning (ICML)*, 2008.

[20] G. Lebanon and J. Lafferty. Cranking: Combining rankings using conditional probability models on permutations. In *Proc. of the International Conference on Machine Learning (ICML)*, 2002.

[21] G. Lebanon and J. Lafferty. Conditional models on the ranking poset. In *The Conference on Advances in Neural Information Processing Systems (NIPS)*, volume 15, pages 431–438, 2003.

[22] D. Lillis, F. Toolan, R. Collier, and J. Dunnion. Probfuse: A probabilistic approach to data fusion. In *Proc. of the International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 139–146, 2006.

[23] Y.-T. Liu, T.-Y. Liu, T. Qin, Z.-M. Ma, and H. Li. Supervised rank aggregation. In *Proc. of the International World Wide Web Conference (WWW)*, 2007.

[24] C. L. Mallows. Non-null ranking models. *Biometrika*, 44:114–130, 1957.

[25] J. I. Marden. *Analyzing and Modeling Rank Data*. CRC Press, 1995.

[26] J. A. Shaw and E. A. Fox. Combination of multiple searches. In *Text REtrieval Conference (TREC)*, pages 243–252, 1994.