A Simple Linear Ranking Algorithm Using Query Dependent Intercept Variables

Nir Ailon
Google Research, 76 Ninth Ave, 4th Floor, New York NY 10011

Abstract. The LETOR website contains three information retrieval datasets used as a benchmark for testing machine learning ideas for ranking. Algorithms participating in the challenge are required to assign score values to search results for a collection of queries, and are measured using standard IR ranking measures (NDCG, precision, MAP) that depend only the relative score-induced order of the results. Similarly to many of the ideas proposed in the participating algorithms, we train a linear classifier. In contrast with other participating algorithms, we define an additional free variable (intercept, or benchmark) for each query. This allows expressing the fact that results for different queries are incomparable for the purpose of determining relevance. The cost of this idea is the addition of relatively few nuisance parameters. Our approach is simple, and we used a standard logistic regression library to test it. The results beat the reported participating algorithms. Hence, it seems promising to combine our approach with other more complex ideas.

1 Introduction

The LETOR benchmark dataset [6] `http://research.microsoft.com/users/LETOR/` (version 2.0) contains three information retrieval datasets used as a benchmark for testing machine learning ideas for ranking. Algorithms participating in the challenge are required to assign score values to search results for a collection of queries, and are measured using standard IR ranking measures (NDCG@n, precision@n and MAP - see [6] for details), designed in such a way that only the relative order of the results matters. The input to the learning problem is a list of query-result records, where each record is a vector of standard IR features together with a relevance label and a query id. The label is either binary (irrelevant or relevant) or trinary (irrelevant, relevant or very relevant).

All reported algorithms used for this task on LETOR website [2, 3, 5, 7–9] rely on the fact that records corresponding to the same query id are in some sense comparable to each other, and cross query records are incomparable. The rationale is that the IR measures are computed as a sum over the queries, where for each query a nonlinear function is computed. For example, RankSVM [5] and RankBoost [3] use pairs of results for the same query to penalize a cost function, but never cross-query pairs of results.

The following approach seems at first too naive compared to others: Since the training information is given as relevance labels, why not simply train a linear classifier to predict the relevance labels, and use prediction confidence as score? Unfortunately this approach fares poorly. The hypothesized reason is that judges' relevance response may depend on the query. To check this hypothesis, we define an additional free variable (*intercept* or *benchmark*) for each query. This allows expressing the fact that results for different queries are incomparable for the purpose of determining relevance. The cost of this idea is the addition of relatively few nuisance parameters. Our approach is extremely simple, and we used a standard logistic regression library to test it on the data. This work is not the first to suggest query dependent ranking, but it is arguably the simplest, most immediate way to address this dependence using linear classification before other complicated ideas should be tested. Based on our judgment, other reported algorithms used for the challenge are more complicated, and our solution is overall better on the given data.

2 Theory and Experiments

Let Q_i, $i = 1, \ldots, n$ be a sequence of queries, and for each i let $R_{i1}, \ldots, R_{im_i}$ denote a corresponding set of retrieved results. For each $i \in [n]$ and $j \in [m_i]$ let $\Phi_{ij} = (\Phi_{ij}(1), \ldots \Phi_{ij}(k)) \in \mathbb{R}^k$ denote a real valued feature vector. Here, the coordinates of Φ_{ij} are standard IR features. Some of these features depend on the result only, and some on the query-result pair, as explained in [6]. Also assume that for each i, j there is a judge's response label $L_{ij} \in \mathcal{O}$, where $\mathcal{O}$ is a finite set of ordinals. In the TREC datasets (TD2003 and TD2004), $\mathcal{O} = \{0, 1\}$. In the OHSUMED dataset $\mathcal{O} = \{0, 1, 2\}$. Higher numbers represent higher relevance.

The Model. We assume the following generalized linear model for L_{ij} given Φ_{ij} using the logit link. Other models are possible, but we chose this one for simplicity. Assume first that the set of ordinals is binary: $\mathcal{O} = \{0, 1\}$. There is a hidden global weight vector $w \in \mathbb{R}^k$. Aside from w, there is a query dependent parameter $\Theta_i \in \mathbb{R}$ corresponding to each query Q_i. We call this parameter a *benchmark* or an *intercept*. The intuition behind defining this parameter is to allow for a different relevance criterion to different queries. The probability distribution $\Pr_{w,\Theta_i}(L_{ij}|Q_i, R_{ij})$ of response to result j for query i is given by

$$\Pr_{w,\Theta_i}(L_{ij} = 1|Q_i, R_{ij}) = \frac{1}{1 + e^{\Theta_i - w\cdot\Phi_{ij}}} \qquad \Pr_{w,\Theta_i}(L_{ij} = 0|Q_i, R_{ij}) = \frac{1}{1 + e^{w\cdot\Phi_{ij} - \Theta_i}}$$

In words, the probability of result j for query i deemed relevant is $\Theta_i - w \cdot \Phi_{ij}$ passed through the logit link, where $w \cdot \Phi_{ij}$ is vector dot product. This process should be thought of as a statistical comparison between the value of a search result R_{ij} (obtained as a linear function of its feature vector Φ_{ij}) to a benchmark Θ_i. In our setting, both the linear coefficients w and the benchmark $\Theta_1, \ldots, \Theta_n$ are variables which can be efficiently learnt in the maximum likelihood (supervised) setting. Note that the total number of variables is n (number of queries) plus k (number of features).

Observation: For any weight vector w, benchmark variable Θ_i corresponding to query Q_i and two result incides j, k,

$$\Pr_{w,\Theta_i}(L_{ij} = 1|Q_i, R_{ij}) > \Pr_{w,\Theta_i}(L_{ik} = 1|Q_i, R_{ik}) \iff w \cdot \Phi_{ij} > w \cdot \Phi_{ik} \ .$$

This last observation means that for the purpose of ranking candidate results for a specific query Q_i in decreasing order of relevance likelihood, the benchmark parameter Θ_i is not needed. Indeed, in our experiments below the benchmark variables will be used only in conjunction with the training data. In testing, this variable will neither be known nor necessary.

The Trinay Case. As stated above, the labels for the OHSUMED case are trinary: $\mathcal{O} = \{0, 1, 2\}$. We chose the following model to extend the binary case. Instead of one benchmark parameter for each query Q_i there are two such parameters, Θ_i^H, Θ_i^L (High/ Low) with $\Theta_i^H > \Theta_i^L$. Giver a candidate result R_{ij} to query Q_i and the parameters, the probability distribution on the three possible ordinals is:

$$\Pr_{w,\Theta_i^H,\Theta_i^L}(L_{ij} = X|Q_i, R_{ij}) = \begin{cases} \frac{1}{\left(1+e^{w\cdot\Phi_{ij}-\Theta_i^H}\right)\left(1+e^{w\cdot\Phi_{ij}-\Theta_i^L}\right)} & X = 0 \\ \frac{1}{\left(1+e^{w\cdot\Phi_{ij}-\Theta_i^H}\right)\left(1+e^{\Theta_i^L-w\cdot\Phi_{ij}}\right)} & X = 1 \\ \frac{1}{\left(1+e^{\Theta_i^H-w\cdot\Phi_{ij}}\right)} & X = 2 \end{cases}$$

In words, the result R_{ij} is statistically compared against benchmark Θ_i^H. If it is deemed higher than the benchmark, the label 2 ("very relevant") is outputted as response. Otherwise, the result is statistically compared against benchmark Θ_i^L, and the resulting comparison is either 0 (irrelevant) or 1 (relevant).[1] The model is inspired by Ailon and Mohri's QuickSort algorithm, proposed as a learning method in their recent paper [1]: Pivot elements (or, benchmarks) are used to iteratively refine the ranking of data.

Experiments. We used an out of the box implementation of logistic regression in R to test the above ideas. Each one of the three datasets includes 5 folds of data, each fold consisting of training, validation (not used) and testing data. From each training dataset, the variables w and Θ_i (or $w, \Theta_i^H, \Theta_i^L$ in the OHSUMED case) were recovered in the maximum likelihood sense (using logistic regression). Note that the constraint $\Theta_i^H > \Theta_i^L$ was not enforced, but was obtained as a byproduct. The weight vector w was then used to score the test data. The scores were passed through an evaluation tool provided by the LETOR website.

Results. The results for OHSUMED are summarized in Tables 1, 2, and 7. The results for TD2003 are summarized in Tables 3, 4, and 7. The results for TD2004 are summarized in Tables 5, 6, and 7. The significance of each score separately is quite small (as can be seen by the standard deviations), but it is clear that overall our method outperforms the others. For convenience, the winning average score (over 5 folds) is marked in red for each table column.

Conclusions and further ideas • In this work we showed that a simple out-of-the-box generalized linear model using logistic regression performs as least as well the state of the art in learning ranking algorithms if a separate intercept variable (benchmark) is defined for each query • In a more eleborate IR system, a separate intercept variable could be attached to each pair of *query* $\times$ *judge* (indeed, in LETOR the separate judges' responses were aggregated somehow, but in general

[1] A natural alternative to this model is the following: Statistically compare against Θ_i^L to decide of the result is irrelevant. If it is not irrelevant, compare against Θ_i^H to decide between relevant and very relevant. In practice, the model proposed above gave better results.

	@2	@4	@6	@8	@10
This	0.491 ± 0.086	0.480 ± 0.058	0.458 ± 0.055	0.448 ± 0.054	0.447 ± 0.047
RankBoost	0.483 ± 0.079	0.461 ± 0.063	0.442 ± 0.058	0.436 ± 0.044	0.436 ± 0.042
RankSVM	0.476 ± 0.091	0.459 ± 0.059	0.455 ± 0.054	0.445 ± 0.057	0.441 ± 0.055
FRank	0.510 ± 0.074	0.478 ± 0.060	0.457 ± 0.062	0.445 ± 0.054	0.442 ± 0.055
ListNet	0.497 ± 0.062	0.468 ± 0.065	0.451 ± 0.056	0.451 ± 0.050	0.449 ± 0.040
AdaRank.MAP	0.496 ± 0.100	0.471 ± 0.075	0.448 ± 0.070	0.443 ± 0.058	0.438 ± 0.057
AdaRank.NDCG	0.474 ± 0.091	0.456 ± 0.057	0.442 ± 0.055	0.441 ± 0.048	0.437 ± 0.046

Table 1. OHSUMED: Mean ± Stdev for NDCG over 5 folds

	@2	@4	@6	@8	@10
This	0.610 ± 0.092	0.598 ± 0.082	0.560 ± 0.090	0.526 ± 0.092	0.511 ± 0.081
RankBoost	0.595 ± 0.090	0.562 ± 0.081	0.525 ± 0.093	0.505 ± 0.072	0.495 ± 0.081
RankSVM	0.619 ± 0.096	0.579 ± 0.072	0.558 ± 0.077	0.525 ± 0.088	0.507 ± 0.096
FRank	0.619 ± 0.051	0.581 ± 0.079	0.534 ± 0.098	0.501 ± 0.091	0.485 ± 0.097
ListNet	0.629 ± 0.080	0.577 ± 0.097	0.544 ± 0.098	0.520 ± 0.098	0.510 ± 0.085
AdaRank.MAP	0.605 ± 0.102	0.567 ± 0.087	0.528 ± 0.102	0.502 ± 0.087	0.491 ± 0.091
AdaRank.NDCG	0.605 ± 0.099	0.562 ± 0.063	0.529 ± 0.073	0.506 ± 0.073	0.491 ± 0.082

Table 2. OHSUMED: Mean ± Stdev for precision over 5 folds

	@2	@4	@6	@8	@10
This	0.430 ± 0.179	0.398 ± 0.146	0.375 ± 0.125	0.369 ± 0.113	0.360 ± 0.105
RankBoost	0.280 ± 0.097	0.272 ± 0.086	0.280 ± 0.071	0.282 ± 0.074	0.285 ± 0.064
RankSVM	0.370 ± 0.130	0.363 ± 0.132	0.341 ± 0.118	0.345 ± 0.117	0.341 ± 0.115
FRank	0.390 ± 0.143	0.342 ± 0.107	0.330 ± 0.087	0.332 ± 0.079	0.336 ± 0.074
ListNet	0.430 ± 0.160	0.386 ± 0.125	0.386 ± 0.106	0.373 ± 0.104	0.374 ± 0.094
AdaRank.MAP	0.320 ± 0.104	0.268 ± 0.120	0.229 ± 0.104	0.206 ± 0.093	0.194 ± 0.086
AdaRank.NDCG	0.410 ± 0.207	0.347 ± 0.195	0.309 ± 0.181	0.286 ± 0.171	0.270 ± 0.161

Table 3. TD2003: Mean ± Stdev for NDCG over 5 folds

	@2	@4	@6	@8	@10
This	0.420 ± 0.192	0.340 ± 0.161	0.283 ± 0.131	0.253 ± 0.115	0.222 ± 0.106
RankBoost	0.270 ± 0.104	0.230 ± 0.112	0.210 ± 0.080	0.193 ± 0.071	0.178 ± 0.053
RankSVM	0.350 ± 0.132	0.300 ± 0.137	0.243 ± 0.100	0.233 ± 0.091	0.206 ± 0.082
FRank	0.370 ± 0.148	0.260 ± 0.082	0.223 ± 0.043	0.210 ± 0.045	0.186 ± 0.049
ListNet	0.420 ± 0.164	0.310 ± 0.129	0.283 ± 0.090	0.240 ± 0.075	0.222 ± 0.061
AdaRank.MAP	0.310 ± 0.096	0.230 ± 0.105	0.163 ± 0.081	0.125 ± 0.064	0.102 ± 0.050
AdaRank.NDCG	0.400 ± 0.203	0.305 ± 0.183	0.237 ± 0.161	0.190 ± 0.140	0.156 ± 0.120

Table 4. TD2003: Mean ± Stdev for precision over 5 folds

	@2	@4	@6	@8	@10
This	0.473 ± 0.132	0.454 ± 0.075	0.450 ± 0.059	0.459 ± 0.050	0.472 ± 0.043
RankBoost	0.473 ± 0.055	0.439 ± 0.057	0.448 ± 0.052	0.461 ± 0.036	0.472 ± 0.034
RankSVM	0.433 ± 0.094	0.406 ± 0.086	0.397 ± 0.082	0.410 ± 0.074	0.420 ± 0.067
FRank	0.467 ± 0.113	0.435 ± 0.088	0.445 ± 0.078	0.455 ± 0.055	0.471 ± 0.057
ListNet	0.427 ± 0.080	0.422 ± 0.049	0.418 ± 0.057	0.449 ± 0.041	0.458 ± 0.036
AdaRank.MAP	0.393 ± 0.060	0.387 ± 0.086	0.399 ± 0.085	0.400 ± 0.086	0.406 ± 0.083
AdaRank.NDCG	0.360 ± 0.161	0.377 ± 0.123	0.378 ± 0.117	0.380 ± 0.102	0.388 ± 0.093

Table 5. TD2004: Mean ± Stdev for NDCG over 5 folds

it is likely that different judges would have different benchmarks as well) • The simplicity of our approach is also its main limitation. However, it can easily be implemented in conjunction with other ranking ideas. For example, recent work by Geng et al. [4] (not evaluated on LETOR) proposes query dependent ranking,

	@2	@4	@6	@8	@10
This	0.447 ± 0.146	0.370 ± 0.095	0.316 ± 0.076	0.288 ± 0.076	0.264 ± 0.062
RankBoost	0.447 ± 0.056	0.347 ± 0.083	0.304 ± 0.079	0.277 ± 0.070	0.253 ± 0.067
RankSVM	0.407 ± 0.098	0.327 ± 0.089	0.273 ± 0.083	0.247 ± 0.082	0.225 ± 0.072
FRank	0.433 ± 0.115	0.340 ± 0.098	0.311 ± 0.082	0.273 ± 0.071	0.256 ± 0.071
ListNet	0.407 ± 0.086	0.357 ± 0.087	0.307 ± 0.084	0.287 ± 0.069	0.257 ± 0.059
AdaRank.MAP	0.353 ± 0.045	0.300 ± 0.086	0.282 ± 0.068	0.242 ± 0.063	0.216 ± 0.064
AdaRank.NDCG	0.320 ± 0.139	0.300 ± 0.082	0.262 ± 0.092	0.232 ± 0.086	0.207 ± 0.082

Table 6. TD2004: Mean ± Stdev for precision over 5 folds

	OHSUMED	TD2003	TD2004
This	0.445 ± 0.065	0.248 ± 0.075	0.379 ± 0.051
RankBoost	0.440 ± 0.062	0.212 ± 0.047	0.384 ± 0.043
RankSVM	0.447 ± 0.067	0.256 ± 0.083	0.350 ± 0.072
FRank	0.446 ± 0.062	0.245 ± 0.065	0.381 ± 0.069
ListNet	0.450 ± 0.063	0.273 ± 0.068	0.372 ± 0.046
AdaRank.MAP	0.442 ± 0.061	0.137 ± 0.063	0.331 ± 0.089
AdaRank.NDCG	0.442 ± 0.058	0.185 ± 0.105	0.299 ± 0.088

Table 7. Mean ± Stdev for MAP over 5 folds

where the category of a query is determined using a k-Nearest Neighbor method. It is immediate to apply the ideas here within each category.

References

1. Nir Ailon and Mehryar Mohri. An efficient reduction of ranking to classification. In *COLT*, 2008.
2. Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. Learning to rank: from pairwise approach to listwise approach. In *ICML '07: Proceedings of the 24th international conference on Machine learning*, pages 129–136, New York, NY, USA, 2007. ACM.
3. Yoav Freund, Raj Iyer, Robert E. Schapire, and Yoram Singer. An efficient boosting algorithm for combining preferences. *J. Mach. Learn. Res.*, 4:933–969, 2003.
4. Xiubo Geng, Tie-Tan Liu, Tao Qin, Hang Li, and Heung-Yeung Shum. Query-dependent ranking with knn. In *SIGIR*, 2008.
5. R. Herbrich, T. Graepel, and K. Obermayer. Support vector learning for ordinal regression. In *ICANN*, 1999.
6. Tie-Yan Liu, Tau Qin, Jun Xu, Wenying Xiong, and Hang Li. Letor: Benchmark dataset for research on learning to rank for information retrieval. In *LR4IR2007, in Conjunction with SIGIR*, 2007.
7. Tao Qin, Xu-Dong Zhang, De-Sheng Wang, Tie-Yan Liu, Wei Lai, and Hang Li. Ranking with multiple hyperplanes. In *SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 279–286, New York, NY, USA, 2007. ACM.
8. Ming-Feng Tsai, Tie-Yan Liu, Tao Qin, Hsin-Hsi Chen, and Wei-Ying Ma. Frank: a ranking method with fidelity loss. In *SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 383–390, New York, NY, USA, 2007. ACM.
9. Jun Xu and Hang Li. Adarank: a boosting algorithm for information retrieval. In *SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 391–398, New York, NY, USA, 2007. ACM.