A Support Vector Method for Optimizing Average Precision

Yisong Yue
Cornell University
Ithaca, NY, USA
yyue@cs.cornell.edu

Thomas Finley
Cornell University
Ithaca, NY, USA
tomf@cs.cornell.edu

Filip Radlinski
Cornell University
Ithaca, NY, USA
filip@cs.cornell.edu

Thorsten Joachims
Cornell University
Ithaca, NY, USA
tj@cs.cornell.edu

ABSTRACT

Machine learning is commonly used to improve ranked retrieval systems. Due to computational difficulties, few learning techniques have been developed to directly optimize for mean average precision (MAP), despite its widespread use in evaluating such systems. Existing approaches optimizing MAP either do not find a globally optimal solution, or are computationally expensive. In contrast, we present a general SVM learning algorithm that efficiently finds a globally optimal solution to a straightforward relaxation of MAP. We evaluate our approach using the TREC 9 and TREC 10 Web Track corpora (WT10g), comparing against SVMs optimized for accuracy and ROCArea. In most cases we show our method to produce statistically significant improvements in MAP scores.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Retrieval Models

General Terms

Algorithm, Theory, Experimentation

Keywords

Machine Learning for Information Retrieval, Support Vector Machines, Ranking

1. INTRODUCTION

State of the art information retrieval systems commonly use machine learning techniques to learn ranking functions. However, most current approaches do not optimize for the evaluation measure most often used, namely Mean Average Precision (MAP).

Instead, current algorithms tend to take one of two general approaches. The first approach is to learn a model that estimates the probability of a document being relevant given a query (e.g., [18, 14]). If solved effectively, the ranking with best MAP performance can easily be derived from the probabilities of relevance. However, achieving high MAP only requires finding a good ordering of the documents. As a result, finding good probabilities requires solving a more difficult problem than necessary, likely requiring more training data to achieve the same MAP performance.

The second common approach is to learn a function that maximizes a surrogate measure. Performance measures optimized include accuracy [17, 15], ROCArea [1, 5, 10, 11, 13, 21] or modifications of ROCArea [4], and NDCG [2, 3]. Learning a model to optimize for such measures might result in suboptimal MAP performance. In fact, although some previous systems have obtained good MAP performance, it is known that neither achieving optimal accuracy nor ROCArea can guarantee optimal MAP performance[7].

In this paper, we present a general approach for learning ranking functions that maximize MAP performance. Specifically, we present an SVM algorithm that globally optimizes a hinge-loss relaxation of MAP. This approach simplifies the process of obtaining ranking functions with high MAP performance by avoiding additional intermediate steps and heuristics. The new algorithm also makes it conceptually just as easy to optimize SVMs for MAP as was previously possible only for accuracy and ROCArea.

In contrast to recent work directly optimizing for MAP performance by Metzler & Croft [16] and Caruana et al. [6], our technique is computationally efficient while finding a globally optimal solution. Like [6, 16], our method learns a linear model, but is much more efficient in practice and, unlike [16], can handle many thousands of features.

We now describe the algorithm in detail and provide proof of correctness. Following this, we provide an analysis of running time. We finish with empirical results from experiments on the TREC 9 and TREC 10 Web Track corpus. We have also developed a software package implementing our algorithm that is available for public use[1].

2. THE LEARNING PROBLEM

Following the standard machine learning setup, our goal is to learn a function $h : \mathcal{X} \to \mathcal{Y}$ between an input space $\mathcal{X}$ (all possible queries) and output space $\mathcal{Y}$ (rankings over a corpus). In order to quantify the quality of a prediction, $\hat{\mathbf{y}} = h(\mathbf{x})$, we will consider a loss function $\Delta : \mathcal{Y} \times \mathcal{Y} \to \Re$. $\Delta(\mathbf{y}, \hat{\mathbf{y}})$ quantifies the penalty for making prediction $\hat{\mathbf{y}}$ if the correct output is $\mathbf{y}$. The loss function allows us to incorporate specific performance measures, which we will exploit

SIGIR'07, July 23–27, 2007, Amsterdam, The Netherlands.
Copyright 2007 ACM 978-1-59593-597-7/07/0007 ...$5.00.

[1] http://svmrank.yisongyue.com

for optimizing MAP. We restrict ourselves to the supervised learning scenario, where input/output pairs $(\mathbf{x}, \mathbf{y})$ are available for training and are assumed to come from some fixed distribution $P(\mathbf{x}, \mathbf{y})$. The goal is to find a function h such that the risk (i.e., expected loss),

$$R_P^{\Delta}(h) = \int_{\mathcal{X}\times\mathcal{Y}} \Delta(\mathbf{y}, h(\mathbf{x}))dP(\mathbf{x}, \mathbf{y}),$$

is minimized. Of course, $P(\mathbf{x}, \mathbf{y})$ is unknown. But given a finite set of training pairs, $S = \{(\mathbf{x}_i, \mathbf{y}_i) \in \mathcal{X} \times \mathcal{Y} : i = 1, \ldots, n\}$, the performance of h on S can be measured by the empirical risk,

$$R_S^{\Delta}(h) = \frac{1}{n}\sum_{i=1}^{n} \Delta(\mathbf{y}_i, h(\mathbf{x}_i)).$$

In the case of learning a ranked retrieval function, $\mathcal{X}$ denotes a space of queries, and $\mathcal{Y}$ the space of (possibly weak) rankings over some corpus of documents $\mathcal{C} = \{d_1, \ldots, d_{|\mathcal{C}|}\}$.

We can define average precision loss as

$$\Delta_{map}(\mathbf{y}, \hat{\mathbf{y}}) = 1 - \text{MAP}(rank(\mathbf{y}), rank(\hat{\mathbf{y}})),$$

where $rank(\mathbf{y})$ is a vector of the rank values of each document in $\mathcal{C}$. For example, for a corpus of two documents, $\{d_1, d_2\}$, with d_1 having higher rank than d_2, $rank(\mathbf{y}') = (1, 0)$. We assume true rankings have two rank values, where relevant documents have rank value 1 and non-relevant documents rank value 0. We further assume that all predicted rankings are complete rankings (no ties).

Let $p = rank(\mathbf{y})$ and $\hat{p} = rank(\hat{\mathbf{y}})$. The average precision score is defined as

$$\text{MAP}(p, \hat{p}) = \frac{1}{rel}\sum_{j:p_j=1} Prec@j,$$

where $rel = |\{i : p_i = 1\}|$ is the number of relevant documents, and $Prec@j$ is the percentage of relevant documents in the top j documents in predicted ranking $\hat{\mathbf{y}}$. MAP is the mean of the average precision scores of a group of queries.

2.1 MAP vs ROCArea

Most learning algorithms optimize for accuracy or ROCArea. While optimizing for these measures might achieve good MAP performance, we use two simple examples to show it can also be suboptimal in terms of MAP.

ROCArea assigns equal penalty to each misordering of a relevant/non-relevant pair. In contrast, MAP assigns greater penalties to misorderings higher up in the predicted ranking. Using our notation, ROCArea can be defined as

$$\text{ROC}(p, \hat{p}) = \frac{1}{rel \cdot (|\mathcal{C}| - rel)} \sum_{i:p_i=1}\sum_{j:p_j=0} \mathbf{1}_{[\hat{p}_i > \hat{p}_j]},$$

where p is the true (weak) ranking, $\hat{p}$ is the predicted ranking, and $\mathbf{1}_{[b]}$ is the indicator function conditioned on b.

Doc ID	1	2	3	4	5	6	7	8
p	1	0	0	0	0	1	1	0
$rank(h_1(\mathbf{x}))$	8	7	6	5	4	3	2	1
$rank(h_2(\mathbf{x}))$	1	2	3	4	5	6	7	8

Table 1: Toy Example and Models

Suppose we have a hypothesis space with only two hypothesis functions, h_1 and h_2, as shown in Table 1. These two hypotheses predict a ranking for query $\mathbf{x}$ over a corpus of eight documents.

Hypothesis	MAP	ROCArea
$h_1(\mathbf{x})$	0.59	0.47
$h_2(\mathbf{x})$	0.51	0.53

Table 2: Performance of Toy Models

Table 2 shows the MAP and ROCArea scores of h_1 and h_2. Here, a learning method which optimizes for ROCArea would choose h_2 since that results in a higher ROCArea score, but this yields a suboptimal MAP score.

2.2 MAP vs Accuracy

Using a very similar example, we now demonstrate how optimizing for accuracy might result in suboptimal MAP. Models which optimize for accuracy are not directly concerned with the ranking. Instead, they learn a threshold such that documents scoring higher than the threshold can be classified as relevant and documents scoring lower as non-relevant.

Doc ID	1	2	3	4	5	6	7	8	9	10	11
p	1	0	0	0	0	1	1	1	1	0	0
$rank(h_1(\mathbf{x}))$	11	10	9	8	7	6	5	4	3	2	1
$rank(h_2(\mathbf{x}))$	1	2	3	4	5	6	7	8	9	10	11

Table 3: Toy Example and Models

We consider again a hypothesis space with two hypotheses. Table 3 shows the predictions of the two hypotheses on a single query $\mathbf{x}$.

Hypothesis	MAP	Best Acc.
$h_1(q)$	0.70	0.64
$h_2(q)$	0.64	0.73

Table 4: Performance of Toy Models

Table 4 shows the MAP and best accuracy scores of $h_1(q)$ and $h_2(q)$. The best accuracy refers to the highest achievable accuracy on that ranking when considering all possible thresholds. For instance, with $h_1(q)$, a threshold between documents 1 and 2 gives 4 errors (documents 6-9 incorrectly classified as non-relevant), yielding an accuracy of 0.64. Similarly, with $h_2(q)$, a threshold between documents 5 and 6 gives 3 errors (documents 10-11 incorrectly classified as relevant, and document 1 as non-relevant), yielding an accuracy of 0.73. A learning method which optimizes for accuracy would choose h_2 since that results in a higher accuracy score, but this yields a suboptimal MAP score.

3. OPTIMIZING AVERAGE PRECISION

We build upon the approach used by [13] for optimizing ROCArea. Unlike ROCArea, however, MAP does not decompose linearly in the examples and requires a substantially extended algorithm, which we describe in this section.

Recall that the true ranking is a weak ranking with two rank values (relevant and non-relevant). Let $\mathcal{C}^x$ and $\mathcal{C}^{\bar{x}}$ denote the set of relevant and non-relevant documents of $\mathcal{C}$ for query $\mathbf{x}$, respectively.

We focus on functions which are parametrized by a weight vector $\mathbf{w}$, and thus wish to find $\mathbf{w}$ to minimize the empirical risk, $R^{\Delta}_{S}(\mathbf{w}) \equiv R^{\Delta}_{S}(h(\cdot;\mathbf{w}))$. Our approach is to learn a *discriminant function* $F : \mathcal{X} \times \mathcal{Y} \to \Re$ over input-output pairs. Given query $\mathbf{x}$, we can derive a prediction by finding the ranking $\mathbf{y}$ that maximizes the discriminant function:

$$h(\mathbf{x};\mathbf{w}) = \operatorname*{argmax}_{\mathbf{y}\in\mathcal{Y}} F(\mathbf{x},\mathbf{y};\mathbf{w}). \tag{1}$$

We assume F to be linear in some *combined feature representation* of inputs and outputs $\Psi(\mathbf{x},\mathbf{y}) \in \mathbb{R}^N$, i.e.,

$$F(\mathbf{x},\mathbf{y};\mathbf{w}) = \mathbf{w}^T\Psi(\mathbf{x},\mathbf{y}). \tag{2}$$

The combined feature function we use is

$$\Psi(\mathbf{x},\mathbf{y}) = \frac{1}{|C^x|\cdot|C^{\bar{x}}|}\sum_{i:d_i\in\mathcal{C}^x}\sum_{j:d_j\in C^{\bar{x}}}\left[y_{ij}\left(\phi(\mathbf{x},d_i)-\phi(\mathbf{x},d_j)\right)\right],$$

where $\phi : \mathcal{X} \times \mathcal{C} \to \Re^N$ is a feature mapping function from a query/document pair to a point in N dimensional space[2]. We represent rankings as a matrix of pairwise orderings, $\mathcal{Y} \subset \{-1,0,+1\}^{|\mathcal{C}|\times|\mathcal{C}|}$. For any $\mathbf{y} \in \mathcal{Y}$, $y_{ij} = +1$ if d_i is ranked ahead of d_j, and $y_{ij} = -1$ if d_j is ranked ahead of d_i, and $y_{ij} = 0$ if d_i and d_j have equal rank. We consider only matrices which correspond to valid rankings (i.e, obeying antisymmetry and transitivity). Intuitively, Ψ is a summation over the vector differences of all relevant/non-relevant document pairings. Since we assume predicted rankings to be complete rankings, y_{ij} is either $+1$ or -1 (never 0).

Given a learned weight vector $\mathbf{w}$, predicting a ranking (i.e. solving equation (1)) given query $\mathbf{x}$ reduces to picking each y_{ij} to maximize $\mathbf{w}^T\Psi(\mathbf{x},\mathbf{y})$. As is also discussed in [13], this is attained by sorting the documents by $\mathbf{w}^T\phi(\mathbf{x},d)$ in descending order. We will discuss later the choices of ϕ we used for our experiments.

3.1 Structural SVMs

The above formulation is very similar to learning a straightforward linear model while training on the pairwise difference of relevant/non-relevant document pairings. Many SVM-based approaches optimize over these pairwise differences (e.g., [5, 10, 13, 4]), although these methods do not optimize for MAP during training. Previously, it was not clear how to incorporate non-linear multivariate loss functions such as MAP loss directly into global optimization problems such as SVM training. We now present a method based on structural SVMs [19] to address this problem.

We use the structural SVM formulation, presented in Optimization Problem 1, to learn a $\mathbf{w} \in \mathbb{R}^N$.

Optimization Problem 1. (Structural SVM)

$$\min_{\mathbf{w},\xi\geq 0} \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{n}\sum_{i=1}^{n}\xi_i \tag{3}$$

$$\text{s.t. } \forall i, \forall \mathbf{y} \in \mathcal{Y} \setminus \mathbf{y}_i :$$

$$\mathbf{w}^T\Psi(\mathbf{x}_i,\mathbf{y}_i) \geq \mathbf{w}^T\Psi(\mathbf{x}_i,\mathbf{y}) + \Delta(\mathbf{y}_i,\mathbf{y}) - \xi_i \tag{4}$$

The objective function to be minimized (3) is a tradeoff between model complexity, $\|\mathbf{w}\|^2$, and a hinge loss relaxation of MAP loss, $\sum\xi_i$. As is usual in SVM training, C is a parameter that controls this tradeoff and can be tuned to achieve good performance in different training tasks.

Algorithm 1 Cutting plane algorithm for solving OP 1 within tolerance ϵ.

```
Input: $(\mathbf{x}_1,\mathbf{y}_1),\ldots,(\mathbf{x}_n,\mathbf{y}_n)$, $C$, $\epsilon$
$\mathcal{W}_i \leftarrow \emptyset$ for all $i = 1,\ldots,n$
repeat
    for $i = 1,\ldots,n$ do
        $H(\mathbf{y};\mathbf{w}) \equiv \Delta(\mathbf{y}_i,\mathbf{y}) + \mathbf{w}^T\Psi(\mathbf{x}_i,\mathbf{y}) - \mathbf{w}^T\Psi(\mathbf{x}_i,\mathbf{y}_i)$
        compute $\hat{\mathbf{y}} = \operatorname{argmax}_{\mathbf{y}\in\mathcal{Y}} H(\mathbf{y};\mathbf{w})$
        compute $\xi_i = \max\{0, \max_{\mathbf{y}\in\mathcal{W}_i} H(\mathbf{y};\mathbf{w})\}$
        if $H(\hat{\mathbf{y}};\mathbf{w}) > \xi_i + \epsilon$ then
            $\mathcal{W}_i \leftarrow \mathcal{W}_i \cup \{\hat{\mathbf{y}}\}$
            $\mathbf{w} \leftarrow$ optimize (3) over $\mathcal{W} = \bigcup_i \mathcal{W}_i$
        end if
    end for
until no $\mathcal{W}_i$ has changed during iteration
```

For each $(\mathbf{x}_i,\mathbf{y}_i)$ in the training set, a set of constraints of the form in equation (4) is added to the optimization problem. Note that $\mathbf{w}^T\Psi(\mathbf{x},\mathbf{y})$ is exactly our discriminant function $F(\mathbf{x},\mathbf{y};\mathbf{w})$ (see equation (2)). During prediction, our model chooses the ranking which maximizes the discriminant (1). If the discriminant value for an incorrect ranking $\mathbf{y}$ is greater than for the true ranking $\mathbf{y}_i$ (e.g., $F(\mathbf{x}_i,\mathbf{y};\mathbf{w}) > F(\mathbf{x}_i,\mathbf{y}_i;\mathbf{w})$), then corresponding slack variable, ξ_i, must be at least $\Delta(\mathbf{y}_i,\mathbf{y})$ for that constraint to be satisfied. Therefore, the sum of slacks, $\sum\xi_i$, upper bounds the MAP loss. This is stated formally in Proposition 1.

Proposition 1. *Let $\xi^*(\mathbf{w})$ be the optimal solution of the slack variables for OP 1 for a given weight vector $\mathbf{w}$. Then $\frac{1}{n}\sum_{i=1}^{n}\xi_i$ is an upper bound on the empirical risk $R^{\Delta}_{S}(\mathbf{w})$. (see [19] for proof)*

Proposition 1 shows that OP 1 learns a ranking function that optimizes an upper bound on MAP error on the training set. Unfortunately there is a problem: a constraint is required for every possible wrong output $\mathbf{y}$, and the number of possible wrong outputs is exponential in the size of $\mathcal{C}$. Fortunately, we may employ Algorithm 1 to solve OP 1. Algorithm 1 is a cutting plane algorithm, iteratively introducing constraints until we have solved the original problem within a desired tolerance ϵ [19]. The algorithm starts with no constraints, and iteratively finds for each example $(\mathbf{x}_i,\mathbf{y}_i)$ the output $\hat{\mathbf{y}}$ associated with the most violated constraint. If the corresponding constraint is violated by more than ϵ we introduce $\hat{\mathbf{y}}$ into the working set $\mathcal{W}_i$ of active constraints for example i, and re-solve (3) using the updated $\mathcal{W}$. It can be shown that Algorithm 1's outer loop is guaranteed to halt within a polynomial number of iterations for any desired precision ϵ.

Theorem 1. *Let $\bar{R} = \max_i \max_{\mathbf{y}} \|\Psi(\mathbf{x}_i,\mathbf{y}_i) - \Psi(\mathbf{x}_i,\mathbf{y})\|$, $\bar{\Delta} = \max_i \max_{\mathbf{y}} \Delta(\mathbf{y}_i,\mathbf{y})$, and for any $\epsilon > 0$, Algorithm 1 terminates after adding at most*

$$\max\left\{\frac{2n\bar{\Delta}}{\epsilon}, \frac{8C\bar{\Delta}\bar{R}^2}{\epsilon^2}\right\}$$

constraints to the working set $\mathcal{W}$. (see [19] for proof)

However, within the inner loop of this algorithm we have

[2] For example, one dimension might be the number of times the query words appear in the document.

to compute $\operatorname{argmax}_{\mathbf{y}\in\mathcal{Y}} H(\mathbf{y};\mathbf{w})$, where

$$H(\mathbf{y};\mathbf{w}) = \Delta(\mathbf{y}_i,\mathbf{y}) + \mathbf{w}^T\Psi(\mathbf{x}_i,\mathbf{y}) - \mathbf{w}^T\Psi(\mathbf{x}_i,\mathbf{y}_i),$$

or equivalently,

$$\underset{\mathbf{y}\in\mathcal{Y}}{\operatorname{argmax}}\, \Delta(\mathbf{y}_i,\mathbf{y}) + \mathbf{w}^T\Psi(\mathbf{x}_i,\mathbf{y}),$$

since $\mathbf{w}^T\Psi(\mathbf{x}_i,\mathbf{y}_i)$ is constant with respect to $\mathbf{y}$. Though closely related to the classification procedure, this has the substantial complication that we must contend with the additional $\Delta(\mathbf{y}_i,\mathbf{y})$ term. Without the ability to efficiently find the most violated constraint (i.e., solve $\operatorname{argmax}_{\mathbf{y}\in\mathcal{Y}} H(\mathbf{y},\mathbf{w})$), the constraint generation procedure is not tractable.

3.2 Finding the Most Violated Constraint

Using OP 1 and optimizing to ROCArea loss (Δ_{roc}), the problem of finding the most violated constraint, or solving $\operatorname{argmax}_{\mathbf{y}\in\mathcal{Y}} H(\mathbf{y},\mathbf{w})$ (henceforth $\operatorname{argmax} H$), is addressed in [13]. Solving $\operatorname{argmax} H$ for Δ_{map} is more difficult. This is primarily because ROCArea decomposes nicely into a sum of scores computed independently on each relative ordering of a relevant/non-relevant document pair. MAP, on the other hand, does not decompose in the same way as ROCArea. The main algorithmic contribution of this paper is an efficient method for solving $\operatorname{argmax} H$ for Δ_{map}.

One useful property of Δ_{map} is that it is invariant to swapping two documents with equal relevance. For example, if documents d_a and d_b are both relevant, then swapping the positions of d_a and d_b in any ranking does not affect Δ_{map}. By extension, Δ_{map} is invariant to any arbitrary permutation of the relevant documents amongst themselves and of the non-relevant documents amongst themselves. However, this reshuffling will affect the discriminant score, $\mathbf{w}^T\Psi(\mathbf{x},\mathbf{y})$. This leads us to Observation 1.

OBSERVATION 1. *Consider rankings which are constrained by fixing the relevance at each position in the ranking (e.g., the 3rd document in the ranking must be relevant). Every ranking which satisfies the same set of constraints will have the same* Δ_{map}. *If the relevant documents are sorted by* $\mathbf{w}^T\phi(\mathbf{x},d)$ *in descending order, and the non-relevant documents are likewise sorted by* $\mathbf{w}^T\phi(\mathbf{x},d)$*, then the interleaving of the two sorted lists which satisfies the constraints will maximize* H *for that constrained set of rankings.*

Observation 1 implies that in the ranking which maximizes H, the relevant documents will be sorted by $\mathbf{w}^T\phi(\mathbf{x},d)$, and the non-relevant documents will also be sorted likewise. By first sorting the relevant and non-relevant documents, the problem is simplified to finding the optimal interleaving of two sorted lists. For the rest of our discussion, we assume that the relevant documents and non-relevant documents are both sorted by descending $\mathbf{w}^T\phi(\mathbf{x},d)$. For convenience, we also refer to relevant documents as $\{d^x_1,\ldots d^x_{|\mathcal{C}^x|}\} = \mathcal{C}^x$, and non-relevant documents as $\{d^{\bar{x}}_1,\ldots d^{\bar{x}}_{|\mathcal{C}^{\bar{x}}|}\} = \mathcal{C}^{\bar{x}}$.

We define $\delta_j(i_1,i_2)$, with $i_1 < i_2$, as the change in H from when the highest ranked relevant document *ranked after* $d^{\bar{x}}_j$ is $d^x_{i_1}$ to when it is $d^x_{i_2}$. For $i_2 = i_1+1$, we have

$$\delta_j(i,i+1) = \frac{1}{|\mathcal{C}^x|}\left(\frac{j}{j+i} - \frac{j-1}{j+i-1}\right) - 2\cdot(s^x_i - s^{\bar{x}}_j), \quad (5)$$

where $s_i = \mathbf{w}^T\phi(\mathbf{x},d_i)$. The first term in (5) is the change in Δ_{map} when the ith relevant document has j non-relevant documents ranked before it, as opposed to $j-1$. The second term is the change in the discriminant score, $\mathbf{w}^T\Psi(\mathbf{x},\mathbf{y})$, when y_{ij} changes from $+1$ to -1.

$$\begin{array}{c}
\hline
\ldots, d^x_i, d^{\bar{x}}_j, d^x_{i+1}, \ldots \\
\ldots, d^{\bar{x}}_j, d^x_i, d^x_{i+1}, \ldots \\
\hline
\end{array}$$

Figure 1: Example for $\delta_j(i,i+1)$

Figure 1 gives a conceptual example for $\delta_j(i,i+1)$. The bottom ranking differs from the top only where $d^{\bar{x}}_j$ slides up one rank. The difference in the value of H for these two rankings is exactly $\delta_j(i,i+1)$.

For any $i_1 < i_2$, we can then define $\delta_j(i_1,i_2)$ as

$$\delta_j(i_1,i_2) = \sum_{k=i_1}^{i_2-1} \delta_j(k,k+1), \quad (6)$$

or equivalently,

$$\delta_j(i_1,i_2) = \sum_{k=i_1}^{i_2-1}\left[\frac{1}{|\mathcal{C}^x|}\left(\frac{j}{j+k} - \frac{j-1}{j+k-1}\right) - 2\cdot(s^x_k - s^{\bar{x}}_j)\right].$$

Let $o_1,\ldots,o_{|\mathcal{C}^{\bar{x}}|}$ encode the positions of the non-relevant documents, where $d^x_{o_j}$ is the highest ranked relevant document ranked *after* the jth non-relevant document. Due to Observation 1, this encoding uniquely identifies a complete ranking. We can recover the ranking as

$$y_{ij} = \begin{cases} 0 & \text{if } i = j \\ sign(s_i - s_j) & \text{if } d_i, d_j \text{ equal relevance} \\ sign(o_{j'} - i' - 0.5) & \text{if } d_i = d^x_{i'}, d_j = d^{\bar{x}}_{j'} \\ sign(j' - o_{i'} + 0.5) & \text{if } d_i = d^{\bar{x}}_{i'}, d_j = d^x_{j'} \end{cases}. \quad (7)$$

We can now reformulate H into a new objective function,

$$H'(o_1,\ldots,o_{|\mathcal{C}^{\bar{x}}|}|\mathbf{w}) = H(\bar{\mathbf{y}}|\mathbf{w}) + \sum_{k=1}^{|\mathcal{C}^{\bar{x}}|}\delta_k(o_k,|\mathcal{C}^x|+1),$$

where $\bar{y}$ is the true (weak) ranking. Conceptually H' starts with a perfect ranking $\bar{\mathbf{y}}$, and adds the change in H when each successive non-relevant document slides up the ranking.

We can then reformulate the $\operatorname{argmax} H$ problem as

$$\operatorname{argmax} H' = \underset{o_1,\ldots,o_{|\mathcal{C}^{\bar{x}}|}}{\operatorname{argmax}} \sum_{k=1}^{|\mathcal{C}^{\bar{x}}|}\delta_k(o_k,|\mathcal{C}^x|+1) \quad (8)$$

s.t.

$$o_1 \le \ldots \le o_{|\mathcal{C}^{\bar{x}}|}. \quad (9)$$

Algorithm 2 describes the algorithm used to solve equation (8). Conceptually, Algorithm 2 starts with a perfect ranking. Then for each successive non-relevant document, the algorithm modifies the solution by sliding that document up the ranking to locally maximize H' while keeping the positions of the other non-relevant documents constant.

3.2.1 Proof of Correctness

Algorithm 2 is greedy in the sense that it finds the best position of each non-relevant document independently from the other non-relevant documents. In other words, the algorithm maximizes H' for each non-relevant document, $d^{\bar{x}}_j$,

Algorithm 2 Finding the Most Violated Constraint ($\text{argmax}\, H$) for Algorithm 1 with Δ_{map}

```
1: Input: $\mathbf{w}$, $\mathcal{C}^x$, $\mathcal{C}^{\bar{x}}$
2: sort $\mathcal{C}^x$ and $\mathcal{C}^{\bar{x}}$ in descending order of $\mathbf{w}^T\phi(x,d)$
3: $s_i^x \leftarrow \mathbf{w}^T\phi(x,d_i^x)$, $i = 1,\ldots,|\mathcal{C}^x|$
4: $s_i^{\bar{x}} \leftarrow \mathbf{w}^T\phi(x,d_i^{\bar{x}})$, $i = 1,\ldots,|\mathcal{C}^{\bar{x}}|$
5: for $j = 1,\ldots,|\mathcal{C}^{\bar{x}}|$ do
6:     $opt_j \leftarrow \text{argmax}_k\, \delta_j(k, |\mathcal{C}^x|+1)$
7: end for
8: encode $\hat{\mathbf{y}}$ according to (7)
9: return $\hat{\mathbf{y}}$
```

without considering the positions of the other non-relevant documents, and thus ignores the constraints of (9).

In order for the solution to be feasible, the jth non-relevant document must be ranked after the first $j-1$ non-relevant documents, thus satisfying

$$opt_1 \leq opt_2 \leq \ldots \leq opt_{|\mathcal{C}^{\bar{x}}|}. \qquad (10)$$

If the solution is feasible, the it clearly solves (8). Therefore, it suffices to prove that Algorithm 2 satisfies (10). We first prove that $\delta_j(\cdot,\cdot)$ is monotonically decreasing in j.

Lemma 1. *For any $1 \leq i_1 < i_2 \leq |\mathcal{C}^x|+1$ and $1 \leq j < |\mathcal{C}^{\bar{x}}|$, it must be the case that*

$$\delta_{j+1}(i_1,i_2) \leq \delta_j(i_1,i_2).$$

Proof. Recall from (6) that both $\delta_j(i_1,i_2)$ and $\delta_{j+1}(i_1,i_2)$ are summations of $i_2 - i_1$ terms. We will show that each term in the summation of $\delta_{j+1}(i_1,i_2)$ is no greater than the corresponding term in $\delta_j(i_1,i_2)$, or

$$\delta_{j+1}(k,k+1) \leq \delta_j(k,k+1)$$

for $k = i_1,\ldots,i_2-1$.

Each term in $\delta_j(k,k+1)$ and $\delta_{j+1}(k,k+1)$ can be further decomposed into two parts (see (5)). We will show that each part of $\delta_{j+1}(k,k+1)$ is no greater than the corresponding part in $\delta_j(k,k+1)$. In other words, we will show that both

$$\frac{j+1}{j+k+1} - \frac{j}{j+k} \leq \frac{j}{j+k} - \frac{j-1}{j+k-1} \qquad (11)$$

and

$$-2\cdot(s_k^x - s_{j+1}^{\bar{x}}) \leq -2\cdot(s_k^x - s_j^{\bar{x}}) \qquad (12)$$

are true for the aforementioned values of j and k.

It is easy to see that (11) is true by observing that for any two positive integers $1 \leq a < b$,

$$\frac{a+1}{b+1} - \frac{a}{b} \leq \frac{a}{b} - \frac{a-1}{b-1},$$

and choosing $a = j$ and $b = j+k$.

The second inequality (12) holds because Algorithm 2 first sorts $d^{\bar{x}}$ in descending order of $s^{\bar{x}}$, implying $s_{j+1}^{\bar{x}} \leq s_j^{\bar{x}}$.

Thus we see that each term in δ_{j+1} is no greater than the corresponding term in δ_j, which completes the proof. □

The result of Lemma 1 leads directly to our main correctness result:

Theorem 2. *In Algorithm 2, the computed values of opt_j satisfy (10), implying that the solution returned by Algorithm 2 is feasible and thus optimal.*

Proof. We will prove that

$$opt_j \leq opt_{j+1}$$

holds for any $1 \leq j < |\mathcal{C}^{\bar{x}}|$, thus implying (10).

Since Algorithm 2 computes opt_j as

$$opt_j = \underset{k}{\text{argmax}}\, \delta_j(k, |\mathcal{C}^x|+1), \qquad (13)$$

then by definition of δ_j (6), for any $1 \leq i < opt_j$,

$$\delta_j(i, opt_j) = \delta_j(i, |\mathcal{C}^x|+1) - \delta_j(opt_j, |\mathcal{C}^x|+1) < 0.$$

Using Lemma 1, we know that

$$\delta_{j+1}(i, opt_j) \leq \delta_j(i, opt_j) < 0,$$

which implies that for any $1 \leq i < opt_j$,

$$\delta_{j+1}(i, |\mathcal{C}^x|+1) - \delta_{j+1}(opt_j, |\mathcal{C}^x|+1) < 0.$$

Suppose for contradiction that $opt_{j+1} < opt_j$. Then

$$\delta_{j+1}(opt_{j+1}, |\mathcal{C}^x|+1) < \delta_{j+1}(opt_j, |\mathcal{C}^x|+1),$$

which contradicts (13). Therefore, it must be the case that $opt_j \leq opt_{j+1}$, which completes the proof. □

3.2.2 Running Time

The running time of Algorithm 2 can be split into two parts. The first part is the sort by $\mathbf{w}^T\phi(x,d)$, which requires $O(n \log n)$ time, where $n = |\mathcal{C}^x| + |\mathcal{C}^{\bar{x}}|$. The second part computes each opt_j, which requires $O(|\mathcal{C}^x|\cdot|\mathcal{C}^{\bar{x}}|)$ time. Though in the worst case this is $O(n^2)$, the number of relevant documents, $|\mathcal{C}^x|$, is often very small (e.g., constant with respect to n), in which case the running time for the second part is simply $O(n)$. For most real-world datasets, Algorithm 2 is dominated by the sort and has complexity $O(n \log n)$.

Algorithm 1 is guaranteed to halt in a polynomial number of iterations [19], and each iteration runs Algorithm 2. Virtually all well-performing models were trained in a reasonable amount of time (usually less than one hour). Once training is complete, making predictions on query $\mathbf{x}$ using the resulting hypothesis $h(\mathbf{x}|\mathbf{w})$ requires only sorting by $\mathbf{w}^T\phi(\mathbf{x},d)$.

We developed our software using a Python interface[3] to SVM^{struct}, since the Python language greatly simplified the coding process. To improve performance, it is advisable to use the standard C implementation[4] of SVM^{struct}.

4. EXPERIMENT SETUP

The main goal of our experiments is to evaluate whether directly optimizing MAP leads to improved MAP performance compared to conventional SVM methods that optimize a substitute loss such as accuracy or ROCArea. We empirically evaluate our method using two sets of TREC Web Track queries, one each from TREC 9 and TREC 10 (topics 451-500 and 501-550), both of which used the WT10g corpus. For each query, TREC provides the relevance judgments of the documents. We generated our features using the scores of existing retrieval functions on these queries. While our method is agnostic to the meaning of the features, we chose to use existing retrieval functions as a simple yet effective way of acquiring useful features. As such, our

[3] http://www.cs.cornell.edu/~tomf/svmpython/

[4] http://svmlight.joachims.org/svm_struct.html

Dataset	Base Funcs	Features
TREC 9 Indri	15	750
TREC 10 Indri	15	750
TREC 9 Submissions	53	2650
TREC 10 Submissions	18	900

Table 5: Dataset Statistics

experiments essentially test our method's ability to re-rank the highly ranked documents (e.g., re-combine the scores of the retrieval functions) to improve MAP.

We compare our method against the best retrieval functions trained on (henceforth *base functions*), as well as against previously proposed SVM methods. Comparing with the best base functions tests our method's ability to learn a useful combination. Comparing with previous SVM methods allows us to test whether optimizing directly for MAP (as opposed to accuracy or ROCArea) achieves a higher MAP score in practice. The rest of this section describes the base functions and the feature generation method in detail.

4.1 Choosing Retrieval Functions

We chose two sets of base functions for our experiments. For the first set, we generated three indices over the WT10g corpus using Indri[5]. The first index was generated using default settings, the second used Porter-stemming, and the last used Porter-stemming and Indri's default stopwords.

For both TREC 9 and TREC 10, we used the description portion of each query and scored the documents using five of Indri's built-in retrieval methods, which are Cosine Similarity, TFIDF, Okapi, Language Model with Dirichlet Prior, and Language Model with Jelinek-Mercer Prior. All parameters were kept as their defaults.

We computed the scores of these five retrieval methods over the three indices, giving 15 base functions in total. For each query, we considered the scores of documents found in the union of the top 1000 documents of each base function.

For our second set of base functions, we used scores from the TREC 9 [8] and TREC 10 [9] Web Track submissions. We used only the non-manual, non-short submissions from both years. For TREC 9 and TREC 10, there were 53 and 18 such submissions, respectively. A typical submission contained scores of its top 1000 documents.

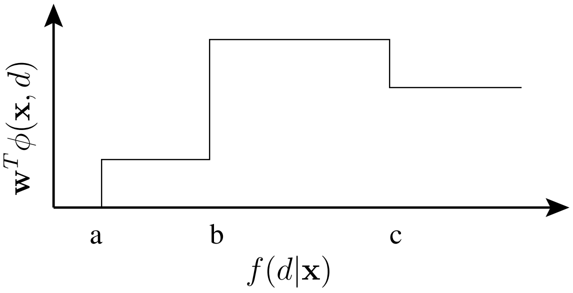

Figure 2: Example Feature Binning

4.2 Generating Features

In order to generate input examples for our method, a concrete instantiation of ϕ must be provided. For each document d scored by a set of retrieval functions $\mathcal{F}$ on query $\mathbf{x}$, we generate the features as a vector

$$\phi(\mathbf{x},d) = \langle \mathbf{1}_{[f(d|\mathbf{x})>k]} : \forall f \in \mathcal{F}, \forall k \in K_f \rangle,$$

where $f(d|\mathbf{x})$ denotes the score that retrieval function f assigns to document d for query $\mathbf{x}$, and each K_f is a set of real values. From a high level, we are expressing the score of each retrieval function using $|K_f|+1$ bins.

	TREC 9		TREC 10	
Model	MAP	W/L	MAP	W/L
$\text{SVM}^{\Delta}_{map}$	0.242	–	0.236	–
Best Func.	0.204	39/11 **	0.181	37/13 **
2nd Best	0.199	38/12 **	0.174	43/7 **
3rd Best	0.188	34/16 **	0.174	38/12 **

Table 6: Comparison with Indri Functions

Since we are using linear kernels, one can think of the learning problem as finding a good piecewise-constant combination of the scores of the retrieval functions. Figure 2 shows an example of our feature mapping method. In this example we have a single feature $\mathcal{F} = \{f\}$. Here, $K_f = \{a, b, c\}$, and the weight vector is $\mathbf{w} = \langle w_a, w_b, w_c \rangle$. For any document d and query $\mathbf{x}$, we have

$$\mathbf{w}^T\phi(\mathbf{x},d) = \begin{cases} 0 & \text{if } f(d|\mathbf{x}) < a \\ w_a & \text{if } a \le f(d|\mathbf{x}) < b \\ w_a + w_b & \text{if } b \le f(d|\mathbf{x}) < c \\ w_a + w_b + w_c & \text{if } c \le f(d|\mathbf{x}) \end{cases}.$$

This is expressed qualitatively in Figure 2, where w_a and w_b are positive, and w_c is negative.

We ran our main experiments using four choices of $\mathcal{F}$: the set of aforementioned Indri retrieval functions for TREC 9 and TREC 10, and the Web Track submissions for TREC 9 and TREC 10. For each $\mathcal{F}$ and each function $f \in \mathcal{F}$, we chose 50 values for K_f which are reasonably spaced and capture the sensitive region of f.

Using the four choices of $\mathcal{F}$, we generated four datasets for our main experiments. Table 5 contains statistics of the generated datasets. There are many ways to generate features, and we are not advocating our method over others. This was simply an efficient means to normalize the outputs of different functions and allow for a more expressive model.

5. EXPERIMENTS

For each dataset in Table 5, we performed 50 trials. For each trial, we train on 10 randomly selected queries, and select another 5 queries at random for a validation set. Models were trained using a wide range of C values. The model which performed best on the validation set was selected and tested on the remaining 35 queries.

All queries were selected to be in the training, validation and test sets the same number of times. Using this setup, we performed the same experiments while using our method ($\text{SVM}^{\Delta}_{map}$), an SVM optimizing for ROCArea ($\text{SVM}^{\Delta}_{roc}$) [13], and a conventional classification SVM (SVM_{acc}) [20]. All SVM methods used a linear kernel. We reported the average performance of all models over the 50 trials.

5.1 Comparison with Base Functions

In analyzing our results, the first question to answer is, can $\text{SVM}^{\Delta}_{map}$ learn a model which outperforms the best base

[5] http://www.lemurproject.org

	TREC 9		TREC 10	
Model	MAP	W/L	MAP	W/L
$\text{SVM}^{\Delta}_{map}$	0.290	–	0.287	–
Best Func.	0.280	28/22	0.283	29/21
2nd Best	0.269	30/20	0.251	36/14 **
3rd Best	0.266	30/20	0.233	36/14 **

Table 7: Comparison with TREC Submissions

	TREC 9		TREC 10	
Model	MAP	W/L	MAP	W/L
$\text{SVM}^{\Delta}_{map}$	0.284	–	0.288	–
Best Func.	0.280	27/23	0.283	31/19
2nd Best	0.269	30/20	0.251	36/14 **
3rd Best	0.266	30/20	0.233	35/15 **

Table 8: Comparison with TREC Subm. (w/o best)

functions? Table 6 presents the comparison of $\text{SVM}^{\Delta}_{map}$ with the best Indri base functions. Each column group contains the macro-averaged MAP performance of $\text{SVM}^{\Delta}_{map}$ or a base function. The W/L columns show the number of queries where $\text{SVM}^{\Delta}_{map}$ achieved a higher MAP score. Significance tests were performed using the two-tailed Wilcoxon signed rank test. Two stars indicate a significance level of 0.95. All tables displaying our experimental results are structured identically. Here, we find that $\text{SVM}^{\Delta}_{map}$ significantly outperforms the best base functions.

Table 7 shows the comparison when trained on TREC submissions. While achieving a higher MAP score than the best base functions, the performance difference between $\text{SVM}^{\Delta}_{map}$ the base functions is not significant. Given that many of these submissions use scoring functions which are carefully crafted to achieve high MAP, it is possible that the best performing submissions use techniques which subsume the techniques of the other submissions. As a result, $\text{SVM}^{\Delta}_{map}$ would not be able to learn a hypothesis which can significantly out-perform the best submission.

Hence, we ran the same experiments using a modified dataset where the features computed using the best submission were removed. Table 8 shows the results (note that we are still comparing against the best submission though we are not using it for training). Notice that while the performance of $\text{SVM}^{\Delta}_{map}$ degraded slightly, the performance was still comparable with that of the best submission.

5.2 Comparison w/ Previous SVM Methods

The next question to answer is, does $\text{SVM}^{\Delta}_{map}$ produce higher MAP scores than previous SVM methods? Tables 9 and 10 present the results of $\text{SVM}^{\Delta}_{map}$, $\text{SVM}^{\Delta}_{roc}$, and SVM_{acc} when trained on the Indri retrieval functions and TREC submissions, respectively. Table 11 contains the corresponding results when trained on the TREC submissions without the best submission.

To start with, our results indicate that SVM_{acc} was not competitive with $\text{SVM}^{\Delta}_{map}$ and $\text{SVM}^{\Delta}_{roc}$, and at times underperformed dramatically. As such, we tried several approaches to improve the performance of SVM_{acc}.

5.2.1 Alternate SVM_{acc} Methods

One issue which may cause SVM_{acc} to underperform is the severe imbalance between relevant and non-relevant documents. The vast majority of the documents are not relevant. SVM_{acc2} addresses this problem by assigning more penalty to false negative errors. For each dataset, the ratio of the false negative to false positive penalties is equal to the ratio of the number non-relevant and relevant documents in that dataset. Tables 9, 10 and 11 indicate that SVM_{acc2} still performs significantly worse than $\text{SVM}^{\Delta}_{map}$.

	TREC 9		TREC 10	
Model	MAP	W/L	MAP	W/L
$\text{SVM}^{\Delta}_{map}$	0.242	–	0.236	–
$\text{SVM}^{\Delta}_{roc}$	0.237	29/21	0.234	24/26
SVM_{acc}	0.147	47/3 **	0.155	47/3 **
SVM_{acc2}	0.219	39/11 **	0.207	43/7 **
SVM_{acc3}	0.113	49/1 **	0.153	45/5 **
SVM_{acc4}	0.155	48/2 **	0.155	48/2 **

Table 9: Trained on Indri Functions

	TREC 9		TREC 10	
Model	MAP	W/L	MAP	W/L
$\text{SVM}^{\Delta}_{map}$	0.290	–	0.287	–
$\text{SVM}^{\Delta}_{roc}$	0.282	29/21	0.278	35/15 **
SVM_{acc}	0.213	49/1 **	0.222	49/1 **
SVM_{acc2}	0.270	34/16 **	0.261	42/8 **
SVM_{acc3}	0.133	50/0 **	0.182	46/4 **
SVM_{acc4}	0.233	47/3 **	0.238	46/4 **

Table 10: Trained on TREC Submissions

Another possible issue is that SVM_{acc} attempts to find just one discriminating threshold b that is query-invariant. It may be that different queries require different values of b. Having the learning method trying to find a good b value (when one does not exist) may be detrimental.

We took two approaches to address this issue. The first method, SVM_{acc3}, converts the retrieval function scores into percentiles. For example, for document d, query q and retrieval function f, if the score $f(d|q)$ is in the top 90% of the scores $f(\cdot|q)$ for query q, then the converted score is $f'(d|q) = 0.9$. Each K_f contains 50 evenly spaced values between 0 and 1. Tables 9, 10 and 11 show that the performance of SVM_{acc3} was also not competitive with $\text{SVM}^{\Delta}_{map}$.

The second method, SVM_{acc4}, normalizes the scores given by f for each query. For example, assume for query q that f outputs scores in the range 0.2 to 0.7. Then for document d, if $f(d|q) = 0.6$, the converted score would be $f'(d|q) = (0.6 - 0.2)/(0.7 - 0.2) = 0.8$. Each K_f contains 50 evenly spaced values between 0 and 1. Again, Tables 9, 10 and 11 show that SVM_{acc4} was not competitive with $\text{SVM}^{\Delta}_{map}$

5.2.2 MAP vs ROCArea

$\text{SVM}^{\Delta}_{roc}$ performed much better than SVM_{acc} in our experiments. When trained on Indri retrieval functions (see Table 9), the performance of $\text{SVM}^{\Delta}_{roc}$ was slight, though not significantly, worse than the performances of $\text{SVM}^{\Delta}_{map}$. However, Table 10 shows that $\text{SVM}^{\Delta}_{map}$ did significantly outperform $\text{SVM}^{\Delta}_{roc}$ when trained on the TREC submissions.

Table 11 shows the performance of the models when trained on the TREC submissions with the best submission removed. The performance of most models degraded by a small amount, with $\text{SVM}^{\Delta}_{map}$ still having the best performance.

	TREC 9		TREC 10	
Model	MAP	W/L	MAP	W/L
$\text{SVM}^{\Delta}_{map}$	0.284	–	0.288	–
$\text{SVM}^{\Delta}_{roc}$	0.274	31/19 **	0.272	38/12 **
SVM_{acc}	0.215	49/1 **	0.211	50/0 **
SVM_{acc2}	0.267	35/15 **	0.258	44/6 **
SVM_{acc3}	0.133	50/0 **	0.174	46/4 **
SVM_{acc4}	0.228	46/4 **	0.234	45/5 **

Table 11: Trained on TREC Subm. (w/o Best)

6. CONCLUSIONS AND FUTURE WORK

We have presented an SVM method that directly optimizes MAP. It provides a principled approach and avoids difficult to control heuristics. We formulated the optimization problem and presented an algorithm which provably finds the solution in polynomial time. We have shown empirically that our method is generally superior to or competitive with conventional SVMs methods.

Our new method makes it conceptually just as easy to optimize SVMs for MAP as was previously possible only for Accuracy and ROCArea. The computational cost for training is very reasonable in practice. Since other methods typically require tuning multiple heuristics, we also expect to train fewer models before finding one which achieves good performance.

The learning framework used by our method is fairly general. A natural extension of this framework would be to develop methods to optimize for other important IR measures, such as Normalized Discounted Cumulative Gain [2, 3, 4, 12] and Mean Reciprocal Rank.

7. ACKNOWLEDGMENTS

This work was funded under NSF Award IIS-0412894, NSF CAREER Award 0237381, and a gift from Yahoo! Research. The third author was also partly supported by a Microsoft Research Fellowship.

8. REFERENCES

[1] B. T. Bartell, G. W. Cottrell, and R. K. Belew. Automatic combination of multiple ranked retrieval systems. In *Proceedings of the ACM Conference on Research and Development in Information Retrieval (SIGIR)*, 1994.

[2] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2005.

[3] C. J. C. Burges, R. Ragno, and Q. Le. Learning to rank with non-smooth cost functions. In *Proceedings of the International Conference on Advances in Neural Information Processing Systems (NIPS)*, 2006.

[4] Y. Cao, J. Xu, T.-Y. Liu, H. Li, Y. Huang, and H.-W. Hon. Adapting ranking SVM to document retrieval. In *Proceedings of the ACM Conference on Research and Development in Information Retrieval (SIGIR)*, 2006.

[5] B. Carterette and D. Petkova. Learning a ranking from pairwise preferences. In *Proceedings of the ACM Conference on Research and Development in Information Retrieval (SIGIR)*, 2006.

[6] R. Caruana, A. Niculescu-Mizil, G. Crew, and A. Ksikes. Ensemble selection from libraries of models. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2004.

[7] J. Davis and M. Goadrich. The relationship between precision-recall and ROC curves. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2006.

[8] D. Hawking. Overview of the TREC-9 web track. In *Proceedings of TREC-2000*, 2000.

[9] D. Hawking and N. Craswell. Overview of the TREC-2001 web track. In *Proceedings of TREC-2001*, Nov. 2001.

[10] R. Herbrich, T. Graepel, and K. Obermayer. Large margin rank boundaries for ordinal regression. *Advances in large margin classifiers*, 2000.

[11] A. Herschtal and B. Raskutti. Optimising area under the ROC curve using gradient descent. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2004.

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

[13] T. Joachims. A support vector method for multivariate performance measures. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 377–384, New York, NY, USA, 2005. ACM Press.

[14] J. Lafferty and C. Zhai. Document language models, query models, and risk minimization for information retrieval. In *Proceedings of the ACM Conference on Research and Development in Information Retrieval (SIGIR)*, pages 111–119, 2001.

[15] Y. Lin, Y. Lee, and G. Wahba. Support vector machines for classification in nonstandard situations. *Machine Learning*, 46:191–202, 2002.

[16] D. Metzler and W. B. Croft. A markov random field model for term dependencies. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 472–479, 2005.

[17] K. Morik, P. Brockhausen, and T. Joachims. Combining statistical learning with a knowledge-based approach. In *Proceedings of the International Conference on Machine Learning*, 1999.

[18] S. Robertson. The probability ranking principle in ir. journal of documentation. *Journal of Documentation*, 33(4):294–304, 1977.

[19] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research (JMLR)*, 6(Sep):1453–1484, 2005.

[20] V. Vapnik. *Statistical Learning Theory.* Wiley and Sons Inc., 1998.

[21] L. Yan, R. Dodier, M. Mozer, and R. Wolniewicz. Optimizing classifier performance via approximation to the Wilcoxon-Mann-Witney statistic. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2003.