Learning To Rank Resources

Zhuyun Dai
Carnegie Mellon University
zhuyund@cs.cmu.edu

Yubin Kim
Carnegie Mellon University
yubink@cs.cmu.edu

Jamie Callan
Carnegie Mellon University
callan@cs.cmu.edu

ABSTRACT

We present a learning-to-rank approach for resource selection. We develop features for resource ranking and present a training approach that does not require human judgments. Our method is well-suited to environments with a large number of resources such as selective search, is an improvement over the state-of-the-art in resource selection for selective search, and is statistically equivalent to exhaustive search even for recall-oriented metrics such as MAP@1000, an area in which selective search was lacking.

KEYWORDS

selective search, resource selection, federated search

ACM Reference format:
Zhuyun Dai, Yubin Kim, and Jamie Callan. 2017. Learning To Rank Resources. In *Proceedings of SIGIR '17, August 07-11, 2017, Shinjuku, Tokyo, Japan, ,* 4 pages.
DOI: 10.1145/3077136.3080657

1 INTRODUCTION

Selective search is a federated search architecture where a collection is clustered into topical *shards*. At query time, a resource selection algorithm is used to select a small subset of shards to search.

Recent work showed that while selective search is equivalent to exhaustive search for shallow metrics (e.g. P@10), it performs worse for recall-oriented metrics (e.g. MAP) [5]. This is a problem because modern retrieval systems apply re-ranking operations to a base retrieval, which can require deep result lists [10].

In this paper, we present learning to rank resources, a resource selection method based on learning-to-rank. While learning-to-rank has been widely studied for ranking documents, its application to ranking resources has not been studied in depth. We take advantage of characteristics of the resource ranking problem that are distinct from document ranking; we present new features; and we propose a training approach that uses exhaustive search results as the gold standard and show that human judgments are not necessary.

Our approach is suitable for efficiently ranking the hundreds of shards produced by selective search and is an improvement over the state-of-the-art in resource selection for selective search. In addition, our approach is statistically equivalent to exhaustive search in MAP@1000, a deep recall-oriented metric.

SIGIR '17, August 07-11, 2017, Shinjuku, Tokyo, Japan

DOI: 10.1145/3077136.3080657

2 RELATED WORK

There are three main classes of resource selection algorithms: term-based, sample-based, and supervised approaches. Term-based algorithms models the language distribution of each shard. At query time, they determine the relevance of a shard by comparing the query to the stored language model [1, 12]. Sample-based algorithms estimate the relevance of a shard by querying a small sample index of the collection, known as the centralized sample index (CSI) [9, 11, 13, 14]. Supervised methods use training data to learn models to evaluate shard relevance, with most methods training a classifier per shard [2, 4]. However, training a classifier for every shard is expensive in selective search, where shards number in hundreds. Thus, supervised methods have not been used for selective search. Techniques that train a single classifier would be more suitable for selective search. Balog [3] trained a learning-to-rank algorithm for a TREC task and Hong et al. [6] learned a joint probabilistic classifier. The latter is used as a baseline in this work.

3 MODEL

Let q denote a query and $\Phi(q, s_i)$ denote features extracted from the ith shard for the query. The goal of learning-to-rank is to find a shard scoring function $f(\Phi(q,s))$ that can minimize the loss function defined as: $L(f) = \int_{q \in Q} l(q,f) dP(q)$. We use $l(q,f) = \sum_{s_i >_q s_j} 1\{f(\Phi(q,s_i)) < f(\Phi(q,s_j))\}$, where $s_i >_q s_j$ denotes shard pairs for which s_i is ranked higher than s_j in the gold standard shard ranking w.r.t query q.

We used SVM^{rank} [7], which optimizes pair-wise loss. List-wise algorithms such as ListMLE [16] produced similar results, thus we only report results with SVM^{rank}.

The training process requires a gold standard shard ranking for each training query. We propose two definitions of the ground truth, *relevance-based* and *overlap-based*. In the *relevance-based* approach, the optimal shard ranking is determined by the number of relevant documents a shard contains. Thus, the training data require queries with relevance judgments, which can be expensive to obtain. The *overlap-based* approach assumes that the goal of selective search is to reproduce the document ranking of exhaustive search. The optimal shard ranking is determined by the number of documents in a shard that were ranked highly by exhaustive search. This does not require manual relevance judgments.

4 FEATURES

4.1 Query-Independent Information

Shard Popularity: Indicates how often the shard had relevant (relevance-based) or top-ranked (overlap-based) documents for training queries. It is query-independent and acts as a shard prior.

4.2 Term-Based Statistics

Term-based features can be easily precomputed, thus are efficient.

Taily Features: One feature is the Taily [1] score calculated for query q and shard s. However, Taily scores can vary greatly across shards and queries. For robustness, we add two additional features. If shard s is ranked r_s for query q, the *inverse rank* is $1/r_s$, which directly describes the importance of s relative to other shards. The *binned rank* is $ceiling(r_s/b)$, where b is a bin-size. We use $b = 10$, meaning that every 10 consecutive shards are considered equally relevant. This feature helps the model to ignore small differences between shards with similar rankings.

Champion List Features: For each query term, the top-k best documents were found. The number of documents each shard contributes to the top-k was stored for each shard-term pair. For multi-term queries, the feature values of each query term were summed. We use two values of $k = \{10, 100\}$, generating 2 features.

Query Likelihood Features: The log-likelihood of a query with respect to the unigram language model of each shard is: $L(q|s) = \sum_{t \in q} \log p(t|s)$, where $p(t|s)$ is the shard language model, the average of all document language models $p(t|d)$ in the shard. Document language model $p(t|d)$ is estimated using MLE with Jelinek-Mercer smoothing. Query likelihood, inverse query likelihood, and binned query likelihood features are created for *body*, *title*, and *inlink* representations, yielding a total of 9 features.

Query Term Statistics: The maximum and minimum shard term frequency across query terms, e.g. $stf_{max}(q,s) = \max_{t \in q} stf(t,s)$, where $stf(t,s)$ is the frequency of term t in shard s. We include the maximum and minimum of $stf \cdot idf$ where idf is the inverse document frequency over the collection. These 4 features are created for *body*, *title*, and *inlink* representations, yielding 12 features.

Bigram Log Frequency: The frequency of each bigram of the query in a shard is $bf_q(s) = \sum_{b \in q} \log bf_b(s)$, where $bf_b(s)$ is the frequency of bigram b in shard s. This feature can estimate term correlation. To save storage, we only store bigrams that appear more than 50 times in the collection.

4.3 Sample-Document (CSI-Based) Features

These features are based on retrieval from the centralized sample index (CSI), which may provide term co-occurrence information. CSI retrieval is expensive, and thus is slower to calculate.

Rank-S and ReDDE Features: Similar to Taily features, the shard scores given by Rank-S [9] and ReDDE [13], as well as the inverse rank and binned rank features for a total of 6 features.

Average Distance to Shard Centroid: The distance between the top-k documents retrieved from the CSI to their respective shards' centroids. Intuitively, if the retrieved documents are close to the centroid, the shard is more likely to contain other similar, highly-scoring documents. For multiple documents from the same shard, the distances are averaged. We use two distance metrics: KL divergence and cosine similarity Note that because KL divergence measures distance rather than similarity, we use the inverse of the averaged KL divergence as the metric. We generated features for $k = \{10, 100\}$ and also a feature measuring the distance between the shard's centroid to its single highest scoring document in the top 100 of the CSI results, for a total of 6 features.

5 EXPERIMENTAL METHODOLOGY

Datasets: Experiments were conducted with ClueWeb09-B and Gov2. ClueWeb09-B (CW09-B) consists of 50 million pages from the ClueWeb09 dataset. Gov2 is 25 million web pages from the US government web domains. For *relevance-based* models, 200 queries from the TREC 09-12 Web Track topics were used for CW09-B, and 150 queries from the TREC 04-06 Terabyte Track topics were used for Gov2. Models were trained by 10-fold cross-validation. For *overlap-based* models, training queries were sampled from the AOL and Million Query Track query logs. Models were tested with the TREC queries. Optimal shard ranking for the overlap method was defined by the number of documents each shard contains that were within the top $N = 2K$ retrieved from exhaustive search. We found $N \in [1K, 3K]$ produced stable results.

Proposed methods and baselines: We used three sources of training data: relevance-based training data (`L2R-TREC`), and overlap-based training data (`L2R-AOL` and `L2R-MQT`). We used linear SVM^{rank}, where C was chosen by cross validation. Our method was compared against state-of-the-art unsupervised models (`Taily` [1], `ReDDE` [13], and `Rank-S` [9]); and a supervised model `Jnt` [6]. `Jnt` was trained and tested using TREC queries with 10-fold cross-validation.

Evaluation Metrics: Search accuracy was measured by P@10, NDCG@30 and MAP@1000. To test the proposed methods' superiority to baselines, a query-level permutation test with $p < 0.05$ was used. To test the equivalence to exhaustive search, a non-inferiority test [15] was used to assert that results of the more efficient selective search were at least as accurate as exhaustive search. The equivalence is established by rejecting the null hypothesis that selective search is at least 5% worse than exhaustive search with a 95% confidence interval.

Selective Search Setup: We used 123 shards for CW09-B and 199 shards for Gov2 [5]. A 1% central sample index (CSI) was created for `ReDDE` and `Rank-S` baselines and CSI based features. `Jnt` followed the original implementation and used a 3% CSI.

Search Engine Setup: Retrieval was performed with Indri, using default parameters. Queries were issued using the sequential dependency model (SDM) with parameters $(0.8, 0.1, 0.1)$. For CW09-B, documents with a Waterloo spam score below 50 were removed [1].

6 EXPERIMENTS

6.1 Overall Comparison

Our method was compared to four baselines and exhaustive search. We tested shard rank cutoffs from 1–8% of total shards; 10 for CW09-B and 16 for Gov2. The automatic cutoffs of Rank-S and Taily performed similarly to fixed cutoffs and are not shown. Shard rankings by `L2R` enabled more accurate search than all baselines in both datasets (Figure 1). The search accuracy of `L2R` models is higher than the baselines at nearly every shard rank cutoff.

Table 1 compares `L2R` models and the baselines at two shard rank cutoffs. The first cutoff is the point where the shallow metrics (P@10 and NDCG@30) stablize: 4 for CW09-B and 6 for Gov2. The second cutoff is where MAP@1000 become stable: 8 for CW09-B and 12 for Gov2. `L2R` models improve over the baselines at both shard cutoffs. For shallow metrics, `L2R` reaches exhaustive search

[1] https://plg.uwaterloo.ca/ gvcormac/

at the first cutoff. Furthermore, searching the first 8 out of 12 shards ranked by L2R is statistically non-inferior to searching all shards exhaustively, even for the recall-oriented MAP@1000. All the baselines have a 10% gap from exhaustive search in MAP@1000.

6.2 Effect of Training Data

One might expect the relevance-based model (L2R-TREC) to be better than overlap-based models (L2R-AOL and AOL-MQT), because it uses manual relevance judgments. A model trained with overlap data might favor shards that contain false-positive documents. However, there is little difference between the two training methods. L2R-TREC was statistically better than TREC or AOL for MAP@1000 in Gov2, but the relative gain is only 2%; in all other cases, there is no statistically significant differences among the three models. Furthermore, models trained with relevance and overlap data agreed on which features are important (not shown due to space constraints). This analysis indicates that unlike learning to rank document models, we can train a learning to rank resource selector on a new dataset before we have relevance judgments.

6.3 Query Length

We compare L2R-MQT to the baselines using MAP@1000 for queries with different lengths on CW09-B, shown in Figure 1. Gov2 and other training data produced similar results and are not shown. For single-term queries, existing methods are already equivalent to or better than exhaustive search, and L2R-MQT retains this good performance. The advantage of L2R-MQT comes from multi-term queries, where the best baseline Jnt still has a 10% gap from exhaustive search. For these queries, the improvement of L2R-MQT over the Taily is expected, because Taily does not model term co-occurrence. However, L2R-MQT also out-performs ReDDE and Rank-S, which account for term co-occurrence by retrieving documents from the CSI, but are limited by only having a sample view of the collection. L2R draws evidence from both the sample and the whole collection. Jnt also fuses sample- and term-based features, but most of its features are derived from ReDDE or Taily-like methods and do not carry new information. L2R improved over Jnt by using novel features that encode new evidence.

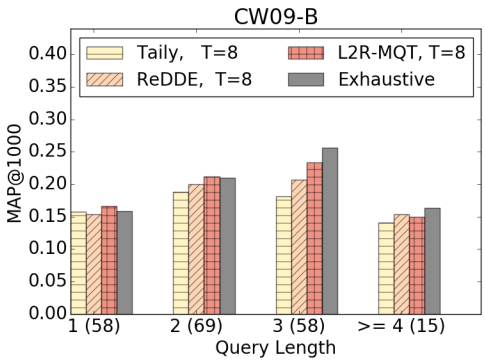

Figure 1: MAP@1000 for queries on CW09-B, grouped by query length. Parentheses on the X axis present the number of queries in each group. T is the shard rank cutoff.

Table 2: Effectiveness and efficiency of FAST features. ALL uses all features. FAST does not use sample-document features. T: shard rank cutoff. $*$: non-inferiority to exhaustive.

	Method	P @10	NDCG @30	MAP @1000	Average Cost
Cw09 -B (T=8)	Redde	0.363*	0.275*	0.187	156,180
	Taily	0.346	0.260	0.175	470
	Jnt	0.367*	0.277*	0.192	468,710
	ALL	0.375*	0.286*	0.202*	158,529
	FAST	0.373*	0.285*	0.201*	2,349
Gov2 (T=12)	Redde	0.579*	0.445*	0.289	105,080
	Taily	0.518	0.403	0.256	758
	Jnt	0.588*	0.465*	0.292	315,875
	ALL	0.593*	0.474*	0.309*	108,306
	FAST	0.587*	0.471*	0.310*	3,226

6.4 Feature Analysis

The L2R approach uses three classes of features: query-independent, term-based, and sample-document (CSI). These three feature classes have substantially different computational costs and contributions.

Fast vs. Slow features: Sample-document (CSI-based) features have a high computational cost, because they search a sample (typically 1-2%) of the entire corpus. Term-based features have a low computational cost, because they lookup just a few statistics per query term per shard. Costs for query-independent features are lower still. The third experiment compares a *slow* model that uses all features (ALL) to a *fast* version that does not use sample-document features (FAST).

We estimate the resource selection cost by the amount of data retrieved from storage. For CSI-based features, the cost is the size of postings of every query term in the CSI. For term-based features, the cost is the amount of *sufficient* statistics required to derive all term-based features. The query-independent feature only looks up the shard popularity, so the cost is one statistic per shard.

Table 2 compares FAST with ALL and baselines by their accuracy and average resource selection cost per query. ReDDE results were similar to Rank-S and are not shown. Taily has been the state-of-the-art term-based ('faster') resource selection algorithm. However, FAST is substantially more accurate. FAST also outperformed Jnt with over 100× speed up. Compared to ALL, FAST is 67 times faster on CW09-B and 34 times faster on Gov2. Although FAST has slightly lower search accuracy than ALL, the gap is not large and is not statistically significant, indicating that the information from the CSI features can be covered by the more efficient features.

We conclude that a resource ranker composed of only query-independent and term-based features is as accurate as exhaustive search and a ranker that includes CSI features. CSI features improve accuracy slightly, but at a significant additional computational cost.

Importance of Feature Types: We investigate the contribution of other types of features: query-independent features and term-based features, where the term-based features were sub-divided into unigram and bigram features. Table 3 presents the results for the *leave-one-out* analysis conducted on FAST. On CW09-B, removing any feature set from FAST led to lower performance. This indicates that each set of features covers different types of information,

Table 1: Search accuracy comparison between 3 L2R models and baselines at two rank cutoffs for two datasets. ▲: statistically significant improvement compared to Jnt, the best resource selection baseline. ∗: non-inferiority to exhaustive search .

Method	CW09-B						Gov2					
	T=4			T=8			T=6			T=12		
	P@10	NDCG@30	MAP@1000	P@10	NDCG@30	MAP@1000	P@10	NDCG@30	MAP@1000	P@10	NDCG@30	MAP@1000
Redde	0.355	0.262	0.176	0.363*	0.275*	0.187	0.580*	0.445	0.267	0.587*	0.4600*	0.289
Rank-S	0.350	0.259	0.175	0.360*	0.268	0.183	0.570	0.440	0.263	0.585*	0.461*	0.286
Taily	0.346	0.260	0.172	0.346	0.260	0.175	0.518	0.403	0.235	0.530	0.418	0.256
Jnt	0.370*	0.269	0.178	0.367*	0.277*	0.192	0.582*	0.459	0.278	0.588*	0.465*	0.292
L2R-TREC	0.374*	0.281*	0.192$^{\blacktriangle}$	0.377*	0.286$^{\blacktriangle *}$	0.202$^{\blacktriangle *}$	0.593*	0.469*	0.299$^{\blacktriangle}$	0.591*	0.475$^{\blacktriangle *}$	0.313$^{\blacktriangle *}$
L2R-AOL	0.374*	0.281$^{\blacktriangle *}$	0.191$^{\blacktriangle}$	0.375*	0.287$^{\blacktriangle *}$	0.202$^{\blacktriangle *}$	0.593*	0.470$^{\blacktriangle *}$	0.291$^{\blacktriangle}$	0.587*	0.470*	0.307$^{\blacktriangle *}$
L2R-MQT	0.382*	0.285$^{\blacktriangle *}$	0.193$^{\blacktriangle}$	0.375*	0.286$^{\blacktriangle *}$	0.202$^{\blacktriangle *}$	0.586*	0.465*	0.292$^{\blacktriangle}$	0.593*	0.474$^{\blacktriangle *}$	0.309$^{\blacktriangle *}$
Exh	0.372	0.288	0.208	0.372	0.288	0.208	0.585	0.479	0.315	0.585	0.479	0.315

Table 3: Performance of L2R-MQT using feature sets constructed with leave-one-out. '- X' means the feature was excluded from FAST. Text in bold indicates the lowest value in the column.

	Feature Set	P@10	NDCG@30	MAP@1000
CW09 -B (T=8)	FAST	0.373	0.285	0.201
	- Unigram	**0.303**	**0.226**	**0.138**
	- Bigram	0.364	0.275	0.187
	- Independent	0.368	0.282	0.199
Gov2 (T=12)	FAST	0.592	0.471	0.310
	- Unigram	0.592	0.468	0.301
	- Bigram	**0.582**	**0.462**	**0.296**
	- Independent	0.591	0.471	0.303

and all are necessary for accurate shard ranking. Among these features, unigram features were most important because CW09-B has many single-term queries. On Gov2, the only substantial difference is observed when bigram features are excluded.

7 CONCLUSION

This paper investigates a learning-to-rank approach to resource selection for selective search. Much attention has been devoted to learning-to-rank documents, but there has been little study of learning-to-rank resources such as index shards. Our research shows that training data for this task can be generated automatically using a slower system that searches all index shards for each query. This approach assumes that the goal of selective search is to mimic the accuracy of an exhaustive search system, but with lower computational cost. This assumption is not entirely true—we would like selective search to also be more accurate—but it is convenient and effective.

We show that the learned resource selection algorithm produces search accuracy comparable to exhaustive search down to rank 1,000. This paper is the first that we know of to demonstrate results that are statistically significantly equivalent to exhaustive search for MAP@1000 on an index that does not have badly skewed shard sizes. Accuracy this deep in the rankings opens up the possibility of using a learned reranker on results returned by a selective search system, which was not practical in the past.

Most prior research found that sample-document algorithms such as ReDDE and Rank-S are a little more accurate than term-based algorithms such as Taily for selective search resource selection; however, sample-document resource selection algorithms have far higher computational costs that increases query latency in some configurations [8]. This work suggests that sample-document features provide only a small gain when combined with other types of features. It may no longer be necessary to choose between accuracy and query latency when using a learned resource ranker.

8 ACKNOWLEDGMENTS

This research was supported by National Science Foundation (NSF) grant IIS-1302206. Yubin Kim is the recipient of the Natural Sciences and Engineering Research Council of Canada PGS-D3 (438411). Any opinions, findings, and conclusions in this paper are the authors' and do not necessarily reflect those of the sponsors.

REFERENCES

[1] R. Aly, D. Hiemstra, and T. Demeester. Taily: shard selection using the tail of score distributions. pages 673–682, 2013.
[2] J. Arguello, F. Diaz, J. Callan, and J. Crespo. Sources of evidence for vertical selection. pages 315–322, 2009.
[3] K. Balog. Learning to combine collection-centric and document-centric models for resource selection. In *TREC*, 2014.
[4] S. Cetintas, L. Si, and H. Yuan. Learning from past queries for resource selection. pages 1867–1870, 2009.
[5] Z. Dai, X. Chenyan, and J. Callan. Query-biased partitioning for selective search. pages 1119–1128, 2016.
[6] D. Hong, L. Si, P. Bracke, M. Witt, and T. Juchcinski. A joint probabilistic classification model for resource selection. pages 98–105. ACM, 2010.
[7] T. Joachims. Training linear SVMs in linear time. In *Proc. SIGKDD*, pages 217–226, 2006.
[8] Y. Kim, J. Callan, J. S. Culpepper, and A. Moffat. Load-balancing in distributed selective search. pages 905–908, 2016.
[9] A. Kulkarni, A. S. Tigelaar, D. Hiemstra, and J. Callan. Shard ranking and cutoff estimation for topically partitioned collections. pages 555–564, 2012.
[10] C. Macdonald, R. L. T. Santos, and I. Ounis. The whens and hows of learning to rank for web search. *Inf. Retr.*, 16(5):584–628, 2013.
[11] I. Markov and F. Crestani. Theoretical, qualitative, and quantitative analyses of small-document approaches to resource selection. 32(2):9, 2014.
[12] H. Nottelmann and N. Fuhr. Evaluating different methods of estimating retrieval quality for resource selection. pages 290–297, 2003.
[13] L. Si and J. P. Callan. Relevant document distribution estimation method for resource selection. pages 298–305, 2003.
[14] P. Thomas and M. Shokouhi. SUSHI: scoring scaled samples for server selection. pages 419–426, 2009.
[15] E. Walker and A. S. Nowacki. Understanding equivalence and noninferiority testing. *Journal of General Internal Medicine*, 26(2):192–196, 2011.
[16] F. Xia, T.-Y. Liu, J. Wang, W. Zhang, and H. Li. Listwise approach to learning to rank: theory and algorithm. In *Proc. ICML*, pages 1192–1199, 2008.