Journal of Machine Learning Research 15 (2014) 981-1022 Submitted 10/12; Revised 9/13; Published 3/14

New Learning Methods for Supervised and Unsupervised Preference Aggregation

Maksims N. Volkovs MVOLKOVS@CS.TORONTO.EDU
Richard S. Zemel ZEMEL@CS.TORONTO.EDU
University of Toronto
40 St. George Street
Toronto, ON M5S 2E4

Editor: William Cohen

Abstract

In this paper we present a general treatment of the preference aggregation problem, in which multiple preferences over objects must be combined into a single consensus ranking. We consider two instances of this problem: unsupervised aggregation where no information about a target ranking is available, and supervised aggregation where ground truth preferences are provided. For each problem class we develop novel learning methods that are applicable to a wide range of preference types.[1] Specifically, for unsupervised aggregation we introduce the Multinomial Preference model (MPM) which uses a multinomial generative process to model the observed preferences. For the supervised problem we develop a supervised extension for MPM and then propose two fully supervised models. The first model employs SVD factorization to derive effective item features, transforming the aggregation problems into a learning-to-rank one. The second model aims to eliminate the costly SVD factorization and instantiates a probabilistic CRF framework, deriving unary and pairwise potentials directly from the observed preferences. Using a probabilistic framework allows us to directly optimize the expectation of any target metric, such as NDCG or ERR. All the proposed models operate on pairwise preferences and can thus be applied to a wide range of preference types. We empirically validate the models on rank aggregation and collaborative filtering data sets and demonstrate superior empirical accuracy.

Keywords: preference aggregation, meta-search, learning-to-rank, collaborative filtering

1. Introduction

Many areas of study, such as information retrieval (IR), collaborative filtering, and social web analysis face the preference aggregation problem, in which multiple preferences over objects must be combined into a single consensus ranking. Early developments in preference aggregation and analysis originated in social science (Arrow, 1951) and statistics (Luce, 1959), giving rise to the field of social choice. Research in social choice concentrates on measuring individual interests, values, and/or welfare as an aggregate towards collective decision. Common problems explored in this field include vote aggregation in elections and other domains as well as player/team ranking based on observed game outcomes. Most of these problems are relatively small in size and can be analyzed thoroughly, resulting in models that have well-explored properties and theoretical guarantees. These models

1. The code for all models introduced in this paper is available at `www.cs.toronto.edu/~mvolkovs`.

now decide the outcomes of such crucial events as presidential and government elections as well as legal decisions. However, the theoretical guarantees for many of these models typically come at the expense of complicated inference and/or strong assumptions about the preference data (Chevaleyre et al., 2007; Rossi et al., 2011).

The recent explosion of web technologies has generated immense amounts of new preference data. Several properties of this data make it difficult to apply many of the existing aggregation models. First, the ease with which people can access and generate content on the web has resulted in a drastic increase in the quantity of data. For instance, where before the majority of sports data had on the order of a thousand players that participated in several tournaments per year, now, online gaming has millions of users that participate in tens of millions of games daily. Recent statistics on the popular game Halo indicate that over 2 billion multiplayer games are played annually,[2] with hundreds of thousands of games happening at any given moment. Consequently, while the aggregation problem in online gaming remains similar to the traditional one in sports—combine preference in the form of game outcomes to generate reliable estimates of players' skills—any model developed for this task now has to be able to process large amounts of data quickly and handle diverse evidence types ranging from one-on-one games to elimination team tournaments.

Second, the diversity of online applications has led to many new preference types. In addition to the common direct evidence in the form of votes and ratings we now also have a variety of indirect evidence, including web page clicks, dwell time (time spent on a page), and viewing patterns. While these evidence forms do not directly indicate preference, when aggregated across many users, they have been found to closely correlate with it (Joachims, 2002; Joachims et al., 2007). Methods that mine these preferences are now extensively used in search engine optimization (Agichtein et al., 2006; Joachims et al., 2007; Guo et al., 2009) and other domains. Moreover, for some of the new problems the preferences are no longer generated by people. An example of this is meta-search where an issued query is sent to several search engines and the (often partial) document rankings returned by them are aggregated by the meta-search engine to generate more comprehensive ranking results. In this problem there is no human interaction and all the preferences are generated by the machines. Consequently social theories on user behavior and models based on these theories are less applicable here.

Finally, new types of aggregation problems have also recently emerged. In the past the majority of the aggregation problems were unsupervised, that is, no ground truth preference information about the items was available. For these problems the aim is typically to produce a ranking that satisfies as many of the observed preferences (majority or another related objective) as possible. Due to the popularity of such problems almost all of the existing research in preference aggregation has concentrated on the unsupervised aggregation. However, many of the recent problems are amenable to the supervised setting, as ground truth preference information is available. The meta-search problem mentioned above is one example of supervised preference aggregation. Often, to train/evaluate the aggregating function the documents retrieved by the search engines are given to human annotators who assign relevance labels to each document. The relevance labels provide ground truth preference information about the documents, that is, the documents with higher relevance

2. Full article can be found at `http://www.pcmag.com/article2/0,2817,2402479,00.asp`.

label are to be ranked above those with lower one. Another example is crowdsourcing (e.g., MechanicalTurk), where 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 labeling of the data. To ensure consistency in generated labels a domain expert typically labels a subset of the data shown to the "crowd". The labels are then used to evaluate the quality of annotations submitted by each worker. In these problems the aim is not to satisfy the majority but rather to learn a mapping from the observed preferences to the ground truth ones. Consequently, methods that aim to satisfy the majority often lead to suboptimal results since they lack the "specialization" property: they cannot identify cases where the majority is wrong and only a small subset of the preferences should be used. Such a property is impossible to achieve without referring to the ground truth labels.

The scale and variety of the preference data generated by the social and other web domains discussed above show that the field of preference aggregation is rapidly evolving and expanding. Almost every user-oriented web application ranging from web shops and social networks to web search and gaming is now using preference aggregation techniques, the accuracy of which has a direct and significant impact on the generated revenue and business decisions. There is thus an evident need to develop effective aggregation methods that are able to scale to the large web data sets and handle diverse preference types.

As the field has evolved a new trend has recently emerged where machine learning methods are starting to be used to automatically learn the aggregating models. While these methods typically lack the theoretical support of the social choice models they often show excellent empirical performance and are able to handle large and diverse preference data. These models have now been applied successfully to preference aggregation problems in collaborative filtering (Gleich and Lim, 2011; Jiang et al., 2011; Guiver and Snelson, 2009), information retrieval (Cormack et al., 2009; Liu et al., 2007b; Chen et al., 2011) and online gaming (Dangauthier et al., 2007), as well as others. Inspired by these results the work presented in this paper also takes a machine learning approach and develops new models for both supervised and unsupervised preference aggregation problems. In the following sections we describe existing approaches and open challenges for both problems. We then introduce and empirically validate new models for each problem type.

2. Unsupervised Preference Aggregation

Unsupervised preference aggregation is the problem of combining multiple preferences over objects into a single consensus ranking when no ground truth preference information is available. As mentioned above, the majority of research in preference aggregation has concentrated on this problem and a number of models have been developed. Given the underlying correspondence between ranking and permutation, considerable work on unsupervised preference aggregation has exploited probabilistic models on permutations, many of which originate in statistics and psychology. Mallows (Mallows, 1957) and Plackett-Luce (Plackett, 1975; Luce, 1959) are particularly popular models, each with many extensions (Guiver and Snelson, 2009; Quin et al., 2010; Lu and Boutilier, 2011). However, research has largely concentrated on learning a consensus ranking based on a set of observed full, or partial rankings. These models are thus inadequate for problems where preferences are

expressed in other forms, and where inconsistencies exist in the observed preferences, such as "a beat b", "b beat c", and "c beat a".

In this section we address this problem by developing a flexible probabilistic model over pairwise comparisons. 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. Our model can thus be applied to a wide spectrum of preference aggregation problems and does not impose any restrictions on the type of evidence. The score-based approach that we adopt allows for rapid learning and inference, which makes the model applicable to large-scale aggregation problems. We experimentally validate our model on a rank aggregation and collaborative filtering tasks using Microsoft's LETOR4.0 (Liu et al., 2007a) and the MovieLens (Herlocker et al., 1999) data sets.

2.1 Framework

We assume a set of N instances where for every instance n we have a set of M_n items $\mathbf{X}_n = \{x_{n1}, ..., x_{nM_n}\}$ and a set of Ψ experts. Each expert $\psi \in \{1, ..., \Psi\}$ generates a list of preferences for items in $\mathbf{X}_n$. We assume that the same set of experts generate preferences for items in each instance. The preferences can be in the form of full or partial rankings, top-K 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. We use $\{x_{ni} \succ x_{nj}\}$ to denote the preference of x_{ni} over x_{nj}. We allow the same pairwise preferences to occur multiple times, and use the pairwise count matrix $\mathbf{Y}_n^{\psi}(i,j) : M_n \times M_n$ to count the number of times preference $\{x_{ni} \succ x_{nj}\}$ is produced by the expert ψ, with $\mathbf{Y}_n^{\psi}(i,j) = 0$ if $\{x_{ni} \succ x_{nj}\}$ is not expressed by ψ.

The most straightforward way to convert rankings into pairwise preferences is through binary comparisons. Given two rankings r_{ni}^{ψ} and r_{nj}^{ψ} assigned by ψ to x_{ni} and x_{nj} we set $\mathbf{Y}_n^{\psi}(i,j) = I[r_{ni}^{\psi} < r_{nj}^{\psi}]$ where I is an indicator function, similarly $\mathbf{Y}_n^{\psi}(j,i) = I[r_{nj}^{\psi} < r_{ni}^{\psi}]$. This representation, however, completely ignores the strength of preference expressed by the magnitude of the rankings. For example, the partial ranking $\{1, 200, 300\}$ will have the same count matrix as the ranking $\{1, 2, 3\}$, but the first ranking expresses significantly more confidence about the ordering of the items than the second one. To account for this we instead use $\mathbf{Y}_n^{\psi}(i,j) = (r_{nj}^{\psi} - r_{ni}^{\psi})I[r_{ni}^{\psi} < r_{nj}^{\psi}]$ and $\mathbf{Y}_n^{\psi}(j,i) = (r_{ni}^{\psi} - r_{nj}^{\psi})I[r_{nj}^{\psi} < r_{ni}^{\psi}]$. In this form we assume that ranking $\{r_{ni} = 1, r_{nj} = 200\}$ is equivalent to observing the pairwise preference $\{x_{ni} \succ x_{nj}\}$ 199 times, whereas ranking $\{r_{ni} = 1, r_{nj} = 2\}$ is equivalent to observing $\{x_{ni} \succ x_{nj}\}$ only once. This method of accounting for preference strength is not new and the reader can refer to Gleich and Lim (2011) and Jiang et al. (2011) for more extensive treatment of this and other approaches for converting rankings to pairwise matrices. We summarize these pairwise preference representations below:

1. **Binary Comparison**:
$$\mathbf{Y}_n^{\psi}(i,j) = I[r_{ni}^{\psi} < r_{nj}^{\psi}],$$

2. **Rank Difference**:
$$\mathbf{Y}_n^{\psi}(i,j) = (r_{nj}^{\psi} - r_{ni}^{\psi})I[r_{ni}^{\psi} < r_{nj}^{\psi}].$$

A ranking of items in $\mathbf{X}_n$ can be represented as a permutation of $\mathbf{X}_n$. A permutation π is a bijection $\pi : \{1, ..., M_n\} \rightarrow \{1, ..., M_n\}$ mapping each item x_{ni} to its rank $\pi(i) = j$, and $i = \pi^{-1}(j)$. Given the observed (partial) preference instance n consisting of count matrices $\mathbf{Y}_n = \{\mathbf{Y}_n^1, ..., \mathbf{Y}_n^\Psi\}$ the goal is to come up with a single ranking π of items in $\mathbf{X}_n$ that maximally satisfies this instance.

Most preference aggregation problems fit this framework. For instance in meta-search instances correspond to queries and $\mathbf{X}_n$ is the set of documents retrieved for a given query q_n. Each expert ψ represents a search engine which generates either partial or complete ranking of the documents in $\mathbf{X}_n$. As before we can let $\mathbf{Y}_n^\psi(i,j) = (r_{nj}^\psi - r_{ni}^\psi) I[r_{ni}^\psi < r_{nj}^\psi]$ if documents x_{ni} and x_{nj} are both ranked by the search engine ψ and set $\mathbf{Y}_n^\psi(i,j) = 0$ otherwise. In collaborative filtering $\mathbf{X}$ is the set of movies/songs/books etc., and an instance of the rank aggregation problem aims to infer the consensus ranking of movies given the (partial) ratings of users ψ (Guiver and Snelson, 2009; Gleich and Lim, 2011). The pairwise approach provides a natural way to model this problem. We can define $\mathbf{Y}^\psi(i,j) = (\ell_i^\psi - \ell_j^\psi) I[\ell_i^\psi > \ell_j^\psi]$ where ℓ_i and ℓ_j are the ratings assigned to movies x_{ni} and x_{nj} by user ψ. If ψ did not rate either x_{ni} or x_{nj} we set $\mathbf{Y}^\psi(i,j) = 0$.

2.2 Previous Work

Relevant previous work in this area can be divided into two categories: permutation based and score based. In this section we describe both types of models.

2.2.1 Permutation-Based Models

Permutation based models work directly in the permutation space. The most common and well explored such model is the Mallows model (Mallows, 1957). Mallows defines a distribution over permutations and is typically parametrized by a central permutation σ and a dispersion parameter $\phi \in (0, 1]$; the probability of a permutation π is given by:

$$P(\pi|\phi,\sigma) = \frac{1}{Z(\phi,\sigma)} \phi^{-\mathcal{D}(\pi,\sigma)},$$

where $\mathcal{D}(\pi,\sigma)$ is a distance between π and σ. For rank aggregation problems inference in this model amounts to finding the permutation σ 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 $Z(\phi,\sigma)$ can be found exactly. However, finding the central permutation σ that maximizes the likelihood is typically very difficult and in many cases is intractable (Meila et al., 2007).

Recent work extends the Mallows model to define distributions over partial rankings (Lu and Boutilier, 2011). 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 such as the Cranking model (Lebanon and Lafferty, 2002) the Aggregation model (Klementiev et al., 2008), and the CPS model (Quin et al., 2010). The Cranking model extends the Mallows distribution to model several diverse preference profiles. Each preference profile i is modeled by its own central permutation σ_i and "importance" θ_i:

$$P(\pi|\boldsymbol{\sigma},\boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\sigma},\boldsymbol{\theta})} e^{-\sum_i \theta_i \mathcal{D}(\pi,\sigma_i)},$$

where $\boldsymbol{\sigma} = \{\sigma_i\}$ and $\boldsymbol{\theta} = \{\theta_i\}$ are the profile parameters. The above model can be viewed as a mixture of Mallows models where each component is parametrized by (σ_i, θ_i) pair. Using this alternative representation the work of Klementiev et al. (2008) further generalizes the Cranking model to partial top-K lists and derives an Expectation Minimization (EM) algorithm to learn the parameters $\boldsymbol{\theta}$.

Another recent extension of the Mallows model, the CPS model, defines a sequential generative process, similar to the Plackett-Luce model described below, which draws the items without replacement to form a permutation; the probability of a given permutation π is:

$$P(\pi|\sigma,\phi) = \prod_{i=1}^{M} \frac{\exp(-\theta \sum_{\pi_{1:i}} \mathcal{D}(\pi_{1:i},\sigma))}{Z(i,\pi)},$$

where the summation in the numerator is over all permutations $\pi_{1:i}$ that have the first i elements fixed to π; $Z(i,\pi)$'s are the normalizing constants that ensure that $\sum_\pi P(\pi|\sigma,\phi) = 1$. For several distance metrics such as Spearman's rank correlation and footrule as well as Kendall's τ, the summation $\sum_{\pi_{1:i}} \mathcal{D}(\pi_{1:i},\sigma)$ over $(M-i)!$ elements, can be found in $O(M^2)$, allowing the normalizing constants $Z(i,\pi)$ to be computed in polynomial time. However, during inference one must still consider nearly all of the $M!$ possible permutations to find an optimal π. A greedy approximation avoids this search, which reduces the complexity to $O(M^2)$, but provides no guarantee with respect to the optimal solution.

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.

2.2.2 Score-Based Models

In score-based approaches the goal is to learn 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.

Early score based methods for rank aggregation in meta-search are heuristic based. For example, BordaCount (Aslam and Montague, 2001), Condorcet (Montague and Aslam, 2002) and median rank aggregation (Fagin et al., 2003) 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 (Bradley and Terry, 1952) model:

$$P(\mathbf{Y}_n^{\psi}|\mathbf{S}_n) = \prod_{i \neq j} \left(\frac{\exp(s_{ni})}{\exp(s_{ni}) + \exp(s_{nj})} \right)^{\mathbf{Y}_n^{\psi}(i,j)},$$

where $\frac{\exp(s_{ni})}{\exp(s_{ni})+\exp(s_{nj})}$ can be interpreted as the probability that item x_{ni} beats x_{nj} in the pairwise contest. In logistic form the Bradley-Terry model is very similar to another popular pairwise model, the Thurstone model (Thurstone, 1927). Extensions of these models include the Elo Chess rating system (Elo, 1978), adopted by the World Chess Federation FIDE in 1970, and Microsoft's TrueSkill (Dangauthier et al., 2007) rating system for player matching in online games, used extensively in Halo and other games. Furthermore, the popular supervised learning-to-rank model RankNet (Burges et al., 2005) is also based on this approach.

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 x_{ni} 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 ∞ (Mase, 2003). 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 (Plackett, 1975; Luce, 1959) to a model for permutations:

$$P(\pi|\mathbf{S}_n) = \prod_{i=1}^{M_n} \frac{\exp(\mathbf{S}_n(\pi^{-1}(i)))}{\sum_{j=i}^{M_n} \exp(\mathbf{S}_n(\pi^{-1}(j)))},$$

where $\mathbf{S}_n(\pi^{-1}(i))$ is the score of the item in position i in π. The generative process behind the Plackett-Luce model assumes that items are selected sequentially without replacement. Initially item $\pi^{-1}(1)$ is selected from the set of M_n items and placed first, then item $\pi^{-1}(2)$ 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 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 (Guiver and Snelson, 2009). 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 significant computational overhead required during score inference. In this work we show that the

model we develop achieves similar improvement over the Plackett-Luce model without the additional computational overhead and preference type restrictions.

Recently several score based approaches have been developed to model the joint pairwise matrix (Gleich and Lim, 2011; Jiang et al., 2011). In these methods the preferences expressed by each of the experts are combined into a single preference matrix $\mathbf{Y}_n^{tot} = \sum_{\psi=1}^{\Psi} \mathbf{Y}_n^{\psi}$, which is then factorized 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 then used to rank the items. The main drawback of this approach is that by combining all preferences into a single $\mathbf{Y}_n^{tot}$ the individual expert information is lost. Consequently outlier experts with preferences substantially deviating from the consensus can significantly influence both $\mathbf{Y}_n^{tot}$ and the resulting scores.

2.3 Multinomial Preference Model (MPM)

In this section we develop a new score based model for pairwise preferences, the Multinomial Preference Model (MPM) (Volkovs and Zemel, 2012). A key motivating idea behind our approach is that when absolute preferences such as rankings are converted into pairwise counts using the rank difference approach described above, we interpret the resulting counts as conveying two forms of information: a binary preference, simply based on which item is ranked higher, and a confidence, based on the magnitude of the rank difference. Consider for example three items x_1, x_2 and x_3 with ranks $r_1 = 1$, $r_2 = 2$ and $r_3 = 3$ respectively. Figure 1(a) shows the resulting count matrix after these ranks are converted to pairwise preferences. Item x_1 is preferred to both x_2 and x_3 with $\mathbf{Y}(1,2) = r_2 - r_1 = 1$ and $\mathbf{Y}(1,3) = r_3 - r_1 = 2$, x_2 is preferred only to x_3 with $\mathbf{Y}(2,3) = r_3 - r_2 = 1$, and x_3 is not preferred to any item. Note that preference $\{x_1 \succ x_3\}$ where both items are at the extremes of the ranking has the largest rank difference and consequently the biggest count.

Now consider the second example with partial ranking $r_1 = 30$, $r_2 = 20$ and $r_3 = 1$ yielding the pairwise count matrix shown in Figure 1(b). Comparing this with the previous example we see that the preference $\{x_3 \succ x_1\}$ with items at the extremes of the ranking also has the highest count, however in this case we are significantly more certain of it. The count $\mathbf{Y}(3,1) = 29$ is considerably higher than the highest count from the previous example, strongly indicating that x_3 should be placed above x_1. The two examples demonstrate how large values of $\mathbf{Y}(i,j)$ may be interpreted as providing more evidence to conclude that $\{x_i \succ x_j\}$ is correct.

In MPM we model the count matrix $\mathbf{Y}_n^{\psi}$ as an outcome of multiple draws from the joint consensus distribution Q_n over pairwise preferences defined by the scores $\mathbf{S}_n$. For instance in the second example above after observing $\mathbf{Y}$ we can infer that $P(x_3 \succ x_1)$ should have the most mass under Q. We use $\mathbf{B}_n$ to denote the random variable distributed as Q_n. A draw from Q_n can be represented as a vector $\mathbf{b}_{ij}$ of length $M_n * (M_n - 1)$ (all possible pairs), with 1 on the entry corresponding to preference $\{x_{ni} \succ x_{nj}\}$ and zeros everywhere else, that is, a one-hot encoding. Given $\mathbf{S}_n$ we define the consensus distribution as follows:

Definition 1 *The consensus distribution* $Q_n = \{P(\mathbf{B}_n = \mathbf{b}_{ij}|\mathbf{S}_n)\}_{i \neq j}$ *is a collection of pairwise probabilities* $P(\mathbf{B}_n = \mathbf{b}_{ij}|\mathbf{S}_n)$*, where* $P(\mathbf{B}_n = \mathbf{b}_{ij}|\mathbf{S}_n) = \frac{\exp(s_{ni} - s_{nj})}{\sum_{k \neq l} \exp(s_{nk} - s_{nl})}$.

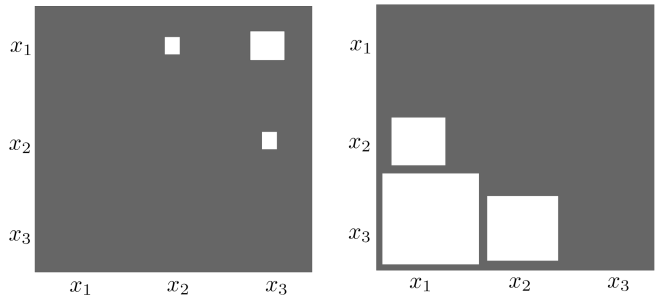


Figure 1: Figure 1(a) displays the count matrix with the contests won by each of the 3 items x_1, x_2 and x_3 after their ranking $\{r_1 = 1, r_2 = 2, r_3 = 3\}$ is converted to pairwise counts using the rank difference method. A count is displayed in each (x_i, x_j) entry if $r_i < r_j$, and the size of the square represents the count magnitude. Figure 1(b) shows the same matrix for the ranking $\{r_1 = 30, r_2 = 20, r_3 = 1\}$.

Q_n defines a multinomial distribution over pairwise preferences. Parametrization through $\mathbf{S}_n$ controls the shape of Q_n, lending considerable flexibility in distributions over preferences, which can be tailored to many different problems. To generate the observed aggregated counts $\mathbf{Y}_n^\psi$ we assume that T_n^ψ independent samples are drawn from Q_n where $T_n^\psi = \sum_{i \neq j} \mathbf{Y}_n^\psi(i,j)$ so that:

$$\mathbf{Y}_n^\psi(i,j) = \sum_{t=1}^{T_n^\psi} I[\mathbf{B}_n = \mathbf{b}_{ij}^{(t)}],$$

where $I[\mathbf{B}_n = \mathbf{b}_{ij}^{(t)}]$ is 1 if preference $\{x_{ni} \succ x_{nj}\}$ was sampled on the t'th draw and 0 otherwise. Under this model the probability of the observed counts is given by:

$$\begin{aligned} P(\mathbf{Y}_n^\psi | \mathbf{S}_n) &= \frac{T_n^\psi!}{\prod_{i \neq j} \mathbf{Y}_n^\psi(i,j)!} \prod_{i \neq j} P(\mathbf{B}_n = \mathbf{b}_{ij} | \mathbf{S}_n)^{\mathbf{Y}_n^\psi(i,j)} \\ &= \frac{T_n^\psi!}{\prod_{i \neq j} \mathbf{Y}_n^\psi(i,j)!} \prod_{i \neq j} \left(\frac{\exp(s_{ni} - s_{nj})}{\sum_{k \neq l} \exp(s_{nk} - s_{nl})} \right)^{\mathbf{Y}_n^\psi(i,j)}. \end{aligned}$$

Note that in MPM the pairwise probabilities depend on the entire item set $\mathbf{X}$ and the observed counts matrix is modeled jointly. The magnitude of the score s_{ni} is directly related to the count $\mathbf{Y}_n^\psi(i,j)$. When the scores are fitted via maximum likelihood the gradient of the log probability with respect to s_{ni} is given by:

$$\frac{\partial \log(P(\mathbf{Y}_n^\psi | \mathbf{S}_n))}{\partial s_{ni}} = \left(\sum_j \mathbf{Y}_n^\psi(i,j) - \sum_j \mathbf{Y}_n^\psi(j,i) \right) - T_n^\psi \left(\frac{\partial \log(\sum_{k \neq l} e^{s_{nk} - s_{nl}})}{\partial s_{ni}} \right). \quad (1)$$

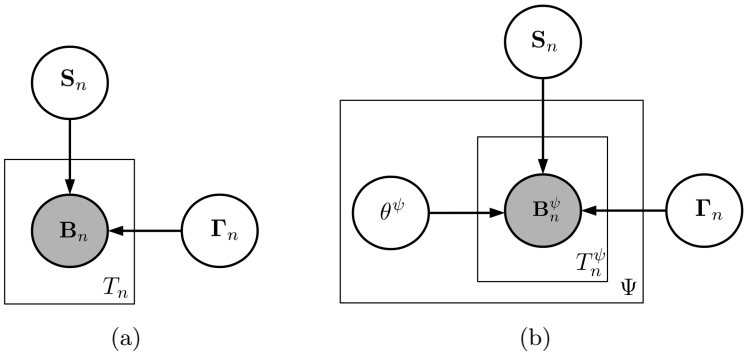


Figure 2: Graphical model representation of the generative process in MPM and its θ extension for a single instance n. Here $T_n^{\psi} = \sum_{i \neq j} \mathbf{Y}_n^{\psi}(i,j)$ is the total number of preferences observed for expert ψ and $T_n = \sum_{\psi=1}^{\Psi} T_n^{\psi}$ is the total number of preferences across all experts for instance n.

Note that when x_{ni} is strongly preferred to other items the first term in Equation 1 will be large leading to an increase in s_{ni}. This will in turn raise the probability of preferences where x_{ni} beats the other items. Raising the probability for some preferences must simultaneously lower it for others since the probabilities always sum to 1. The second term, the derivative of the partition function, accounts for this. The scores thus compete with each other and the ones with the most positive/negative evidence get pushed to the extremes. This is exactly the effect we wanted to achieve because it will allow us to accurately model the count matrices as illustrated by the toy examples above. In contrast with MPM, in the Bradley-Terry model there is no joint interaction amongst scores and pairs are modeled independently so a single preference is sufficient to push the score to infinity.

2.4 Incorporating Prediction Confidence

In the base MPM model it is difficult to judge the model's *confidence* for a given score combination. Aside from the relative score magnitudes, it is hard to measure the uncertainty associated with the score assigned to each item and the aggregate ranking that the scores impose. Such a measure can be very useful during inference and can influence the decision process. For instance, it can be used to further filter and/or reorder the items in the aggregate ranking. Moreover, for problems where the accuracy is extremely important, the recommender system can inform the user if the produced ranking has high/low degree of uncertainty.

To address this problem we introduce a set of *variance* parameters $\mathbf{\Gamma}_n = \{\gamma_{n1}, ..., \gamma_{nM_n}\}$, $\gamma_{ni} > 0 \ \forall i$. Each γ_{ni} models the uncertainty associated with the score s_{ni} inferred for the item x_{ni}. The consensus distribution now becomes:

$$P(\mathbf{B}_n = \mathbf{b}_{ij} | \mathbf{S}_n, \mathbf{\Gamma}_n) = \frac{\exp((s_{ni} - s_{nj})/(\gamma_{ni} + \gamma_{nj}))}{\sum_{k \neq l} \exp((s_{nk} - s_{nl})/(\gamma_{nk} + \gamma_{nl}))}.$$

Note that the probability of x_{ni} beating x_{nj} decreases (increases) if the variance for *either* x_{ni} or x_{nj} increases (decreases). Through $\mathbf{\Gamma}_n$ we can effectively express the variance over the preferences for each item x_{ni} and translate this variance into uncertainty over pairwise probabilities. Moreover, measures such as the average variance, $\bar{\gamma}_n = \frac{1}{M_n}\sum_{i=1}^{M}\gamma_{ni}$, can be used to infer the variance for the entire aggregate ranking produced by the model.

In this setting the γ's can either be learned in combination with scores via maximum likelihood or set using some inference procedure. The generative process for MPM with both $\mathbf{S}_n$ and $\mathbf{\Gamma}_n$ parameters is shown in Figure 2(a).

2.5 Modelling Deviations from the Consensus

The assumption in MPM that the preferences generated by the Ψ experts are independent and identically distributed is likely to be false in many domains. Often one would expect to find preferences which either completely or partially deviate from the general consensus. For example in collaborative filtering most people tend to like popular movies such as *Harry Potter* and *Forrest Gump*, but in almost all cases one can find a number of outlier users who would give these movies low ratings. Assuming that the preferences of the outliers have the same distribution as the consensus, as is done in the base MPM model, can skew the aggregation especially if the outliers are severe.

To introduce the notion of outliers into our model we define an additional set of *adherence* parameters $\mathbf{\Theta} = \{\theta^1, ..., \theta^\Psi\}$, $\theta^\psi \in [0, 1]$. Here we assume that each expert ψ has its own distribution over preferences Q_n^ψ, whose adherence to the global consensus distribution Q_n (see Definition 1) is described by θ^ψ. Associated with each expert ψ is a random variable $\mathbf{B}_n^\psi \sim Q_n^\psi$, where we define Q_n^ψ as:

$$Q_n^\psi = \{P(\mathbf{B}_n^\psi = \mathbf{b}_{ij} | \mathbf{S}_n, \mathbf{\Gamma}_n, \theta^\psi)\}_{i \neq j},$$
$$P(\mathbf{B}_n^\psi = \mathbf{b}_{ij} | \mathbf{S}_n, \mathbf{\Gamma}_n, \theta^\psi) = \frac{\exp(\theta^\psi (s_{ni} - s_{nj})/(\gamma_{ni} + \gamma_{nj}))}{\sum_{k \neq l} \exp(\theta^\psi (s_{ni} - s_{nj})/(\gamma_{ni} + \gamma_{nj}))}.$$

Note that if $\theta^\psi = 0$, Q_n^ψ becomes a uniform distribution indicating that the preferences of the expert ψ deviate completely from the consensus (is an outlier), and will not be modeled by it. Values between 0 and 1 indicate different degrees of agreement, with $\theta^\psi = 1$ indicating complete agreement. Hence, by introducing θ^ψ we make the model robust, allowing it to control the extent to which each expert's preferences are modeled by the scores, effectively eliminating the outliers.

In the generative process we now assume that at each of the $T_n = \sum_\psi T_n^\psi$ draws an expert ψ is picked at random and a preference is generated from Q_n^ψ; Figure 2(b) demonstrates this process. Under this process the probability of the observed instance n with count matrices

$\mathbf{Y}_n = \{\mathbf{Y}_n^1, ..., \mathbf{Y}_n^\Psi\}$ is given by:

$$
\begin{aligned}
&P(\mathbf{Y}_n|\mathbf{S}_n, \mathbf{\Gamma}_n, \mathbf{\Theta}) = \\
&\quad = \prod_{\psi=1}^{\Psi} \left[\frac{T_n^\psi!}{\prod_{i \neq j} \mathbf{Y}_n^\psi(i,j)!} \prod_{i \neq j} P(\mathbf{B}_n^\psi = \mathbf{b}_{ij}|\mathbf{S}_n, \mathbf{\Gamma}, \theta^\psi)^{\mathbf{Y}_n^\psi(i,j)} \right] \\
&\quad = \prod_{\psi=1}^{\Psi} \left[\frac{T_n^\psi!}{\prod_{i \neq j} \mathbf{Y}_n^\psi(i,j)!} \prod_{i \neq j} \left(\frac{\exp(\theta^\psi(s_{ni} - s_{nj})/(\gamma_{ni} + \gamma_{nj}))}{\sum_{k \neq l} \exp(\theta^\psi(s_{nk} - s_{nl})/(\gamma_{nk} + \gamma_{nl}))} \right)^{\mathbf{Y}_n^\psi(i,j)} \right].
\end{aligned}
$$

The preferences are modeled by a mixture of Ψ multinomials that share the same score vector $\mathbf{S}_n$ but differ in the adherence parameter θ^ψ. Both $\mathbf{S}_n$ and $\mathbf{\Theta}$ can be efficiently learned by maximizing the log likelihood, and the consensus ranking can then be obtained by sorting the scores.

As noted above, in many preference aggregation problems the input typically consists of several preference instances n, and the goal is to infer a separate set of scores $\mathbf{S}_n$ and variances $\mathbf{\Gamma}_n$ for each instance n. The log likelihood of the entire corpus under the model is given by:

$$
\begin{aligned}
&\mathcal{L}(\{\mathbf{Y}_n\}|\{\mathbf{S}_n\}, \{\mathbf{\Gamma}_n\}, \mathbf{\Theta}) = \\
&\qquad = \log \prod_{n=1}^{N} \prod_{\psi=1}^{\Psi} \left[\frac{T_n^\psi!}{\prod_{i \neq j} \mathbf{Y}_n^\psi(i,j)!} \prod_{i \neq j} P(\mathbf{B}_n^\psi = \mathbf{b}_{ij}|\mathbf{S}_n, \mathbf{\Gamma}, \theta^\psi)^{\mathbf{Y}_n^\psi(i,j)} \right].
\end{aligned}
$$

Here $\mathbf{\Theta}$ is shared across the instances and the original MPM model is recovered by setting $\mathbf{\Theta} \equiv \mathbf{1}$. When two of the three parameters $\{\mathbf{S}_n, \mathbf{\Gamma}_n, \mathbf{\Theta}\}$ are fixed it is not difficult to show that $\mathcal{L}$ is concave with respect to the third parameter. Therefore simple gradient descent can be used to efficiently find a globally optimal setting. Furthermore, even though joint optimization is no longer convex, in the experiments we found that by using gradient descent jointly good local optimum solutions can still be found efficiently.

When training examples are available the inference proceeds as follows: first training examples are used to set $\mathbf{\Theta}$; then keeping $\mathbf{\Theta}$ fixed the scores and the variances are optimized on the test examples by maximizing the log likelihood. The advantage of this approach is that it is conceptually simple and can be applied to most extensions/generalizations of the model. The disadvantage is that it requires computing parameter gradients for every test instance which can be computationally intensive. Moreover, due to the non-convexity we do not have any guarantees on the types of solutions found by this approach since gradient optimization can converge to any local optimum. These disadvantages however are shared by most score-based aggregation models. For many of these models (aside from the simple ones) finding the maximum a posteriori score vectors is intractable so one has to resort to approximate variational or gradient-based methods that have similar complexities. We empirically found gradient-based inference to be stable provided that the model parameters are initialized with small values. Throughout all experiments we used samples from a Gaussian with mean 0 and standard deviation of 0.01 to initialize the parameters and found that the difference in results across multiple restarts was negligible.

2.6 Rank Aggregation Experiments

For rank aggregation problem we use the LETOR (Liu et al., 2007a) benchmark data sets. 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 with 69,623 documents and MQ2008-agg contains 784 queries and a total of 15,211 documents. Each query contains several lists of partial rankings of the documents under that query. There are 21 such lists in MQ2007-agg and 25 in MQ2008-agg. These are the outputs of 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 data set 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 data set is approximately 35% sparse, meaning that for an average query the partial ranking matrix of documents by search engines will be missing 35% of its entries. MQ2008-agg is significantly more sparse with the sparsity factor of approximately 65%.

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 we use standard information retrieval metrics: Normalized Discounted Cumulative Gain (N@K) (Jarvelin and Kekalainen, 2000), Precision (P@K) and Mean Average Precision (MAP) (Baeza-Yates and Ribeiro-Neto, 1999). Given an aggregate ranking π, and relevance levels $\mathbf{L}_n$, NDCG is defined as:

$$NDCG(\pi, \mathbf{L}_n)@K = \frac{1}{G(\mathbf{L}_n, K)} \sum_{i=1}^{K} \frac{2^{\mathbf{L}_n(\pi^{-1}(i))} - 1}{\log(1+i)},$$

where $\mathbf{L}_n(\pi^{-1}(i))$ is the relevance level of the document in position i in π, and $G(\mathbf{L}_n, K)$ is a normalizing constant that ensures that a perfect ordering has an NDCG value of 1. The normalizing constant allows an NDCG measure averaged over multiple instances with different numbers of items to be meaningful. Furthermore, K is a truncation constant and is generally set to a small value to emphasize the utmost importance of getting the top ranked items correct.

MAP only allows binary (relevant/not relevant) document assignments, and is defined in terms of average precision (AP):

$$AP(\pi, \mathbf{L}_n) = \frac{\sum_{i=1}^{M_n} P@i * \mathbf{L}_n(\pi^{-1}(i))}{\sum_{i=1}^{M_n} \mathbf{L}_n(\pi^{-1}(i))},$$

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

$$P@i = \sum_{j=1}^{i} \frac{\mathbf{L}_n(\pi^{-1}(j))}{i}.$$

MAP is then computed by averaging AP over all queries. To compute P@k and MAP on the MQ data sets the relevance levels are binarised with 1 converted to 0 and 2 converted

	NDCG					Precision					
	N@1	N@2	N@3	N@4	N@5	P@1	P@2	P@3	P@4	P@5	MAP
MQ2008											
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
Condorcet	35.67	37.39	39.11	40.50	41.59	40.94	37.43	34.73	32.08	29.59	42.63
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
θ-MPM	37.07	**40.29**	**41.78**	**42.76**	**43.69**	43.62	**40.94**	**37.24**	**33.64**	**30.81**	44.32
MQ2007											
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
Condorcet	37.31	37.63	38.03	38.37	38.66	43.26	42.14	40.94	39.85	38.75	42.56
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
θ-MPM	**41.13**	**41.21**	**41.09**	**41.41**	**41.53**	**47.35**	**45.78**	**44.17**	**43.01**	**41.97**	**44.35**

Table 1: MQ2008-agg and MQ2007-agg results; statistically significant results are underlined.

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.[3].

To investigate the properties of MPM we conducted extensive experiments with various versions of the model. Through these experiments we found that the θ version (see Section 2.5) had the best performance; below we refer to this model as θ-MPM. To learn this model we first used the training data to learn the adherence parameters $\mathbf{\Theta}$. Then keeping $\mathbf{\Theta}$ fixed we inferred the scores and variances on each test query via maximum likelihood and sorted the scores to produce a predicted ranking. This is similar to the framework used by the CPS model (Quin et al., 2010) where the training data is used to estimate the θ parameter. In all experiments we did not take the variances into account during the sort.

We compare the results of θ-MPM 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 data sets. We also compare with the Condorcet, Bradley-Terry and Plackett-Luce models, as well as the singular value decomposition based method SVP (Gleich and Lim, 2011). These models cover most of the primary leading approaches in unsupervised preference aggregation research. The Bradley-Terry model is fit using the same count matrices $\mathbf{Y}_n^{\psi}$ that are used for MPM.

For all models we found that 100 steps of gradient descent were enough to obtain the optimal results. To avoid constrained optimization we reparametrized the variance parameters as $\gamma_{ni} = \exp(\beta_{ni})$ and optimized β_{ni} instead. This reparametrization was done for all the reported experiments.

3. LETOR data set can be found at `http://research.microsoft.com/en-us/um/beijing/projects/letor/`.

	N@1	N@2	N@3	N@4	N@5	N@6	N@7	N@8	N@9	N@10
Ratings imputed by PMF										
Bradley-Terry	40.09	36.00	35.20	34.96	34.49	34.40	31.63	32.08	32.46	32.35
Plackett-Luce	**69.56**	54.17	48.97	46.58	44.89	43.44	42.50	41.25	40.64	40.03
Neighbor-based	61.48	49.96	44.66	42.87	40.98	39.74	39.01	37.94	37.94	37.73
MPM	69.15	**54.29**	**49.72**	**46.98**	**45.52**	**44.13**	**43.25**	**42.62**	**42.04**	**41.57**
Ratings imputed by neighbor model										
Bradley-Terry	34.66	34.07	34.09	34.11	34.02	34.22	32.73	33.14	33.47	33.48
Plackett-Luce	70.81	56.33	50.97	48.27	46.64	45.17	44.17	43.01	42.23	41.74
PMF	69.17	55.93	50.90	48.22	46.65	**45.42**	**44.58**	**43.88**	**43.22**	**42.72**
MPM	**71.90**	**56.34**	**51.21**	**48.55**	**46.73**	45.41	44.34	43.58	42.94	42.43

Table 2: NDCG results for the MovieLens data set, for each user the missing ratings are filled using the probabilistic matrix factorization model (top half), and the neighbor-based approach (bottom half); statistically significant results are underlined.

The results[4] for MPM together with the baselines on MQ2008-agg and MQ2007-agg data sets are shown in the top and bottom halves of Table 1 respectively. For each data set we conducted a paired T-test between θ-MPM and the best baseline at each of the 5 truncations for NDCG and precision as well as MAP; the statistically significant results at the 0.05 level are underlined.

From the table we see that the θ-MPM models significantly outperforms the baselines on the MQ2007-agg data set on both NDCG and MAP metrics. θ-MPM is also the best model On MQ2008-agg, significantly improving over the baselines on truncations 2-5 for NDCG and 2,3 for Precision.

2.7 Collaborative Filtering Experiments

For collaborative filtering experiments we used the MovieLens data set,[5] a collection of 100,000 ratings (1-5) from 943 users on 1682 movies. This data set was chosen because it provides demographic information such as age and occupation for each user, as well as movie information such as genre, title and release year. Each user in this data set rated at least 20 movies, but the majority of ratings for each movie are missing and the rating matrix is more than 94% sparse. We formulate the preference aggregation as follows: given users' ratings the goal is to come up with a single ranking of the movies that accurately summarizes the majority of user preferences expressed in the data. This ranking could be used as an initial recommendation for a new user who has not provided any ratings yet, as well as in a summary page. Note that the aggregation can be further personalized by only aggregating over users that share similar demographic and/or other factors with the target user.

4. All NDCG and precision values in this and other tables were multiplied by 100 to make them more readable.
5. MovieLens data set can be found at `http://www.grouplens.org/node/73`.

To convert ratings into preferences we can either sort them (resolving ties), to obtain a partial ranking for each user, or use the pairwise method to obtain the count matrices $\mathbf{Y}^{\psi}$, where $\mathbf{Y}^{\psi}(i,j) = (\ell_i^{\psi} - \ell_j^{\psi})I[\ell_i^{\psi} > \ell_j^{\psi}]$ if movies x_i and x_j were rated by user ψ and 0 otherwise. We use the sort method for the permutation-based Plackett-Luce model and use the rating difference method for the pair-based Bradley-Terry and MPM models.

In collaborative filtering and in most other applications the primary goal of aggregation is to recommend items to a new or existing user. Items ranked in the top few positions are of particular interest because they are the ones that will typically be shown to the user. Intuitively a top ranked item should have ratings from many users (*high support*) and most who rated it should prefer it to other items (*strong preference*). Consequently NDCG is a good metric to evaluate the rankers for this problem because of its emphasis on the top ranked items and the truncation level structure. Unlike rank aggregation the ground truth aggregate ratings are not available for most collaborative filtering data. To get around this problem we complete the rating matrix by imputing the missing ratings for every user. We investigate two methods of imputing the ratings: a user independent method, where all the missing ratings are filled in by the same value, and two user dependent methods. The first method is neighbor-based and predicts the ratings using the nearest neighbors for a given user ψ. The second method uses the probabilistic matrix factorization model (PMF) (Salakhutdinov and Mnih, 2008) to factorize the rating matrix and predict missing entries. Both are commonly used for CF and have shown good empirical performance on various CF tasks such as the Netflix challenge. After completing the rating matrix we compute the NDCG value for every user by sorting the items according to scores:

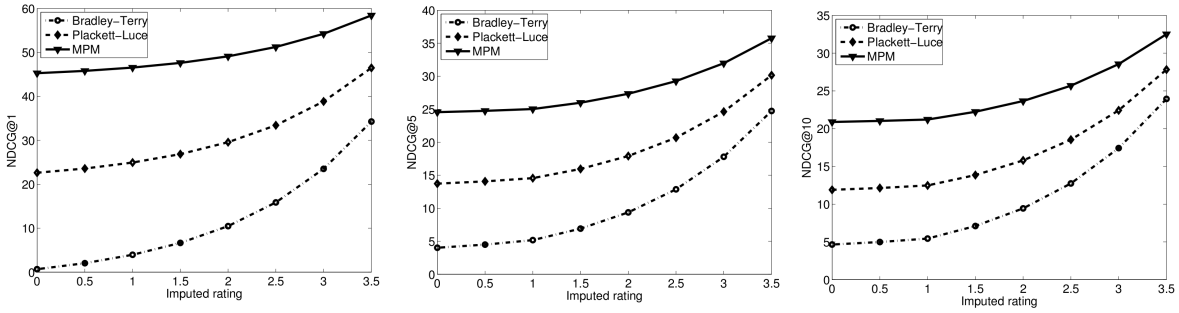
$$NDCG(\pi, \mathbf{L}^{\psi})@K = \frac{1}{G(\mathbf{L}^{\psi}, K)} \sum_{i=1}^{K} \frac{2^{\mathbf{L}^{\psi}(\pi^{-1}(i))} - 1}{\log(i+1)}. \tag{2}$$

Here π is the aggregated ranking obtained by sorting the items according to scores, and $\pi^{-1}(i)$ is the index of the item in position i in π; $\mathbf{L}^{\psi}$ is a (completed) vector of ratings for user ψ. $G(\mathbf{L}^{\psi}, K)$ is the normalizing constant and represents the maximum DCG value that could be obtained for ψ:

$$G(\mathbf{L}^{\psi}, K) = \sum_{i=1}^{K} \frac{2^{\mathbf{L}^{\psi}(\sigma^{-i}(i))} - 1}{\log(i+1)},$$

where σ is a permutation of $\mathbf{L}^{\psi}$ with the ratings sorted from largest to smallest. In this form, if an item in position i in π has a rating lower than the rating $\mathbf{L}^{\psi}(\sigma^{-1}(i))$ of the i'th highest rated item by user ψ, the corresponding term in the NDCG summation will decrease exponentially with the difference between $\mathbf{L}^{\psi}(\sigma^{-1}(i))$ and $\mathbf{L}^{\psi}(\pi^{-1}(i))$. We use this metric (averaged across all users) to evaluate the performance of the models.

We compare the results of MPM to the Bradley-Terry and Plackett-Luce models, the two best baselines on the rank aggregation task. For all models we found that 100 steps of gradient descent was enough to reach convergence. In addition to rank aggregation, we also examine CF methods directly to aggregate the items. To avoid biased evaluation in both cases we use a different CF method to aggregate the items, that is, if ratings are imputed with PMF (neighbor-based) then the neighbor-based (PMF) model is used to aggregate the items. We use Borda count to aggregate the items by first sorting the completed rating



(a) (b) (c)

Figure 3: Plots of NDCG at truncations 1, 5 and 10; in this setting all the missing ratings were repeatedly imputed by one of the constants shown on the x-axis and the rankings given by each method were evaluated using NDCG (Equation 2). All the differences are statistically significant.

vector for each user to get a user-dependent item ranking. The resulting rankings are then aggregated across all users using the Borda rule. We chose to use Borda count here because it is a well-explored approach that has stable performance and is commonly applied to social choice problems such as CF.

The NDCG results from the user dependent rating imputation method are shown in Table 2. From this table we see that MPM outperforms the best aggregation method, Plackett-Luce, both when ratings are imputed by the neighbor-based approach (top half of the table) and PMF (bottom half of the table). We also see that the neighbor-based CF model performs considerably worse than both MPM and Plackett-Luce while PMF has competitive performance, outperforming MPM at higher truncations. These results are consistent with the CF literature where PMF is typically found to perform better since it can capture more complex correlations that extend beyond simple neighbor relationships. However, unlike most aggregation methods that only learn one parameter (two for MPM) per item, in PMF we have to fit a set of parameters for each user and item. This significant increase in the number of parameters makes optimization more complex as PMF models are typically highly prone to overfitting. Moreover, it is unclear how learned models should be used to get the aggregate ranking as they only provide user-dependent rating predictions. This introduces an additional optimization step that needs to be run before ranking predictions can be made. Overall, for the MovieLens data we found that these models did not give significant gains while being considerably slower.

The NDCG plots for the user independent rating imputation method are shown in Figure 3. Here we concentrate on comparison with other aggregation methods and exclude CF methods for reasons mentioned above. The plots show NDCG at truncations 1, 5 and 10 for the three methods, when each of the values in $\{0, 0.5, 1, 1.5, 2, 2.5, 3, 3.5\}$ was used to fill the missing ratings. Here, the value 3.5 was chosen as the upper boundary because it is the average rating for the MovieLens data set. A number of studies have shown that users tend to rate items that they like so the average of the observed ratings is typically significantly

Bradley-Terry	#u	#won	#lost	Plackett-Luce	#u	#won	#lost	MPM	#u	#won	#lost
Pather Panchali	8	1431	89	Shawshank Red.	283	32592	5943	Star Wars	583	49290	10112
Wallace & Gromit	67	7448	962	Wallace & Gromit	67	7448	962	Raiders of the L.	420	40057	10644
Casablanca	243	26837	4633	Usual Suspects	267	30779	6666	Godfather	413	36531	8040
Close Shave	112	11219	1963	Star Wars	583	49290	10112	Silence of the L.	390	38192	9125
Rear Window	209	22590	4513	Wrong Trousers	118	13531	2291	Shawshank Red.	283	32592	5943
.				.				.			
.				.				.			
.				.				.			
Children of Corn	19	62	3161	Barb Wire	30	462	5507	Cable Guy	106	3469	14377
Lawnmower Man 2	21	129	3868	Robocop 3	11	125	2535	Striptease	67	1347	9909
Free Willy 3	27	171	3912	Gone Fishin'	11	123	1098	Very Brady	93	3353	12509
Kazaam	10	128	2041	Highlander III	16	881	2826	Jungle2Jungle	132	2375	11086
Best of the Best 3	6	33	1445	Ready to Wear	18	289	3785	Island of Dr.	57	1176	9415

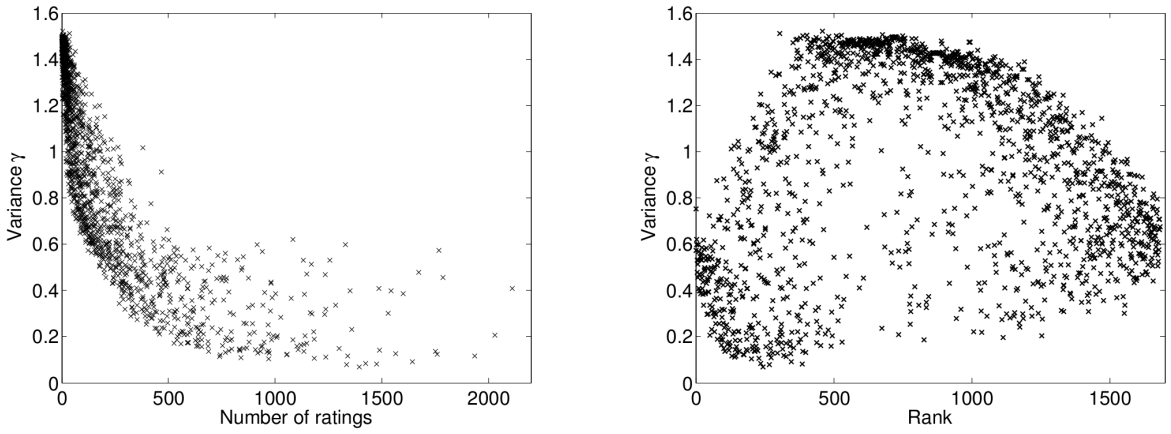
Table 3: Top 5 and bottom 5 movies found by each model. For each movie the table shows the number of users that rated it (#u) and the total number of pairwise contests that the movie won (#won) and lost (#lost) across all users.

higher than the average of the unobserved ones (Marlin et al., 2005). From the figure we see that MPM significantly outperforms both the Bradley-Terry and Plackett-Luce models. The differences are especially large when low values are imputed for the missing ratings. This indicates that the Bradley-Terry and Plackett-Luce models place items that were rated by very few users (*low support*) at the top of the list. This causes the imputed ratings to dominate the numerator in the NDCG summation making the results very sensitive to the magnitude of the imputed rating.

This effect can also be observed from Table 3, which shows the top and bottom 5 movies generated by each model together with statistics on the number of users that rated each movie and the number of pairwise contests lost and won by the movie (summed across all users). For a given user ψ and movie i with rating ℓ_i^ψ we find the number of pairwise wins by counting the number of pairs (i, j) with $\ell_i^\psi > \ell_j^\psi$; losses are found in a similar way. From the table we see that the Bradley-Terry model places the movie *Pather Panchali* at the top of the list. This movie is only rated by 8 out of 943 users and even though most users who rated it preferred it to other movies (#lost is low) there is still very little evidence that this movie represents the top preference for the majority of users. Due to its pairwise independence assumption the Bradley-Terry model is always likely to place movies with few ratings near the top/bottom of the list.

The Plackett-Luce model partially fixes this problem by considering items jointly, and places the frequently rated movie *Shawshank Redemption* first. However the model does not fully eliminate the problem, placing the very infrequently rated *Wallace & Gromit* (also ranked second by Bradley-Terry) in the second spot. Part of the reason for this comes from the fact that Plackett-Luce is a permutation based model and as such cannot model the strength of preferences, treating the preferences given for example by ratings $\{5, 2, 1\}$ the same as $\{5, 4, 3\}$.

On the other hand for the Multinomial Preference Model we see that the position of the item is related to both the number of observed preferences and the strength of those



(a) (b)

Figure 4: 4(a) shows the number of ratings versus the learned variance γ_i for each movie x_i. 4(b) shows the rank for each movie obtained after sorting the scores versus the learned γ_i.

preference. The top three movies are all rated by more than 400 users and are strongly preferred by the majority of those users.

A more severe pattern can be observed for the bottom 5 movies. Both Bradley-Terry and Plackett-Luce place movies rated by fewer than 30 users in the bottom 5 positions, labeling them the worst movies in the entire data set. This selection has very little evidence in the data and has a high probability of being wrong if more ratings are collected. For MPM all of the bottom 5 movies are rated by more than 50 users with 3 out of 5 movies rated by more than 90 users.

In addition to the retrieval accuracy we investigated the properties of the learned variance parameters γ. Figure 4(a) shows the learned variances together with the number of ratings for each movie. Note that the variance is inversely proportional to the number of ratings, so as the number of ratings increases the model becomes increasingly more certain in the preferences, decreasing the variance. In Figure 4(b) we plot γ against the aggregate rank for each movie. The general pattern is clear: the variance decreases towards the extremes of the ranking, indicating that the model is more certain in the movies that are placed near the top and near the bottom of the aggregate ranking. As shown above, this is due to the fact that the movies at the extremes of the ranking have many comparisons, allowing accurate inference of strong negative or positive preferences.

The plot however also shows outliers, which are the movies placed in the middle of the aggregate ranking with low variance/high confidence. After further inspection we found that each such movie had many positive as well as negative preferences. Examples of these include *Sabrina* (#u:190 #won:10190 #lost:12347), *Mrs. Doubtfire* (#u:192 #won:13251 #lost:17551) and *Ghost* (#u:170 #won:11785 #lost:14452). Note that all three movies were rated by more than 150 users and overall were neither strongly preferred nor strongly

disliked. The model thus appropriately placed them in the middle of the ranking with strong confidence. Moreover, note that it is impossible to express this confidence with scores alone since all the movies in the middle of the ranking have similar scores. The variances thus provide additional information about the decisions made by the model during the aggregation, which could be very useful for post-processing and evaluation.

3. Supervised Preference Aggregation

Research in preference aggregation has largely concentrated on the unsupervised aggregation problem described above. However, many of the recent aggregation problems are amenable to supervised learning, as ground truth preference information is available. The meta-search and crowdsourcing problems are both examples of supervised preference aggregation. Due to the popularity of these and other supervised problems the supervised aggregation framework has received a lot of attention recently, with a number of competitions conducted in TREC[6] as well as other conferences, and several meta-search data sets(Liu et al., 2007a) have been released to encourage research in the area. Despite these efforts, to the best of our knowledge most of the proposed models are still unsupervised and the supervised methods are unable to fully use the labeled data and optimize the aggregating function for the target metric. There is thus an evident need to develop an effective supervised aggregation framework.

To address this problem we first develop a supervised extension of the MPM model introduced in the previous section. We show how the labeled training data can be used to set the adherence parameters $\mathbf{\Theta}$ and experimentally verify that this approach improves the test accuracy of the model.

We then develop a general framework for supervised preference aggregation. Our framework builds on the idea of converting the observed preferences into pairwise matrices described in the previous section. We first show how these matrices can be used to derive effective fixed length item representations that make it possible to apply any learning-to-rank method to optimize the parameters of the aggregating function for the target IR metric. We then show how the pairwise matrices can also be employed as potentials in a ranking Conditional Random Field (CRF), and develop efficient learning and inference procedure to optimize the CRF for the target IR metric.

We validate all of the introduced models on two supervised rank aggregation data sets from Microsoft's LETOR4.0 data collection.

3.1 Framework

As in unsupervised preference aggregation, a typical supervised problem also consists of training instances where for each instance we are given a set of items. The experts generate preferences for the items and the preferences can be in the variety of forms ranging from full/partial rankings and ratings to relative item comparisons and combinations of these. However, unlike the unsupervised problem, we now also have access to the ground truth preference information over the items for each instance. The goal is to learn an aggregating function which maps expert preferences to an aggregate ranking that maximally agrees with the ground truth preferences.

6. TREC 2013 crowdsourcing track can be found at `https://sites.google.com/site/treccrowd/`.

	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 5: (a) An example rank matrix $\mathbf{R}$ 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 $\mathbf{R}$. (b) The resulting pairwise matrix $\mathbf{Y}^2$ for expert 2 (second column of $\mathbf{R}$) after the ranks are transformed to pairwise preferences using the log rank difference method.

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 an $M_n \times \Psi$ rank matrix $\mathbf{R}_n$ where $\mathbf{R}_n(i, \psi)$ denotes the rank assigned to document d_{ni} by the expert ψ. Note that the same Ψ experts rank items for all the queries so Ψ is query independent. Furthermore, $\mathbf{R}_n$ can be sparse, as experts might not assign ranks to every document in $\mathbf{D}_n$; we use $\mathbf{R}_n(i, \psi) = 0$ to indicate that document d_{ni} was not ranked by expert ψ. The sparsity arises in problems like meta-search where q_n is sent to 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 $\mathbf{L}_n = \{\ell_{n1}, ..., \ell_{nM_n}\}$ which are typically assigned to the documents by human annotators.

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 use the ideas behind the MPM approach and convert the rank matrix into a pairwise preference matrix. We then show how this conversion can be used to develop an effective supervised framework for this problem.

3.2 Pairwise Preferences

Given the $M_n \times \Psi$ ranking matrix $\mathbf{R}_n$ our aim is to convert it into Ψ $M_n \times M_n$ pairwise matrices $\mathbf{Y}_n = \{\mathbf{Y}_n^1, ..., \mathbf{Y}_n^\Psi\}$, where each $\mathbf{Y}_n^\psi$ expresses the preferences between pairs of documents based on the expert ψ. In Section 2.1 we considered binary and count-based

transformations; here we extend these ideas and consider a general transformation of the form $\mathbf{Y}_n^{\psi}(i,j) = g(\mathbf{R}_n(i,\psi), \mathbf{R}_n(j,\psi))$. We experiment with three versions for g that were proposed by Gleich and Lim (2011): Binary Comparison (Equation 1), and normalized and logarithmic versions of Rank Difference (Equation 2).

3. **Normalized Rank Difference**

   Here the normalized rank difference is used:

$$\mathbf{Y}_n^{\psi}(i,j) = I[\mathbf{R}_n(i,\psi) < \mathbf{R}_n(j,\psi)] \frac{\mathbf{R}_n(j,\psi) - \mathbf{R}_n(i,\psi)}{\max(\mathbf{R}_n(:,\psi))}.$$

   Normalizing by the maximum rank assigned by the expert ψ ($\max(\mathbf{R}_n(:,\psi))$) ensures that the entries of $\mathbf{Y}_n^{\psi}$ have comparable ranges across experts.

4. **Log Rank Difference**

   This method uses the normalized log rank difference:

$$\mathbf{Y}_n^{\psi}(i,j) = I[\mathbf{R}_n(i,\psi) < \mathbf{R}_n(j,\psi)] \frac{\log(\mathbf{R}_n(j,\psi)) - \log(\mathbf{R}_n(i,\psi))}{\log(\max(\mathbf{R}_n(:,\psi)))}.$$

In all cases both $\mathbf{Y}_n^{\psi}(i,j)$ and $\mathbf{Y}_n^{\psi}(j,i)$ are set to 0 if either $\mathbf{R}_n(i,\psi) = 0$ or $\mathbf{R}_n(j,\psi) = 0$ (missing ranking). Non-zero entries $\mathbf{Y}_n^{\psi}(i,j)$ represent the strength of the pairwise preference $\{d_{ni} \succ d_{nj}\}$ expressed by expert ψ. Figure 5 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-K lists can easily be converted into $\mathbf{Y}_n^{\psi}$ using the same transformations. Moreover, if pairwise preferences are observed, we simply skip the transformation step and fill the entries $\mathbf{Y}_n^{\psi}(i,j)$ directly.

As mentioned above, working with pairwise comparisons has a number of advantages, and models over pairwise preferences have been extensively used in areas such as social choice (David, 1988), information retrieval (Joachims, 2002; Burges, 2010), and collaborative filtering (Lu and Boutilier, 2011; Gleich and Lim, 2011). 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.

3.3 Previous Work

The majority of research on preference aggregation has concentrated on the unsupervised problem and was covered in detail in Section 2.2. In this section we review the supervised approaches for preference aggregation.

In meta-search a heuristic method called Reciprocal Rank Fusion (RRF) (Cormack et al., 2009) has recently been proposed. RRF uses the rank matrix to derive the document scores:

$$s_{ni} = \sum_{\psi=1}^{\Psi} \frac{1}{\alpha + \mathbf{R}_n(i,\psi)},$$

where α is a constant which mitigates the impact of high rankings from outlier experts and is set by cross validation. Once the scores are computed they are used to sort the documents. We place this method into the supervised category because of the need to set the constant α which has a significant effect on ranking accuracy (Cormack et al., 2009). Despite its simplicity RRF has obtained excellent empirical accuracy; however it also has some disadvantages. First, RRF relies on complete rankings and it is unclear how to extend it to problems like meta-search where many rankings are typically missing. Second, this method is designed specifically for rank aggregation and cannot be applied to other types of preferences.

A supervised score-based rank aggregation approach based on a Markov Chain was also recently introduced by Liu et al. (2007b). 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. The pairwise constraint idea was also recently extended by Chen et al. (2011) to a semi-supervised setting where the ground truth preferences are available for only a subset of the documents.

Extensive work in the learning-to-rank domain has demonstrated that optimizing the ranking function for the target metric produces significant gains in test accuracy (Li, 2011). However, to the best of our knowledge none of the models developed for supervised preference aggregation take the target metric into account during the optimization of the aggregating function. The models that we develop in the following sections aim to bridge this gap.

3.4 Supervised Extension for MPM

Before delving into the new models we first develop a simple supervised extension for the Multinomial Preference Model introduced above. The MPM can be considered fully unsupervised, as the adherence parameters $\mathbf{\Theta}$, the consensus scores and the variances are inferred from the observed preferences. This produces a predicted ranking for a given set of observed preferences by sorting the inferred scores, without ever using any known consensus rankings or relevance labels in the data. In this section we describe an approach to incorporate this ground truth information into this model.

Each θ^{ψ} models the adherence of the expert ψ to the consensus. For the labeled training instances the consensus is explicitly given by the relevance levels $\mathbf{L}_n$. This allows us to evaluate the adherence of each expert to the consensus exactly by computing the match between the preferences given by the expert and those expressed by the ground truth relevances. Using this we can set θ^{ψ} to the average distance between the preferences of the

expert ψ and the ground truth labels across the instances:

$$\theta^{\psi} = \frac{1}{N}\sum_{n=1}^{N} 1 - \mathcal{D}(\mathbf{L}_n, \mathbf{Y}_n^{\psi}),$$

where $\mathcal{D}$ is a normalized distance metric between preferences, such as Kendall's τ. Note that $\theta^{\psi} \to 1(\to 0)$ indicates that the preferences of expert ψ agree with (deviate from) the ground truth preferences (target consensus) across the training examples.

To apply the model we now (1) use the labeled training instances to find $\mathbf{\Theta}$ and (2) keeping $\mathbf{\Theta}$ fixed infer scores and variances for each test instance by maximizing the likelihood of the observed preferences.

3.5 Feature-Based Approach

We now introduce the first of the two fully supervised models for preference aggregation. The main idea behind this approach is to summarize the relative preferences for each document across the experts by a fixed length feature vector (Volkovs et al., 2012). 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 following section we describe an approach to extract the document features.

3.5.1 Feature-Based Approach: Feature Extraction

Given the rank matrix $\mathbf{R}_n$ and the resulting pairwise matrix $\mathbf{Y}_n^{\psi}$ for expert ψ (as shown in Figure 5), our aim is to convert $\mathbf{Y}_n^{\psi}$ into a fixed length feature vector for each of the documents in $\mathbf{D}_n$. Singular Value Decomposition (SVD) based approaches for document summarization such as Latent Semantic Indexing (Deerwester et al., 1990) 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 $M_n \times M_n$ pairwise matrix $\mathbf{Y}_n^{\psi}$ the rank-p SVD factorization has the form:

$$\mathbf{Y}_n^{\psi} \approx \mathbf{U}_n^{\psi}\mathbf{\Sigma}_n^{\psi}\mathbf{V}_n^{\psi},$$

where $\mathbf{U}_n^{\psi}$ is an $M_n \times p$ matrix, $\mathbf{\Sigma}_n^{\psi}$ is an $p \times p$ diagonal matrix of singular values and $\mathbf{V}_n^{\psi}$ is an $M_n \times p$ matrix. The full SVD factorization has $p = M_n$, however, to reduce the noise and other undesirable artifacts of the original space most applications that use SVD typically set $p \ll M_n$. 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_n.

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

$$\phi(d_{ni}, \mathbf{Y}_n^{\psi}) = [\mathbf{U}_n^{\psi}(i,:), \mathrm{diag}(\mathbf{\Sigma}_n^{\psi}), \mathbf{V}_n^{\psi}(i,:)],$$

here $\mathbf{U}_n^{\psi}(i,:)$ and $\mathbf{V}_n^{\psi}(i,:)$ are the i'th rows of $\mathbf{U}_n^{\psi}$ and $\mathbf{V}_n^{\psi}$ respectively and $\mathrm{diag}(\mathbf{\Sigma}_n^{\psi})$ is the main diagonal of $\mathbf{\Sigma}_n^{\psi}$ represented as a $1 \times p$ vector. Note that the $\mathrm{diag}(\mathbf{\Sigma}_n^{\psi})$ component is

independent of i and will be the same for all the documents in $\mathbf{D}_n$. We include the singular values to preserve as much information from the SVD factorization as possible. The features $\phi(d_{ni}, \mathbf{Y}_n^\psi)$ summarize the relative preference information for d_{ni} expressed by the expert ϕ. To get a complete view across the experts we concatenate together the features extracted for each expert:

$$\phi(d_{ni}) = [\phi(d_{ni}, \mathbf{Y}_n^1), ..., \phi(d_{ni}, \mathbf{Y}_n^\Psi)].$$

Each $\phi(d_{ni}, \mathbf{Y}_n^\psi)$ contains $3p$ features so the entire representation will have $3\Psi p$ features. Moreover, note that since the experts 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_{ni}, effectively transforming the aggregation problem into a standard learning-to-rank one. During training our aim is now to learn a scoring function $f : \mathbb{R}^{3\Psi p} \rightarrow \mathbb{R}$ which maximizes the target IR metric such as NDCG. The feature representation allows us to fully use all of 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.

It is worth mentioning here that the SVD factorization of pairwise matrices has been used in the context of preference aggregation (see (Gleich and Lim, 2011) for example). However, previous approaches largely concentrated on applying SVD to fill the missing entries in the joint pairwise matrix $\mathbf{Y}_n^{tot} = \sum_{\psi=1}^{\Psi} \mathbf{Y}_n^\psi$ and then use the completed matrix to infer the aggregate ranking. Our approach on the other hand uses SVD to compress each pairwise matrix $\mathbf{Y}_n^\psi$ and produce fixed length feature vector for each document.

3.5.2 Feature-Based Approach: Learning and Inference

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_{ni} is not ranked by the expert ψ the row $\mathbf{Y}_n^\psi(i,:)$ and column $\mathbf{Y}_n^\psi(:,i)$ 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 Ψ experts:

$$f(\phi(d_{ni}), \mathbf{W}) = \sum_{\psi \in \Psi} \mathbf{w}^\psi \cdot \phi(d_{ni}, \mathbf{Y}_n^\psi) + I[\mathbf{R}_n(i,\psi) = 0] b^\psi, \tag{3}$$

where $\mathbf{W} = \{\mathbf{w}^\psi, b^\psi\}_\psi$ is the set of free parameters to be learned with each $\mathbf{w}^\psi$ having the same dimension as $\phi(d_{ni}, \mathbf{Y}_n^\psi)$, and $I[]$ is an indicator function. The bias term b^ψ provides a base score for d_{ni} if d_{ni} is not ranked by expert ψ. The weights $\mathbf{w}^\psi$ control how much emphasis is given to preferences from expert ψ. It is important to note here that the scoring function can easily be made non-linear by adding additional hidden layer(s), as in conventional multilayer neural nets. In the form given by Equation 3 our model has a total of $(3p+1)\Psi$ parameters to be learned. We can use any of the developed learning-to-rank approaches (see Li, 2011 for a detailed description of many popular learning-to-rank methods) to optimize $\mathbf{W}$; in this work we chose to use the LambdaRank method. We chose LambdaRank because it has shown excellent empirical performance, for example, winning the Yahoo! Learning To Rank Challenge (Chapelle et al., 2010). We briefly describe

Algorithm 1 Feature-Based 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
for $n = 1$ **to** N **do** {feature extraction}
  from $\mathbf{R}_n$ compute $\mathbf{Y}_n = \{\mathbf{Y}_n^1, ..., \mathbf{Y}_n^\Psi\}$
  for $i = 1$ **to** M_n **do**
    compute features $\phi(d_{ni})$ using SVD
  end for
end for
initialize weights: $\mathbf{W}$
repeat {scoring function optimization}
  for $n = 1$ **to** N **do**
    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: $\mathbf{W}$

LambdaRank here and refer the reader to Burges et al. (2006) and Burges (2010) 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_{ni}, d_{nj}) with $\ell_{ni} \neq \ell_{nj}$, the target probability that d_{ni} should be ranked higher than d_{nj} is defined as:

$$P_{nij} = \begin{cases} 1 & \text{if } \ell_{ni} > \ell_{nj} \\ 0 & \text{otherwise} \end{cases} .$$

The model's probability is then obtained by passing the difference in scores between d_{ni} and d_{nj} through a logistic:

$$\begin{aligned} Q_{nij} &= \frac{1}{1 + \exp(-(f(\phi(d_{ni}), \mathbf{W}) - f(\phi(d_{nj}), \mathbf{W})))} \\ &= \frac{1}{1 + \exp(-(s_{ni} - s_{nj}))}. \end{aligned}$$

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:

$$O_n = -\sum_{\ell_{ni} > \ell_{nj}} P_{nij} \log(Q_{nij}).$$

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

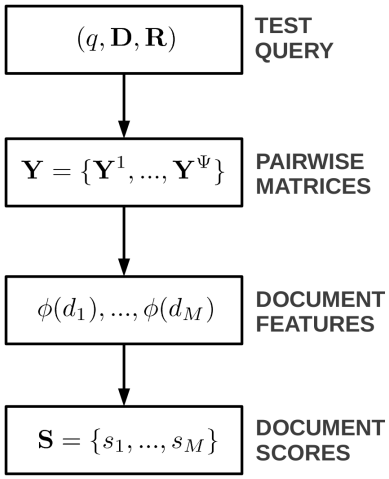


Figure 6: The inference diagram for the feature-based preference aggregation approach. Given a test query $(q, \mathbf{D}, \mathbf{R})$ the inference proceeds in three steps: (1) The ranking matrix $\mathbf{R}$ is converted to a set of pairwise matrices $\mathbf{Y} = \{\mathbf{Y}^1, ..., \mathbf{Y}^\Psi\}$. (2) SVD is used to extract document features from each pairwise matrix $\mathbf{Y}^\psi$. (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.

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_{ni}, 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:

$$\lambda_{nij} = |\Delta NDCG@K(s_{ni}, s_{nj})| \frac{\partial O_n}{\partial(s_{ni} - s_{nj})}.$$

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 Burges et al., 2006 for more details on λ calculation). The λ-gradient for document d_{ni} is computed by summing the λ's for all pairs of documents (d_{ni}, d_{nj}) for query q_n:

$$\lambda_{ni} = \sum_{j:\ell_{nj} \neq \ell_{ni}} \lambda_{nij}.$$

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 $O(M_n \Psi p)$ for each query q_n. The complete stochastic gradient descent learning procedure is summarized in Algorithm 1.

Once the parameters of the scoring function are learned then at test time, given a new query q with rank matrix $\mathbf{R}$, we (1) convert $\mathbf{R}$ into a set of pairwise matrices $\mathbf{Y} = \{\mathbf{Y}^1, ..., \mathbf{Y}^\Psi\}$; (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 6.

3.6 CRF Approach

The SVD-based feature approach introduced above is effective at producing compact document representation which we experimentally found to work well for this task. These representations also allow to apply any learning-to-rank approach making it very easy to incorporate the model into many existing IR frameworks. However, while the SVD model is robust, it requires applying SVD factorization at test time which can be computationally intensive. Moreover, SVD representation significantly compresses the pairwise matrices and might throw away useful information potentially affecting performance and making the model less interpretable. To avoid these disadvantages we develop another fully supervised model which uses the pairwise matrices directly. As mentioned above, the variable number of documents per query and absence of the fixed-length document representation make it difficult to apply the majority of supervised methods to this problem, since they require fixed-length item representations. CRFs 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 (Sha and Pereira, 2003; Roth and Yih, 2005), computational biology (Sato and Sakakibara, 2005; Liu et al., 2006), and information retrieval (Qin et al., 2008; Volkovs and Zemel, 2009). Moreover, CRFs are very flexible and can be used optimize the parameters of the model for the target metric. For these reasons we develop a CRF framework for preference aggregation (Volkovs and Zemel, 2013).

3.6.1 CRF Approach: Model

The main idea behind this 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 π by comparing the order induced by the ranking with the pairwise preferences from each expert. This leads to an energy function:

$$E(\pi, \mathbf{Y}_n; \mathbf{W}) = \\ = -\frac{1}{M_n^2} \sum_{i=1}^{M_n} \frac{\sum_{\psi=1}^{\Psi} \left(b^\psi \varphi^\psi(\pi^{-1}(i)) + w_{pos}^\psi \sum_{j \neq i} \mathbf{Y}_n^\psi(\pi^{-1}(i), j) - w_{neg}^\psi \sum_{j \neq i} \mathbf{Y}_n^\psi(j, \pi^{-1}(i)) \right)}{\log(i+1)}, \quad (4)$$

where $\mathbf{Y}_n^\psi(\pi^{-1}(i), j)$ is the pairwise preference for the document in position i in π over document d_{nj} expressed by expert ψ, and $\mathbf{W} = \{b^\psi, w_{pos}^\psi, w_{neg}^\psi\}_{\psi=1}^\Psi$ is the set of free parameters to be learned.

This energy function contains a binary unary potential $\varphi^{\psi}(i, \mathbf{R}_n) = I[\mathbf{R}_n(i, \psi) = 0]$, where $I[]$ is an indicator function. This potential is active only when document d_{ni} is not ranked by the expert ψ, in which case b^{ψ} provides a base preference score for the item. The energy also contains pairwise potentials that directly use pairwise matrices $\mathbf{Y}$. Note that from the definition of $\mathbf{Y}$ in Section 3.2 it follows that only one of $\mathbf{Y}_n^{\psi}(\pi^{-1}(i), j)$ or $\mathbf{Y}_n^{\psi}(j, \pi^{-1}(i))$ can be non-zero for any pair of documents. Consequently, if $\mathbf{Y}_n^{\psi}(\pi^{-1}(i), j)$ is on (non-zero) then expert ψ "agrees" with the relative order induced by π (lowering the energy) and the strength of this agreement is given by w_{pos}^{ψ}. Similarly, if $\mathbf{Y}_n^{\psi}(j, \pi^{-1}(i))$ is on then expert ψ "disagrees" with the relative order, and raises the energy. The weights w_{pos}^{ψ} and w_{neg}^{ψ} thus control how much emphasis is given to positive and negative relative preferences from expert ψ. $1/\log(i+1)$ 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.

The model assigns a probability to every ranking π based on this energy function:

$$P(\pi|\mathbf{Y}_n) = \frac{1}{Z(\mathbf{Y}_n)} \exp(-E(\pi, \mathbf{Y}_n; \mathbf{W})) \qquad Z(\mathbf{Y}_n) = \sum_{\pi} \exp(-E(\pi, \mathbf{Y}_n; \mathbf{W})),$$

where the partition function $Z(\mathbf{Y}_n)$ sums over all valid rankings π.

It is important to note here that in the proposed model a separate set of weights $\{b^{\psi}, w_{pos}^{\psi}, w_{neg}^{\psi}\}$ is learned for each expert ψ, which allows the model to effectively capture the correlations between individual expert preferences and the ground truth ones. However, this framework 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 b, w_{pos} and w_{neg}, removing the dependence on ψ. The resulting *consensus* model only takes into account the net preference across all Ψ 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 $\{b^{\psi}, w_{pos}^{\psi}, w_{neg}^{\psi}\}$ for the known experts ψ, and consensus-based weights $\{b, w_{pos}, w_{neg}\}$ 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.

3.6.2 CRF Approach: Learning and Inference

The most popular approach to training CRFs is maximum likelihood. However, this approach does not take the target metric into account and thus might be ineffective at optimizing it. Recent work has explored different empirical and theoretical approaches to incorporate the target metric during CRF training (Volkovs and Zemel, 2009; Gimpel and Smith, 2010; McAllester and Keshet, 2011) Inspired by this work we follow the approach of

Algorithm 2 CRF 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, cut-off ϵ
for $n = 1$ **to** N **do** {pairwise matrices}
  from $\mathbf{R}_n$ compute $\mathbf{Y}_n = \{\mathbf{Y}_n^1, ..., \mathbf{Y}_n^\Psi\}$
end for
initialize weights: $\mathbf{W}$
repeat {CRF optimization}
  for $n = 1$ **to** N **do**
    if $M_n > \epsilon$ **then**
      downsample documents to get $\mathbf{L}_{n\epsilon}$, $\mathbf{Y}_{n\epsilon}$
    else
      $\mathbf{L}_{n\epsilon} = \mathbf{L}_n$ and $\mathbf{Y}_{n\epsilon} = \mathbf{Y}_n$
    end if
    compute exact gradients with $\{\mathbf{L}_{n\epsilon}, \mathbf{Y}_{n\epsilon}\}$:
      $\nabla\mathbf{W} = \partial O(\mathbf{L}_{n\epsilon}, \mathbf{Y}_{n\epsilon})/\partial\mathbf{W}$
    update weights: $\mathbf{W} = \mathbf{W} + \eta\nabla\mathbf{W}$
  end for
until convergence
Output: $\mathbf{W}$

Volkovs and Zemel (2009) and use the expected metric as the target objective to maximize:

$$O(\mathbf{L}_n, \mathbf{Y}_n) = \sum_\pi \mathcal{G}(\pi, \mathbf{L}_n) P(\pi|\mathbf{Y}_n), \tag{5}$$

where $\mathcal{G}$ can be any IR metric such as the NDCG or MAP. Note that even even the cases where $\mathcal{G}$ is non-smooth (e.g., NDCG, MAP) the above objective remains smooth with respect to $\mathbf{W}$ and can be maximized using standard gradient-based procedure. However, to optimize this objective we need to calculate $P(\pi|\mathbf{Y}_n)$ for all $M_n!$ rankings π of the items in each query q_n. This computation 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 avoid these problems we use an approach similar to the one suggested by Caetano et al. (2009) which we empirically found to work well. Every time a query q_n is visited and the number of documents is greater than ϵ, we randomly select a subset of ϵ documents and use the corresponding relevance labels $\mathbf{L}_{n\epsilon}$ and pairwise matrices $\mathbf{Y}_{n\epsilon} = \{\mathbf{Y}_{n\epsilon}^1, ..., \mathbf{Y}_{n\epsilon}^\Psi\}$ to compute the gradients for $\mathbf{W}$. Here each pairwise matrix $\mathbf{Y}_{n\epsilon}^\psi$ is a slice of the original matrix $\mathbf{Y}_n^\psi$ which includes only the selected ϵ documents. Choosing ϵ sufficiently small allows the gradients to be computed exactly by enumerating all possible $\epsilon!$ permutations of the documents.

Similarly to the feature-based model, this learning procedure can be made more efficient by precomputing both unary and pairwise potentials. This reduces the complexity of computing the model's energy from $O(M_n^2\Psi)$ to $O(M_n\Psi)$ at the cost of additional storage

requirement of $O(M_n)$ per query. The complete stochastic gradient descent learning procedure is summarized in Algorithm 2. Note that this algorithm can be used to optimize the parameters of the CRF for *any* of the target IR metrics.

Once the CRF is learned then given a new test query $(q, \mathbf{D}, \mathbf{R})$, the goal is to predict a single ranking π for the M documents in $\mathbf{D}$ based on the expert preferences $\mathbf{R}$. Fortunately, once $\mathbf{R}$ is converted into $\mathbf{Y} = \{\mathbf{Y}^1, ..., \mathbf{Y}^\Psi\}$ such inference can be done very efficiently in this CRF. Since $1/\log(i+1)$ is a monotonically decreasing function it is easy to verify that the ranking with the highest probability is obtained by sorting the items according to their total "score" given by the potentials:

$$\hat{\pi} = \arg\min_{\pi} E(\pi, \mathbf{Y}; \mathbf{W}) = \arg\operatorname{sort}([\vartheta_1, ..., \vartheta_M])$$

where the scores are:

$$\vartheta_i = -\sum_{\psi=1}^{\Psi} \left(b^\psi \varphi^\psi(i) + w^\psi_{pos} \sum_{j \neq i} \mathbf{Y}^\psi(i,j) - w^\psi_{neg} \sum_{j \neq i} \mathbf{Y}^\psi(j,i) \right).$$

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 the feature-based approach which requires SVD factorization to be done for every test query.

3.7 Experiments

For our experiments we use the MQ2007-agg and MQ2008-agg rank aggregation data sets from the LETOR4.0 (Liu et al., 2007a) collection. Detailed description of the data sets is given is Section 2.6.

3.7.1 MPM Experiments

In the first set of experiments we compare the performances of the supervised and the unsupervised versions of the MPM model. Both versions are fit using the pairwise count matrix described in Section 2.1. Note that this method of converting rankings to pairwise preferences is very similar to the rank difference method discussed in Section 3.2 with the only difference being the exclusion of the normalization term $\max(\mathbf{R}_n(:, \psi))$. As before we use 100 steps of the gradient descent to fit the models and reparametrize the variance parameters as $\gamma_{ni} = \exp(\beta_{ni})$ to avoid constrained optimization. The results for the unsupervised (θ-MPM) and the supervised (θ_{sup}-MPM) models on both data sets are shown in Table 4. From the table we see that the supervised version of the MPM model significantly outperforms the unsupervised one on both data sets. This supports the expected conclusion that when ground truth preference data is available using it during model training improves performance.

To further investigate the differences between the two MPM models we plotted the adherence parameters $\mathbf{\Theta}$ found by each model. Figure 7 shows the adherence parameters $\mathbf{\Theta}$ set based on the labeled training examples, together with the one learned in an unsupervised fashion by doing gradient descent. From the figure we see many similarities in the two vectors, indicating that the model is able to capture the notion of "outliers" which correlates

	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											
θ-MPM	37.07	40.29	41.78	42.76	43.69	43.62	40.94	37.24	33.64	30.81	44.32
θ_{sup}-MPM	**38.17**	**40.57**	**42.19**	**43.07**	**43.99**	**44.89**	**41.13**	**37.67**	**33.80**	**31.17**	**44.71**
MQ2007-agg											
θ-MPM	41.13	41.21	41.09	41.41	42.53	47.35	45.78	44.17	43.01	41.97	44.35
θ_{sup}-MPM	**41.77**	**41.91**	**41.92**	**42.34**	**42.79**	**48.35**	**46.64**	**44.53**	**43.52**	**42.72**	**45.71**

Table 4: MQ2008-agg and MQ2007-agg MPM results; statistically significant results are underlined.

Figure 7: Top row: normalized $\mathbf{\Theta}$, found by the supervised procedure outlined in Section 3.4, for training Fold 1 of MQ2007-agg. Bottom row: learned $\mathbf{\Theta}$ on the same Fold. Here white = 1 and black = 0.

closely with the training labels. There are however a number of differences, such as the first three components being switched from on to off in the learned $\mathbf{\Theta}$. In our experiments we consistently found that setting $\mathbf{\Theta}$ using the training labels produced better performance.

3.7.2 SVD and CRF Experiments

In the second set of experiments we compare the performance of the SVD-based feature model trained with LambdaRank (denoted as SVDsup) and the CRF one. For each model we experimented with binary, rank difference, and log rank difference methods to compute the pairwise matrices (see Section 3.2) and selected the method that gave the best validation NDCG@10 results. For the SVD-based model we found through cross-validation that setting $p = 1$ (SVD rank) gave the best performance which is expected considering the sparsity level of the pairwise matrices. The LambdaRank training of the scoring function was run for 200 iterations with a learning rate of 0.01, and validation NDCG@10 was used to choose the best model. For the CRF model we used expected NDCG (see Equation 5) as the target objective and set $\epsilon = 6$ ensuring that at least one document of every relevance label was chosen each time.

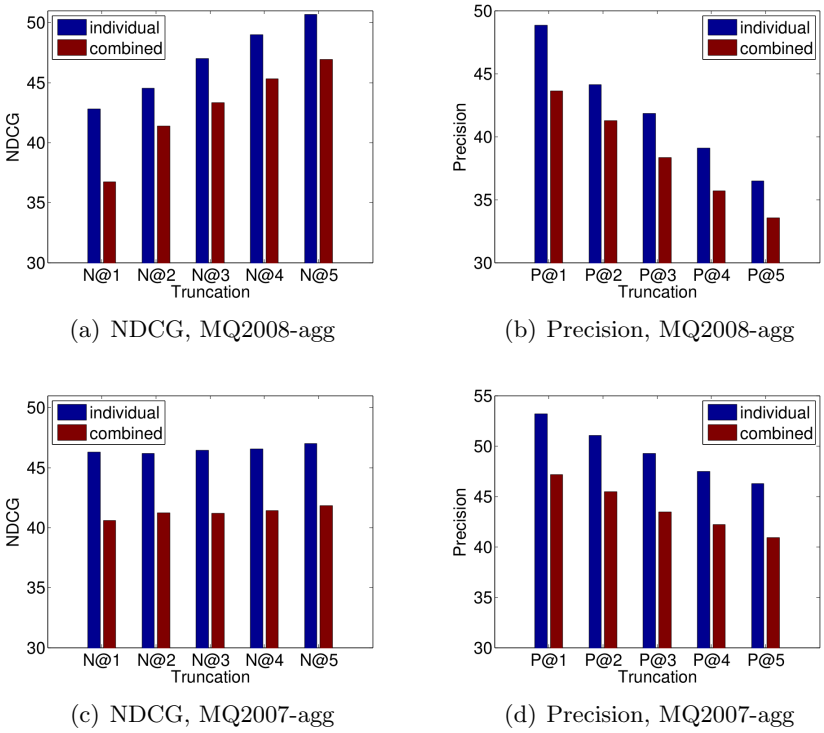
We compare the results of our model against the unsupervised baselines used in the MPM experiments (see Section 2.6). We also compare with the established meta-search standard Reciprocal Rank Fusion (RRF). To the best of our knowledge these models cover all of the primary leading approaches in the (supervised) rank aggregation research except for the Markov Chain model (Liu et al., 2007b) discussed above. 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 difficult.

	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
θ_{sup}-MPM	38.17	40.57	42.19	43.07	43.99	44.89	41.13	37.67	33.80	31.17	44.71
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											
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
θ_{sup}-MPM	41.77	41.91	41.92	42.34	42.79	48.35	46.64	44.53	43.52	42.72	45.71
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**

Table 5: MQ2008-agg and MQ2007-agg results; statistically significant differences between CRF and SVDsup are underlined. All the differences between both CRF and SVDsup models and the best baseline are statistically significant.

The NDCG and Precision results for truncations 1-5 as well as MAP results are shown in Table 5. From the table we see that both SVDsup and CRF model significantly outperform all the other aggregation methods, improving by as much as 5 NDCG points over the best baseline on each of the MQ-agg data sets. Moreover, we also see that the CRF model has very strong performance producing similar results to the best SVDsup model. While both models use the same pairwise matrices $\mathbf{Y}$, inference in CRF is orders of magnitude faster than in SVDsup. Finally, we note that the results on the MQ2008-agg data set, which is more than 65%, sparse demonstrate that both models are robust and generalize well even in the sparse setting.

To investigate the utility of representing each expert's preferences with a separate set of parameters we ran experiments with a combined approach. For each query q_n we combined all pairwise preference matrices into a single matrix $\mathbf{Y}_n^{tot} = \sum_{\psi=1}^{\Psi} \mathbf{Y}_n^{\psi}$ and trained the SVDsup model on the SVD features from $\mathbf{Y}_n^{tot}$ only. Note that this model has 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 $\mathbf{Y}_n^{\psi}$ referred to as "individual". The results for the two data sets are shown in Figure 8. From the figure we see that the individual model significantly outperforms the combined one on both data sets. The performance of



(a) NDCG, MQ2008-agg

(b) Precision, MQ2008-agg

(c) NDCG, MQ2007-agg

(d) Precision, MQ2007-agg

Figure 8: Test NDCG@1-5 and Precision@1-5 results for SVDsup trained on features extracted from each $\mathbf{Y}_n^{\psi}$ (individual), versus SVDsup trained on features extracted only from the joint preference matrix $\mathbf{Y}_n^{tot}$ (combined). Results for the CRF model are similar and are not shown.

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 experts, 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. To further validate this conclusion we conducted the same experiment with the CRF model. We removed the expert specific weights w^{ψ}, w_{pos}^{ψ} and w_{neg}^{Ψ} and learned a single set of parameters $\mathbf{W} = \{b, w_{pos}, w_{neg}\}$ for all the experts. The results were similar to those in Figure 8 and we observed a significant drop in both NDCG and Precision accuracies.

Figure 9 further demonstrates the importance of specialization. The figure shows the expert ranking matrix together with the scores produced by the Reciprocal Rank Fusion and SVDsup for an example test query from MQ2008-agg. From the ground truth relevance levels ($\mathbf{L}$) 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

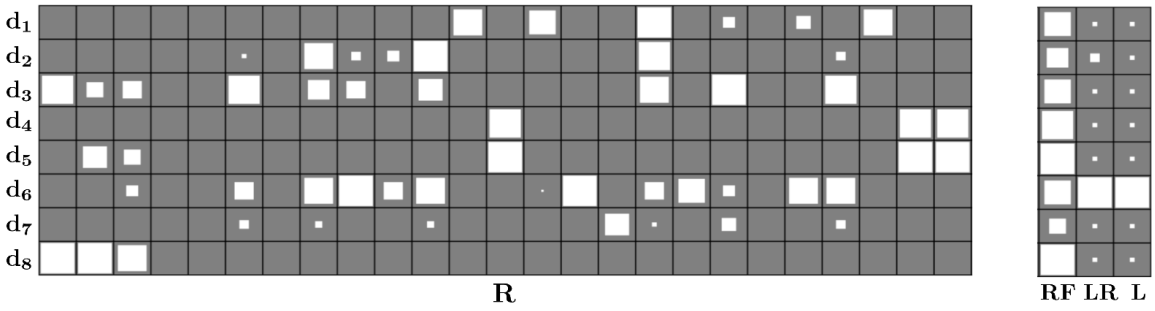


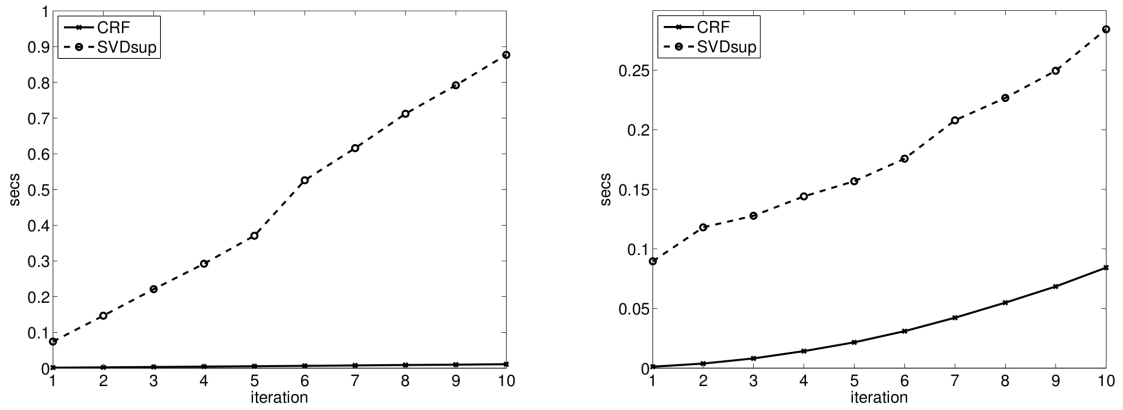
Figure 9: The matrix on the left shows the expert ranking matrix $\mathbf{R}$ for a test query in Fold 1 of the MQ2008-agg data set. 8 documents $\{d_1, ..., d_8\}$ were retrieved for this query and the normalized expert rankings are represented by white squares. The size of each square reflects the 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 LambdaRank SVDsup (LR) models, as well as the ground truth relevance levels ($\mathbf{L}$). Note that only d_6 is relevant to this query.

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. SVDsup 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 SVDsup are significantly lower than the score for d_6, so the model is confident in this ranking. The query shown in Figure 9 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 supervised approaches are able to fix this problem through specialization. By examining the queries in both MQ2008-agg and MQ2007-agg we found that both data sets contain a number of such queries and that both SVDsup and CRF performs significantly better on those queries than the consensus-based baselines.

3.7.3 Runtime Comparison

In the previous sections we demonstrated that fully supervised models SVDsup and CRF significantly outperform all baselines on two rank aggregation tasks. We also mentioned that the CRF model is considerably faster at inference time. In this section we quantify this difference.

We use test Fold 1 of the MQ2008-agg data set 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 $t \times \Psi$ columns. Concatenating



(a) Expert expansion runtimes

(b) Item expansion runtimes

Figure 10: Average per query runtimes (in seconds) for test Fold 1 of the MQ2008-agg. Figure 10(a) shows runtimes for the expert expansion experiment. Figure 10(b) shows runtimes for the item expansion experiment.

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 Ψ 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 10 and 10(b) show, averaged across queries, runtimes (in seconds) for both methods at each expansion iteration. Figure 10(a) shows runtimes for the expert expansion while Figure 10(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 10(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 10(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 the CRF model is more than 3.5 times faster. From these results we can conclude that when inference speed is important CRF is a better model choice especially if the number of experts is large.

3.7.4 Assessing Expert Quality

An additional advantage of using preference matrices directly, as done in CRF, is model interpretability. By analyzing the learned potential weights we can gain insight into which

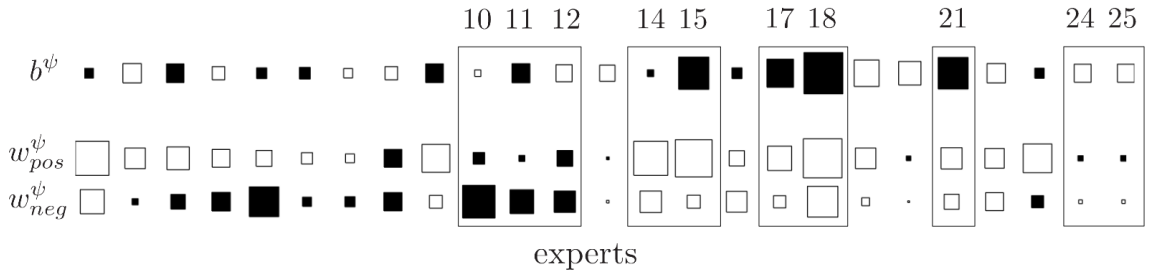


Figure 11: Learned b^ψ, w^ψ_{pos} and w^ψ_{neg} expert weights for training Fold 1 of MQ2008-agg; weights for other folds look analogous. White squares represent positive weights while black squares represent negative ones. The area of each square is proportional to weight magnitude.

experts are useful and how their preferences are combined. Figure 11 shows an example weight matrix learned by CRF model on the training Fold 1 of MQ2008-agg. Before delving into the figure we note that negative b^ψ raises the energy (lowering the probability, see Equation 4). Hence large negative values indicate that when preference from expert ψ is missing for a given document it is pushed down in the aggregate ranking, that is, expert ψ is important for aggregation. Similarly, positive w^ψ_{pos} lower the energy (upping the probability) while positive w^ψ_{neg} raise the energy. Consequently, when both weights are positive for a given expert ψ, documents d_{ni} strongly preferred by k (i.e., $\sum_{j\neq i} \mathbf{Y}^\psi_n(i,j) \gg \sum_{j\neq i} \mathbf{Y}^\psi_n(j,i)$) get pushed up in the ranking while those not preferred get pushed down.

Taking these relationships into account we see from Figure 11 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 b^ψ. Moreover, large positive values for both w^ψ_{pos} and w^ψ_{neg} 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 b^ψ's meaning that when their preferences are absent the rank of a document actually improves. Each of these experts also has near-zero w^ψ_{pos} and w^ψ_{neg} indicating that when their preferences are present the model does not use them.

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 w^ψ_{pos} and w^ψ_{neg} 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 noting 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 11 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

	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

Table 6: 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.

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 6, from the table we see that retrained model CRF* either performs comparably or outperforms the original model. This further supports the conclusion that useful insight into expert quality can be gained by analyzing weights learned by the model. Such analysis can be particularly useful in crowdsourcing and related domains where the goal is often to identify most accurate/reliable labelers from the crowd.

4. Conclusion and Future Work

In this work we have investigated the preference aggregation problem. The explosion in online technology has generated an immense amount of preference data and introduced many new ways to express and mine preferences. As the social web activity continues to grow effective preference aggregation techniques become increasingly more important as they have a direct impact on the success or failure of many web applications.

Machine learning has recently become the technique of choice to automatically mine preferences and learn effective aggregating functions. Building on the success of existing methods we have investigated both supervised and unsupervised aggregation problems and introduced new machine learning models for each problem type.

In the unsupervised problem we introduced a new probabilistic model over preferences based on a multinomial generative process. Preferences over items are expressed through real valued scores resulting in a convex optimization problem during inference which can be solved efficiently with standard gradient based techniques. Modeling the general partial pairwise preferences makes the model applicable to a wide range of preference aggregation problems. Empirically we have shown that our approach outperforms existing unsupervised aggregation methods on two unrelated problems: rank aggregation and collaborative filtering. Future work includes in this area includes investigating how the learned variances can be used to improve the final ranking. Another interesting direction is to explore mixtures of the MPM distributions where each mixing component is parametrized by its own set of scores. This idea is similar to the mixture of Mallows models discussed above and could be employed to learn a model for different user preference types and used for personalized recommendation.

In the supervised domain we have introduced a supervised extension to the Multinomial Preference Model as well as two fully supervised preference aggregation models. All of the presented approaches are based on pairwise preference matrices, and can also be applied to a variety of problems with different preference types. Both supervised models fully use the

labeled training data and can optimize the aggregating function for any target IR metric. The first model allows to apply any learning-to-rank algorithm during optimization and can thus be easily incorporated into many existing IR frameworks. The second model has a more involved learning procedure but is significantly faster during inference time and produces interpretable results. The two models thus offer a trade-off between ease of use and speed allowing the user to make an appropriate choice based on systems requirements. Future work in this domain involves applying these models to preference aggregation problems with other forms of expert and ground truth preferences. We also plan to investigate other ways of producing effective document representations from full or partial preferences.

Acknowledgments

We would like to thank Craig Boutilier, Hugo Larochelle, and Ruslan Salakhutdinov for many thoughtful discussions and suggestions. This research was supported by the Canadian Natural Sciences and Engineering Council (NSERC) and the Canadian Institute for Advanced Research (CIFAR).

References

E. Agichtein, E. Brill, and S. Dumais. Improving Web search ranking by incorporating user behavior information. In *International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2006.

K. J. Arrow. *Social Choice and Individual Values*. Yale University Press, 1951.

J. A. Aslam and M. Montague. Models for metasearch. In *International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2001.

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

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

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

C. J. C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. In *International Conference on Machine Learning*, 2005.

C. J. C. Burges, R. Ragno, and Q. V. Le. Learning to rank with nonsmooth cost functions. In *Neural Information Processing Systems*, 2006.

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

O. Chapelle, Y. Chang, and T.-Y. Liu. The Yahoo! learning to rank challenge, 2010. URL `http://learningtorankchallenge.yahoo.com/`.

S. Chen, F. Wang, Y. Song, and C. Zhang. Semi-supervised ranking aggregation. *Information Processing and Management*, 47, 2011.

Y. Chevaleyre, U. Endriss, J. Lang, and N. Maudet. A short introduction to computational social choice. In *International Conference on Current Trends in Theory and Practice of Computer Science*, 2007.

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

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

H. A. David. *The Method of Paired Comparissons*. Hodder Arnold, 1988.

S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41, 1990.

A. E. Elo. *The Rating of Chess Players: Past and Present*. Acro Publishing, 1978.

R. Fagin, R. Kumar, and D. Sivakumar. Efficient similarity search and classification via rank aggregation. In *International Conference on Management of Data*, 2003.

K. Gimpel and N. A. Smith. Softmax-margin CRFs: Training log-linear models with cost functions. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2010.

D. F. Gleich and L.-H. Lim. Rank aggregation via nuclear norm minimization. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2011.

J. Guiver and E. Snelson. Bayesian inference for Plackett-Luce ranking models. In *International Conference on Machine Learning*, 2009.

F. Guo, C. Liu, A. Kannan, T. Minka, M. Taylor, Y.-M. Wang, and C. Faloutsos. Click chain model in Web search. In *International World Wide Web Conference*, 2009.

J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl. An algorithmic framework for performing collaborative filtering. In *International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1999. URL `http://www.grouplens.org/node/73`.

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

X. Jiang, L.-H. Lim, Y. Yao, and Y. Ye. Statistical ranking and combinatorial hodge theory. *Mathematical Programming*, 127, 2011.

T. Joachims. Optimizing search engines using clickthrough data. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2002.

T. Joachims, L. Granka, B. Pan, H. Hembrooke, F. Radlinski, and G. Gay. Evaluating the accuracy of implicit feedback from clicks and query reformulations in web search. *ACM Transactions on Information Science*, 25, 2007.

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

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

H. Li. *Learning to Rank for Information Retrieval and Natural Language Processing*. Morgan Claypool, 2011.

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

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

Y.-T. Liu, T.-Y. Liu, T. Qin, Z.-M. Ma, and H. Li. Supervised rank aggregation. In *International World Wide Web Conference*, 2007b.

T. Lu and C. Boutilier. Learning Mallows models with pairwise preferences. In *International Conference on Machine Learning*, 2011.

R. D. Luce. *Individual Choice Behavior: A Theoretical Analysis*. Wiley, 1959.

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

B. M. Marlin, R. S. Zemel, and S. T. Roweis. Unsupervised learning with non-ignorable missing data. In *International Conference on Artificial Intelligence and Statistics*, 2005.

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

D. McAllester and J. Keshet. Generalization bounds and consistency for latent structural probit and ramp loss. In *Neural Information Processing Systems*, 2011.

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

M. Montague and J. A. Aslam. Condorcet fusion for improved retrieval. In *International Conference on Information and Knowledge Management*, 2002.

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

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

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

F. Rossi, K. Brent Venable, and T. Walsh. *A Short Introduction to Preferences: Between Artificial Intelligence and Social Choice*. Morgan & Claypool Publishers, 2011.

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

R. Salakhutdinov and A. Mnih. Restricted boltzmann machines for collaborative filtering. In *Neural Information Processing Systems*, 2008.

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

F. Sha and F. Pereira. Shallow parsing with conditional random fields. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2003.

L. L. Thurstone. The method of paired comparisons for social values. *Abnormal and Social Psychology*, 21, 1927.

M. N. Volkovs and R. S. Zemel. BoltzRank: Learning to maximize expected ranking gain. In *International Conference on Machine Learning*, 2009.

M. N. Volkovs and R. S. Zemel. A flexible generative model for preference aggregation. In *International World Wide Web Conference*, 2012.

M. N. Volkovs and R. S. Zemel. CRF framework for supervised preference aggregation. In *International Conference on Information and Knowledge Management*, 2013.

M. N. Volkovs, H. Lorochelle, and R. S. Zemel. Learning to rank by aggregating expert preferences. In *International Conference on Information and Knowledge Management*, 2012.