Direct Maximization of Rank-based Metrics **DRAFT IN PROGRESS**

Donald Metzler

Abstract

The ability to rank instances is important for a number of applications, such as information retrieval and collaborative filtering. It is often the case that the underlying task attempts to maximize some evaluation metric, such as mean average precision, over rankings. Most past work on learning how to rank has focused on likelihood- or margin-based approaches. In this work we explore directly maximizing *rank-based metrics*, which are a family of metrics that only depend on the order of ranked items. This allows us to maximize different metrics for the same training data. We show how the parameter space of linear (and monotonically linear) scoring functions can be reduced to a multinomial manifold. Parameter estimation is accomplished by solving an optimization procedure that directly maximizes the evaluation metric over the manifold. Results from *ad hoc* information retrieval are given that show significant improvements in effectiveness using the proposed model.

1 Introduction

For many information retrieval tasks, such as *ad hoc* retrieval, named-page finding, and question answering systems are evaluated based on some metric of how well they rank results. Examples of such metrics include mean average precision, mean reciprocal rank, and precision at 10, among many others [2]. A retrieval system designer must choose a metric appropriate to the underlying task. The end goal of the system is then to maximize the given metric. This is often accomplished by hand tuning system parameters until a given performance level is achieved. However, such an approach is only feasible for a small number of parameters and relatively simplistic models. The goal of this work is to develop methods for automatically setting parameters for a large family of retrieval models and evaluation metrics in a supervised fashion. That is, given a set of training data and a suitable retrieval model we explore how to find the parameters that maximize any *rank-based metric*. A rank-based metric is any metric that depends only on the ranking of the items under consideration. Therefore, for a fixed training set and model our training process will yield different parameter estimates for different metrics. This is intuitively appealing, as we expect one set of parameters will maximize average precision and another (different) set will maximize mean reciprocal rank. Ultimately, this allows for a single retrieval model to be used for a variety of different tasks. Although this work focuses mostly on information retrieval it is applicable to any task that involves maximizing a rank-based metric, such as collaborative filtering and certain classification tasks.

In this work we consider ranking categories (documents) for a set of input instances (queries). For each input instance, a complete ranking of the categories is generated. We focus on linear and monotonically linear models and show that their parameter spaces can be reduced to a multinomial manifold. As we show throughout the remainder of the paper, such a reduction has several advantages. Parameter estimation then attempts to maximize the evaluation metric of interest over the manifold. Since we are no longer dealing with a Euclidean search space, some care is necessary when carrying out such optimizations. Three straightforward, easy to implement, derivative-free optimization procedures over the multinomial manifold are described.

The remainder of this work is laid out as follows. Section 2 gives a broad overview past work done on the topic of learning rankings. Section 3 develops a theory of linear and monotonically linear ranking functions and shows the reduction of the parameter space to the multinomial manifold. Next, Section 4 describes three approaches to directly optimizing rank-based metrics over the multinomial manifold. Sections 5 and 6 describes the role of ranking in *ad hoc* information retrieval and empirically evaluates the model developed here. Finally, we conclude and detail possible future work in Section 7.

2 Related Work

Most past work on learning how to rank objects has been a variant of ordinal regression. Ordinal regression is similar to standard statistical regression, but allows ordinal, rather than nominal or real-valued, responses. Ordinal variables are coarse grained quantitative measures [1]. For example, in information retrieval, documents can be thought as being "highly relevant", "somewhat relevant", or "non-relevant". Such categories are ordinal because a "highly relevant" document is (quantitatively) *better than* a "somewhat relevant" document, but it is impossible to specify *how much* better it is. We note that this is different than assigning relevance scores on an integer scale of 1 to 10. In this case it is possible to determine the exact quantitative difference between the relevance of two documents. It is this type of response that standard regression models implicitly assume.

There are two popular approaches to ordinal regression. The most simple, yet naive, approach is to treat the ordinal responses as either class labels or real-valued targets and then apply standard classification or regression techniques. Such an approach completely ignores the ordinal nature of the responses and throws away a great deal of information. The other approach, which makes use of the ordinal structure, maintains weights and category boundary parameters. The boundary parameters are used to denote the boundary between two adjacent ordinal responses.

New instances are assigned to a category based the proximity of their response to the category boundaries.

There have been many variations of this scheme, all based on different ways to estimate the weights and boundary parameters. One of the earliest approaches used a perceptron-based learning technique [5], whereas more recent techniques have made use of large margin ideas, such as those used in SVMs [7, 8, 18].

Although many of the large margin methods have theoretically strong error rate bounds, they are often infeasible to train on large training data sets. The reason for this is that most add one (or more) constraints for every pair of instances in adjacent classes. For small training sets this generates a feasible number of constraints. However, for more realistic training sets, such as those that arise in information retrieval, this can lead to on the order of one million or more constraints. Clearly, such an optimization problem is both computationally and resource intensive.

3 Maximizing Rank-based Metrics

This section develops a novel approach to learning how to rank. The motivation for this task is information retrieval, where ranking large sets of documents is inherently important. As discussed in the previous section, most state of the art ordinal regression approaches can not efficiently handle such tasks. Therefore, we do not attempt to impose an ordinal response, such as "relevant" and "non-relevant" on documents, since the end goal of most ranking systems is to maximize some evaluation metric based on the rankings produced. Thus, we focus on directly maximizing this evaluation metric. As we will show, this approach can easily handle large training sets, such as those that typically arise in information retrieval, as long as rankings can be efficiently evaluated using the metric. Our approach can be applied to any task that induces a ranking attempts to maximize some metric over the rankings. It is readily applicable to information retrieval, collaborative filtering, and even general classification tasks.

3.1 Problem Description

Suppose we are given a set of categories (documents) $\mathcal{C} = \{C_i\}_{i=1}^{K}$, inputs (queries) $\mathcal{X} = \{X_i\}_{i=1}^{N}$, and training data $\mathcal{T}$. In addition, we are given a real-valued scoring function $S_\Lambda(C; X)$ parameterized by Λ. Given an input X_i, the score function $S_\Lambda(C; X_i)$ is computed for each $C \in \mathcal{C}$. The categories are then ranked in descending order according to their score. Therefore, the scoring function induces a partial ordering (ranking) $R(\mathcal{C}, X_i, S_\Lambda)$ on $\mathcal{C}$ for each input X_i. For simplicity we rewrite $R(\mathcal{C}, X_i, S_\Lambda)$ as $R_i(\Lambda)$ and let $\mathcal{R}_\Lambda = \{R_i(\Lambda)\}_{i=1}^{N}$ be the set of rankings induced over the inputs. Finally, we need a rank-based metric (evaluation function) $E(\mathcal{R}_\Lambda; \mathcal{T})$ that produces real valued output given a set of ranked lists and the training data. It should be noted that we require that E only considers the category *orderings* (rankings) and not the category scores.

The scores are only used to order (rank) the categories and not used to evaluate the ranking whatsoever.

Therefore, our goal is to find the Λ that maximizes E over the parameter space. Formally, this can be stated as:

$$\begin{aligned} \hat{\Lambda} &= \arg\max_{\Lambda} E(\mathcal{R}_\Lambda; \mathcal{T}) \\ s.t. & \quad \mathcal{R}_\Lambda \sim S_\Lambda(C; X) \\ & \quad \Lambda \in M_\Lambda \end{aligned}$$

where $\mathcal{R}_\Lambda \sim S_\Lambda(C; X)$ denotes that the orderings in $\mathcal{R}$ are induced using scoring function S, and M_Λ is the parameter space over Λ.

3.2 Monotonically Linear Scoring Functions

Rather than tackle the general optimization problem proposed above, we aim to solve a more constrained version. We restrict our focus to strictly monotonically increasing linear scoring functions. That is, we consider scoring functions from the following family:

$$\begin{aligned} \mathcal{S} = \{S_\Lambda(C; X) \quad : & \quad \exists l(\cdot) \text{ s.t. } l \text{ is strictly monotonically increasing and} \\ & \quad l(S_\Lambda(C; X)) = \Lambda^T f(C, X) + Z\} \end{aligned}$$

where $f(\cdot, \cdot)$ is a feature function that maps category/input pairs to real-valued vectors in $\mathbb{R}^d$, Z is a constant that does not depend on C (but may depend on Λ or X). That is, we require there to exist some strictly monotonically increasing function l that, when applied to S, yields a function that is linear in Λ. This family of models is strongly related to Generalized Linear Models from statistics, with l acting as the *link function*.

Trivial examples of functions within this family include linear discriminants, such as those used with perceptrons or support vector machines [3]. Another example includes the so-called maximum entropy (MaxEnt) distribution given by [15]:

$$\begin{aligned} S_\Lambda(C; X) &= \frac{1}{Z_\Lambda} \exp\left[\Lambda^T f(C, X)\right] \\ \log S_\Lambda(C; X) &= \Lambda^T f(C, X) - Z_\Lambda \end{aligned}$$

Therefore, distributions of this form are also in $\mathcal{S}$, with $l(\cdot) = log(\cdot)$.

By definition, every $S \in \mathcal{S}$ can can be reduced to a linear form via a strictly monotonically increasing function. Since such functions are rank preserving and subsequently evaluation metric preserving, we can always write the optimization problem for any scoring function in $\mathcal{S}$ as:

$$\begin{aligned} \hat{\Lambda} &= \arg\max_{\Lambda} E(\mathcal{R}_\Lambda; \mathcal{T}) \\ s.t. & \quad \mathcal{R}_\Lambda \sim \Lambda^T f(C, X) + Z \\ & \quad \Lambda \in M_\Lambda \end{aligned}$$

where the parameter space over Λ is unconstrained but not degenerate (i.e. $M_\Lambda = \mathbb{R}^d \setminus \{0\mathbf{I}\}$). Applying numerical optimization algorithms to unconstrained parameter domains can be problematic, especially in the presence of many repeated local and global extrema as is the case here. Also, many techniques require an intelligently chosen starting point or that an extrema can be bracketed, both of which may be difficult in general. For these reasons, we wish to further reduce the problem to a constrained optimization problem where the number of repeated extrema is significantly reduced and it is always possible to bracket a *global* extrema.

To facilitate the theory, we must introduce a slight modification to the feature vectors $f(C, X)$. We augment $f(C, X)$ with an additional component $f(C, X)_{d+1}$, such that $f(C, X)_{d+1} = K - \sum_{j=1}^{d} f(C, X)_j$ for some arbitrary K and denote the modified feature vector as $\tilde{f}(C, X)$. By augmenting each feature vector with this additional component we require that the sum of the feature vectors is constant ($= K$). We note that this additional component introduces redundant information but has no impact on the final result, as will be shown. Finally, let $\tilde{\Lambda}$ be the $d + 1$ dimension version of Λ. Using the augmented feature vectors, the unconstrained optimization problem can be written as:

$$\begin{aligned} \hat{\Lambda} &= \arg\max_{\tilde{\Lambda}} E(\mathcal{R}_{\tilde{\Lambda}}; \mathcal{T}) \\ s.t. \quad & \mathcal{R}_{\tilde{\Lambda}} \sim \tilde{\Lambda}^T \tilde{f}(X, C) + Z \\ & \tilde{\Lambda} \in M_{\tilde{\Lambda}} \\ & \tilde{\lambda}_{d+1} = 0 \end{aligned}$$

It is easy to see why this optimization problem is equivalent to non-augmented input case. The condition $\tilde{\lambda}_{d+1} = 0$ requires the weight associated with the augmented feature to be 0, thus eliminating any influence the augmented feature may have on the outcome.

3.3 Reduction to Multinomial Manifold

We will now show that this optimization problem, for the family of scoring functions $\mathcal{S}$, is equivalent to the following constrained optimization problem:

$$\begin{aligned} \hat{\Lambda} &= \arg\max_{\tilde{\Lambda}} E(\mathcal{R}_{\tilde{\Lambda}}; \mathcal{T}) \\ s.t. \quad & \mathcal{R}_{\tilde{\Lambda}} \sim \tilde{\Lambda}^T \tilde{f}(C, X) + Z \\ & \tilde{\Lambda} \in \mathbb{P}^d \end{aligned}$$

where $\mathbb{P}^d$ is a multinomial manifold (also known as a d-simplex) described by:

$$\mathbb{P}^d = \left\{ \Lambda \in \mathbb{R}^{d+1} : \forall j \ \lambda_j \geq 0, \sum_{i=1}^{n+1} \lambda_i = 1 \right\}$$

Theorem. Any solution to the non-augmented optimization problem over $\mathbb{R}^d \setminus \{0\mathbf{I}\}$ has en equivalent solution to the augmented optimization over $\mathbb{P}^d$.

Proof. Suppose that $\hat{\Lambda}$ is the solution to the augmented optimization problem. This is equivalent to saying that the first d components of $\hat{\Lambda}$

are a solution to the original unconstrained optimization problem. Now, consider the following transformation to $\hat{\Lambda}$:

$$\hat{\lambda}'_i = \frac{\hat{\lambda}_i - k}{W}$$

where $k = \min\{0, \min\{\hat{\lambda}_i\}\}$ and $W = \sum_i (\hat{\lambda}_i - k)$. Thus, there are two cases: $\min\{\hat{\lambda}_i\} \geq 0$ and $\min\{\hat{\lambda}_i\} < 0$.

If $\min\{\hat{\lambda}_i\} \geq 0$, then $k = 0$ by definition and therefore $\hat{\lambda}'_i = \frac{\hat{\lambda}_i}{\sum_j \hat{\lambda}_j}$. Since all of the parameter values are scaled by the same positive constant (W), the parameter $\hat{\Lambda}'$ ranks categories exactly the same as using parameter $\hat{\Lambda}$. In addition, it is easy to verify that $\hat{\Lambda}' \in \mathbb{P}^d$. Therefore, for any solution $\hat{\Lambda}$ such that $\min\{\hat{\lambda}_i\} \geq 0$, there exists an equivalent solution to the constrained problem in $\mathbb{P}^d$.

Next, if $\min\{\hat{\lambda}_i\} < 0$, then $k = \min\{\hat{\lambda}_i\}$. Under $\hat{\Lambda}$, the scoring function becomes:

$$\begin{aligned}
S_\Lambda(C; \tilde{X}) &= \sum_i \hat{\lambda}'_i \tilde{f}(C, X)_i + Z \\
&= \sum_i \left(\frac{\hat{\lambda}_i - k}{W} \right) \tilde{f}(C, X)_i + Z \\
&= \frac{1}{W} \sum_i \hat{\lambda}_i \tilde{f}(C, X)_i - \frac{k}{W} \sum_i \tilde{f}(C, X)_i + Z \\
&\overset{rank}{=} \sum_i \hat{\lambda}_i \tilde{f}(C, X)_i
\end{aligned}$$

Where the last step follows from the fact that $\sum_i \tilde{f}(C, X)_i$ is constant ($= K$) by definition. Again, it is straightforward to see that $\hat{\Lambda}' \in \mathbb{P}^d$. Therefore, we see that for any solution $\hat{\Lambda}$ such that $\min\{\hat{\lambda}_i\} < 0$, there exists an equivalent solution to the constrained problem in $\mathbb{P}^d$. We have therefore shown that for any solution to the unconstrained problem there also exists as a solution to the constrained problem, thus completing the proof $\square$

Therefore, for a given training set and any scoring function in $\mathcal{S}$, we have reduced the unconstrained problem of maximizing $E(\mathcal{R}_\Lambda; \mathcal{T})$ to a constrained optimization over the multinomial manifold, where categories are ranked according to a simple linear function. This allows us to always bracket a global extremum, because we know that *any* solution to the original unconstrained problem has an equivalent solution on the manifold, including the global extrema.

Searching over the manifold provides an augmented solution $\tilde{\Lambda}$, which requires augmented feature vectors. Fortunately, it is always possible to transform the augmented parameter vector to one that is rank equivalent to a scoring function over the non-augmented parameter space. This can be done by transforming each weight by $\hat{\lambda}'_i = \hat{\lambda}_i - \hat{\lambda}_{d+1}$ for all i. The proof that this is valid (i.e. does not impact how things are rankings) is similar to the proof given above. This results in the weight associated with the augmented feature being set to 0, thus removing any influence the augmented feature may have on our scoring function. The solution is again implicitly over the original input space.

3.4 Summary

The following summarizes the proposed procedure:

1. Construct $\tilde{f}(C, X)$ from $f(C, X)$ for all of the feature vectors.
2. Solve the constrained optimization problem over the multinomial manifold to find $\hat{\Lambda}$.
3. Construct Λ by translating $\hat{\Lambda}$ such that the augmented component has weight 0.
4. Order (rank) new inputs according to scoring function $\Lambda^T f(C, X)$.

Steps 1, 3 and 4 are straightforward. Step 2 is the most important part of the method. In the next section we explore numerical techniques for solving the optimization procedure required in this step.

4 Parameter Estimation

In this section we describe three simple numerical optimization techniques for solving the optimization problem described in the last section. Unlike most optimization techniques that treat the parameter space as $\mathbb{R}^d$, we must solve an optimization problem over the multinomial manifold $\mathbb{P}^d$. Since the manifold is coordinate-free, we must take care when devising an optimization strategy. Furthermore, we make no explicit assumptions about the continuity or differentiability of E, the function we are attempting to maximize. In those cases that E is continuous and differentiable over $\mathbb{R}^d$ the optimization is straightforward. However, for most applications of interest, E will not have such a nice form and certain provisions will have to be made.

4.1 Grid Search

The most naive approach to solving the optimization problem is to perform an exhaustive grid search over the manifold. That is, we place a grid over the manifold and evaluate $E(\mathcal{R}_\Lambda; \mathcal{T})$ at every grid intersection. More formally, given a parameter $\epsilon = \frac{1}{K}$ for $K \in \mathbb{Z}^+$ that controls how fine grained our grid is, we define:

$$\begin{aligned} \mathcal{G} &= \left\{ \Lambda = (k_1\epsilon \dots k_d\epsilon) : \sum_i k_i\epsilon = 1, k_i \in \mathbb{N} \right\} \\ &= \left\{ \Lambda = (k_1\epsilon \dots k_d\epsilon) : \sum_i k_i = K, k_i \in \mathbb{N} \right\} \end{aligned}$$

As we see $|\mathcal{G}|$, the number of parameter values we must evaluate E at, depends both on d (the number of parameters) and K (how fine grained our grid is). A grid search is feasible only if both d and K are relatively small. For larger values we must turn to more sophisticated training methods. However, we should note that the grid search method has the nice property that it is guaranteed to find a global maximum as K gets large. This allows exact global convergence to be traded off for faster training time.

4.2 Coordinate Ascent

Coordinate ascent is a commonly used optimization technique for unconstrained optimization problems. The algorithm iteratively optimizes a multivariate objective function by solving a series of one dimensional searches. It repeatedly cycles through each parameter, holding all other parameters fixed, and maximizes over the free parameter. The technique is known to converge slowly on objective functions with long ridges. Variations of the method, including Powell's method, have been proposed to overcome this issue [17].

Coordinate ascent can be applied to the optimization problem under consideration with minor modifications necessary to handle the fact we are optimizing over the multinomial manifold rather than a Euclidean space. All one dimensional searches done by the algorithm will be performed as if they were being done in $\mathbb{R}^d$. However, this does not ensure that the update parameter estimate will be a point on the manifold. Therefore, after a step is taken in $\mathbb{R}^d$, we project the point back onto the manifold, which we showed is always possible. Note that this projection preserves the function value since the unnormalized and projected parameter estimates lead to rank-equivalent rankings. Therefore, the optimization is implicitly being done in a space that we know how to optimize over ($\mathbb{R}^d$), but is continually being projected back onto to the manifold.

More concretely, suppose that λ_i is the current free parameter and all other parameters are held fixed. Then, the update rule is given by:

$$\lambda_i' \quad = \quad \arg\max_{\lambda_i} E(\mathcal{R}_\Lambda; \mathcal{T})$$

After λ_i' is updated the entire parameter vector is then projected back onto the manifold. This process is iteratively done over all parameters until some convergence criteria is met. Finally, we note that if E is partially differentiable with respect to each parameter then the update rule is straightforward. For those functions where E is not partially differentiable, such as the ones considered in the remainder of this paper, a line search must be done to find the $\arg\max$.

4.3 Steepest Ascent

Another common optimization technique, similar in nature to coordinate ascent, is steepest ascent. This technique aims at finding a maxima by iteratively taking steps in the direction of steepest ascent. The update rule is given by:

$$\lambda_i' \quad = \quad \lambda_i + \alpha \frac{\partial E(\mathcal{R}_\Lambda; \mathcal{T})}{\partial \lambda_i}$$

where α is the step size that maximizes the objective function in the direction of the gradient. Just like with the coordinate ascent algorithm, after each update the parameter vector must be projected onto the manifold.

The basic technique requires the function being optimized to be differentiable. Since we do not assume our objective function is differentiable we must use approximate partial derivatives. This is accomplished via finite difference approximations. The parameter update rule is then stated by:

$$\lambda_i' \quad \approx \quad \lambda_i + \alpha \frac{E(\mathcal{R}_{\Lambda_t + h\mathbf{e}_i}; \mathcal{T}) - E(\mathcal{R}_{\Lambda_t}; \mathcal{T})}{h}$$

where h is width of the finite difference, α is the step size that maximizes the objective function in the direction of the approximate gradient, and $\mathbf{e}_i$ is a d-vector with entry i equal to 1 and all other entries equal to 0. In this case α can be found by a simple line search in the direction of the approximate gradient. As we see, the finite difference approximation of the partial derivative attempts to measure what change in E is achieved when the current parameter setting is perturbed slightly. Other finite difference approximations exist and may lead to better approximations.

For cases when E is differentiable and the gradient can be computed exactly, steepest ascent may converge faster than coordinate ascent. However, when E is not differentiable, steepest ascent may perform poorly due to the roughly approximated gradients. In our experiments using non-differentiable objective functions we have found coordinate ascent to be far superior to the steepest ascent algorithm.

4.4 Discussion

In this section the details of three optimization techniques were given. Finding the maximum of an arbitrary evaluation function E can be very difficult, especially in high-dimensional space. Only the grid search method, with a suitably chosen K, is a guaranteed to find a global maxima. Both coordinate and gradient ascent are local search techniques that are only guaranteed to find a global maxima if the evaluation function E is concave. Previous work in information retrieval has shown that for a certain set of term and phrase features both average precision and precision at 10 are approximately concave over a wide range of collections [9]. This may be the case for many related applications and feature sets, but is not true in general. For functions with many local maxima, a multiple random restart strategy can be used to increase the chances of finding a global solution. Throughout the remainder of this work, all optimization is carried out using coordinate ascent with 10 random restarts.

5 Ranking in *Ad Hoc* IR

Ad hoc retrieval is the standard information retrieval task of generating a ranked list of documents that are topically relevant to some information need, where the information need is typically expressed as a textual query. This is the main task that we explore here. There are many ways of evaluating information retrieval systems, nearly all of which are rank-based metrics over ranked lists. Therefore, the approach developed in this paper is easily applicable to training models for most information retrieval tasks. One of the most widely used evaluation metrics, and the one used here, is mean average precision. Training data for this task consists of

	Feature		Feature
1	$\sum_{w \in Q \cap D} \log(tf_{w,D})$	4	$\sum_{w \in Q \cap D} \log(\frac{\|C\|}{cf_w})$
2	$\sum_{w \in Q \cap D} \log(1 + \frac{tf_{w,D}}{\|D\|})$	5	$\sum_{w \in Q \cap D} \log(1 + \frac{tf_{w,D}}{\|D\|} \frac{N}{df_w})$
3	$\sum_{w \in Q \cap D} \log(\frac{N}{df_w})$	6	$\sum_{w \in Q \cap D} \log(1 + \frac{tf_{w,D}}{\|D\|} \frac{\|C\|}{cf_w})$

Table 1: Features used in the small collections *ad hoc* retrieval experiments. $tf_{w,D}$ is the number of times term w occurs in document D, cf_w is the number of times term w occurs in the entire collection, df_w is the number of documents term w occurs in, $|D|$ is the length (in terms) of document D, $|C|$ is the length (in terms) of the collection, and N is the number of documents in the collection.

a set of queries and relevance judgments. The relevance judgments are binary. That is, a document is either relevant to a given query or it is not.

Information retrieval has long been concerned with ranking documents. In fact, most retrieval models are based on the Probability Ranking Principle (PRP), which states that an optimal retrieval model will rank documents in order of decreasing usefulness to a given information need. Since relevance is most often treated as binary, this is most often interpreted probabilistically as ranking documents in decreasing order of their likelihood of being relevant, as in the Binary Independence Model (BIM), where relevance is treated as a binary random variable. Therefore, most probabilistic retrieval models either implicitly or explicitly model relevance and attempt to rank documents this way.

One proposed machine learning-based retrieval model ranks documents according to $P(R = 1|D, Q)$, where R is a binary random variable, D is the document random variable, and Q is the query random variable [6, 14]. For a fixed query q, documents in the collection are ranked according to their likelihood of being relevant ($R = 1$). In [14], Nallapati assumes the following:

$$P(R = 1|D = d, Q = q) \quad \propto \quad \Lambda^T f(d, q)$$

Table 1 shows the six features considered by Nallapati. The parameter vector is estimated using a linear SVM, with relevant documents considered the "positive class" and non-relevant documents the "negative class". Therefore, the ranking task is being treated as a classification problem.

Is it very often the case that there are many more relevant documents compared to non-relevant documents for a given query. For this reason, the training data is very unbalanced. Nallapati found that the data needed to be balanced in order to achieve good generalization performance. Balancing was done by undersampling the majority (non-relevant) class. Although this led to improved performance over the unbalanced case, it had the negative effect of throwing away valuable training data. We note that other solutions to the unbalanced data problem for SVMs exist that do not require training data to be compromised, such as allowing separate costs for training errors in the positive and negative classes [13].

We feel that estimating $P(R = 1|D, Q)$ and/or $P(R = 0|D, Q)$ may actually be inappropriate. Such models are solving an inherently more difficult problem than necessary. The PRP is based entirely on *ranking* documents. The popular BIM model has guided the way for much of the history of information retrieval, but it may have also led researchers into the pitfall of studying a problem that is more difficult than necessary. Estimating joint or conditional models over large spaces of random variables is a daunting task. However, as we showed in this paper, the space of ranking functions is well defined and lends itself to a direct maximization.

Another issue is that past models have lent themselves to estimation via maximization of likelihood or maximization of margin. These approaches are inherently maximizing the incorrect metric. Information retrieval is not concerned with likelihood, nor classification accuracy. Instead it is entirely concerned with how well a model ranks documents. It can be argued that estimating parameters by maximizing the likelihood of some training data or minimizing classification error (as in the Nallapati SVM model) can be thought of as optimizing a function that is correlated with the underlying retrieval metric, such as mean average precision. However, this has been shown experimentally to be invalid [12] and it can also be shown theoretically to be invalid, as well.

Therefore, we argue for estimation techniques that are entirely rank-based. We note that Joachims studied this problem when applied to information retrieval using clickthrough data as a form of relevance judgment [8]. In his approach parameter estimation was achieved using ranking SVMs, which is an SVM formulation aimed entirely at learning rankings, rather than classification, and therefore is optimizing the correct metric. Another advantage of ranking SVMs is the fact that they are based on the strong computational learning theory of SVMs and yield a relatively straightforward quadratic programming optimization problem. However, given a large number of queries and a large number of relevance judgments the ranking SVM approach is infeasible in an offline setting due to the number of "preference constraints" generated. Planned experiments carried out using this approach were cancelled due to their long running times. Whereas ranking SVMs scale both with the number of training instances and number of features, the approach presented in this paper only scales with the number of features and the time it takes to evaluate and generate a ranking. Therefore, arbitrarily complex, large, and dense preferences/rankings can be handled without a computational explosion.

6 Experimental Results

This section describes experiments carried out using the approach described in this work on a number of *ad hoc* retrieval experiments. Results are compared against Nallapati's SVM model [14] and language modeling [16]. The results show that the proposed approach is not only feasible for large collections and large training sets, but also that the parameters estimated in the model lead to superior retrieval effectiveness.

	Disks 1,2	Disk 3	Disks 4,5
Num. Docs	741,856	336,310	556,077
Training topics	101-150	51-100	301-350
Test topics	151-200	101-150	401-450

Table 2: Summary of TREC collections used in small collection experiments.

6.1 Small Collection Experiments

In this section we compare the models on three standard TREC collections. For each collection, 50 topics (queries) are used for training and 50 for testing. Only the title portion of the TREC topics are used. All documents are stemmed using Krovetz Stemmer and stopped using a standard list of common terms. A summary of the collections used and the training and test topics are given in Table 2.

Each model is trained using only the data in the relevance judgments. That is, when the model is being trained it only 'knows' about the documents contained in the relevance judgments and not about any of the unjudged documents in the collection. In the case of the balanced SVM model, the non-relevant judgments from the relevance file were undersampled. Our model, denoted by RankMax, is trained using the same exact features as the SVM, and therefore has no additional "power". The feature-based models are compared against a language modeling baseline. The language modeling run ranks documents via query likelihood, with document models estimated using Bayesian (Dirichlet) smoothing. The language model is trained by finding the smoothing parameter that maximizes the mean average on the training data.

The results of the experiments are given in Table 3. For the numbers in the table, the trained model is used to rank queries against the *entire collection*, not just the documents found in the relevance judgments. The mean average precision is computed at a depth of 1000 retrieved documents. As we see from the results our parameter estimation technique consistently leads to statistically significant improvements over the SVM estimates. Furthermore, it significantly outperforms language modeling on 4 out of 6 runs. Language modeling, on the other hand, significantly outperforms the SVM model on 4 out of the 6 runs.

The results indicate that language modeling, despite its simplicity, stands up very well compared to sophisticated feature-based machine learning techniques. The results also provide empirical proof that SVM parameter estimation is simply not the correct paradigm here, mainly because it is optimizing the wrong objective function. Our estimation technique, however, is directly maximizing the evaluation metric under consideration and results in stable, effective parameter estimates across the collections.

When it comes to implementation, our method is much easier to implement than SVMs, but more complex than language modeling. For this reason, a number of issues should be considered before choosing a retrieval model. We feel that for this simple case, using simple term statistic fea-

	Disks 1,2		Disk 3		Disks 4,5	
	Train	Test	Train	Test	Train	Test
SVM (unbalanced)	0.0955	0.1091	0.1501	0.1336	0.1421	0.1434
SVM (balanced)	0.1577	0.1849	0.1615	0.1361	0.1671	0.1897
RankMax	0.1955‡	0.2327†‡	0.2080†‡	0.1773‡	0.2238†‡	0.2328†‡
Language modeling	0.1883‡	0.2155‡	0.1875‡	0.1642‡	0.1819	0.1995

Table 3: Training and test set mean average precision values for various *ad hoc* retrieval data sets and training methods. A † represents a statistically significant improvement over language modeling and ‡ denotes significant improvement over the balanced SVM model. Tests done using a one tailed paired t-test at the 95% confidence level.

	WT10G	GOV2
Num. Docs	1,692,096	25,205,179
Size	11 GB	427GB
Topics	451-550	701-750

Table 4: Summary of TREC collections used in small collection experiments.

tures, language modeling is very likely the best practical choice. The real power of feature-based methods comes when more complex features, such as those that the language modeling framework fails to handle directly, are used. A good example of the power of feature-based models over language modeling is given in the next section, where we explore *ad hoc* retrieval on large web collections.

6.2 Large Collection Experiments

In this section we consider *ad hoc* retrieval experiments on two large TREC web collections. Table 4 summarizes the data sets considered. There has been recent evidence that using term proximity information for *ad hoc* retrieval on web collections can provide significant improvements in effectiveness over models that only consider terms to be a simple bag of words [9, 11].

Therefore, we consider an extremely simple set of features that account for different kinds of proximity between terms within the query. Table 5 explains the three features used. As we see, there are three features — a single term feature, an exact phrase feature, and an unordered phrase feature. These features are meant to capture the fact that the order that query terms appear provides important information. For example, the queries “white house rose garden” and “white rose house garden” seek completely different pieces of information, yet are viewed as the same query in the bag of words representation.. The features also attempt to capture the fact that

Name	Feature
Term	$\sum_{q_i \in Q} \log \left[(1-\alpha_D) \frac{tf_{q_i,D}}{\lvert D \rvert} + \alpha_D \frac{cf_{q_i}}{\lvert C \rvert} \right]$
Ordered Phrase	$\sum_{q_i, q_{i+1} \cdots, q_{i+k} \in Q} \log \left[(1-\alpha_D) \frac{tf_{\#1(q_i \cdots q_{i+k}),D}}{\lvert D \rvert} + \alpha_D \frac{cf_{\#1(q_i \cdots q_{i+k})}}{\lvert C \rvert} \right]$
Unordered Phrase	$\sum_{q_i, \ldots, q_j \in Q} \log \left[(1-\alpha_D) \frac{tf_{\#uw8(q_i \cdots q_j),D}}{\lvert D \rvert} + \alpha_D \frac{cf_{\#uwN(q_i \cdots q_j)}}{\lvert C \rvert} \right]$

Table 5: Features used in the large collections *ad hoc* retrieval experiments. The ordered phrase sum is over every contiguous subset of query terms of length two or more within the query, and the unordered phrase sum is over every subset of two or more query terms. $tf_{\#1}$ is the count of the number of times the expression occurs as an exact phrase within D, and $tf_{\#uw8}$ is the count of the number of times the terms within the expression appear ordered or unordered within a window of length 8 within D.

Train \ Test	WT10G	GOV2
WT10G	0.2231†	0.2783†
GOV2	0.2201†	0.2844†
Language modeling	0.2030	0.2502

Table 6: Mean average precision results for the large collections experiments. The † denotes statistically significant improvements over the language modeling baseline at the 95% confidence level.

most web queries are made up of one or more implicit phrases, such as “white house” and “rose garden” in the example above. Since we consider all subphrases, we are likely to pick up on such phrases and retrieve more relevant documents.

Here, we train and test our model on both collections and compare against a simple bag of words model (language modeling) baseline. The α_D used in the features is set to $\frac{\mu}{|D|+\mu}$ (Dirichlet smoothing), with μ fixed at a value approximately equal to double the average document length.

The results of the experiments are given in Table 6. As the results show, the models learned generalize well and achieve statistically significant improvements over the baseline language modeling system. These results provide evidence that both proximity and the training method developed in this work can be leveraged to significantly improve effectiveness. As further evidence of the power of both term proximity for web retrieval and the proposed training method, a model similarly trained achieved the best title-only run at the TREC 2004 Terabyte Track [10, 4].

Finally, Figure 1 illustrates the nearly concave surface that arises by imposing the mean average precision metric over the multinomial simplex of ranking function parameters. Although there is no guar-

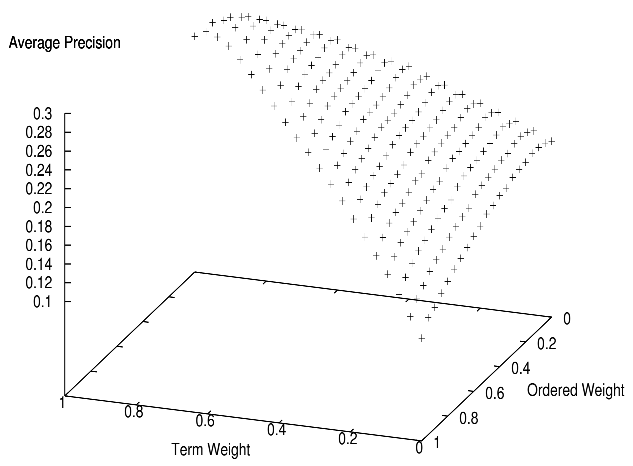

Figure 1: Mean average precision surface over multinomial simplex for GOV2 using full dependence model.

antee that such a nicely concave surface will exist for all features and all evaluation metrics, it provides some evidence that the functions we are maximizing over the simplex are not too difficult to optimize using easy to implement machinery, such as coordinate or steepest ascent algorithms.

7 Conclusions and Future Work

In this paper we have investigated the properties of linear and monotonically linear scoring functions when used in conjunction with the class of rank-based evaluation metrics. It was shown that the parameter space of these scoring functions lies on a multinomial manifold. Not only does this provide a theoretically elegant representation of the parameter space, but it also provides a parameter space with a significantly reduced number of local extrema and a guaranteed bracket of a global maxima. Three simple optimization techniques that make no assumptions about the underlying objective function were presented. Finally, we applied the methods developed to the task of *ad hoc* information retrieval. Both small and large collections were examined and the mean average precision metric was explored. It was shown that our parameter estimation paradigm significantly outperforms other models in terms of retrieval effectiveness.

Potential areas of future work include applying the model using other evaluation metrics and to other application domains. In particular, collaborative filtering is an area that other ranking algorithms have been applied to in the past. It would be beneficial to compare the performance of our model with models tested in that domain. Another area that needs explored is how well the simple optimization algorithms scale to large numbers of parameters and how easy or difficult different classes of features/evaluation metrics are to successfully optimize.

Acknowledgments

This work was supported in part by the Center for Intelligent Information Retrieval and in part by Advanced Research and Development Activity and NSF grant #CCF-0205575. Any opinions, findings and conclusions or recommendations expressed in this material are the author(s) and do not necessarily reflect those of the sponsor.

References

[1] A. Agresti. *Analysis of Ordinal Categorical Data*. John Wiley and Sons, Inc., 1984.

[2] R. Baeza-Yates and G. Navarro. *Modern Information Retrieval*. Addison-Wesley, 1999.

[3] C. J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

[4] C. Clarke, N. Craswell, and I. Soboroff. Overview of the trec 2004 terabyte track. In *Online proceedings of the 2004 Text REtrieval Conference (TREC 2004)*, 2004.

[5] K. Crammer and Y. Singer. Pranking with ranking. In *Proceedings of Advances in Neural Information Processing Systems (NIPS 2001)*, 2001.

[6] F. Gey. Inferring probability of relevance using the method of logistic regression. In *Proceedings of the ACM SIGIR*, 1994.

[7] R. Herbrich, T. Graepel, and K. Obermayer. *Advances in Large Margin Classifiers*, chapter Large Margin Rank Boundaries for Ordinal Regression, pages 115–132. The MIT Press, 2000.

[8] T. Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining (KDD 02)*, pages 133–142, 2002.

[9] D. Metzler and W. B. Croft. A markov random field model for term dependencies. In *To appear in the Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 05)*, 2004.

[10] D. Metzler, T. Strohman, H. Turtle, and W. B. Croft. Indri at terabyte track 2004. In *Text REtrieval Conference (TREC 2004)*, 2004.

[11] G. Mishne and M. de Rijke. Boosting web retrieval through query operators. In *Proceedings of the 27th European Conference on Information Retrieval (ECIR '05)*, 2005.

[12] W. Morgan, W. Greiff, and J. Henderson. Direct maximization of average precision by hill-climbing with a comparison to a maximum entropy approach. Technical report, MITRE, 2004.

[13] K. Morik, P. Brockhausen, and T. Joachims. Combining statistical learning with a knowledge-based approach - a case study in intensive care monitoring. In *Proceedings of the 16th International Conference on Machine Learning (ICML '99)*, 1999.

[14] R. Nallapati. Discriminative models for information retrieval. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 04)*, pages 64–71, 2004.

[15] S. D. Pietra, V. D. Pietra, and J. Lafferty. Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(4):380–393, 1997.

[16] J. M. Ponte and W. B. Croft. A language modeling approach to information retrieval. In *Proceedings of the 21st annual international ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 98)*, pages 275–281, 1998.

[17] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing.* Cambridge University Press, 1992.

[18] A. Shashua and A. Levin. Ranking with large margin principle: Two approaches. In *Proceedings of Advances in Neural Information Processing Systems (NIPS 2002)*, 2002.