Direct Maximization of Rank-Based Metrics for Information Retrieval

Donald A. Metzler, W. Bruce Croft, and Andrew McCallum
Department of Computer Science
Center for Intelligent Information Retrieval
University of Massachusetts, Amherst
{metzler, croft, mccallum}@cs.umass.edu

Abstract

Ranking is an essential component for a number of tasks, 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 scoring functions can be reduced to a multinomial manifold. Parameter estimation is accomplished by optimizing the evaluation metric over the manifold. Results from *ad hoc* information retrieval are given that show our model yields significant improvements in effectiveness over other approaches.

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. 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 set of features 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 documents for a set of queries[1]. For each query, a total ordering of the documents 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 is done by maximizing 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. A general coordinate ascent procedure is 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 ranking functions and shows the reduction of the parameter space to the multinomial manifold. Section 4 describes the role of ranking in *ad hoc* information retrieval and empirically evaluates our model. Finally, we conclude and detail possible future work in Section 5.

2 Related Work

Most past work on learning how to rank objects have been variants 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 on 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 [3], whereas more recent techniques have made use of large margin ideas, such as those used in SVMs [5, 6, 8].

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.

[1] We use the terms 'documents' and 'queries' to adhere to common information retrieval terminology, but these should be thought of more abstractly as 'objects' and 'input instances', respectively.

3 Maximizing Rank-based Metrics

This section develops a novel way of looking at ranking. Our motivation 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 cannot efficiently handle such tasks. Rather than explicitly imposing an ordinal response, such as "relevant" and "non-relevant" on documents, we focus on learning the parameters of a scoring function that implicitly imposes an ordering on them. Since the end goal of most ranking systems is to maximize some evaluation metric based on the rankings produced, we focus on finding the parameters that maximize this evaluation metric. As we will show, this approach can easily handle large training sets, such as those that typically arise in information retrieval. The approach can be applied to any task that induces a ranking via a scoring function and seeks 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 documents $\mathcal{D}$, queries $\mathcal{Q} = \{Q_i\}_{i=1}^N$, and training data $\mathcal{T}$. In addition, we are given a real-valued scoring function $S_\Lambda(D; Q)$ parameterized by Λ. Given a query Q_i, the score function $S_\Lambda(D; Q_i)$ is computed for each $D \in \mathcal{D}$. The documents are then ranked in descending order according to their score. Therefore, the scoring function induces a total ordering (ranking) $R(\mathcal{D}, Q_i, S_\Lambda)$ on $\mathcal{D}$ for each query Q_i[2]. For simplicity we rewrite $R(\mathcal{D}, Q_i, S_\Lambda)$ as $R_i(\Lambda)$ and let $\mathcal{R}_\Lambda = \{R_i(\Lambda)\}_{i=1}^N$ be the set of rankings induced over the queries. 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 document *rankings* and not the document scores. The scores are only used to rank the documents 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:

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

where $\mathcal{R}_\Lambda \sim S_\Lambda$ 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 increasing linear scoring functions. That is, we consider scoring functions from the following family:

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

where $f(\cdot, \cdot)$ is a feature function that maps document/query pairs to real-valued vectors in $\mathbb{R}^d$, Z is a constant that does not depend on D (but may depend on Λ or Q). That is, we require there to exist some strictly 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*.

[2]We assume ties are broken by category (document) id.

By definition, every $S \in \mathcal{S}$ can can be reduced to a linear form via a strictly increasing function. Since such functions are rank preserving and subsequently evaluation metric preserving, we can always write the scoring function linearly as $\Lambda^T f(D,Q) + Z$.

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 $f(D,Q)$. We augment $f(D,Q)$ with an additional component $f(D,Q)_{d+1}$, such that $f(D,Q)_{d+1} = K - \sum_{j=1}^{d} f(D,Q)_j$ for some arbitrary K and denote the modified feature vector as $\tilde{f}(D,Q)$. 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 optimization problem can be written as:

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

where we assume $\tilde{\Lambda}$ is non-trivial. It is easy to see why this optimization problem is equivalent to non-augmented feature 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:

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

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 augmented optimization problem over $\mathbb{R}^{d+1} \setminus \{0\mathbf{I}\}$ has a rank-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 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 $W = \sum_i (\hat{\lambda}_i - k)$. It is easy to see that $\hat{\Lambda}' \in \mathbb{P}^d$, so the transformation maps points onto the manifold. We must now show the transformation preserves rank-equivalence.

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 documents exactly the same as using parameter $\hat{\Lambda}$. Next, if $\min\{\hat{\lambda}_i\} < 0$, then $k = \min\{\hat{\lambda}_i\}$. Under $\hat{\Lambda}'$, the scoring function becomes:

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

Where the last step follows from the fact that $\sum_i \tilde{f}(D,Q)_i$ is constant (= K) by definition. Thus, the transformation preserves the rankings. We have therefore shown that any solution to the augmented problem over $\mathbb{R}^{d+1} \setminus \{0\mathbf{I}\}$ can be transformed into a rank-equivalent solution over $\mathbb{P}^d$, thus completing the proof $\square$

Since the two solutions are rank-equivalent, they produce the same value for E. Therefore, for a given training set and any scoring function in $\mathcal{S}$, we only need to solve the optimization over the multinomial manifold and rank documents according to a simple linear function. This allows us to always bracket a global extrema, 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 original parameter space. This can be done by transforming each weight by $\hat{\lambda}'_i = \hat{\lambda}_i - \hat{\lambda}_{d+1}$ for all i and renormalizing. The proof that this is valid is similar to the proof given above. This results in the weight associated with the augmented feature being set to 0. Thus the solution is again over the original space.

3.4 Summary

The following summarizes the proposed procedure:

1. Construct $\tilde{f}(D,Q)$ from $f(D,Q)$ for each feature vector in $\mathcal{T}$.
2. Solve 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. Rank documents for query Q using scoring function $\Lambda^T f(D,Q)$.

Steps 1, 3 and 4 are straightforward. Step 2 is the most important part of the method. We now describe one possible numerical technique for solving the optimization procedure required in this step.

	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 *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 of document D, $|C|$ is the length of the collection, and N is the number of documents in the collection.

3.5 Parameter Estimation

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. 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 desirable form and certain provisions will have to be made.

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 updated parameter estimate will be a point on the manifold. Therefore, after a step is taken in $\mathbb{R}^d$, we transform the point back onto the manifold, which we showed is always possible in Section 3.3. Recall that this transformation preserves the function value. Therefore, the optimization is implicitly being done in a space that we know how to optimize over ($\mathbb{R}^d$), but is continually being transformed 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 transformed 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 simple line search must be done to find the $\arg\max$.

4 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 [7] and language modeling [4]. 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.

Language modeling for document retrieval is a state-of-the-art probabilistic retrieval model inspired by similar models used in speech recognition. The model is straightforward to implement and use, but lacks the explicit ability to represent arbitrary query-document features. On the other hand, Nallapati's SVM model casts ranking as a classification problem,

	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 and topics (queries).

with relevant documents making up the the positive class and non-relevant documents making up the negative class. Such a model allows arbitrary query-document features. After a linear SVM is trained on the training data, new documents are ranked using the resulting classifier. This is an example of an ordinal regression approach that ignores the ordinal nature of the data. We attempted to train a model using so-called ranking SVMs, described in [6], which considers the ordinal relationships in the training data, but stopped after five days of training. We note that language modeling, Nallapati's SVM approach, and our approach each take less than an hour to train[3].

We evaluate and 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. Mean average precision (MAP), a popular evaluation metric in information retrieval, is used to evaluate the effectiveness of the models [2].

When each 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. For the SVM model, we follow [7] and balance the training data by undersampling the majority (non-relevant) class. Our model, denoted by RankMax, is trained using coordinate ascent (see Section 3.5) with $E = \text{MAP}(\mathcal{R}_\Lambda; T)$ and the same features as the SVM model. The features are given in Table 1 and discussed in [7]. The language modeling run ranks documents via query likelihood, with document models estimated using Bayesian (Dirichlet) smoothing [4]. The language model is trained by finding the smoothing parameter that maximizes the mean average on the training data.

The results are given in Table 3. For the numbers in the table, the trained model is used to rank documents from 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 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.

5 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

[3] All SVM experiments use SVM^{light} (http://svmlight.joachims.org/).

	Disks 1,2		Disk 3		Disks 4,5	
	Train	Test	Train	Test	Train	Test
SVM	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: Mean average precision results for various *ad hoc* retrieval data sets and training methods. The † and ‡ denotes significant improvement over language modeling and the SVM model, respectively. Tests done using a one tailed paired t-test at the 0.05 level.

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 significantly reduces the number of local extrema is guaranteed to bracket a global extrema. A simple optimization technique that makes no assumption about the underlying objective function was presented. Finally, we applied the methods developed to the task of *ad hoc* information retrieval. It was shown that our parameter estimation paradigm significantly outperforms other models, including those based on SVMs, 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. Another area that needs explored is how well the simple coordinate ascent algorithm scales to large numbers of parameters and how easy or difficult different classes of features/evaluation metrics are to optimize.

Acknowledgments

This work was supported in part by the CIIR, the CIA, the NSA and NSF under NSF grant #IIS-0326249, in part by ARDA and NSF grant #CCF-0205575 , and in part by DARPA, through the Department of the Interior, NBC, Acquisition Services Division, under contract number NBCHD030010. Any opinions, findings and conclusions or recommendations expressed in this material are the authors 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] K. Crammer and Y. Singer. Pranking with ranking. In *Proceedings of Advances in Neural Information Processing Systems*, 2001.

[4] W. B. Croft and J. Lafferty. *Language Modeling for Information Retrieval*. Kluwer Academic Publishers, 2003.

[5] 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.

[6] T. Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the eighth ACM SIGKDD*, pages 133–142, 2002.

[7] R. Nallapati. Discriminative models for information retrieval. In *Proceedings of the 27th Annual International ACM SIGIR*, pages 64–71, 2004.

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