Learning to Rank with Labeled Features

Fernando Diaz
Microsoft
fdiaz@microsoft.com

ABSTRACT

Classic learning to rank algorithms are trained using a set of labeled documents, pairs of documents, or rankings of documents. Unfortunately, in many situations, gathering such labels requires significant overhead in terms of time and money. We present an algorithm for training a learning to rank model using a set of labeled features elicited from system designers or domain experts. Labeled features incorporate a system designer's belief about the correlation between certain features and relative relevance. We demonstrate the efficacy of our model on a public learning to rank dataset. Our results show that we outperform our baselines even when using as little as a single feature label.

CCS Concepts

•Information systems → Learning to rank;

Keywords

learning to rank

1. INTRODUCTION

Ranking is a fundamental subproblem of information retrieval research. Much of the early work in information retrieval studied text ranking in general and resulted in the development of several core retrieval functions. Ranking signals such as BM25 [16] have the advantage of portability to new domains, with little or no parametric tuning necessary for effective performance. Conversely, modern retrieval systems often employ machine learning to tune a large number of ranking function parameters by using a set of labeled query-document pairs [11].

Despite the success of machine learned ranking models, acquiring labeled data remains a pain point for system designers and researchers. Gathering editorially judged data requires nontrivial overhead in terms of time (e.g. drafting relevance guidelines, training assessors, waiting for judgments) and money (e.g. developing an annotation system, paying assessors). By the same token, gathering user interaction data (e.g. clicks) requires an existing ranking system that already performs well, usually as a result of training from editorial data. This is not to mention the prerequisite of already having users. Enterprise search scenarios highlight the problems with traditional methods of learning to rank. Enterprise search providers must design systems with little or no knowledge about the target queries or corpus; often they default to a standard ranking signal such as BM25. Enterprise search customers, on the other hand, do not have the expertise or resources to gather or train models based on editorial or behavioral data. As a result, customers must either settle for a default ranking function or handtune elaborate ranking heuristics.

While existing solutions such as active sampling focus on reducing the cost of data acquisition [2, 3], these methods all require the development of an assessment infrastructure, including drafting relevance guidelines and training assessors.

In this paper, we contribute a new learning to rank paradigm where a system must estimate model parameters given a set of labeled ranking features. This problem, which we refer to as *learning to rank with labeled features*, offers an opportunity for saving money and time. In the context of enterprise search, systems that learning to rank with labeled features provide a means for customers to indicate which ranking signals they believe are important for their domain, without explicitly reasoning about the functional form of the final ranking function.

2. RELATED WORK

Much of the foundational work in information retrieval concentrated on understanding the fundamental principles involved in text ranking. These included notions of saturating term frequency, inverse document frequency, and term proximity. Fang *et al.* present many of these principles as information retrieval axioms common across several retrieval models [8]. As such, we have an understanding of, in general, what ranking signals are appropriate in different retrieval situations.

Initializing a model with prior knowledge has received attention in information retrieval. Mohan *et al.* present an algorithm for incorporating domain knowledge into the first stage of boosting [13]. Similarly, in some cases, knowledge from a related vertical or market can be exploited to warm start a ranking model [4, 12]. However, all of these techniques require editorially labeled queries and documents.

Our work is most similar to learning classifiers from labeled features. While Schapire *et al.* introduced the prob-

ICTIR '16, September 12-16, 2016, Newark, DE, USA
© 2016 ACM. ISBN 978-1-4503-4497-5/16/09…$15.00
DOI: http://dx.doi.org/10.1145/2970398.2970435

lem of learning from feature labels [17], others have extended this work to other modeling frameworks [15, 7]. Our work can be seen as extending this to thread of research from classification to learning to rank.

3. LEARNING TO RANK WITH LABELED FEATURES

Given a query q, each document in the corpus can be represented as a vector m feature values including factors such as the BM25 score, PageRank, and other ranking signals. The design matrix $\mathbf{\Phi}^q$ is the $n \times m$ matrix of feature values for all n documents in the corpus. The (unobserved) relevance of every document in the corpus to q is represented in the $n \times 1$ vector $\mathbf{y}^q$. For clarity, we will omit q from the notation.

Domain experts provide the system with direction through a set of feature labels. Each label encodes the confidence of i being ranked above j given a difference in the feature values of i and j. For example, the expert may believe, "if the value of the BM25 score of i is higher than that of j, then my confidence that $i \succ j$ scales linearly with the difference in the BM25 scores." More generally, a feature label is *a*) a ranking feature name and *b*) a scaling factor indicating the confidence in the judgment. In other words, if the candidate ranking feature is k, then for two documents i and j, we would like the expert to provide a scalar grade $\tilde{w}_k$ such that $\tilde{w}_k \times (\Phi_{ik} - \Phi_{jk})$ is proportional to the confidence that i is preferred to j. A positive score indicates a positive correlation between the feature difference and document ordering. A negative score indicates a negative correlation between the feature difference and document ordering. The magnitude reflects the confidence in the judgment (i.e. a score of zero indicates no preference). The set of feature labels is represented as a sparse $m \times 1$ vector $\tilde{\mathbf{w}}$.

As with most learning to rank problems, the goal is to learn a function $f : \Re^m \to \Re$ such that, when documents are sorted by the value of this function, some retrieval evaluation metric is maximized.

Whereas traditional learning to rank domains involve labeled documents (i.e. $\{\langle \mathbf{\Phi}^q, \mathbf{y}^q \rangle\}_{q \in \mathcal{Q}}$), we only provide the system with a set of unlabeled documents (i.e. $\{\mathbf{\Phi}^q\}_{q \in \mathcal{Q}}$) in addition to $\tilde{\mathbf{w}}$. An unlabeled dataset can be gathered offline with an indexed corpus and a set of queries.

4. AN ALGORITHM

Our approach to learning to rank with labeled features builds on existing work. Specifically, our functional form and training method follows LambdaRank [5]. We will begin with a review of this approach for labeled documents before extending it to labeled features.

4.1 Maximum Entropy Learning to Rank

Given a document i, we would like to learn a linear model, $\mathbf{w} \in \Re^m$, such that ranking documents by $f(\mathbf{\Phi}_i) = \mathbf{w}^\mathsf{T}\mathbf{\Phi}_i$ places relevant documents near the top. In learning to rank with labeled documents, we can train a linear function by using a pairwise ranking model. For example, we can estimate the probability of an ordering of two documents as a logistic function of the individual document retrieval scores [5],

$$p(\mathbf{y}_i > \mathbf{y}_j | \mathbf{\Phi}_i, \mathbf{\Phi}_j) = \frac{1}{1 + \exp(-(f(\mathbf{\Phi}_i) - f(\mathbf{\Phi}_j)))} \quad (1)$$

$$= \frac{1}{1 + \exp(\sum_k w_k(\Phi_{jk} - \Phi_{ik}))} \quad (2)$$

Notice that Equation 2 highlights that such a model is learning a maximum entropy model based on the difference in feature values for pairs of instances. This allows us to train a model with observed preferences between documents instead of absolute document labels. The trained parameters, $\mathbf{w}$, for this model, then, can also be used to rank documents by $\mathbf{w}^\mathsf{T}\mathbf{\Phi}_i$. A similar intuition is used in the derivation of SVM$^{\text{rank}}$[10], although in this case we are optimizing for logistic loss. It is important to recognize that this framework optimizes the Kendall's τ between the model ranking and the ideal ranking [10, Section 3] instead of a core retrieval metric like NDCG.

When using labeled documents, we can infer a set of preference labels from the document labels (i.e. documents with high labels are preferred to those with low labels) and construct a new training set of the form, $\{\langle i \succ j, \mathbf{\Phi}_i, \mathbf{\Phi}_j \rangle\}$. With this preference data, we can learn the parameters $\mathbf{w}$ using stochastic gradient descent where the gradient magnitude is merely the prediction error,

$$\Delta_{ij} = \mathrm{I}(i \succ j) - p(\mathbf{y}_i > \mathbf{y}_j | \mathbf{\Phi}_i, \mathbf{\Phi}_j, \mathbf{w}) \quad (3)$$

This is the approach underlying RankNet [5].

The problem with the pairwise loss is that, although related, information retrieval evaluation metrics are often subtler than Kendall's τ. Metrics such as NDCG incorporate rank preference and graded relevance. Burges *et al.* introduce the notion of weighting the gradients in Equation 3 with the magitude of the rank loss [5]. In practice, we multiply the gradient by the change in the metric,

$$\alpha_{i,j} \equiv |\delta \mathcal{N}^{\mathbf{y}}_{i,j}| \times \Delta_{ij} \quad (4)$$

where we set $\delta \mathcal{N}^{\mathbf{y}}_{i,j}$ to be the change in NDCG from swapping documents in positions i and j.

As a result update rule for stochastic gradient descent over pairs of ordered instances is,

$$\mathbf{w}^t = \mathbf{w}^{t-1} + \gamma \alpha_{i,j}(\mathbf{\Phi}_i - \mathbf{\Phi}_j) \quad (5)$$

where γ is a learning rate.

4.2 Maximum Entropy with Labeled Features

Although we cannot infer preferences from document labels, we can estimate the preference with a simple model derived from $\tilde{\mathbf{w}}$. Specifically, we adopt a functional form consistent with Equation 2,

$$p(\mathbf{y}_i > \mathbf{y}_j | \mathbf{\Phi}_i, \mathbf{\Phi}_j, \tilde{\mathbf{w}}) = \frac{1}{1 + \exp(\sum_k \tilde{w}_k(\Phi_{jk} - \Phi_{ik}))} \quad (6)$$

where the score for document i is $f(\mathbf{\Phi}_i) = \tilde{\mathbf{w}}^\mathsf{T}\mathbf{\Phi}_i$. While there are many other methods for combining feature labels, we leave an exploration of this for future work.

If we replace the indicator function in Equation 3 with Equation 6, our gradient magnitude is,

$$\tilde{\Delta}_{ij} = p(\mathbf{y}_i > \mathbf{y}_j | \mathbf{\Phi}_i, \mathbf{\Phi}_j, \tilde{\mathbf{w}}) - p(\mathbf{y}_i > \mathbf{y}_j | \mathbf{\Phi}_i, \mathbf{\Phi}_j, \mathbf{w}) \quad (7)$$

As with labeled documents, we multiply the gradient by the change in the metric. Unfortunately, we cannot adopt a metric based on labeled documents. Instead, we adopt a preference-based version of NDCG [6] define as,[1]

$$\mathcal{N}^{\tilde{\mathbf{w}}}(\mathcal{O}_k) = \frac{1}{\mathcal{Z}} \sum_{\{(i,j):i\in\mathcal{O}_k\}} \frac{p(\mathbf{y}_i > \mathbf{y}_j | \mathbf{\Phi}_i, \mathbf{\Phi}_j, \tilde{\mathbf{w}})}{\log_2(\min\{\rho_i, \rho_j\} + 1)} \tag{8}$$

where the set $\mathcal{O}_k$ is defined as the top k set of documents ordered by the model, ρ_i is the rank of document i in the model ordering, and $\mathcal{Z}$ is a discounted preference gain of the optimal ordering. We can compute a $\delta\mathcal{N}^{\tilde{\mathbf{w}}}_{i,j}$, and as a result $\tilde{\alpha}_{i,j}$, in the same way as done with document-based NDCG.

Given a large pool of unlabeled documents, $\mathcal{U}$, we can use stochastic gradient descent to train the parameters with Equation 5, replacing $\alpha_{i,j}$ with $\tilde{\alpha}_{i,j}$.

4.3 Regularization

Readers may note that, unconstrained and with the right initialization of $\mathbf{w}$, following the gradient in Equation 7 will recover the original weights $\tilde{\mathbf{w}}$, rendering the whole process pointless. We can avoid this degenerate solution by introducing a regularization constraint that prevents the model from concentrating too much of $\mathbf{w}$ in a few features. We adopt an ℓ_2 penalty since it is of standard practice in gradient descent. This coverts our update rule into,

$$\mathbf{w}^t = (1 - \lambda\gamma)\mathbf{w}^{t-1} + \gamma\tilde{\alpha}_{i,j}(\mathbf{\Phi}_i - \mathbf{\Phi}_j) \tag{9}$$

where λ controls how much we penalize a model for having concentrated feature weights.

5. METHODS

We use the Microsoft Learning to Rank Dataset [14], consisting of 30,000 queries, with a standard training, validation, and testing splits. Documents are represented as 136-dimensional feature vectors, each rated on a five point relevance scale. We only use relevance labels for evaluation, never for training.[2]

We considered feature labels following a symmetric five point scale, $\sigma \in \{-2, -1, 0, 1, 2\}$. We then encode this feature level information into $\tilde{\mathbf{w}}$. We consider two feature label sets. The first consists of a single feature, the BM25 score of the title field, with a score of 2. In other words, this would represent an expert being very confident that a document with a higher BM25 score in the title should be preferred to another with a lower BM25 score in the title. The second feature label set consists of five features chosen by an information retrieval expert. The features include the BM25 score of the title, anchor text, and body as well as two query independent features, PageRank and a document quality score. Our expert has a PhD in information retrieval and is familiar with the various feature names but has never previously conducted research with the Microsoft dataset. We plan on study the sensitivity of our approach to expertise level in future work.

We use a γ of 10^{-5} and a regularization weight λ of 0.5. We leave as future work the automatic tuning of these parameters absent a labeled validation set. As a baseline, we rank documents by $\tilde{\mathbf{w}}^\mathsf{T}\mathbf{\Phi}_i$, representing a naïve linear combination of labeled features. In all cases, including for baselines, we perform per-query feature normalization as is customary in the learning to rank literature.

We measure performance as a function of the number of *unlabeled* queries used for estimating the model parameters (Equation 9). For space reasons, we only present results for the mean NDCG.

6. RESULTS

We present the performance of our model as a function of unlabeled training set size in Figure 1. As we can see, our trained model are able to use $\tilde{\mathbf{w}}$ in order to learn a superior dense model $\mathbf{w}$. Interestingly, we are able to significantly outperform the straightforward baseline with as few as 100 queries for our single feature label condition and 500 queries for our multiple feature label condition. Recall that using handcrafted ranking function such as $\tilde{\mathbf{w}}^\mathsf{T}\mathbf{\Phi}_i$ remains the best approach for many domains. That said, we found performance to be roughly 65% of that when using the full set of labeled documents, suggesting that, if available, system designers should use these judgments.

In order to understand the model, we can inspect the feature weights $\mathbf{w}$ in Table 1. Both of our models demonstrate that highly weighted features tend to be correlated with the labeled features. For example, labeling the BM25 title results in a model which also highly weights other representations of the title field. Similarly, the model learned from expert feature labels heavily weights anchor text features, not only because these are correlated with the labeled anchor feature but also because they are correlated with the combination of other labeled features. The PageRank feature remains highly weighted because it is unlikely to be correlated with other features based mostly on the content. Interestingly, across both conditions we notice negative weights for features associated with document length and URL length. This suggests that very long documents or those deep in a website hierarchy are unlikely to be preferred by our labeled features.

7. CONCLUSION

Because unlabeled documents feature vectors can be easily gathered offline, we believe methods based on labeled features point to a compelling direction for low cost learning to rank model development.

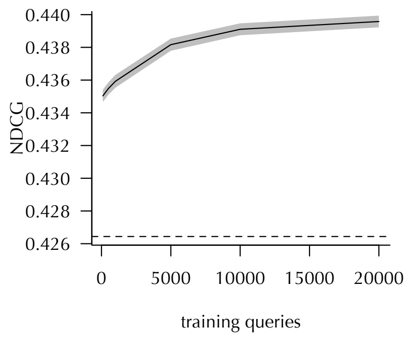
Our preliminary algorithm outperforms a straightforward baseline but there are many opportunities for further research. From a modeling perspective, we believe that novel optimization objectives or functional forms can further improve performance. This may include the development of new preference-based evaluation metrics for training or setting hyperparameters. In addition to core modeling, there is opportunity to develop novel interfaces for eliciting feature labels such as those developed for text classification [1].

8. REFERENCES

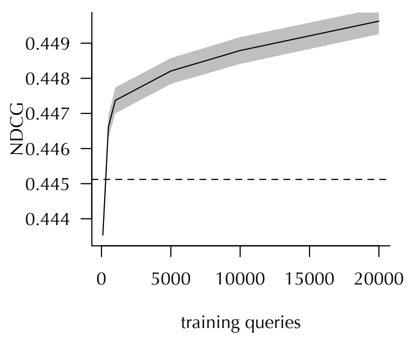
[1] S. Amershi, M. Chickering, S. M. Drucker, B. Lee, P. Simard, and J. Suh. Modeltracker: Redesigning performance analysis tools for machine learning. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pages 337–346, 2015.

[1]We adjust the gain to use the probability of preference instead of an absolute label difference.

[2]We avoid using the LETOR 3.0 and 4.0 as they have methodological issues [9]. Other datasets such as those from Yahoo and Yandex only provide numeric feature indexes, not interpretable names required for feature assessment.



(a) BM25 title feature label



(b) Expert feature labels

Figure 1: Results for MSLR-30k dataset with simple and expert feature labels. The dashed line indicates the performance of our baseline $\tilde{\mathbf{w}}^{\mathsf{T}}\mathbf{\Phi}_i$. Shaded regions represent one standard error across twenty five random training samples of the given size. All improvements for training size great than or equal to 100 are statistically significant (t-test: $p < 0.001$).

Table 1: Sorted feature weights for our two feature label sets. Models shown for 5000 unlabeled instances and $\lambda = 1$. Labeled features are bolded.

BM25 title		expert	
feature$_k$	w_k	feature$_k$	w_k
BM25-title	**0.547**	**pageRank**	**0.817**
normTF-title	0.469	**BM25-anchor**	**0.392**
meanNormTF-title	0.469	VSM-anchor	0.382
LMABS-title	0.403	QTermRatio-anchor	0.330
LMJM-title	0.370	QTerm-anchor	0.330
maxNormTF-title	0.362	**BM25-title**	**0.319**
minNormTF-title	0.299	normTF-anchor	0.311
VSM-title	0.253	meanNormTF-anchor	0.311
QTerm-title	0.231	maxNormTF-anchor	0.270
QTermRatio-title	0.231	**quality**	**0.263**
		⋮	
bool-URL	-0.015	bool-URL	-0.001
DL-anchor	-0.016	siteRank	-0.006
DL-body	-0.016	minTFIDF-URL	-0.007
lengthURL	-0.018	minTF-URL	-0.008
DL-all	-0.018	DL-body	-0.039
DL-URL	-0.023	DL-all	-0.040
outLinks	-0.026	DL-title	-0.092
pageRank	-0.033	URL-length	-0.129
siteRank	-0.054	DL-URL	-0.131
DL-title	-0.164	URL-slashes	-0.134

[2] J. A. Aslam, E. Kanoulas, V. Pavlu, S. Savev, and E. Yilmaz. Document selection methodologies for efficient and effective learning-to-rank. In *SIGIR*, pages 468–475, 2009.

[3] J. A. Aslam, V. Pavlu, and E. Yilmaz. A statistical method for system evaluation using incomplete judgments. In *SIGIR*, pages 541–548, 2006.

[4] J. Bai, F. Diaz, Y. Chang, Z. Zheng, and K. Chen. Cross-market model adaptation with pairwise preference data for web search ranking. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 18–26, 2010.

[5] C. J. Burges. From ranknet to lambdarank to lambdamart: An overview. Technical Report MSR-TR-2010-82, Microsoft Research, 2010.

[6] B. Carterette and P. N. Bennett. Evaluation measures for preference judgments. In *SIGIR*, pages 685–686, 2008.

[7] G. Druck, G. Mann, and A. McCallum. Learning from labeled features using generalized expectation criteria. In *SIGIR*, pages 595–602, 2008.

[8] H. Fang and C. Zhai. An exploration of axiomatic approaches to information retrieval. In *SIGIR*, pages 480–487, 2005.

[9] G. C. M. Gomes, V. C. Oliveira, J. M. Almeida, and M. A. Gonçalves. Is learning to rank worth it? a statistical analysis of learning to rank methods. *Journal of Information and Data Management*, 4(1):57–66, February 2013.

[10] T. Joachims. Optimizing search engines using clickthrough data. In *KDD*, pages 133–142, 2002.

[11] T.-Y. Liu. *Learning to Rank for Information Retrieval*. Springer, 2011.

[12] B. Long, Y. Chang, A. Dong, and J. He. Pairwise cross-domain factor model for heterogeneous transfer ranking. In *WSDM*, pages 113–122, 2012.

[13] A. Mohan, Z. Chen, and K. Q. Weinberger. Web-search ranking with initialized gradient boosted regression trees. *Journal of Machine Learning Research, Workshop and Conference Proceedings*, 14:77–89, 2011.

[14] T. Qin, T.-Y. Liu, W. Ding, J. Xu, and H. Li. Microsoft learning to rank datasets, May 2010.

[15] H. Raghavan, O. Madani, and R. Jones. Active learning with feedback on features and instances. *J. Mach. Learn. Res.*, 7:1655–1686, Dec. 2006.

[16] S. E. Robertson, S. Walker, S. Jones, M. Hancock-Beaulieu, and M. Gatford. Okapi at trec-3. In *Proceedings of the Third Text REtrieval Conference*, 1994.

[17] R. E. Schapire, M. Rochery, M. G. Rahim, and N. Gupta. Incorporating prior knowledge into boosting. In *ICML*, pages 538–545, 2002.