Learning to Rank By Aggregating Expert Preferences

Maksims N. Volkovs
University of Toronto
40 St. George Street
Toronto, ON M5S 2E4
mvolkovs@cs.toronto.edu

Hugo Larochelle
Université de Sherbrooke
2500 boul. de l'Université
Sherbrooke, QC J1K 2R1
hugo.larochelle@usherbrooke.ca

Richard S. Zemel
University of Toronto
40 St. George Street
Toronto, ON M5S 2E4
zemel@cs.toronto.edu

ABSTRACT

We present a general treatment of the problem of aggregating preferences from several experts into a consensus ranking, in the context where information about a target ranking is available. Specifically, we describe how such problems can be converted into a standard learning-to-rank one on which existing learning solutions can be invoked. This transformation allows us to optimize the aggregating function for any target IR metric, such as Normalized Discounted Cumulative Gain, or Expected Reciprocal Rank. When applied to crowdsourcing and meta-search benchmarks, our new algorithm improves on state-of-the-art preference aggregation methods.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Retrieval models*; I.2.6 [**Artificial Intelligence**]: Learning

General Terms

Algorithms, Experimentation

Keywords

Preference Aggregation, Meta-search, Crowdsourcing

1. INTRODUCTION

Preference aggregation is the problem of combining multiple preferences over objects into a single consensus ranking. This problem is crucially important in many applications, such as information retrieval (IR), collaborative filtering and crowdsourcing. Across various domains, the preferences over objects are expressed in many different ways, ranging from full and partial rankings to arbitrary comparisons. For instance, in meta-search an issued query is sent to several search engines and the (often partial) rankings returned by them are aggregated to generate more comprehensive ranking results. In crowdsourcing, tasks often involve assigning ratings to objects or pairs of objects ranging from images to text. The ratings from several users are then aggregated to produce a single labelling of the data.

Research in preference aggregation has largely concentrated on unsupervised aggregation where the ground truth ranking is unknown and the aim is to produce an aggregate ranking that satisfies the majority of the observed preferences. However, many of the recent aggregation problems are amenable to supervised learning, as ground truth ranking information is available. For example, in meta-search the documents retrieved by the search engines are often given to annotators who assign relevance labels to each document. The relevance labels provide ground truth preference information about the documents i.e., the documents with higher relevance label are to be ranked above those with lower one. Similarly, in crowdsourcing, a domain expert typically labels a subset of the data shown to the "crowd". These labels are then used to evaluate the quality of annotations submitted by each worker. In these settings aggregating methods that aim to satisfy the majority often lead to suboptimal results since the majority preference can often be wrong. In meta-search for instance, many of the search engines are likely to return incorrect results for difficult long-tail queries, and we expect high quality rankings only from a few search engines that were specifically optimized for these queries. Consequently, we need the aggregating function to be able to "specialize" and infer when to use the majority preference and when to concentrate only on a small subset of preferences. The specialization property is impossible to achieve with unsupervised preference aggregation techniques. There is thus an evident need to develop an effective supervised aggregation method that is able to fully utilize the labeled data.

In this paper we address this problem by developing a general framework for supervised preference aggregation. Our framework is based on a simple yet powerful idea of transforming the aggregation problem into a learning-to-rank one. This transformation makes it possible to apply any learning-to-rank method to optimize the parameters of the aggregating function for the target metric. Experiments on crowdsourcing task from TREC2011 [17] and meta-search tasks with Microsoft's LETOR4.0 [20] data sets show that our model significantly outperforms existing aggregation methods.

2. PREFERENCE AGGREGATION

A typical supervised preference aggregation problem consists of N training instances where for each instance n we are

CIKM'12, October 29–November 2, 2012, Maui, HI, USA.

given a set of M_n items. We also have a set of K "experts" and each expert k generates a list of preferences for items in each instance. The preferences can be in the form of full or partial rankings, top-T lists, ratings, relative item comparisons, or combinations of these. All of these forms can be converted to a set of *partial pairwise preferences*, which in most cases will be neither complete nor consistent. Moreover, for each instance n we are also given a set of ground truth preferences over the items. The goal is to learn an aggregating function which takes expert preferences as input and produces an aggregate ranking that maximally agrees with the ground truth preferences.

In this work we concentrate on the rank aggregation instance of this problem from the information retrieval domain. However, the framework that we develop is general and can be applied to any supervised preference aggregation problem in the form defined above. In information retrieval the instances correspond to queries $\mathbf{Q} = \{q_1, ..., q_N\}$ and items to documents $\mathbf{D}_n = \{d_{n1}, ..., d_{nM_n}\}$ where M_n is the number of documents retrieved for q_n. For each query q_n the experts' preferences are summarized in a rank matrix $\mathbf{R}_n$ where $\mathbf{R}_n(i,k)$ denotes the rank assigned to document d_{ni} by the expert $k \in \{1, .., K\}$. Note that the same K experts rank items for all the instances so K is instance independent. Furthermore, $\mathbf{R}_n$ can be sparse, as experts might not assign ranks to every document in $\mathbf{D}_n$; we use $\mathbf{R}_n(i,k) = 0$ to indicate that document d_{ni} was not ranked by expert k. The sparsity arises in problems like meta-search where q_n is sent to K different search engines and each search engine typically retrieves and ranks only a portion of the documents in $\mathbf{D}_n$. The ground truth preferences are expressed by the relevance levels $\mathbf{L}_n = \{l_{n1}, ..., l_{nM_n}\}$ which are typically assigned to the documents by human annotators.

A ranking of documents in $\mathbf{D}_n$ can be represented as a permutation of $\mathbf{D}_n$. A permutation π is a bijection $\pi : \mathbf{D}_n \rightarrow \{1, ..., M_n\}$ mapping each document d_{ni} to its rank $\pi(d_{ni}) = j$, and $d_{ni} = \pi^{-1}(j)$. Given the training queries the goal is to create an aggregating function such that, given a set of documents $\mathbf{D}$ with relevance levels $\mathbf{L}$ and ranking matrix $\mathbf{R}$ retrieved for a new query q, the aggregate ranking π produced by the function has maximal agreement with $\mathbf{L}$.

Normalized Discounted Cumulative Gain (NDCG) [14] is typically used to evaluate the agreement between the aggregate permutation and the relevance levels. Given the aggregate permutation π, relevances $\mathbf{L}$, and truncation level $T \leq M$, NDCG is defined as:

$$NDCG(\pi, \mathbf{L})@T = \frac{1}{G_T(\mathbf{L})} \sum_{i=1}^{T} \frac{2^{\mathbf{L}(\pi^{-1}(i))} - 1}{\log(i+1)} \qquad (1)$$

where $\mathbf{L}(\pi^{-1}(i))$ is the relevance level of the document in position i in π and $G_T(\mathbf{L})$ is a normalizing constant.

In contrast to the learning-to-rank problem where each document is represented by a fixed length, query dependent, and typically heavily engineered feature vector, in rank aggregation the rank matrix $\mathbf{R}$ is the only information available to train the aggregating function. Additionally this matrix is typically very sparse. Hence there is no fixed length document description, as is required by most supervised methods. To overcome this problem we convert the rank matrix into a pairwise preference matrix and demonstrate that through this conversion the rank aggregation problem can be cast into a learning-to-rank one which is well explored with many developed approaches readily available.

	EXPERTS		
d_1	2	7	0
d_2	0	0	1
d_3	10	5	0
d_4	0	15	3

(a) $\mathbf{R}$

	d_1	d_2	d_3	d_4
d_1	0	0	0	0.28
d_2	0	0	0	0
d_3	0.12	0	0	0.41
d_4	0	0	0	0

(b) $\mathbf{Y}_2$

Figure 1: (a) An example rank matrix R where 4 documents are ranked by 3 experts. Note that in meta-search the rank for a given document can be greater than the number of documents in R. (b) The resulting pairwise matrix $\mathbf{Y}_2$ for expert 2 (second column of R) after the ranks are transformed to pairwise preferences using the log rank difference method.

2.1 Pairwise Preferences

Given the $M_n \times K$ ranking matrix $\mathbf{R}_n$ our aim is to convert it into K $M_n \times M_n$ pairwise matrices $\mathbf{Y}_n = \{\mathbf{Y}_{n1}, ..., \mathbf{Y}_{nK}\}$, where each $\mathbf{Y}_{nk}$ expresses the preferences between pairs of documents based on the expert k. To convert $\mathbf{R}_n$ to $\mathbf{Y}_n$ we use a transformation of the form $\mathbf{Y}_{nk}(i,j) = g(\mathbf{R}_n(i,k), \mathbf{R}_n(j,k))$. In this work we experiment with three versions for g that were proposed by Gleich & Lim [12]:

1. **Binary Comparison**

   The rank magnitude is ignored:

   $$\mathbf{Y}_{nk}(i,j) = I[\mathbf{R}_n(i,k) < \mathbf{R}_n(j,k)]$$

   here $I[x]$ is an indicator function evaluating to 1 if x is true and to 0 otherwise.

2. **Rank Difference**

   Here the normalized rank difference is used:

   $$\mathbf{Y}_{nk}(i,j) = I[\mathbf{R}_n(i,k) < \mathbf{R}_n(j,k)] \frac{\mathbf{R}_n(j,k) - \mathbf{R}_n(i,k)}{\max(\mathbf{R}_n(:,k))}$$

   Normalizing by the maximum rank assigned by the expert k ($\max(\mathbf{R}_n(:,k))$) ensures that the entries of $\mathbf{Y}_{nk}$ have comparable ranges.

3. **Log Rank Difference**

   This method uses the normalized log rank difference:

   $$\mathbf{Y}_{nk}(i,j) = I[\mathbf{R}_n(i,k) < \mathbf{R}_n(j,k)] \frac{\log(\mathbf{R}_n(j,k)) - \log(\mathbf{R}_n(i,k))}{\log(\max(\mathbf{R}_n(:,k)))}$$

In all cases both $\mathbf{Y}_{nk}(i,j)$ and $\mathbf{Y}_{nk}(j,i)$ are set to 0 if either $\mathbf{R}_n(j,k) = 0$ or $\mathbf{R}_n(i,k) = 0$ (missing ranking). Non-zero entries $\mathbf{Y}_{nk}(i,j)$ represent the strength of the pairwise preference of d_{ni} over d_{nj} expressed by expert k. Figure 1 shows an example ranking matrix and the resulting pairwise matrix after the ranks are transformed to pairwise preferences using the log rank difference method. Note that preferences in the form of ratings and top-N lists can easily be converted into

Table 1: A summary of notation.

Variable	Description
$\mathbf{Q} = \{q_1, ..., q_N\}$	input queries
$\mathbf{D}_n = \{d_{n1}, ..., d_{nM_n}\}$	documents for q_n
$\mathbf{L}_n = \{l_{n1}, ..., l_{nM_n}\}$	relevance levels for q_n
$\mathbf{R}_n(i,k)$	ranking for d_{ni} by expert k
$\mathbf{Y}_n = \{\mathbf{Y}_{n1}, ..., \mathbf{Y}_{nK}\}$	pairwise representation of $\mathbf{R}_n$
$\psi(d_{ni})$	feature vector for d_{ni}
$f(\psi(d_{ni}), \mathbf{W})$	scoring function
$\mathbf{S}_n = \{s_{n1}, ..., s_{nM_n}\}$	scores given by f to D_n

$\mathbf{Y}_{nk}$ using the same transformations. Moreover, if pairwise preferences are observed, we simply skip the transformation step and fill the entries $\mathbf{Y}_{nk}(i,j)$ directly. Table 1 summarizes the notation used throughout this paper.

Working with pairwise comparisons has a number of advantages, and models over pairwise preferences have been extensively used in areas such as social choice [10], information retrieval [16, 5], and collaborative filtering [22, 12]. First, pairwise comparisons are the building blocks of almost all forms of evidence about preference and subsume the most general models of evidence proposed in literature. A model over pairwise preferences can thus be readily applied to a wide spectrum of preference aggregation problems and does not impose any restrictions on the input type. Second, pairwise comparisons are a relative measure and help reduce the bias from the preference scale. In meta-search for instance, each of the search engines that receives the query can retrieve diverse lists of documents significantly varying in size. By converting the rankings into pairwise preferences we reduce the list size bias emphasizing the importance of the relative position.

3. RELEVANT WORK

Relevant previous work on rank aggregation can be divided into two categories: permutation-based and score-based. In this section we briefly describe both types of models.

3.1 Permutation-Based Models

Permutation based models work directly in the permutation space. The most common and well explored such model is the Mallows model [24]. Mallows defines a distribution over permutations and is typically parametrized by a central permutation σ and a dispersion parameter $\phi \in (0, 1]$; the likelihood of a ranking matrix $\mathbf{R}$ under this model is given by:

$$P(\mathbf{R}_n|\phi, \sigma_n) = \prod_{k=1}^{K} \frac{1}{Z(\phi, \sigma_n)} \phi^{-d(\mathbf{R}_n(:,k), \sigma_n)}$$

where $d(\mathbf{R}(:,k), \sigma_n)$ is a distance between the ranking given by the expert k and the σ_n. For rank aggregation problems inference in this model amounts to finding the aggregate permutation σ_n that maximizes the likelihood of the observed rankings $\mathbf{R}$. However, finding the most likely permutation is typically very difficult and in many cases is intractable [25]. A number of extensions of the Mallows model, e.g., [28, 22, 18, 19], have also recently been explored.

Another disadvantage of these models is that they are difficult to adapt to the fully supervised rank aggregation problem considered here. The dispersion parameter ϕ and the parameters that define the distance function d are typically estimated via maximum likelihood with σ_n clamped to the ground truth ranking. This learning procedure cannot be used to optimize for a particular target metric, such as NDCG.

3.2 Score-Based Models

In score-based approaches the goal is to learn a set of real valued scores (one per document) $\mathbf{S}_n = \{s_{n1}, ..., s_{nM_n}\}$ which are then used to sort the documents. Working with scores avoids the discontinuity problems of the permutation space.

A number of heuristic score-based methods for rank aggregation have been proposed. For example, BordaCount [1], Condorcet [26] and Reciprocal Rank Fusion [8] derive the document scores by averaging (weighted) ranks across the experts or counting the number of pairwise wins. In statistics a popular pairwise score model is the Bradley-Terry [4] model:

$$P(\mathbf{R}_n|\mathbf{S}_n) = \prod_{k=1}^{K} \prod_{i \neq j} I[\mathbf{R}_n(i,k) < \mathbf{R}_n(j,k)] \frac{e^{s_{ni}}}{e^{s_{ni}} + e^{s_{nj}}}$$

where $\frac{\exp(s_{ni})}{\exp(s_{ni})+\exp(s_{nj})}$ can be interpreted as the probability that document d_{ni} beats d_{nj} in the pairwise contest and I is an indicator function. Note that the scores do not depend on the expert k and thus represent the consensus preference expressed by the experts. In logistic form the Bradley-Terry model is very similar to another popular pairwise model, the Thurstone model [29]. Extensions of these models include the Elo Chess rating system [11], adopted by the World Chess Federation FIDE in 1970, and Microsoft's TrueSkill rating system [9] for player matching in online games, used extensively in Halo and other games. The popular learning-to-rank model RankNet [5] is also based on this approach. The Bradley-Terry model was later generalized by Plackett and Luce to a Plackett-Luce model for permutations [23, 27]. A Bayesian framework was also recently introduced for the Plackett-Luce model by placing a Gamma prior on the selection probabilities [13].

Recently, factorization approaches have been developed to model the joint pairwise matrix [12, 15]. There the pairwise matrices are aggregated into a single matrix $\mathbf{Y}_n^{tot} = \sum_{k=1}^{K} \mathbf{Y}_{nk}$ which is then approximated by a low rank factorization such as: $\mathbf{Y}_n^{tot} \approx \mathbf{S}_n\mathbf{e}^T - \mathbf{e}\mathbf{S}_n^T$. The resulting scores $\mathbf{S}_n$ are used to rank the documents.

Finally, a supervised score-based rank aggregation approach based on a Markov Chain was recently introduced [21]. In this model the authors use the ground truth preferences to create a pairwise constraint matrix and then learn a scoring function such that the produced aggregate rankings satisfy as many pairwise constraints as possible. The main drawbacks of this approach are that it is computationally very intensive requiring constrained optimization (semidefinite programming), and it does not incorporate the target IR metric into the optimization.

In general, none of the models mentioned above take the target metric into account during the optimization of the aggregating function. To the best of our knowledge no such approach has been developed.

4. SUPERVISED FRAMEWORK

In this section we describe our supervised framework for preference aggregation. To simplify notation, for the remainder of this section, we drop the query index n and work with a general query q with M documents $\mathbf{D} = \{d_1, ..., d_M\}$, relevance labels $\mathbf{L} = \{l_1, ..., l_M\}$ and ranking matrix $\mathbf{R}$ that has been converted to a set of pairwise matrices $\mathbf{Y} = \{\mathbf{Y}_1, ..., \mathbf{Y}_K\}$ using one of the techniques mentioned earlier.

The main idea behind our approach is to summarize the relative preferences for each document across the K experts by a fixed length feature vector. This transforms the preference aggregation problem into a learning-to-rank one, and any of the standard methods can then be applied to optimize the aggregating function for the target IR metric such as NDCG. In the next section we describe an approach to extract the document features.

4.1 Feature Extraction

Given the rank matrix $\mathbf{R}$ and the resulting pairwise matrix $\mathbf{Y}_k$ for expert k (as shown in Figure 1), our aim is to convert $\mathbf{Y}_k$ into a fixed length feature vector for each of the documents in $\mathbf{D}$. Singular Value Decomposition (SVD) based approaches for document summarization such as Latent Semantic Indexing are known to produce good descriptors even for sparse term-document matrices. Another advantage of SVD is that it requires virtually no tuning and can be used to automatically generate the descriptors once the pairwise matrices are computed. Because of these advantages we chose to use SVD to extract the features. For a given $M \times M$ pairwise matrix $\mathbf{Y}_k$ the rank-p SVD factorization has the form:

$$\mathbf{Y}_k \approx \mathbf{U}_k \mathbf{\Sigma}_k \mathbf{V}'_k$$

where $\mathbf{U}_k$ is an $M \times p$ matrix, $\mathbf{\Sigma}_k$ is an $p \times p$ diagonal matrix of singular values and $\mathbf{V}_k$ is an $M \times p$ matrix. The full SVD factorization has $p = M$, however, to reduce the noise and other undesirable artifacts of the original space most applications that use SVD typically set $p \ll M$. Reducing the rank is also an important factor in our approach as pairwise matrices tend to be very sparse with ranks significantly smaller than M.

Given the SVD factorization we use the resulting matrices as features for each document. It is important to note here that *both* $\mathbf{U}$ and $\mathbf{V}$ contain useful document information since $\mathbf{Y}_k$ is a pairwise document by document matrix. To get the features for document d_i and expert k we use:

$$\psi(d_i, \mathbf{Y}_k) = [\mathbf{U}_k(i,:), \text{diag}(\mathbf{\Sigma}_k), \mathbf{V}_k(i,:)] \quad (2)$$

here $\mathbf{U}_k(i,:)$ and $\mathbf{V}_k(i,:)$ are the i'th rows of $\mathbf{U}_k$ and $\mathbf{V}_k$ respectively and $\text{diag}(\mathbf{\Sigma}_k)$ is the main diagonal of $\mathbf{\Sigma}_k$ represented as a $1 \times p$ vector. Note that the $\text{diag}(\mathbf{\Sigma}_k)$ component is independent of i and will be the same for all the documents in $\mathbf{D}$. We include the singular values to preserve as much information from the SVD factorization as possible. The features $\psi(d_i, \mathbf{Y}_k)$ summarize the relative preference information for d_i expressed by the expert k. To get a complete view across the K experts we concatenate together the features extracted for each expert:

$$\psi(d_i) = [\psi(d_i, \mathbf{Y}_1), ..., \psi(d_i, \mathbf{Y}_K)] \quad (3)$$

Each $\psi(d_i, \mathbf{Y}_k)$ contains $3p$ features so the entire representation will have $3Kp$ features. Moreover, note that since both K and p are fixed across queries this representation will have the *same* length for every document in each query. We have thus created a fixed length *feature representation* for every document d_i, effectively transforming the aggregation problem into a standard learning-to-rank one. During training our aim is now to learn a scoring function $f : \mathbb{R}^{3Kp} \rightarrow \mathbb{R}$ which maximizes the target IR metric such as NDCG. At test time, given a new query q' with rank matrix $\mathbf{R}'$, we (1) convert $\mathbf{R}'$ into a set of pairwise matrices $\mathbf{Y}' = \{\mathbf{Y}'_1, ..., \mathbf{Y}'_K\}$; (2) extract the document features using rank-p SVD; and (3) apply the learned scoring function to get the score for every document. The scores are then sorted to get the aggregate ranking. This process is shown in Figure 2. The feature representation allows us to fully utilize the available labeled training data to optimize the aggregating function for the target metric, which is not possible to do with the existing aggregation methods.

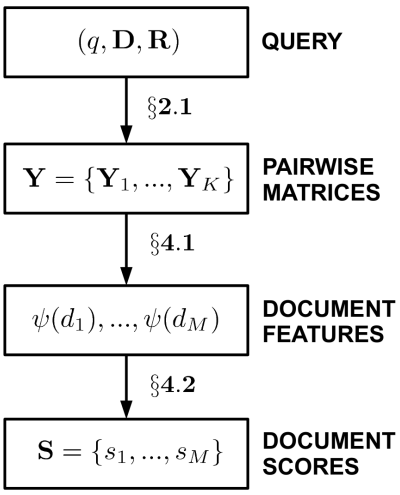


Figure 2: The flow diagram for the feature-based preference aggregation approach. (1) The ranking matrix R is converted to a set of pairwise matrices Y. (2) SVD is used to extract document features from each pairwise matrix $\mathbf{Y}_k$. (3) The learned scoring function is applied to the features to produce the scores for each document. The scores are then sorted to get the aggregate ranking.

It is worth mentioning here that the SVD factorization of pairwise matrices has been used in the context of preference aggregation (see [12] for example). However, previous approaches largely concentrated on applying SVD to fill the missing entries in the joint pairwise matrix $\mathbf{Y}^{tot} = \sum_{k=1}^{K} \mathbf{Y}_k$ and then use the completed matrix to infer the aggregate ranking. Our approach on the other hand uses SVD to compress each pairwise matrix $\mathbf{Y}_k$ and produce fixed length feature vector for each document.

4.2 Learning the Scoring Function

Given the document features extracted via the SVD approach our goal is to use the labeled training queries to optimize the parameters of the scoring function for the target IR metric. The main difference between the introduced feature-based rank aggregation approach and the typical learning-to-rank setup is the possibility of missing features. When a given document d_i is not ranked by the expert k the row

$\mathbf{Y}_k(i,:)$ and column $\mathbf{Y}_k(:,i)$ will both be missing (i.e., 0). To account for this we modify the conventional linear scoring function to include a bias term for each of the K experts:

$$f(\psi(d_i), \mathbf{W}) = \sum_{k=1}^{K} w_k \cdot \psi(d_i, \mathbf{Y}_k) + I[\mathbf{R}(i,k) = 0] b_k \quad (4)$$

where $\mathbf{W} = \{w_k, b_k\}_{k=1}^{K}$ is the set of free parameters to be learned with each w_k having the same dimension as $\psi(d_i, \mathbf{Y}_k)$, and I is an indicator function. The bias term b_k provides a base score for d_i if d_i is not ranked by expert k. The weights w_k control how much emphasis is given to preferences from expert k. It is important to note here that the scoring function can easily be made non-linear by adding additional hidden layer(s) as done in conventional multilayer neural nets. In the form given by Equation 4 our model has a total of $(3p+1)K$ parameters to be learned. We can use any of the developed learning-to-rank approaches to optimize $\mathbf{W}$; in this work we chose to use the LambdaRank method. We chose LambdaRank because it has shown excellent empirical performance recently winning the Yahoo! Learning To Rank Challenge [7]. We briefly describe LambdaRank here and refer the reader to [6] and [5] for a more extensive treatment.

LambdaRank learns pairwise preferences over documents with emphasis derived from the NDCG gain found by swapping the rank position of the documents in any given pair, so it is a listwise algorithm (in the sense that the cost depends on the sorted list of documents). Formally, given a pair of documents (d_i, d_j) with $l_i \neq l_j$, the target probability that d_i should be ranked higher than d_j is defined as:

$$P_{ij} = \begin{cases} 1 & \text{if } l_i > l_j \\ 0 & \text{otherwise} \end{cases}$$

The model's probability is then obtained by passing the difference in scores between d_i and d_j through a logistic:

$$\begin{aligned} Q_{ij} &= \frac{1}{1 + \exp(-(f(\psi(d_i), \mathbf{W}) - f(\psi(d_j), \mathbf{W})))} \\ &= \frac{1}{1 + \exp(-(s_i - s_j))} \end{aligned}$$

The aim of learning is to match the two probabilities for every pair of documents with different labels. To achieve this a cross entropy objective is used:

$$O_q = -\sum_{l_i > l_j} P_{ij} \log(Q_{ij})$$

This objective weights each pair of documents equally thus placing equal importance on getting the relative order of document correctly both at the top and at the bottom of the ranking. However, most target evaluation metrics including NDCG are heavily concentrated on the top of the ranking. To take this into account the LambdaRank framework uses a smooth approximation to the gradient of a target evaluation measure with respect to the score of a document d_i, and we refer to this approximation as λ-gradient. The λ-gradient for NDCG is defined as the derivative of the cross entropy objective weighted by the difference in NDCG obtained when a pair of documents swap rank positions:

$$\lambda_{ij} = |\Delta NDCG@M(s_i, s_j)| \frac{\partial O_q}{\partial s_i - s_j}$$

Algorithm 1 Learning Algorithm

Input: $\{(q_1, \mathbf{D}_1, \mathbf{L}_1, \mathbf{R}_1), ..., (q_N, \mathbf{D}_N, \mathbf{L}_N, \mathbf{R}_N)\}$
Parameters: learning rate η
initialize weights: $\mathbf{W}$
for $n = 1$ **to** N **do** {feature extraction: §4.1}
  from $\mathbf{R}_n$ compute $\mathbf{Y}_n = \{\mathbf{Y}_{n1}, ..., \mathbf{Y}_{nK}\}$
  for $i = 1$ **to** M_n **do**
    compute features $\psi(d_{ni})$
  end for
end for
repeat {scoring function optimization: §4.2}
  for $n = 1$ **to** N **do**
    calculate query cross entropy objective: $O_{q_n} = \sum_{l_{ni} > l_{nj}} \log(1 + e^{s_{nj} - s_{ni}})$
    compute λ-gradients: $\nabla\mathbf{W} = \sum_i \lambda_{ni} \frac{\partial s_{ni}}{\partial \mathbf{W}}$
    update weights: $\mathbf{W} = \mathbf{W} - \eta \nabla \mathbf{W}$
  end for
until convergence
Output: $\mathbf{W}$

Thus, at the beginning of each iteration, the documents are sorted according to their current scores, and the difference in NDCG is computed for each pair of documents by keeping the ranks of all of the other documents constant and swapping only the rank positions for that pair (see [6] for more detail on λ calculation). The λ-gradient for document d_i is computed by summing the λ's for all pairs of documents (d_i, d_j) for query q:

$$\lambda_i = \sum_{j: l_j \neq l_i} \lambda_{ij}$$

The $|\Delta NDCG|$ factor emphasizes those pairs that have the largest impact on NDCG. Note that the truncation in NDCG is relaxed to the entire document set to maximize the use of the available training data.

To make the learning algorithm more efficient the document features can be precomputed a priori and re-used throughout learning. This significantly reduces the computational complexity at the cost of additional storage requirement of $O(M_n K p)$ for each query q_n. The complete stochastic gradient descent learning procedure is summarized in Algorithm 1.

5. CROWDSOURCING EXPERIMENTS

Our approach is primarily designed for supervised preference aggregation problems where sufficient expert preference data is available to learn a feature-based model for each expert. However, to further examine the utility of the proposed method we also conducted experiments on the crowdsourcing problem where very limited preference data is typically available for each expert.

For crowdsourcing experiments we use the dataset from Task 2 of the TREC2011 Crowdsourcing Track[1] [17]. The dataset is a collection of binary (relevant or not relevant) judgements from 762 workers for 19,033 documents. The documents are split into 100 topics and for each topic ground truth or "gold" labels are provided for a subset of the documents. The topics in this task can be thought of as queries in the traditional search problem. For each topic n the judgements for the documents are represented as a matrix $\mathbf{R}_n$

[1]https://sites.google.com/site/treccrowd/home

Table 2: TREC2011 crowdsourcing dataset NDCG results.

	N@1	N@2	N@3	N@4	N@5	N@6	N@7	N@8	N@9	N@10
Condorcet	42.00	50.00	49.28	49.87	49.04	49.53	49.35	50.34	52.16	51.41
Majority vote	59.00	64.00	62.80	63.95	63.96	64.45	64.93	66.34	66.46	64.83
LR-consensus	70.00	72.00	70.80	69.56	69.85	69.47	68.86	68.79	68.83	66.79
LR-labeled	80.00	79.50	77.46	75.63	74.10	73.80	73.29	73.18	72.78	71.13
LR-unlabeled	**83.00**	**82.00**	**80.80**	**78.76**	**77.94**	**76.77**	**75.93**	**75.34**	**75.36**	**73.37**

where each entry $\mathbf{R}_n(i,k)$ is either 1 (relevant), -1 (not relevant) or 0 (not observed). The ground truth (gold) labels $\mathbf{L}_n$ are available only for a subset of the documents under each topic and are given by one of two values: 1 = relevant and 0 = not relevant. There are a total of 2275 gold labels: 1275 relevant and 1000 not relevant. The aim is to use the worker judgements to accurately rank the documents under each topic (note that this is a different problem from the Task 2 in TREC2011 where the goal was to predict each document's relevance label). To evaluate the models we created 5 random splits of topics into 60/20/20 sets for training/validation/testing. The results shown for each model are the average test set results for the five folds. For each topic the models are evaluated using only those documents for which the gold labels are available.

Across all topics the data contains 89,624 judgements and each document is judged by 5 workers on average. Moreover, out of the 762 workers 547 have less than 40 judgements so the dataset is very imbalanced with the majority of judgements coming from a small subset of the workers. Given the label sparsity there is not enough preference data to learn an accurate feature-based model for each worker. Instead of extracting preference features for every worker, we used a hybrid approach. Separate features were extracted for workers that had 500 or more preferences in the training set, there were around 20 such workers in each fold. Preferences for all the other workers were combined into a single preference matrix for which we extracted an additional set of features. Using $\mathbf{\Gamma}_{500}$ to denote the set of workers that have 500 or more preferences in the training set we can write the complete feature vector as:

$$\psi(d_{ni}) = \{\psi(d_{ni}, \mathbf{Y}_k) \;\mid\; k \in \mathbf{\Gamma}_{500}\} \cup \psi(d_{ni}, \mathbf{Y}_n^{<500}) \quad (5)$$

where $\mathbf{Y}_n^{<500}$ is given by $\mathbf{Y}_n^{<500} = \sum_{k \notin \mathbf{\Gamma}_{500}} \mathbf{Y}_{nk}$. The hybrid approach allows us to learn accurate feature-based models for those workers that have enough preferences in the training data and combine these models with a consensus-based model for workers with few training ratings. Each pairwise matrix was computed using the binary comparison approach (see Section 2.1) since it is the only suitable approach for binary judgements.

5.1 Results

We conducted experiments with three versions of the LambdaRank model. The first model, which we refer to as LR-consensus, did not use the per-worker features and only used features from the SVD factorization of the combined matrix $\mathbf{Y}_n^{tot} = \sum_{k=1}^{K} \mathbf{Y}_{nk}$. This model thus does not take the individual worker preferences into account and ranks only based on consensus features. The second model, LR-labeled, uses the individual worker features as per equation Equation 5 but the features were computed using preferences only for the documents that had ground truth labels. The last model, LR-unlabeled, included the preferences for the unlabeled documents during feature computation. Through cross validation we found that setting $p = 1$ (SVD rank) gave the best performance which is expected considering the sparsity level of the pairwise matrices $\mathbf{Y}$. For all settings, LambdaRank was run for 200 iterations with a learning rate of 0.01, and validation NDCG@10 was used to choose the best model.

We compare our feature based model with the condorcet and the majority vote approach which was shown to work well on the related TREC2011 Task 2 [3]. Table 2 shows test NDCG (N@T) at truncations $1-10$. From the table it is seen that all LambdaRank models significantly outperform the majority vote baseline on the ranking task. It is also seen that the consensus based model LR-consensus performs significantly worse than the models that take the individual worker preferences into account. This indicates that ranking based on consensus alone leads to suboptimal performance, as we further demonstrate in the meta-search experiments in the next section. Finally, we see that using features computed from preferences for only the labeled documents (LR-labeled) does not perform as well as using all the preferences (LR-unlabeled). The LR-unlabeled model performs significantly better than the LR-labeled at all truncations. This indicates that preferences from the unlabeled documents do contain useful information and should be included during the feature extraction. This can be especially relevant when the number of labeled documents and/or preferences is small which is the case here.

6. META-SEARCH EXPERIMENTS

For our meta-search experiments we use the LETOR4.0 benchmark datasets. These data sets were chosen because they are publicly available, include several baseline results, and provide evaluation tools to ensure accurate comparison between methods. In LETOR4.0 there are two meta-search data sets, MQ2007-agg and MQ2008-agg. MQ2007-agg contains 1692 queries with 69,623 documents and MQ2008-agg contains 784 queries and a total of 15,211 documents. Each query q_n contains a ranking matrix $\mathbf{R}_n$ with partial expert rankings of the documents under that query. There are 21 experts in MQ2007-agg and 25 experts in MQ2008-agg. In this setting the experts correspond to the search engines to which the query was submitted. In addition, in both data sets, each document is assigned one of three relevance levels: 2 = highly relevant, 1 = relevant and 0 = irrelevant. Finally, each dataset comes with five precomputed folds with 60/20/20 splits for training/validation/testing. The results shown for each model are the averages of the test set results for the five folds. The MQ2007-agg dataset is approximately 35% sparse, meaning that for an average query the ranking matrix is missing 35% of its entries. MQ2008-agg is significantly sparser, with $\sim 65\%$ entries missing.

Table 3: MQ2008-agg and MQ2007-agg results; all differences between LambdaRank and the best baseline are statistically significant.

	NDCG					Precision					
	N@1	N@2	N@3	N@4	N@5	P@1	P@2	P@3	P@4	P@5	MAP
MQ2008-agg											
BordaCount	23.68	28.06	30.80	34.32	37.13	29.72	30.42	29.38	29.75	29.03	39.45
CPS-best	26.52	31.38	34.59	37.63	40.04	31.63	32.27	32.27	31.66	30.64	41.02
SVP	32.49	36.20	38.62	40.17	41.85	38.52	36.42	34.65	32.01	30.23	43.61
Bradley-Terry	38.05	39.24	40.77	41.79	42.62	44.77	39.73	36.26	33.19	30.28	44.35
Plackett-Luce	35.20	38.49	39.70	40.49	41.55	41.32	38.96	35.33	32.02	29.62	42.20
Condorcet	35.67	37.39	39.11	40.50	41.59	40.94	37.43	34.73	32.08	29.59	42.63
RRF	38.77	40.73	43.48	45.70	47.17	44.89	41.32	38.82	36.51	34.13	47.71
LR-I	40.22	43.94	46.33	48.30	50.16	46.94	44.00	41.45	38.52	36.28	49.54
LR-R	40.90	42.75	45.83	47.93	49.68	47.45	42.92	41.20	38.77	36.22	49.26
LR-logR	**42.81**	**44.53**	**47.02**	**49.00**	**50.69**	**48.85**	**44.13**	**41.84**	**39.09**	**36.50**	**50.32**
MQ2007-agg											
BordaCount	19.02	20.14	20.81	21.28	21.88	24.88	25.24	25.69	25.80	25.97	32.52
CPS-best	31.96	33.18	33.86	34.09	34.76	38.65	38.65	38.14	37.19	37.02	40.69
SVP	35.82	35.91	36.53	37.16	37.50	41.61	40.28	39.50	38.88	38.10	42.73
Bradley-Terry	39.88	39.86	40.40	40.60	40.91	46.34	44.65	43.48	41.98	40.95	43.98
Plackett-Luce	40.63	40.39	40.26	40.71	40.96	46.93	45.10	43.09	42.32	41.09	43.64
Condorcet	37.31	37.63	38.03	38.37	38.66	43.26	42.14	40.94	39.85	38.75	42.56
RRF	41.93	42.66	42.42	42.73	43.13	48.70	47.20	44.84	43.52	42.52	46.72
LR-I	46.13	**46.76**	**46.71**	**46.87**	**47.28**	52.90	**51.39**	**49.33**	**47.80**	**46.66**	50.05
LR-R	45.02	45.40	45.68	45.87	46.12	51.83	50.36	48.88	47.25	45.98	49.55
LR-logR	**46.30**	46.18	46.46	46.56	47.02	**53.19**	51.04	49.26	47.48	46.29	**50.39**

The goal is to use the rank lists to infer an aggregate ranking of the documents for each query which maximally agrees with the held-out relevance levels. To evaluate this agreement, in addition to NDCG (N@T), we also compute Precision (P@T) and Mean Average Precision (MAP) [2]. MAP only allows binary (relevant/not relevant) document assignments, and is defined in terms of average precision (AP). For an aggregate ranking π, and relevance levels $\mathbf{L}$, AP is given by:

$$AP(\pi, \mathbf{L}) = \frac{\sum_{t=1}^{M} P@t * \mathbf{L}(\pi^{-1}(t))}{\sum_{t=1}^{M} \mathbf{L}(\pi^{-1}(t))} \quad (6)$$

where $L(\pi^{-1}(t))$ is a relevance label for document in position t in π; and $P@t$ is the precision at t:

$$P@t = \frac{\sum_{i=1}^{t} \mathbf{L}(\pi^{-1}(i))}{t} \quad (7)$$

MAP is then computed by averaging AP over all queries. To compute P@t and MAP on the MQ-agg datasets the relevance levels are binarized with 1 converted to 0 and 2 converted to 1 (as per LETOR4.0 evaluation guidelines). All presented NDCG, Precision and MAP results are averaged across the test queries and were obtained using the evaluation script available on the LETOR4.0 website[2].

6.1 Results

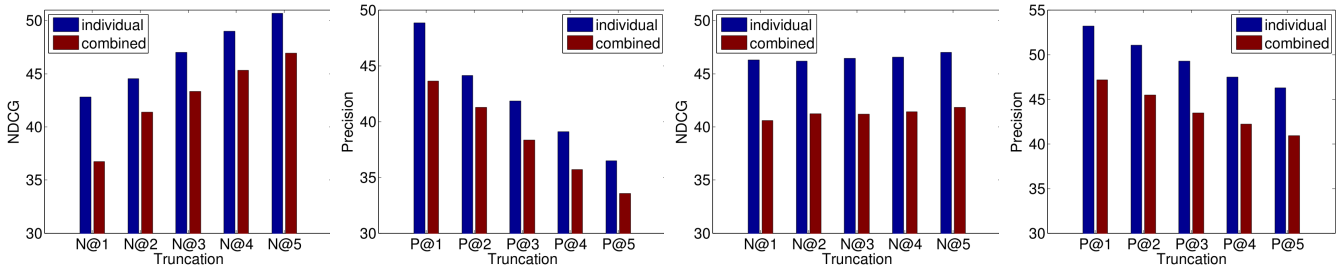
To investigate the properties of our approach we conducted extensive experiments with various versions of the model. We experimented with binary, rank difference, and log rank difference methods to compute the pairwise matrices (see Section 2.1) and refer to these models as LR-I, LR-R and LR-logR respectively. Through cross validation we found that setting $p = 1$ (SVD rank) gave the best performance. For all settings, LambdaRank was run for 200 iterations with a learning rate of 0.01, and validation NDCG@10 was used to choose the best model.

We compare the results of our model against the best methods currently listed on the LETOR4.0 website, namely the BordaCount model and the best of the three CPS models (combination of Mallows and Plackett-Luce models) on each of the MQ-agg datasets. We also compare with the established meta-search standards Condorcet and Reciprocal Rank Fusion (RRF) as well as the Bradley-Terry and Plackett-Luce models and the SVD-based method SVP. The above models cover all of the primary leading approaches in the rank aggregation research except for the Markov Chain model [21]. We were unable to compare with this method because it is neither publicly available nor listed as one of the baselines on LETOR, making standardized comparison impossible.

The NDCG and Precision results for truncations 1-5 as well as MAP results are shown in Table 3. From the table we see that all versions of the LambdaRank model significantly outperform the other aggregation methods, improving by as much as 5 NDCG points over the best baseline on each of the MQ-agg datasets. The results on the MQ2008-agg which is more than 65% sparse demonstrate that features extracted by the SVD approach are robust and generalize well even in the sparse setting. From the table we also see that the binary and log rank transformations lead to the best performance, with LR-logR significantly outperforming the other LambdaRank models on MQ2008-agg and having comparable performance to LR-I on MQ2007-agg.

To investigate the utility of representing each expert's preferences with a separate set of features we ran experiments with a combined approach. For each query q_n we combined all pairwise preference matrices into a single matrix $\mathbf{Y}_n^{tot} = \sum_{k=1}^{K} \mathbf{Y}_{nk}$ and trained the LR-logR model on the SVD features from $\mathbf{Y}_n^{tot}$ only. Note that this model has

[2]research.microsoft.com/en-us/um/beijing/projects/letor/



(a) NDCG, MQ2008-agg (b) Precision, MQ2008-agg (c) NDCG, MQ2007-agg (d) Precision, MQ2007-agg

Figure 3: Test NDCG@1-5 and Precision@1-5 results for LR-logR trained on SVD features extracted from each $\mathbf{Y}_{nk}$ (individual), versus LR-logR trained on SVD features extracted only from the joint preference matrix $\mathbf{Y}_n^{tot}$ (combined).

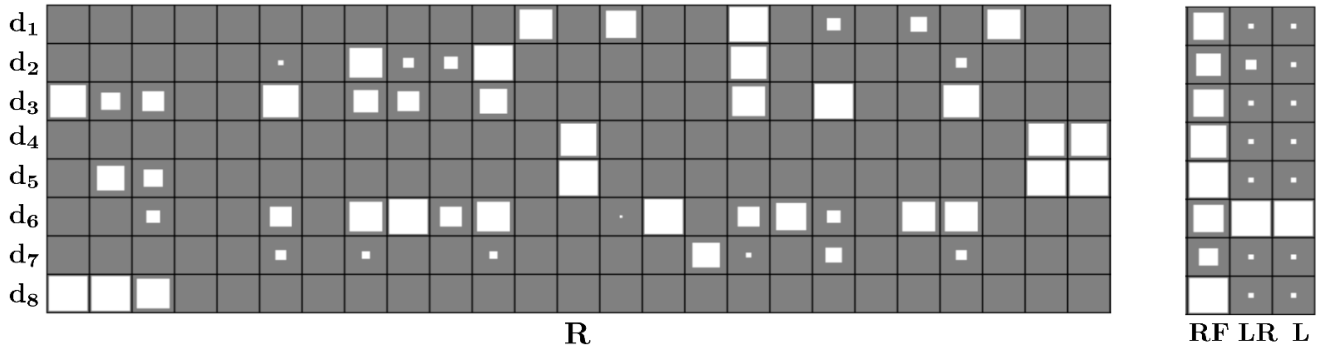


Figure 4: The matrix on the left shows the expert ranking matrix R for a test query in Fold 1 of the MQ2008-agg dataset. 8 documents $\{d_1, ..., d_8\}$ were retrieved for this query and the normalized expert rankings are represented by white squares. The size of each square reflects the preference strength, so that large squares correspond to high rankings (strong preference). Missing rankings are represented by empty cells. The matrix on the right shows the normalized scores for each document produced by the Reciprocal Rank Fusion (RF) and the LR-logR (LR) models, as well as the ground truth relevance levels (L). Note that only d_6 is relevant to this query.

no information about the individual expert preferences. We refer to this model as "combined" and compare it to the full model which uses SVD features from each $\mathbf{Y}_{nk}$ referred to as "individual". The results for the two datasets are shown in Figure 3. From the figure we see that the individual model significantly outperforms the combined one on both datasets. The performance of the combined model is comparable to the best baseline consensus method, the Reciprocal Rank Fusion. This is not surprising since $\mathbf{Y}_n^{tot}$ summarizes the consensus preference across the agents, and without access to the individual preferences the ranker can only learn to follow the consensus. Note however, that the gain from using the individual expert features is very significant which further supports the conclusion that specialization is very important for the supervised preference aggregation.

Figure 4 further demonstrates the importance of specialization. The figure shows the expert ranking matrix together with the scores produced by the Reciprocal Rank Fusion and LR-logR for an example test query from MQ2008-agg. From the the ground truth relevance levels (**L**) it is seen that only document d_6 is relevant to this query. However, from the ranking matrix we see that the experts express strong net preference for documents d_1, d_4, d_5 and d_8, whereas the preferences for d_6 are mixed with many positive and negative preferences. The Reciprocal Rank Fusion is a consensus-based approach and as such ranks documents d_1, d_4, d_5 and d_8 above d_6 with NDCG@1-4 of 0. Other consensus-based methods produce similar rankings. LR-logR on the other hand, is able to ignore the consensus and concentrate on a small subset of the preferences, placing d_6 on top and producing a perfect ranking. Moreover, note that the scores for the other documents produced by the LR-logR are significantly lower than the score for d_6, so the model is confident in this ranking. The query shown in Figure 4 is an example of a difficult query (often these correspond to long-tail queries) where the majority of experts generate incorrect preferences. For these queries the aggregated rankings produced by the consensus-based methods will also be incorrect. Our feature-based supervised approach is able to fix this problem through specialization. By examining the queries in both MQ2008-agg and MQ2007-agg we found that both datasets contain a number of such queries and that our approach performs significantly better on those queries than all of the consensus-based baselines.

7. CONCLUSION

We presented a fully supervised rank aggregation framework. Our approach transforms the rank aggregation problem into a supervised learning-to-rank one. The transformation is based on feature summarization of pairwise preference matrices. To compute the features we employ a low-rank SVD factorization. The extracted features allow us to use any supervised learning-to-rank approach to learn the aggregating function. This in turn allows us to optimize the aggregating function for any target IR metric. Experimental results with LambdaRank as the learning-to-rank approach show that our method outperforms the existing methods on supervised aggregation tasks. The improvements are especially significant on difficult queries where the majority of preferences are wrong. Future work includes investigating other ways of producing effective fixed length representations from full/partial preferences.

8. REFERENCES

[1] J. A. Aslam and M. Montague. Models for metasearch. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 276–284, 2001.

[2] R. Baeza-Yates and B. Ribeiro-Neto. *Information Retrieval*. Addison-Wesley, 1999.

[3] P. N. Bennett, E. Kamar, and G. Kazai. MSRC at TREC 2011 Crowdsourcing Track. In *To Appear in Proceedings of the Text Retrieval Conference*, 2011.

[4] R. Bradley and M. Terry. Rank analysis of incomplete block designs. I. The method of paired comparisons. *Biometrika*, 39:324–345, 1952.

[5] C. J. C. Burges. From RankNet to LambdaRank to LambdaMART: An overview. Technical Report MSR-TR-2010-82, Microsoft Research, 2010.

[6] C. J. C. Burges, R. Rango, and Q. V. Le. Learning to rank with nonsmooth cost functions. In *NIPS*, 2007.

[7] O. Chapelle, Y. Chang, and T.-Y. Liu. The Yahoo! learning to rank challenge, 2010.

[8] G. V. Cormack, C. L. A. Clarke, and S. Büttcher. Reciprocal rank fusion outperforms condorcet and individual rank learning methods. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2009.

[9] P. Dangauthier, R. Herbrich, T. Minka, and T. Graepel. TrueSkill through time: Revisiting the history of chess. In *Proceedings of the Neural Information Processing Systems*, 2007.

[10] H. A. David. *The method of paired comparissons*. Hodder Arnold, 1988.

[11] A. E. Elo. *The rating of chess players: Past and present*. Acro Publishing, 1978.

[12] D. F. Gleich and L.-H. Lim. Rank aggregation via nuclear norm minimization. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 60–68, 2011.

[13] J. Guiver and E. Snelson. Bayesian inference for Plackett-Luce ranking models. In *Proceedings of the International Conference on Machine Learning*, pages 377–384, 2009.

[14] K. Jarvelin and J. Kekalainen. IR evaluation methods for retrieving highly relevant documents. In *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 41–48, 2000.

[15] X. Jiang, L.-H. Lim, Y. Yao, and Y. Ye. Statistical ranking and combinatorial hodge theory. *Mathematical Programming*, 127:203–244, 2011.

[16] T. Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 133–142, 2002.

[17] G. Kazai and M. Lease. TREC2011 Crowdsourcing Track, 2011.

[18] A. Klementiev, D. Roth, and K. Small. Unsupervised rank aggregation with distance-based models. In *Proceedings of the International Conference on Machine Learning*, pages 472–479, 2008.

[19] G. Lebanon and J. Lafferty. Cranking: Combining rankings using conditional probability models on permutations. In *Proceedings of the International Conference on Machine Learning*, pages 363–370, 2002.

[20] T. Liu, J. Xu, W. Xiong, and H. Li. LETOR: Benchmark dataset for search on learning to rank for information retrieval. In *ACM SIGIR Workshop on Learning to Rank for Information Retrieval*, 2007.

[21] Y.-T. Liu, T.-Y. Liu, T. Qin, Z.-M. Ma, and H. Li. Supervised rank aggregation. In *Proceedings of the International Conference on World Wide Web*, pages 481–489, 2007.

[22] T. Lu and C. Boutilier. Learning Mallows models with pairwise preferences. In *Proceedings of the International Conference on Machine Learning*, 2011.

[23] R. D. Luce. *Individual choice behavior: A theoretical analysis*. Wiley, 1959.

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

[25] M. Meila, K. Phadnis, A. Patterson, and J. Bilmes. Consensus ranking under the exponential model. In *Proceedings of the Uncertainty in Artificial Intelligence Conference*, 2007.

[26] M. Montague and J. A. Aslam. Condorcet fusion for improved retrieval. In *Proceedings of the International ACM CIKM Conference on Information and Knowledge Management*, 2002.

[27] R. Plackett. The analysis of permutations. *Applied Statistics*, 24:193–302, 1975.

[28] T. Quin, X. Geng, and T.-Y. Liu. A new probabilistic model for rank aggregation. In *Proceedings of the Neural Information Processing Systems*, pages 681–689, 2010.

[29] L. L. Thurstone. The method of paired comparisons for social values. *Journal of Abnormal and Social Psychology*, 21:384–400, 1927.