Efficient Optimization for Rank-based Loss Functions Supplementary Material

Pritish Mohapatra*
IIIT Hyderabad

Michal Rolínek*
MPI Tübingen

C.V. Jawahar
IIIT Hyderabad

Vladimir Kolmogorov
IST Austria

M. Pawan Kumar
University of Oxford, Alan Turing Institute

Appendix

The supplementary material is organized as follows. In Section 1, we give a full definition of QS-suitable loss functions and in Section 2 we justify the correctness of Algorithm 1. Section 3 contains proofs of QS-suitability of AP and NDCG losses and Section 4 establishes worst-case complexity of Algorithm 1 as well as a matching lower-bound on the complexity of solving Problem (6). Several remaining proofs are delegated to Section 5 and we use Section 6 for certain clarifications regarding previous use of NDCG in the literature. Finally, we report some additional experimental results in Section 7.

1. Complete characterization of QS-Suitable Loss Functions

A proper loss function $\Delta = \Delta(\mathbf{R}^*, \mathbf{R})$ is called QS-*suitable* if it meets the following three conditions.

(C1) **Negative decomposability with interleaving dependence.** There are functions $\delta_j \colon \{1, \dots, |\mathcal{P}| + 1\} \to \mathbb{R}$ for $j = 1, \dots, |\mathcal{N}|$ such that for a proper ranking $\mathbf{R}$ one can write

$$\Delta(\mathbf{R}^*, \mathbf{R}) = \sum_{\mathbf{x} \in \mathcal{N}} \delta_{ind^-(\mathbf{x})}(rank(\mathbf{x})).$$

(C2) **j-monotonicity of discrete derivative.** For every $1 \leq j < |\mathcal{N}|$ and $1 \leq i \leq |\mathcal{P}|$ we have

$$\delta_{j+1}(i+1) - \delta_{j+1}(i) \geq \delta_j(i+1) - \delta_j(i).$$

(C3) **Fast evaluation of discrete derivative.** For any $j \in \{1, \dots, |\mathcal{N}|\}$ and $i \in \{1, \dots, |\mathcal{P}|\}$, can the value $\delta_j(i+1) - \delta_j(i)$ be computed in constant time.

*The first two authors contributed equally and can be reached at *pritish.mohapatra@research.iiit.ac.in*, *michal.rolinek@tuebingen.mpg.de* respectively.

From (C1), we can see that the loss function depends only on the interleaving ranks of the negative samples. More accurately, it depends on the vector $\mathbf{r} = (r_1, \dots, r_{|\mathcal{N}|})$ where r_i is the interleaving rank of the i-th most relevant negative sample (i.e. with the i-th highest score).

Another way to interpret this type of dependence is by looking at the $\pm$-pattern of a ranking which can be obtained as follows. Given a proper ranking $\mathbf{R}$ (in the form of a permutation of samples), it is the pattern obtained by replacing each positive sample with a "$+$" symbol and each negative sample with a "$-$" symbol. It is easy to see that the $\pm$-pattern uniquely determines the vector $\mathbf{r}$ and vice versa and thus (C1) also implies dependence on the $\pm$-pattern.

2. Justification of Algorithm 1

First key point is that the entire objective function (6) inherits properties (C1) and (C2).

Observation 1 *The following holds:*

(a) There are functions $f_j \colon \{1, \dots, |\mathcal{P}| + 1\} \to \mathbb{R}$ for $j = 1, \dots, |\mathcal{N}|$ such that the objective function in (6) can be written as

$$\sum_{j=1}^{|\mathcal{N}|} f_j(r_j),$$

where r_j is the interleaving rank of the negative sample x with $ind^-(x) = j$.

(b) The functions f_j inherit property (C2). More precisely, for every $1 \leq j < |\mathcal{N}|$ and $1 \leq i \leq |\mathcal{P}|$ we have

$$f_{j+1}(i+1) - f_{j+1}(i) \geq f_j(i+1) - f_j(i).$$

(c) We can compute $\operatorname{argmax}_{l \leq i \leq r} f_j(i)$ in $O(r-l)$ time if we are provided access to the sorted array $\{s_i^+\}$ and to the score of the negative sample x with $ind^-(x) = j$.

As a result, solving Problem (6) reduces to computing the optimal interleaving ranks (or the optimal vector $\mathbf{r}$ from the remark above) [1].

The next vital point is that these interleaving ranks can be optimized independently. This is however not obvious. One certainly can maximize each f_j but the resulting vector $\mathbf{r}$ may not induce any ranking – its entries may not be monotone.

But as a matter of fact, this does not happen and Observation 2 from the main text gives the precise guarantee. This "correctness of greedy maximization" hinges upon condition (C2) as will also be demonstrated with a counterexample given later in Section 6.

All in all, it suffices to compute the vector $\mathbf{opt}$ in which $opt_j = \max \operatorname{argmax} f_j$ (the maximum ensures that ties are broken consistently) as is done in the main text of the paper.

3. Properties of Δ_{AP} and Δ_{NDCG}

In this place, let us prove the aforementioned properties of Δ_{AP} and Δ_{NDCG}.

Proposition 1 *Δ_{NDCG} is QS-suitable.*

Proof. As for (C1), let us first verify that the functions δ_j can be set as

$$\delta_j(i) = \frac{1}{C}\left(D(i+j-1) - D(|\mathcal{P}|+j)\right),$$

where $C = \sum_{i=1}^{|\mathcal{P}|} D(i)$. Indeed, one can check that

$$\begin{aligned}
&\Delta(\mathbf{R}^*, \mathbf{R}) \\
&= 1 - \frac{\sum_{\mathbf{x}\in\mathcal{P}} D(ind(\mathbf{x}))}{\sum_{i=1}^{|\mathcal{P}|} D(i)} \\
&= \frac{1}{C}\sum_{i=1}^{|\mathcal{P}|} D(i) - \sum_{\mathbf{x}\in\mathcal{P}} D(ind^+(\mathbf{x}) + rank(\mathbf{x}) - 1) \\
&= \frac{1}{C}\sum_{\mathbf{x}\in\mathcal{N}} D(ind^-(\mathbf{x}) + rank(\mathbf{x}) - 1) - D(|\mathcal{P}| + ind^-(\mathbf{x})) \\
&= \sum_{\mathbf{x}\in\mathcal{N}} \delta_{ind^-(\mathbf{x})}(rank(\mathbf{x}))
\end{aligned}$$

as desired. As for (C2) and (C3), let us realize that

$$\delta_j(i+1) - \delta_j(i) = \frac{1}{C}\left(D(i+j) - D(i+j-1)\right).$$

Then (C3) becomes trivial and checking (C2) reduces to

$$D(i+j+1) + D(i+j-1) \geq 2D(i+j)$$

which follows from convexity of the function D. □

[1] Note that the value of the objective can be computed efficiently given a vector $\mathbf{r}$ – for example by constructing any ranking $\mathbf{R}$ which respects $\mathbf{r}$.

Proposition 2 *Δ_{AP} is QS-suitable.*

Proof. Regarding (C1), the functions δ_j were already identified in [2] as

$$\delta_j(i) = \frac{1}{|\mathcal{P}|}\sum_{k=i}^{|\mathcal{P}|}\left(\frac{j}{j+k} - \frac{j-1}{j+k-1}\right)$$

so after writing

$$\delta_j(i+1) - \delta_j(i) = \frac{j-1}{j+i-1} - \frac{j}{j+i}$$

we again have (C3) for free and (C2) reduces to

$$2g_i(j) \geq g_i(j-1) + g_i(j+1),$$

where $g_i(x) = \frac{x}{x+i}$, and the conclusion follows from concavity of $g_i(x)$ for $x > 0$. □

4. Computational Complexity

Now is the time to establish the computational complexity of Algorithm 1 as well as the afore-mentioned matching lower bound. efficiency.

Theorem 3 *If Δ is QS-suitable, then the Problem (6) can be solved in time $O(|\mathcal{N}|\log|\mathcal{P}| + |\mathcal{P}|\log|\mathcal{P}| + |\mathcal{P}|\log|\mathcal{N}|)$, which in the most common case $|\mathcal{N}| > |\mathcal{P}|$ reduces to $O(|\mathcal{N}|\log|\mathcal{P}|)$.*

Outside running Algorithm 1, the entire computation also consists of preprocessing (sorting positive samples by their scores) and post processing (computing the output from vector $\mathbf{opt}$). These subroutines have only one non-linear complexity term – $O(|\mathcal{P}|\log|\mathcal{P}|)$ coming from the sorting. Therefore, it remains to establish the complexity of Algorithm 1 as $O(|\mathcal{N}|\log|\mathcal{P}| + |\mathcal{P}|\log|\mathcal{N}|)$.

To this end, let us denote $n = r^- - \ell^- + 1$ and $p = r^+ - \ell^+ + 1$, and set $T_{neg}(n,p)$, $T_{pos}(n,p)$ as the total time spent traversing the arrays of negative and positive sample scores, respectively, including recursive calls. The negative score array is traversed in the MEDIAN and SELECT procedures and the positive scores are traversed when searching for opt_m. The latter has by complexity $O(p)$, due to Observation 1(c), whose assumption are always satisfied during the run of the algorithm.

Proposition 4 *The runtimes $T_{neg}(n,p)$ and $T_{pos}(n,p)$ satisfy the following recursive inequalities*

$$\begin{aligned}
T_{neg}(n,p) &\leq Cn + T_{neg}(n/2,p_1) + T_{neg}(n/2,p_2) \\
&\qquad \textit{for some} \quad p_1 + p_2 = p+1, \\
T_{pos}(n,p) &\leq Cp + T_{pos}(n/2,p_1) + T_{pos}(n/2,p_2) \\
&\qquad \textit{for some} \quad p_1 + p_2 = p+1, \\
T_{neg}(n,1) &\leq Cn, \qquad T_{neg}(1,p) = 0,
\end{aligned}$$

$$T_{pos}(n,1)=0, \qquad T_{pos}(1,p)\le Cp$$

for a suitable constant C. These inequalities imply $T_{neg}(n,p)\le C'n\log(1+p)$ and $T_{pos}(n,p)\le C'(p-1)\log(1+n)$ for another constant C'. Thus the running time of Algorithm 1, where $p=|\mathcal{P}|+1$, $n=|\mathcal{N}|$, is $O(|\mathcal{N}|\log|\mathcal{P}|+|\mathcal{P}|\log|\mathcal{N}|)$.

Proof. The recursive inequalities follow from inspection of Algorithm 1. As for the "aggregated" inequalities, we proceed in both cases by induction. For the first inequality the base step is trivial for high enough constant C' and for the inductive step we may write

$$\begin{aligned}T_{neg}(n,p)&\le Cn+T_{neg}(n/2,p_1)+T_{neg}(n/2,p_2)\\&\le Cn+\frac{1}{2}C'n\log(1+p_1)+\frac{1}{2}C'n\log(1+p_2)\\&=C'n\left(\frac{C}{C'}+\log\sqrt{(1+p_1)(1+p_2)}\right)\\&\le C'n\log(p_1+p_2)=C'n\log(1+p)\end{aligned}$$

where in the last inequality we used that

$$1+(1+p_1)(1+p_2)\le(p_1+p_2)^2$$

for integers p_1,p_2 with $p_1+p_2=p+1\ge 3$. That makes the last inequality true for sufficiently high C' (not depending on n and p).

The proof of the second inequality is an easier variation on the previous technique. □

4.1. Lower Bound on Complexity

In order to prove the matching lower bound (among comparison-based algorithms), we intend to use the classical information theoretical argument: There are many possible outputs and from each comparison we receive one bit of information, therefore we need "many" comparison to shatter all output options.

Proposition 5 *Let Δ be a loss function. Then any comparison-based algorithm for Problem (6) requires $\Omega(|\mathcal{N}|\log|\mathcal{P}|)$ operations.*

Proof. Since the negative samples are unsorted on the input and the scores are arbitrary, every possible mapping from $\{1,\dots,|\mathcal{N}|\}$ to $\{1,\dots,|\mathcal{P}|+1\}$ may induce the (unique) optimal assignment of interleaving ranks. There are $(|\mathcal{P}|+1)^{|\mathcal{N}|}$ possibilities to be distinguished and each comparison has only two possible outcomes. Therefore we need $\log_2\left((|\mathcal{P}|+1)^{|\mathcal{N}|}\right)\in\Omega(|\mathcal{N}|\log|\mathcal{P}|)$ operations. □

5. Remaining proofs

Throughout the text we omitted several proofs, mostly because they are straightforward generalizations of what already appeared in [2] and [1]. For the sake of completeness, we present them here.

Proof of Observation 1 (of main text) **:** Let $\mathbf{R}$ be any optimal solution. We check that $F(\mathbf{X},\mathbf{R};\mathbf{w})$ increases if we swap two samples $x,y\in\mathcal{P}$ in $\mathbf{R}$ with $ind(\mathbf{x})<ind(\mathbf{y})$ and $\phi(\mathbf{x};\mathbf{w})<\phi(\mathbf{y};\mathbf{w})$ (it boils down to $ac+bd>ad+bc$ for $a>b\ge 0$ and $c>d\ge 0$). Since similar argument applies for negative samples, we can conclude that $\mathbf{R}$ already has both negative and positive samples sorted decreasingly. Otherwise, one could perform swaps in $\mathbf{R}$ that would increase the value of the objective, a contradiction with the optimality of $\mathbf{R}$. □

Proof of Observation 2 (of main text) **:** Recall that opt_j is the highest rank with maximal value of the corresponding $f_{j'}$. It suffices to prove that for $i_{j+1}=\max\operatorname{argmax} f_{j+1}$ and $i_j=\max\operatorname{argmax} f_j$, we have $i_{j+1}\ge i_j$. Since by Observation 1 functions f_j inherit property (C2), we can compare the discrete derivatives of f_j and f_{j+1}, all left to do is to formalize the discrete analogue of what seems intuitive for continuous functions.

Assume $i_{j+1}<i_j$. Then since

$$\begin{aligned}f_{j+1}(i_j)-f_{j+1}(i_{j+1})&=\sum_{i=i_{j+1}}^{i_j-1}f_{j+1}(i+1)-f_{j+1}(i)\\&\ge\sum_{i=i_{j+1}}^{i_j-1}f_j(i+1)-f_j(i)\\&=f_j(i_j)-f_j(i_{j+1})\ge 0,\end{aligned}$$

we obtain that $i_j\in\operatorname{argmax} f_{j+1}$ and as $i_j>i_{j+1}=\max\operatorname{argmax} f_{j+1}$ and we reached the expected contradiction. □

Lemma 6 *The objective function $F(\mathbf{X},\mathbf{R};\mathbf{w})$ decomposes into contributions of negative and positive samples as follows:*

$$\begin{aligned}F(\mathbf{X},\mathbf{R};\mathbf{w})&=\frac{1}{|\mathcal{P}|\,|N|}\sum_{\mathbf{x}\in\mathcal{P}}\sum_{\mathbf{y}\in\mathcal{N}}\mathbf{R}_{\mathbf{x},\mathbf{y}}(\phi(\mathbf{x};\mathbf{w})-\phi(\mathbf{y};\mathbf{w}))\\&=\sum_{\mathbf{x}\in\mathcal{P}}c(\mathbf{x})\phi(\mathbf{x};\mathbf{w})+\sum_{\mathbf{y}\in\mathcal{N}}c(\mathbf{y})\phi(\mathbf{y};\mathbf{w}),\end{aligned}$$

where

$$c(\mathbf{x})=\frac{|\mathcal{N}|+2-2rank(\mathbf{x})}{|\mathcal{P}|\,|\mathcal{N}|},\ c(\mathbf{y})=\frac{|\mathcal{P}|+2-2rank(\mathbf{y})}{|\mathcal{P}|\,|\mathcal{N}|}.$$

In particular, assuming already that $\{s_i^+\}$ is sorted, and that $\mathbf{R}$ is induced by a vector of interleaving ranks $\mathbf{r}$, one

has

$$F(\mathbf{X}, \mathbf{R}; \mathbf{w}) = \sum_{i=1}^{|\mathcal{P}|} c_i^+ s_i^+ + \sum_{j=1}^{|\mathcal{N}|} c_j^- s_j^*,$$

where

$$c_i^+ = \frac{|\mathcal{N}| + 2 - 2r_i^+}{|\mathcal{P}|\,|\mathcal{N}|}, \qquad c_j^- = \frac{|\mathcal{P}| + 2 - 2r_j}{|\mathcal{P}|\,|\mathcal{N}|}.$$

Here r_i^+ stands for the interleaving rank of the i-th positive sample, which can be computed as $r_i^+ = 1 + |\{j : r_j \leq i\}|$.

Proof. This is straightforward to verify with a short computation. □

Proof of Observation 1: We slightly modify the decomposition from Lemma 6 in order to incorporate the array $\{s_i^+\}$:

$$\begin{aligned} &F(\mathbf{X}, \mathbf{R}; \mathbf{w}) \\ &= \sum_{\mathbf{y} \in \mathcal{N}} \left(c(\mathbf{y})\phi(\mathbf{y}; \mathbf{w}) + \sum_{\mathbf{x} \in \mathcal{P}} \mathbf{R}_{\mathbf{x},\mathbf{y}} \phi(\mathbf{x}; \mathbf{w}) \right) \\ &= \frac{1}{|\mathcal{N}|\,|\mathcal{P}|} \sum_{j=1}^{|\mathcal{N}|} \left((|\mathcal{P}| + 2 - 2r_j) s_j^* + 2 \sum_{i=1}^{r_j - 1} s_i^+ - \sum_{i=1}^{|\mathcal{P}|} s_i^+ \right). \end{aligned}$$

This, in combination with (C1), defines the functions f_j for $j = 1, \ldots, |\mathcal{N}|$. As for the condition (C2), we have

$$f_j(i+1) - f_j(i) = \frac{2(s_i^+ - s_j^*)}{|\mathcal{N}|\,|\mathcal{P}|} + \delta_j(i+1) - \delta_j(i),$$

where, let us be reminded, $\{s_j^*\}$ is the sorted array of scores of negative samples. After writing analogous equality for $j+1$ and using that (C2) holds for functions δ_j, we can check that the desired inequality

$$f_{j+1}(i+1) - f_{j+1}(i) \geq f_j(i+1) - f_j(i)$$

follows from $s_{j+1}^* \leq s_j^*$.

Note that for computing the $\operatorname{argmax} f_j(i)$ it is sufficient to compute all discrete derivatives (i.e. all the differences $f_j(i+1) - f_j(i)$); the actual values of f_j are in fact not needed. For δ_j we know that one such evaluation is constant time and for f_j this is also the case since we assumed to have access to s_j^*. □

6. NDCG and Discount Functions

Chakrabarti *et al.* [1] use a slightly modified definition of the discount $D(\cdot)$ as

$$D(i) = \begin{cases} 1 & 1 \leq i \leq 2 \\ 1/\log_2(i) & i > 2 \end{cases}.$$

For the resulting NDCG loss, a greedy algorithm is proposed for solving the loss augmented inference problem. This algorithm achieves the runtime of $O(|\mathcal{N}|\,|\mathcal{P}| + |\mathcal{N}| \log |\mathcal{N}|)$. The authors also suggest to use a cut-off k in the definition of discount $D(i)$, setting $D(i) = 0$ for $i \geq k$. With this simplification they achieved a reduced complexity of $O((|\mathcal{N}| + |\mathcal{P}|) \log(|\mathcal{P}| + |\mathcal{N}|) + k^2)$.

However, with the above definition of a discount, it is possible to obtain a corner-case where their proof of correctness of the greedy algorithm is not valid (specifically, there exists a counter-example for Fact 3.4 of [1]). For the greedy algorithm to be correct, it turns out that the convexity of $D(i)$ is essential.

Remark 1 *Observation 2 is not true for Δ_{NDCG} with function $D(i)$ taken as*

$$D(i) = \begin{cases} 1 & 1 \leq i \leq 2 \\ 1/\log_2(i) & i > 2 \end{cases}.$$

Proof. Consider negative samples $\mathbf{x}_1$ and $\mathbf{x}_2$ and a positive sample $\mathbf{x}_3$ with scores $s_1 = 3\varepsilon$, $s_2 = \varepsilon$, $s_3 = 5\varepsilon$, where $\varepsilon > 0$ is small.

Note that the NDCG loss of a ranking $\mathbf{R}$ reduces to $\Delta_{NDCG}(\mathbf{R}^*, \mathbf{R}) = 1 - D(ind(\mathbf{x}_3))$ where we used the fact that $D(1) = 1$.

The decomposition $\Delta_{NDCG}(\mathbf{R}^*, \mathbf{R}) = \delta_1(r_1) + \delta_2(r_2)$ holds if we set

$$\begin{aligned} \delta_1(1) &= \delta_2(1) = 0, \\ \delta_2(1) &= D(2) - D(3), \\ \delta_1(2) &= D(1) - D(2) = 0 \end{aligned}$$

and (possibly by looking at the proof of Observation 1) we also find values of f_1 and f_2 as

$$\begin{aligned} f_1(1) &= \frac{1}{2}(s_1 - s_3) + \delta_1(1) = -\varepsilon < \varepsilon \\ &= \frac{1}{2}(s_3 - s_1) + \delta_1(2) = f_1(2) \\ f_2(1) &= \frac{1}{2}(s_2 - s_3) + \delta_2(1) = -2\varepsilon + D(2) - D(3) > 2\varepsilon \\ &= \frac{1}{2}(s_3 - s_2) + \delta_2(2) = f_2(2). \end{aligned}$$

Hence $opt_1 = 2 > 1 = opt_2$, a contradiction. □

7. Additional Experimental Results

For the action classification experiments on the PASCAL VOC 2011 data set, we report the performance of models trained by optimizing 0-1 loss as well as AP loss in Table 1. Specifically, we report the AP on the test set for each of the 10 action classes. Similarly, we also report the performance

Object class	0-1 loss	AP loss
Jumping	52.580	55.230
Phoning	32.090	32.630
Playing instrument	35.210	41.180
Reading	27.410	26.600
Riding bike	72.240	81.060
Running	73.090	76.850
Taking photo	21.880	25.980
Using computer	30.620	32.050
Walking	54.400	57.090
Riding horse	79.820	83.290

Table 1. *Performance of classification models trained by optimizing 0-1 loss and* AP *loss, in terms of* AP *on the test set for the different action classes of* PASCAL VOC *2011 action dataset.*

of models trained by optimizing 0-1 loss as well as NDCG loss, in terms of NDCG on the test set in Table 2.

For our object detection experiments, we report the detection AP in Table 3 for all the 20 object categories obtained by models trained using 0-1 loss as well as AP loss. For all object categories other than 'bottle', AP loss based training does better than that with 0-1 loss. For 15 of the 20 object categories, we get statistically significant improvement with AP loss trained models compared to those trained using 0-1 loss (using paired t-test with p-value less than 0.05). While optimizing AP loss for learning gives an overall improvement of 7.12% compared to when using 0-1 loss, for 5 classes it gives an improvement of more than 10%. The bottom 2 classes with the least improvement obtained by AP loss based training, 'chair' and 'bottle' seem to be difficult object categories to detect, with detectors registering very low detection APs. In conjunction with the overall superior performance of AP loss for learning model parameters, the efficient method proposed by this paper makes a good case for optimizing AP loss rather than 0-1 loss for tasks like object detection.

Object class	0-1 loss	NDCG loss
Jumping	86.409	87.895
Phoning	73.134	76.733
Playing instrument	81.533	83.666
Reading	74.528	75.588
Riding bike	94.928	95.958
Running	93.766	93.776
Taking photo	74.058	76.701
Using computer	79.518	78.276
Walking	89.789	89.742
Riding horse	96.160	96.875

Table 2. *Performance of classification models trained by optimizing 0-1 loss and* NDCG *loss, in terms of* NDCG *on the test set for the different action classes of* PASCAL VOC *2011 action dataset. We conduct 5-fold cross-validation and report the mean* NDCG *over the five validation sets.*

Object category	0-1 loss	AP loss
Aeroplane	46.60	48.18
Bicycle	48.53	61.45
Bird	33.31	36.73
Boat	15.23	19.66
Bottle	6.10	1.01
Bus	37.01	49.51
Car	61.28	66.78
Cat	38.12	40.77
Chair	2.71	3.23
Cow	21.06	38.52
Dining-table	14.20	39.53
Dog	33.55	36.25
Horse	46.14	53.86
Motorbike	29.97	34.81
Person	29.58	30.41
Potted-plant	21.27	23.03
Sheep	11.65	32.20
Sofa	36.66	42.03
Train	29.71	37.10
TV-monitor	27.31	37.26

Table 3. *Performance of detection models trained by optimizing 0-1 loss and* AP *loss, in terms of* AP *on the test set for the different object categories of* PASCAL VOC *2007 test set.*

References

[1] S. Chakrabarti, R. Khanna, U. Sawant, and C. Bhattacharyya. Structured learning for non-smooth ranking losses. In *KDD*, 2008. 3, 4

[2] Y. Yue, T. Finley, F. Radlinski, and T. Joachims. A support vector method for optimizing average precision. In *SIGIR*, 2007. 2, 3