CRF Framework for Supervised Preference Aggregation

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

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

ABSTRACT

We develop a flexible Conditional Random Field framework for supervised preference aggregation, which combines preferences from multiple experts over items to form a distribution over rankings. The distribution is based on an energy comprised of unary and pairwise potentials allowing us to effectively capture correlations between both items and experts. We describe procedures for learning in this model and demonstrate that inference can be done much more efficiently than in analogous models. Experiments on benchmark tasks demonstrate significant performance gains over existing rank 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 task of combining preferences from multiple experts over items into a single consensus ranking. This problem is crucially important in many applications. 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 audio and text. The ratings from several users are then aggregated to produce a single labeling of the data.

CIKM'13, Oct. 27–Nov. 1, 2013, San Francisco, CA, USA.

http://dx.doi.org/10.1145/2505515.2505713.

Existing approaches to preference aggregation can be divided into two categories: permutation-based and score-based. Permutation-based models work directly in the permutation space and the majority of these methods are based on the Mallows model [23, 31, 21]. Score-based approaches infer a set of real valued scores that are then used to rank the items. A number of heuristic score-based methods for preference aggregation have been proposed. For example, BordaCount [1], Condorcet Fusion [27] and Reciprocal Rank Fusion [8] derive the item scores by averaging (weighted) ranks across the experts, or counting the number of pairwise wins. Several probabilistic models have also been proposed, the majority of which are based on the Bradley-Terry and/or Plackett-Luce models [3, 28, 13].

The vast majority of the proposed methods in both categories are developed for *unsupervised* preference aggregation, where the aim is to produce an aggregate ranking that satisfies the majority of the preferences. However, many of the recent aggregation problems are amenable to supervised learning, as ground truth preference 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, which provides ground truth ranking information. 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 always satisfy the majority often produce suboptimal results. Consequently, we need the aggregating function to be able to "specialize" and infer when to use the majority preference versus when to concentrate only on a small subset of preferences; this specialization property is impossible to achieve without referring to the ground truth labels.

The supervised problem has received considerable attention in recent years and a number of supervised approaches have been proposed [20, 36, 37, 29]. Notably, [37] has recently shown that by applying SVD factorization to pairwise preference matrices effective item features can be extracted. The features transform the problem into a standard learning-to-rank one allowing to apply any of the existing learning-to-rank methods to optimize the aggregating function for the target metric. While the authors of that work have shown superior empirical accuracy of this approach to many existing aggregation methods, it also has a major drawback in that it requires computing SVD factors at test time. For large problems with many items per instance, such as those in crowdsourcing applications, applying SVD

at test time can be prohibitively expensive, limiting the application of this method. A number of other popular supervised aggregation methods share the same disadvantage and also require applying complex optimization procedures such as semidefinite programming [20] at test time.

In this paper we address the complexity problem by developing a flexible Conditional Random Field (CRF) framework for supervised preference aggregation. Our framework uses preference matrices directly thus avoiding costly optimization and only requires computing simple sums during the inference step. We then show how ideas from learning-to-rank and related literature and be used to effectively optimize our model for most existing metrics. Experiments on rank aggregation tasks with Microsoft's LETOR4.0 [18] data sets show that our model achieves performance comparable to state-of-the art aggregation methods while requiring only a small fraction of computational time.

2. PREFERENCE AGGREGATION PROBLEM

A typical supervised preference aggregation problem consists of a set of N training instances $\mathcal{D} = \{\mathbf{r}_n, \mathbf{R}_n\}_{n=1}^N$. Here $\mathbf{R}_n$ is a set of (partial) preferences expressed by the K_n "experts" for the M_n items, and $\mathbf{r}_n$ is a set of (partial) ground truth preferences for the items. Note that the number of experts and items varies across the instances, and there is no information available about the items beyond $\mathbf{R}_n$ and $\mathbf{r}_n$. Across different domains the preferences in $\mathbf{R}_n$ can be in the form of full or partial rankings, top-T lists, ratings, relative item comparisons, or combinations of these. Moreover, the form of ground truth preferences can also vary across domains. For instance, in social choice, ground truth preferences often come in the form of pairwise comparisons and/or (partial) rankings. On the other hand in meta-search and collaborative filtering ground truth preferences are typically expressed via ratings and/or relevance labels. In this work we concentrate on the rank aggregation instance of this problem from the information retrieval (IR) 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 rank aggregation the experts' preferences are summarized in an $M_n \times K_n$ matrix $\mathbf{R}_n$ where $\mathbf{R}_n(i,k)$ denotes the rank assigned to item i by the expert k. Furthermore, $\mathbf{R}_n$ can be sparse, as experts might not assign ranks to every item; we use $\mathbf{R}_n(i,k) = 0$ to indicate that item i was not ranked by expert k.

Irrespective of the preference type the goal in supervised preference aggregation is to use $\mathbf{R}_n$ to predict a ranking $\hat{\mathbf{y}}_n$ of the items with highest "agreement" with the ground truth preferences $\mathbf{r}_n$. We use $\mathbf{y}_n(i) = j$ to denote that rank of item i in $\mathbf{y}_n$ and $i = \mathbf{y}_n^{-1}(j)$ to denote the reverse. Depending on the preference type, agreement between $\hat{\mathbf{y}}_n$ and $\mathbf{r}_n$ can be measured using different metrics.

In social choice the common evaluation metric is Kendall's tau, which measures the number of pairwise disagreements between $\hat{\mathbf{y}}_n$ and $\mathbf{r}_n$. Popular evaluation metrics in IR include Normalized Discounted Cumulative Gain (NDCG)[14] and Expected Reciprocal Rank (ERR)[6]. Both NDCG and ERR take relevance labels as input: the value of $\mathbf{r}_{ni}$ is proportional to the relevance of item i. NDCG is the most commonly used metric in IR and is given by:

$$NDCG(\hat{\mathbf{y}}_n, \mathbf{r}_n)@T = \frac{1}{G(\mathbf{r}_n, T)} \sum_{i=1}^{T} \frac{2^{\mathbf{r}_n(\hat{\mathbf{y}}_n^{-1}(i))} - 1}{\log(1+i)} \quad (1)$$

where $\mathbf{r}_n(\hat{\mathbf{y}}_n^{-1}(i))$ is ground truth preference for item in position i in $\hat{\mathbf{y}}_n$, and $G(\mathbf{r}_n, T)$ is a normalizing constant such that the maximum of $NDCG(\hat{\mathbf{y}}_n, \mathbf{r}_n)@T$ is 1. As is common in CRFs, the learning problem optimizes average training loss $\frac{1}{N}\sum_{n=1}^{N} l(\hat{\mathbf{y}}_n, \mathbf{r}_n)$, where $l(\hat{\mathbf{y}}_n, \mathbf{r}_n)$ is the loss incurred for predicting ranking $\hat{\mathbf{y}}_n$ under $\mathbf{r}_n$. For NDCG we simply define this loss as:

$$\begin{aligned} l(\hat{\mathbf{y}}_n, \mathbf{r}_n) &= \max_{\mathbf{y}} NDCG(\mathbf{y}, \mathbf{r}_n)@T - NDCG(\hat{\mathbf{y}}_n, \mathbf{r}_n)@T \\ &= 1 - NDCG(\hat{\mathbf{y}}_n, \mathbf{r}_n)@T \end{aligned} \quad (2)$$

Other metrics can be converted into a loss in a similar fashion.

The variable number of preferences per expert and the variable number of experts across the instances make it difficult to apply the majority of supervised methods to this problem, since they require fixed-length item representations. CRFs (formally defined in Section 4) on the other hand are well suited for tasks with variable input lengths, and have successfully been applied to problems that have this property, such as natural language processing [35, 33, 32], computational biology [34, 19], and information retrieval [30, 38]. Moreover, CRFs are very flexible and can be used to optimize the parameters of the model for the target loss. For these reasons we develop a CRF framework for preference aggregation.

3. RELEVANT WORK

Before delving into our model, we give a brief overview of existing methods for preference aggregation. Most of the existing approaches in this area can be divided into two categories: permutation-based and score-based. In the following sections we describe both types of models.

3.1 Permutation-Based Methods

Permutation-based models work directly in the permutation space. The most common and well explored such model is the Mallows model [23]. Mallows defines a distribution over permutations and is typically parametrized by a central permutation $\hat{\mathbf{y}}_n$ and a dispersion parameter $\phi \in (0, 1]$; the probability of a permutation $\mathbf{y}$ is given by:

$$P(\mathbf{y}|\phi, \hat{\mathbf{y}}_n) = \frac{1}{Z(\phi, \hat{\mathbf{y}}_n)} \phi^{-d(\mathbf{y}, \hat{\mathbf{y}}_n)} \quad (3)$$

where $d(\mathbf{y}, \hat{\mathbf{y}}_n)$ is a distance between $\mathbf{y}$ and $\hat{\mathbf{y}}_n$. For rank aggregation problems inference in this model amounts to finding the permutation $\hat{\mathbf{y}}_n$ that maximizes the likelihood of the observed rankings. For some distance metrics, such as Kendall's τ and Spearman's rank correlation, the partition function $Z(\phi, \hat{\mathbf{y}}_n)$ can be found exactly. However, finding the central permutation $\hat{\mathbf{y}}_n$ that maximizes the likelihood is typically very difficult and in many cases is intractable [26].

Recent work extends the Mallows model to define distributions over partial rankings [21]. Under partial rankings the partition function can no longer be computed exactly, so these authors introduced a new sampling approach to estimate it. When the number of items is large, however, this sampling approach is typically very slow, which makes

the model impractical for many large scale online problems such as meta-search where aggregation has to be done very quickly. Furthermore, both the proposed pairwise model and the sampling approach rely on the assumption that all pairwise preferences are consistent, which is often violated in real-world preference aggregation problems.

A number of other generalizations of the Mallows model have been proposed [17, 16, 31]; however, to the best of our knowledge none of these extensions address learning and/or inference complexity of this model. In general, due to the extremely large search space (typically $M!$ for M items) and the discontinuity of functions over permutations, exact inference in permutation-based models is often intractable. Thus one must resort to approximate inference methods, such as sampling or greedy approaches, often without guarantees on how close the approximate solution will be to the target optimal one. As the number of items grows, the cost of finding a good approximation increases significantly, which makes the majority of these models impractical for many real world applications where data collections are extremely large. The score-based approach described next avoids this problem by working with real valued scores instead.

3.2 Score-Based Methods

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

A number of popular score-based aggregation methods in meta-search are heuristic based. For example, BordaCount [1], Condorcet [27] and median rank aggregation [10] derive the item scores by averaging ranks across the experts or counting the number of pairwise wins. In statistics a very popular pairwise score model is the Bradley-Terry [3] model:

$$P(\mathbf{r}_n|\mathbf{s}_n) = \prod_{\mathbf{r}_{ni}>\mathbf{r}_{nj}} \left(\frac{\exp(s_{ni})}{\exp(s_{ni})+\exp(s_{nj})}\right)^{\mathbf{r}_{ni}-\mathbf{r}_{nj}} \quad (4)$$

where $\frac{\exp(s_{ni})}{\exp(s_{ni})+\exp(s_{nj})}$ can be interpreted as the probability that item i beats item j in the pairwise contest. The key assumption behind the Bradley-Terry model is that the pairwise probabilities are completely independent of the items not included in the pair. A problem that arises from this assumption is that if a given item i has won all pairwise contests, the likelihood becomes larger as s_{ni} becomes larger. It follows that a maximum likelihood estimate for s_{ni} is ∞ [24]. As a consequence the model will always produce a tie amongst all undefeated items. Often this is an unsatisfactory solution because the contests that the undefeated items participated in, and their opponents' strengths, could be significantly different.

To avoid some of these drawbacks, the Bradley-Terry model was generalized by Plackett and Luce [28, 22] to a Plackett-Luce model for permutations:

$$P(\mathbf{y}|\mathbf{s}_n) = \prod_{i=1}^{M_n} \frac{\exp(\mathbf{s}_n(\mathbf{y}^{-1}(i)))}{\sum_{j=i}^{M_n}\exp(\mathbf{s}_n(\mathbf{y}^{-1}(j)))} \quad (5)$$

where $\mathbf{s}_n(\mathbf{y}^{-1}(i))$ is the score of the item in position i in $\mathbf{y}$. The generative process behind the Plackett-Luce model assumes that items are selected sequentially without replacement. Initially item $\mathbf{y}^{-1}(1)$ is selected from the set of M_n items and placed first, then item $\mathbf{y}_n^{-1}(2)$ is selected from the remaining $M_n - 1$ items and placed second and so on until all M_n items are placed. Note that here inference can be done quickly by doing simple gradient descent on scores, which is a clear advantage over most permutation based models. The Plackett-Luce generalization relaxes the pairwise independence assumption of the Bradley-Terry model but this model is only applicable to consistent full or partial rankings (or consistent pairwise preferences) which significantly limits its application. Moreover, for 2-item rankings the Plackett-Luce model reduces to the Bradley-Terry model and thus suffers from the same infinite score problem. To overcome this problem a Bayesian framework was also recently introduced for the Plackett-Luce model by placing a Gamma prior on the selection probabilities [13]. The authors of that work demonstrated that the Bayesian approach prevented overfitting and produced aggregate rankings that better fitted the observed preference data. This improvement however, comes at the cost of computational overhead required during score inference.

Both Bradley-Terry and Plackett-Luce models are unsupervised and are typically fitted via maximum likelihood. This makes them ill-suited for supervised aggregation problems as they are unable to capture the correlations between observed preferences and the ground-truth ones. To overcome this disadvantage a number of supervised methods have recently been proposed. Several of these methods have explored weighted aggregation rules [36, 29], where a well explored social choice aggregation rule, such as Borda or Kemeny, are applied to weighted expert preferences. The weights are tuned on the training data to reflect each experts "agreement" with the ground truth preferences. While these methods have empirically been shown to give improvements, the weights typically have to be tuned by hand making the models inflexible and expensive to optimize.

Other supervised methods explore pairwise item-item preference matrices. A supervised Markov Chain model based on this framework was recently introduced [20]. In this model ground truth preferences are used to create pairwise constraint matrices and a scoring function is then trained to satisfy as many of these pairwise constraints as possible. This method was recently extended by [7] to a semi-supervised setting where ground truth preferences are available only for a subset of the documents. Another method based on pairwise matrices is the SVD approach [37]. In this model pairwise expert matrices are factorized using low-rank SVD factorization and the resulting SVD representations are then used as item features to train the aggregating function. This approach allows to optimize the model for any target metric, but similarly to the Markov-Chain method which requires solving semi definite programming problem, suffers from expensive inference requiring SVD factorization for every test instance.

Our proposed framework is also based on the pairwise matrix approach which as demonstrated by the strong empirical results of the above methods, is a promising way to approach this problem. Unlike the existing methods however, we focus on making the inference as efficient as possible without affecting the accuracy. We describe our approach in the next section.

4. CRF FRAMEWORK FOR PREFERENCE AGGREGATION

In a typical supervised problem we are interested in learning a relationship between input $\mathbf{x}$ and a target $\mathbf{y}$ for a given training set of instantiated pairs $\mathcal{D} = \{\mathbf{x}_n, \mathbf{y}_n\}$. More specifically, we are interested in learning a predictive mapping for $\mathbf{x}$ to $\mathbf{y}$.

CRFs tackle this problem by defining the conditional distribution $p(\mathbf{y}|\mathbf{x})$ through some energy function $E(\mathbf{y}, \mathbf{x}; \boldsymbol{\theta})$ as follows:

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \exp(-E(\mathbf{y}, \mathbf{x}; \boldsymbol{\theta}))$$

$$Z(\mathbf{x}) = \sum_{\mathbf{y}} \exp(-E(\mathbf{y}, \mathbf{x}; \boldsymbol{\theta}))$$

where $\boldsymbol{\theta}$ is the model's parameter vector. The parametric form of the energy function $E(\mathbf{y}, \mathbf{x}; \boldsymbol{\theta})$ depends on the nature of the problem and typically consists of weighted unary and/or higher order potentials. As mentioned above this framework is very flexible and has successfully been applied to a wide range of diverse problems.

In this work we show that CRFs can also be used to build an effective model for preference aggregation. Supervised aggregation can be put into above framework by noting that our goal is to also learn a predictive mapping from an expert matrix $\mathbf{R} \equiv \mathbf{x}$ to a ranking $\mathbf{y}$ that has the highest agreement with the ground truth preferences $\mathbf{r}$. Our goal is thus to define a conditional distribution $p(\mathbf{y}|\mathbf{R})$ through an energy $E(\mathbf{y}, \mathbf{R}; \boldsymbol{\theta})$ and optimize it for the target metric. In the following sections we show that effective unary and pairwise potentials can be derived directly from the expert preference matrix, and use these potentials to define a smooth energy function over the space of rankings.

4.1 Pairwise Preferences

Given the expert matrix $\mathbf{R}_n$ our aim is to convert it to a set of pairwise preference over the items. There are a number of pairwise functions that we can use here, however, for consistency we chose to use the functions that were utilized in the SVD-based aggregation method [37]:

1. **Binary Comparison**:

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

2. **Normalized Rank Difference**

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

3. **Log Rank Difference**

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

Here $I[]$ is an indicator function; when either $\mathbf{R}_n(i, k)$ or $\mathbf{R}_n(j, k)$ is missing $\phi_k(i, j, \mathbf{R}_n)$ is set to 0 instead. Notice that $\phi_k(i, j, \mathbf{R}_n)$ is also zero if $i = j$. Normalization by the maximum (log-)preference assigned by the expert k ensures that $\phi_k(i, j, \mathbf{R}_n)$ has a comparable range across experts.

Table 1: A summary of notation.

Variable	Description
$\mathcal{D} = \{\mathbf{r}_n, \mathbf{R}_n\}_{n=1}^N$	training instances
$\mathbf{r}_n$	ground truth preferences
$\mathbf{R}_n$	$M_n \times K_n$ expert preference matrix: M_n items, K_n experts
$\mathbf{R}_n(i, k)$	ranking for item i by expert k
$\hat{\mathbf{y}}_n$	ranking predicted by the model
$l(\hat{\mathbf{y}}_n, \mathbf{r}_n)$	target loss
$\varphi_k(i, \mathbf{R}_n)$	unary potential from expert k
$\phi_k(i, j, \mathbf{R}_n)$	pairwise potential from expert k

Working with pairwise comparisons has a number of advantages, and models over pairwise preferences have been extensively used in areas such as social choice [9, 21], information retrieval [15, 4], and collaborative filtering [21, 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. For instance, preferences in the form of ratings can be treated like rankings and the same pairwise difference/comparison functions can be applied. Moreover, top-T lists (and their variations) can also be converted into this framework using the binary comparison function and setting $\phi_k(i, j, \mathbf{R}_n) = 1$ if item i is in the top-T and item j is not. These examples demonstrate the flexibility and wide applicability of a model over pairwise preferences.

Second, pairwise comparisons are a relative measure and help reduce the bias from the preference scale. In metasearch 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.

4.2 Distribution Over Permutations

The main idea behind our approach is based on an observation that the pairwise preference functions defined above naturally translate to pairwise potentials in a CRF model. Using these function we can evaluate the "compatibility" of any ranking $\mathbf{y}$ by comparing the order induced by the ranking with the pairwise preferences from each expert. This leads to an energy function:

$$E(\mathbf{y}, \mathbf{R}_n; \boldsymbol{\theta}) = -\frac{1}{M_n^2} \sum_{i=1}^{M_n} \frac{1}{\log(i+1)} \left(\sum_{k=1}^{K_n} \alpha_k \varphi_k(\mathbf{y}^{-1}(i)) + \beta_k^P \sum_{j \neq i} \phi_k(\mathbf{y}^{-1}(i), j, \mathbf{R}_n) - \beta_k^N \sum_{j \neq i} \phi_k(j, \mathbf{y}^{-1}(i), \mathbf{R}_n) \right)$$

where $\mathbf{y}^{-1}(i)$ is the item in position i in ranking $\mathbf{y}$. This energy function contains a binary unary potential $\varphi_k(i) = I[\mathbf{R}_n(i, k) = 0]$, where $I[]$ is an indicator function. This potential is active only when item i is not ranked by the expert k, in which case ϕ_k is 0, and α_k provides a base preference score for the item.

The energy also contains pairwise potentials ϕ_k. Note that from the definition of ϕ in Section 4.1 it follows that only one of $\phi_k(\mathbf{y}^{-1}(i), j, \mathbf{R}_n)$ or $\phi_k(j, \mathbf{y}^{-1}(i), \mathbf{R}_n)$ can be non-zero for any pair of items. Consequently, if $\phi_k(\mathbf{y}^{-1}(i), j, \mathbf{R}_n)$ is on (non-zero) then expert k "agrees" with the relative order induced by $\mathbf{y}$ (lowering the energy) and the strength of this agreement is given by ϕ_k. Similarly, if $\phi_k(j, \mathbf{y}^{-1}(i), \mathbf{R}_n)$ is on then expert k "disagrees" with the relative order, and raises the energy. The weights β_k^P and β_k^N thus control how much emphasis is given to positive and negative relative preferences from expert k. $1/\log(i+1)$ is the rank discount function similar to the one used in NDCG and other IR metrics, which emphasizes items at the top positions in the ranking. Finally, normalizing by $1/M_n^2$ ensures that the energy ranges are comparable across instances with different numbers of items.

Using the energy we define a conditional probability for a ranking $\mathbf{y}$:

$$p(\mathbf{y}|\mathbf{R}_n) = \frac{1}{Z(\mathbf{R}_n)} \exp(-E(\mathbf{y}, \mathbf{R}_n; \boldsymbol{\theta}))$$
$$Z(\mathbf{R}_n) = \sum_{\mathbf{y}} \exp(-E(\mathbf{y}, \mathbf{R}_n; \boldsymbol{\theta})) \quad (6)$$

where the partition function $Z(\mathbf{R}_n)$ sums over all $M_n!$ valid rankings $\mathbf{y}$. In the proposed model a separate set of weights $\{\alpha_k, \beta_k^P, \beta_k^N\}$ is learned for each expert k, which allows the model to effectively capture the correlations between individual expert preferences and the ground truth ones. The proposed framework can easily handle training/test instances with missing experts by simply dropping the corresponding pairwise potentials from the energy and only using the base scores ψ_k for those experts. Moreover, while existing models rely exclusively on pairwise matrices, our model can be straightforwardly extended to handle any available side information on both items and experts. For instance, by adding extra pairwise and higher order potentials we can go beyond item interaction and, for instance, model interactions between experts correlating them to ground truth preferences.

This framework however, cannot be applied when the expert identity is unknown or when new experts, unseen during training, are introduced at test time. This is often the case in domains like crowdsourcing where the experts must be anonymized due to privacy considerations, and the number of experts is large so new experts are often introduced at test time. To generalize the model to these settings we can simply share the same parameters α, β^P and β^N, removing the dependence on k. The resulting *consensus* model only takes into account the net preference across all K_n experts, ignoring the individual preferences. Though this makes it possible to apply the model to arbitrary expert sets, this may weaken it since preference information from individual experts can contain very useful information, especially in cases where the majority of experts are wrong. When a subset of the experts is known, it is possible to take an intermediate approach and learn individual weights $\{\alpha_k, \beta_k^P, \beta_k^N\}$ for the known experts k, and consensus-based weights $\{\alpha, \beta^P, \beta^N\}$ for the unknown experts. This demonstrates the flexibility of the proposed CRF framework which allows us to effectively learn to aggregate preferences in the settings where both item and expert sets can vary in length.

Algorithm 1 Learning Algorithm

Input: $\{\mathbf{r}_n, \mathbf{R}_n\}_{n=1}^N$
Parameters: learning rate η, cut-off ϵ
initialize weights: $\boldsymbol{\theta}$
repeat {CRF optimization}
  for $n = 1$ **to** N **do**
    if $M_n > \epsilon$ **then**
      subsample items to get $\mathbf{r}_n^\epsilon$ and $\mathbf{R}_n^\epsilon$
    else
      $\mathbf{r}_n^\epsilon = \mathbf{r}_n$ and $\mathbf{R}_n^\epsilon = \mathbf{R}_n$
    end if
    compute exact gradients with $\{\mathbf{r}_n^\epsilon, \mathbf{R}_n^\epsilon\}$:
      $\nabla\theta = \partial O(\mathbf{r}_n^\epsilon, \mathbf{R}_n^\epsilon)/\partial\boldsymbol{\theta}$
    update weights: $\boldsymbol{\theta} = \boldsymbol{\theta} - \eta\nabla\theta$
  end for
until convergence
Output: $\boldsymbol{\theta}$

4.3 Learning and Inference

Given the model our aim is to learn the parameters $\boldsymbol{\theta} = \{\alpha_k, \beta_k^P, \beta_k^N\}_{k=1}^K$ that minimize the average training loss $\frac{1}{N}\sum_{n=1}^N l(\hat{\mathbf{y}}_n, \mathbf{r}_n)$. Unfortunately, direct minimization is typically not possible because $l(\hat{\mathbf{y}}_n, \mathbf{r}_n)$ is not a smooth function of the CRF parameters $\boldsymbol{\theta}$. Specifically, the loss itself $l(\hat{\mathbf{y}}_n, \mathbf{r}_n)$ is not a smooth function of the prediction $\hat{\mathbf{y}}_n$ and $\hat{\mathbf{y}}_n$ itself is also not a smooth function given the model parameters $\boldsymbol{\theta}$. Such non-smoothness makes it impossible to apply gradient-based optimization directly.

To solve this problem recent work explored different approximation methods to incorporate the target loss into CRF training [38, 11, 25]. The most related of these approaches is the learning-to-rank method BoltzRank [38]. The authors of BoltzRank also dealt with a parametrized distribution over permutations and optimized it for the target IR metric. Inspired by this work we follow this approach and use the expected loss as the target objective to minimize:

$$O(\mathbf{r}_n, \mathbf{R}_n) = \sum_{\mathbf{y}} l(\mathbf{y}, \mathbf{r}_n) p(\mathbf{y}|\mathbf{R}_n) \quad (7)$$

Note that even in cases where l is non-smooth (e.g., NDCG, ERR) the above objective remains smooth with respect to $\boldsymbol{\theta}$ and can be minimized using standard gradient-based procedure. However, to optimize this objective for a given training instance we need to calculate $l(\mathbf{y}, \mathbf{r}_n)$ and $p(\mathbf{y}|\mathbf{R}_n, \boldsymbol{\theta})$ for all $M_n!$ rankings. This computation very quickly becomes intractable since even for $M_n = 15$ one needs to sum over more than 10^{12} permutations. Standard MCMC and variational techniques can be used here to estimate the gradients, however, these methods are typically too slow to be applied to the IR domain where data sets often contain thousands of queries. To deal with this problem the authors of [38] suggested pre-computing a fixed sample set for every instance and reusing it throughout learning. While this approach is computationally efficient, it might miss important regions of model's probability space and can thus be ineffective at optimizing the target distribution.

To avoid these problems we opted to use an approach suggested by [5], which we empirically found to work very well. Every time an instance n is visited and the number of items is greater than ϵ, we sample a subset of ϵ items and use the corresponding expert preferences $\mathbf{R}_n^\epsilon$ and targets $\mathbf{r}_n^\epsilon$

Table 2: MQ2008-agg and MQ2007-agg results; statistically significant differences between CRF and SVDsup are underlined. All the differences between CRF and other baselines 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											
CPS	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
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 Fusion	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
SVDsup	**42.81**	44.53	47.02	49.00	50.69	**48.85**	44.13	41.84	**39.09**	36.50	50.32
CRF	42.29	**44.99**	**47.54**	**49.05**	**51.03**	48.67	**44.58**	**42.08**	38.75	**36.55**	**50.41**
MQ2007-agg											
CPS	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
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 Fusion	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
SVDsup	46.13	46.76	46.71	**46.87**	**47.28**	52.90	51.39	**49.33**	**47.80**	46.66	50.05
CRF	**46.93**	**46.81**	**46.75**	46.51	46.93	**54.14**	**51.48**	49.12	47.54	**46.68**	**50.39**

to compute the gradients for $\boldsymbol{\theta}$. When selecting the items we ensure diversity by sampling from different relevance groups. This is especially important for imbalanced datasets, which are common in IR, where most items are irrelevant. For these datasets random sampling often leads to subsets where all items are irrelevant thus providing very little learning signal to the model. Choosing ϵ sufficiently small allows the gradients to be computed exactly by enumerating all possible $\epsilon!$ permutations of the items. Unlike static sample sets, repeated re-sampling together with full enumeration of all permutations allows us to explore all regions of the model's probability space throughout learning albeit for the reduced item sets.

To make the learning more efficient both unary and pairwise potentials can be precomputed a priori and re-used throughout learning. This reduces the complexity of computing the model's energy from $O(M_n^2 K_n)$ to $O(M_n K_n)$ at the cost of additional storage requirement of $O(M_n)$ per training instance. The complete learning algorithm is summarized in Algorithm 1.

Once the model is learned, at test time, given a new instance with corresponding experts' preferences $\mathbf{R}$ our goal is to produce a single aggregate ranking $\hat{\mathbf{y}}$ of the items that has the highest probability (lowest energy) under the model. Fortunately, such inference can be done very efficiently in this CRF. We note that the energy can be rewritten as a sum of discounted "weights":

$$E(\mathbf{y}, \mathbf{R}; \boldsymbol{\theta}) = \frac{1}{M_n^2} \sum_{i=1}^{M_n} \frac{\omega_i}{\log(i+1)}$$

where the weights are given by:

$$\omega_i = -\sum_{k=1}^{K} \alpha_k \varphi_k(i) - \beta_k^P \sum_{j \neq i} \phi_k(i, j, \mathbf{R}) + \beta_k^N \sum_{j \neq i} \phi_k(j, i, \mathbf{R}) \tag{8}$$

Since $1/\log(i+1)$ is a monotonically decreasing function it is easy to verify that the ranking with highest probability is obtained by sorting the items according to the weights:

$$\hat{\mathbf{y}} = \arg\min_{\mathbf{y}} E(\mathbf{y}, \mathbf{R}; \boldsymbol{\theta}) = \arg\text{sort}([\omega_1, ..., \omega_M]) \tag{9}$$

It is important to note here that this inference procedure only requires computing simple sums and can thus be done very efficiently . This is a significant advantage over existing aggregation methods based on pairwise matrices that require complex optimization procedures such as semidefinite programming [20] or SVD [37] to be run at test time. Moreover, while the inference procedure is simple the learning in our model takes full advantage of the target loss function and optimizes the aggregating function for that metric.

5. EXPERIMENTS

For our experiments we use the LETOR4.0 benchmark datasets [18]. 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 rank aggregation data sets, MQ2007-agg and MQ2008-agg.

MQ2007-agg contains 1692 queries (instances) with a total of 69623 documents (items), and MQ2008-agg contains 784 queries and a total of 15211 documents. Each query contains partial expert rankings of the documents under that query. There are 21 experts in MQ2007-agg and 25 in MQ2008-agg. Consequently, for every query n in MQ2007-agg with M_n documents we have a sparse $M_n \times 21$ ($M_n \times 25$ for MQ2008-agg) expert preference matrix $\mathbf{R}_n$. In addition, in both data sets, each document is assigned one of three relevance levels: 2 = highly relevant, 1 = relevant and 0 = irrelevant. These relevance levels correspond to the ground truth preferences $\mathbf{r}_n$. 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 partial ranking matrix $\mathbf{R}_n$ will be missing 35% of its entries. MQ2008-agg is

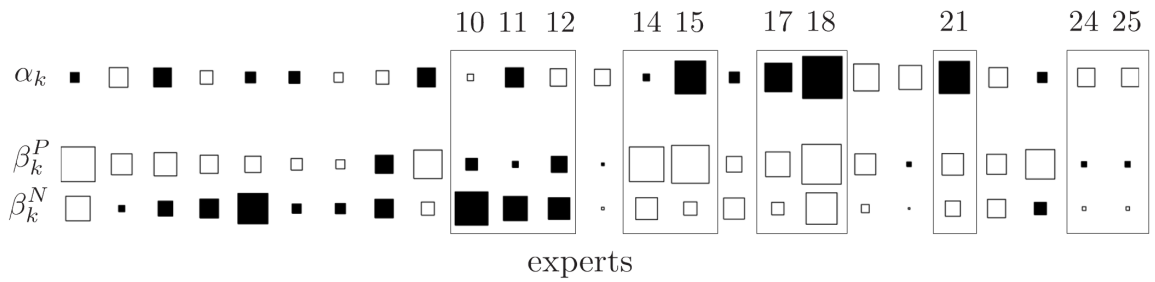


Figure 1: Learned α_k, β_k^P and β_k^N expert weights for training Fold 1 of MQ2008-agg; weights for other folds look analogous. White squares represent positive weights while black squares represent negative ones. The area of each square is proportional to weight magnitude.

significantly more sparse with the sparsity factor of approximately 65%.

The goal is to use the training data to learn a mapping from $\mathbf{R}_n$ to an aggregate ranking $\hat{\mathbf{y}}_n$ that has maximal agreement with the ground truth preferences $\mathbf{r}_n$. In LETOR4.0 this agreement is evaluated using NDCG (N@T, see Equation 2), Precision (P@T) and Mean Average Precision (MAP) [2]. Unlike NDCG, MAP only allows binary (relevant/not relevant) document relevance, and is defined in terms of average precision (AP):

$$AP(\hat{\mathbf{y}}_n, \mathbf{r}_n) = \frac{\sum_{i=1}^{M_n} P@i * \mathbf{r}_n(\hat{\mathbf{y}}_n^{-1}(i))}{\sum_{i=1}^{M_n} \mathbf{r}_n(i)} \tag{10}$$

where $P@i$ is the precision at i:

$$P@i = \sum_{j=1}^{i} \frac{\mathbf{r}_n(\hat{\mathbf{y}}_n(j))}{i} \tag{11}$$

MAP is then computed by averaging AP over all queries. To compute P@k and MAP on the MQ datasets the relevance levels are binarised with 1 converted to 0 and 2 converted to 1. All presented NDCG, Precision and MAP results are averaged across the test queries and were obtained using the evaluation script available on the LETOR website[1].

5.1 Results

To the best of our knowledge the SVD approach of [37] currently has the best published results on the MQ-agg datasets so in experiments we concentrate on comparing our approach with this method. To train our model we use stochastic gradient descent (one query at a time) and do 300 full passes through the training data. We set $\epsilon = 6$ (see Algorithm 1) and ensure that at least one document of every relevance level appears in the same for each query. To choose the type of pairwise potential to use (see Section 4.1) we train separate models with each type and use validation MAP to select the best one. We found that the log rank difference potential generally produced the best results for both datasets.

We compare our model to the best method listed on LETOR website, namely the CPS (combination of Mallows and Plackett-Luce models) [31] on each of the MQ-agg datasets. In addition, we compare with the established meta-search standards Condorcet Fusion [27] and Reciprocal Rank Fusion (RRF) [8] as well as the Plackett-Luce model. Finally, we also compare with two SVD-based approaches that use the same pairwise matrices: unsupervised method SVP [12] and the supervised SVD approach (SVDsup) [37] described above. These models cover all of the primary leading approaches in the rank aggregation research except for the Markov Chain model [20].

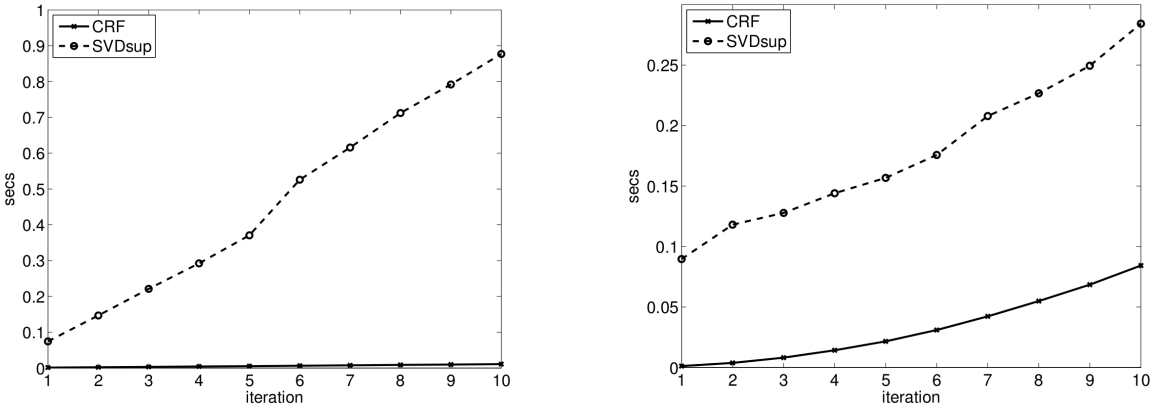
NDCG, Precision and MAP results for both datasets are shown in Table 2. From the tables we see that our model has very strong performance producing similar results to the best baseline SVDsup. It is important to note here that we use the *same* pairwise matrices as SVDsup during both learning and inference. These results indicate that our model is able to achieve highly competitive performance without using expensive optimization procedures during inference. In the following section we quantify the difference in runtimes between the two models.

An additional advantage of using preference matrices directly is model interpretability. By analyzing the learned potential weights we can gain insight into which experts are useful and how their preferences are combined. Figure 1 shows an example weight matrix learned by our model on the training Fold 1 of MQ2008-agg. Before delving into the figure we note that negative α_k raises the energy (lowering the probability). Hence large negative values indicate that when preference from expert k is missing for a given document it is pushed down in the aggregate ranking i.e. expert k is important for aggregation. Similarly, positive β_k^P lower the energy (upping the probability) while positive β_k^N raise the energy. Consequently, when both weights are positive for a given expert k, documents i strongly preferred by k (i.e. $\sum_{j \neq i} \phi_k(i, j, \mathbf{R}_n) \gg \sum_{j \neq i} \phi_k(j, i, \mathbf{R}_n)$) get pushed up in the ranking while those not preferred get pushed down.

Taking these relationships into account we see from Figure 1 that preferences from experts 14, 15, 17, 18 and 21 are good indicators of document relevance. The importance of these experts is shown by large negative values of α_k. Moreover, large positive values for both β_k^P and β_k^N indicate that strong net preference from each of these experts correlates closely with high relevance.

We also see that some experts are not useful for aggregation. For instance experts 24, and 25 all have positive α_k's meaning that when their preferences are absent the rank of a document actually improves. Each of these experts also has near-zero β_k^P and β_k^N indicating that when their preferences are present the model does not use them.

[1] `http://research.microsoft.com/en-us/um/beijing/projects/letor/`



(a) Expert expansion runtimes

(b) Item expansion runtimes

Figure 2: Average per query runtimes (in seconds) for test Fold 1 of the MQ2008-agg. Figure 2(a) shows runtimes for the expert expansion experiment. Figure 2(b) shows runtimes for the item expansion experiment.

Table 3: MQ2008-agg NDCG@1-5 results; CRF is trained on the full data, CRF* is trained on a subset of the data with experts 13, 20, 24 and 25 removed.

	N@1	N@2	N@3	N@4	N@5
CRF	42.29	44.99	47.54	**49.05**	**51.03**
CRF*	**42.64**	**45.07**	**47.63**	49.00	50.90

Finally, some experts are used for aggregation even though their preferences correlate inversely with ground truth. For instance, experts 10, 11 and 12 all have negative β_k^P and β_k^N weights indicating that documents strongly preferred by these experts will be pushed down in the ranking while those strongly opposed will be pushed up. Moreover, most weights for these experts are large indicating that they play an important role in the aggregation process. The model thus learned that these experts often give wrong relative orderings reversing which can still lead to useful predictions. It is worth emphasizing here that this kind of inverse relationship is impossible to capture with unsupervised methods.

To further validate the utility of analyzing experts through CRF's parameters we removed experts whose preferences were found not to be useful by the CRF and retrained the model. Specifically, from Figure 1 we see that experts 13, 20, 24 and 25 are not being used by the model and when preferences from these experts are missing, the corresponding document actually gets a boost in ranking. These experts are clearly not useful for ranking so we removed them and retrained the model on the remaining 21 experts. The results are shown in Table 3, from the table we see that retrained model CRF* either performs comparably or outperforms the original model. This further support the conclusion that useful insight into expert quality can be gained by analyzing the weights learned by our model. Such analysis can be particularly useful in crowdsourcing and related domains where the goal is often to identify the most accurate/reliable labelers from the crowd.

5.2 Runtime Comparison

In the previous section we demonstrated that our model has comparable performance to the state-of-the-art model SVDsup. Moreover, inference in our model only requires computing simple sums and can thus be done considerably more efficiently than in SVDsup which requires SVD factorization. In this section we quantify this difference.

We use test Fold 1 of the MQ2008-agg dataset and conduct two sets of experiments. In the first experiment we repeatedly increase the number of experts. Starting with the initial expert matrix at iteration 1: $\mathbf{R}_n^{(1)} = \mathbf{R}_n$, we concatenate it with the original matrix to get an expanded one for iteration 2: $\mathbf{R}_n^{(2)} = [\mathbf{R}_n^{(1)}, \mathbf{R}_n]$. Thus, after t iterations the resulting matrix $\mathbf{R}_n^{(t)} = [\mathbf{R}_n^{(t-1)}, \mathbf{R}_n]$ contains M_n rows and $t \times K_n$ columns. Concatenating expert matrices allows us to test the inference procedure of each method on an increasingly larger data while preserving sparsity. In the second experiment we repeat this procedure but this time we append the matrices increasing the number of documents. Here, the ranking matrix at iteration t contains $t \times M_n$ rows and K_n columns. The first experiment thus tests for scenarios where the expert set is large (expert expansion), that typically arise in domains like crowdsourcing. While the the second experiment tests for large item sets (item expansion) that often arise in domains like meta-search.

Figures 2 and 2(b) show, averaged across queries, runtimes (in seconds) for both methods at each expansion iteration. Figure 2(a) shows runtimes for the expert expansion while Figure 2(b) shows runtimes for the item expansion. From the figures we see significant differences in runtimes between the two methods. The difference is especially large for the expert expansion (Figure 2(a)) where SVDsup is on average almost 80 times slower than our CRF method at the tenth iteration. This difference is due to the fact that SVDsup has to run SVD factorization for *every* expert. Consequently, the number of SVD factorizations grows linearly with the number of experts significantly slowing down SVDsup. For the item expansion (Figure 2(b)) the number of experts stays constant while the dimension of the preference matrix increases. Since no additional SVD factorizations are required we found the speed of SVDsup to not increase as significantly as in the first experiment. However, even in this setting our approach is more than 3.5 times faster. Moreover, we found that for very large matrices (not shown on this plot) SVD factorization dominated the calculation significantly slowing down SVDsup. From these results we can conclude that our approach is considerably more efficient than SVDsup especially in cases where the number of experts is large.

6. CONCLUSION

We presented a fully supervised CRF approach to preference aggregation. Unlike existing methods our approach uses observed preferences directly and does not require any expensive optimization procedures at test time. The direct use of preferences also allows us to analyze learned models and draw valuable conclusions about preference quality of each expert. Experimental results show that our approach has very competitive performance outperforming existing methods on two supervised rank aggregation tasks.

Going forward a promising direction would be to explore different types of potentials. Specifically, we plan to experiment with adding expert cross correlations and incorporating side information for items and/or experts. Another promising area of research would be to explore CRF models that in addition to experts also condition on queries. In meta-search and other applications it is often the case that different experts perform well for different queries. Adding this extra conditioning can thus help the model to distinguish when to use each expert making it more powerful.

7. 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*, 2001.

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

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

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

[5] T. S. Caetano, L. Cheng, Q. V. Le, and A. J. Smola. Learning graph matching. In *Proceedings of the International Conference on Machine Learning*, 2009.

[6] O. Chapelle, D. Metlzer, Y. Zhang, and P. Grinspan. In *Proceedings of the ACM Conference on Information and Knowledge Management*, 2009.

[7] S. Chen, F. Wang, Y. Song, and C. Zhang. Semi-supervised ranking aggregation. 47, 2011.

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

[10] R. Fagin, R. Kumar, and D. Sivakumar. Efficient similarity search and classification via rank aggregation. In *Proceedings of the ACM SIGMOD Conference*, 2003.

[11] K. Gimpel and N. A. Smith. Softmax-margin CRFs: Training log-linear models with cost functions. In *HLT-NAACL*, 2010.

[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*, 2011.

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

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

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

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

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

[18] T. Liu, J. Xu, W. Xiong, and H. Li. LETOR: Benchmark dataset for search on learning to rank for information retrieval. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2007.

[19] Y. Liu, J. Carbonell, P. Weigele, and V. Gopalakrishnan. Protein fold recognition using segmentation conditional random fields (SCRFs). *Journal of Computational Biology*, 2006.

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

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

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

[23] C. L. Mallows. Non-null ranking models. *Biometrika*, 44, 1957.

[24] D. Mase. A penalized maximum likelihood approach for the ranking of college football teams independent of victory margins. *The American Statistician*, 57, 2003.

[25] D. McAllester and J. Keshet. Generalization bounds and consistency for latent structural probit and ramp loss. In *Proceedings of the Neural Information Processing Systems*, 2011.

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

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

[28] R. Plackett. The analysis of permutations. *Applied Statistics*, 24, 1975.

[29] M. Pujari and R. Kanawati. Supervised rank aggregation approach for link prediction in complex networks. In *Proceedings of the International World Wide Web Conference*, 2012.

[30] T. Qin, T.-Y. Liu, X.-D. Zhang, D.-S. Wang, and H. Li. Global ranking using continuous conditional random fields. In *Proceedings of the Neural Information Processing Systems*, 2008.

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

[32] D. Roth and W.-Y. Yih. Integer linear programming inference for conditional random fields. In *Proceedings of the International Conference on Machine Learning*, 2005.

[33] S. Sarawagi and W. W. Cohen. Semi-Markov conditional random fields for information extraction. In *Proceedings of the Neural Information Processing Systems*. 2005.

[34] K. Sato and Y. Sakakibara. RNA secondary structural alignment with conditional random fields. *Bioinformatics*, 2005.

[35] F. Sha and F. Pereira. Shallow parsing with conditional random fields. In *HLT/NAACL*, 2003.

[36] K. Subbian and P. Melville. Supervised rank aggregation for predicting influencers in twitter. In *SocialCom*, 2011.

[37] M. N. Volkovs, H. Larochelle, and R. S. Zemel. Learning to rank by aggregating expert preferences. In *Proceedings of the ACM Conference on Information and Knowledge Management*, 2012.

[38] M. N. Volkovs and R. S. Zemel. Boltzrank: Learning to maximize expected ranking gain. In *Proceedings of the International Conference on Machine Learning*, 2009.