Yahoo! Learning to Rank Challenge Overview

Olivier Chapelle* CHAP@YAHOO-INC.COM
Yi Chang YICHANG@YAHOO-INC.COM
Yahoo! Labs
Sunnyvale, CA

Abstract

Learning to rank for information retrieval has gained a lot of interest in the recent years but there is a lack for large real-world datasets to benchmark algorithms. That led us to publicly release two datasets used internally at Yahoo! for learning the web search ranking function. To promote these datasets and foster the development of state-of-the-art learning to rank algorithms, we organized the Yahoo! Learning to Rank Challenge in spring 2010. This paper provides an overview and an analysis of this challenge, along with a detailed description of the released datasets.

1. Introduction

Ranking is at the core of information retrieval: given a query, candidates documents have to be ranked according to their relevance to the query. Learning to rank is a relatively new field in which machine learning algorithms are used to learn this ranking function. It is of particular importance for web search engines to accurately tune their ranking functions as it directly affects the search experience of millions of users.

A typical setting in learning to rank is that feature vectors describing a query-document pair are constructed and relevance judgments of the documents to the query are available. A ranking function is learned based on this training data, and then applied to the test data.

Several benchmark datasets, such as LETOR, have been released to evaluate the newly proposed learning to rank algorithms. Unfortunately, their sizes – in terms of number of queries, documents and features – are still often too small to draw reliable conclusions, especially in comparison with datasets used in commercial search engines. This prompted us to released two internal datasets used by Yahoo! search, comprising of 36k queries, 883k documents and 700 different features.

To promote these datasets and encourage the research community to develop new learning to rank algorithms, we organized the Yahoo! Learning to Rank Challenge which took place from March to May 2010. There were two tracks in the challenge: a standard learning to rank track and a transfer learning track where the goal was to learn a ranking function for a small country by leveraging the larger training set of another country.

The challenge drew a huge number of participants with more than thousand teams registered. Winners were awarded cash prizes and the opportunity to present their algorithms

* The challenge was organized by O. Chapelle and Y. Chang. T.-Y. Liu joined us for organizing the workshop and editing these proceedings.

© 2011 O. Chapelle & Y. Chang.

in a workshop at the 27th International Conference on Machine Learning (ICML 2010) in Haifa, Israel. Some major findings include:

1. Decision trees were the most popular class of function among the top competitors;

2. Ensemble methods, including boosting, bagging and random forests, were dominant techniques;

3. The differences in accuracy between the winners were very small.

The paper is organized as follows. An overview of learning to rank is presented in section 2. Then section 3 reviews the different benchmark datasets for learning to rank while the details of our datasets are given in section 4. The challenge is described in section 5 which also includes some statistics on the participation. Finally section 6 presents the outcome of the challenge and an overview of the winning methods.

2. Learning to rank

This section gives an overview of the main types of learning to rank methods; a comprehensive survey of the literature can be found in (Liu, 2009).

Web page ranking has traditionally been based on a manually designed ranking function such as BM25 (Robertson and Walker, 1994). However ranking is currently considered as a supervised learning problem and several machine learning algorithms have been applied to it (Freund et al., 2003; Burges et al., 2005; Cao et al., 2006; Xu and Li, 2007; Cao et al., 2007; Cossock and Zhang, 2008; Liu, 2009; Burges, 2010).

In these methods, the training data is composed of a set of queries, a set of triples of (query, document, grade), and the grade indicates the degree of relevance of this document to its corresponding query. For example, each grade can be one element in the ordinal set,

$$\{\texttt{perfect}, \texttt{excellent}, \texttt{good}, \texttt{fair}, \texttt{bad}\} \tag{1}$$

and is labeled by human editors. The label can also simply be binary: relevant or irrelevant. Each query and each of its documents are paired together, and each query-document pair is represented by a feature vector. Thus the training data can be formally represented as: $\{(\mathbf{x}_j^q, l_j^q)\}$, where q goes from 1 to n, the number of queries, j goes from 1 to m_q, the number of documents for query q, $\mathbf{x}_j^q \in \mathbb{R}^d$ is the d-dimensional feature vector for the pair of query q and the j-th document for this query while l_j^q is the relevance label for $\mathbf{x}_j^q$.

In order to measure the quality of a search engine, some evaluation metrics are needed. The *Discounted Cumulative Gain* (DCG) has been widely used to assess relevance in the context of search engines (Jarvelin and Kekalainen, 2002) because it can handle multiple relevance grades such as (1). Therefore, when constructing a learning to rank approach, it is often beneficial to consider how to optimize model parameters with respect to these metrics during the training procedure. Many machine learning algorithms apply the gradient based techniques for parameter optimization. Unfortunately IR measures are not continuous and it is not possible to directly optimize them with gradient based approaches. Consequently many current ranking algorithms turn to optimize other objectives that can be divided into three categories:

Pointwise The objective function is of the form $\sum_{q,j} \ell(f(\mathbf{x}_j^q), l_j^q)$ where ℓ can for instance be a regression loss (Cossock and Zhang, 2008) or a classification loss (Li et al., 2008).

Pairwise Methods in this category try to order correctly pairs of documents by minimizing

$$\sum_q \sum_{i,j,\ l_i^q > l_j^q}^{m_q} \ell(f(\mathbf{x}_i^q) - f(\mathbf{x}_j^q)).$$

RankSVM (Herbrich et al., 2000; Joachims, 2002) uses $\ell(t) = \max(0, 1-t)$, while RankNet (Burges et al., 2005) uses $\ell(t) = \log(1 + \exp(-t))$. GBRank (Zheng et al., 2008) is similar to RankSVM, but uses a quadratic penalization, $\ell(t) = \max(0, 1-t)^2$ and is combined with functional gradient boosting. Finally in LambdaRank (Burges et al., 2007), the weight of each preference pair is the NDCG difference resulting from swapping that pair.

Listwise The loss function is defined over all the documents associated with the query, $\ell(\{f(\mathbf{x}_j^q)\}, \{l_j^q\})$ for $j = 1 \ldots m_q$. This type of algorithms can further be divided into two different sub-categories. The first one ignores the IR measure during training: for instance ListNet (Cao et al., 2007) and ListMLE (Xia et al., 2008) belong to this category. The second sub-category tries to optimize the IR measure during training: examples include AdaRank (Xu and Li, 2007), several algorithms based on structured estimation (Chapelle et al., 2007; Yue et al., 2007; Chakrabarti et al., 2008) and SoftRank (Taylor et al., 2008) which minimizes a smooth version of the NDCG measure.

For each query-document pair, a set of features is extracted to form a feature vector which typically consists of three parts: query-feature vector (depending only on the query), document-feature vector (depending only on the document), and query-document feature vector (depending on both). We will give in section 4.2 a high-level description of the features used in the datasets released for the challenge.

3. Motivation

A lot of papers have been published in the last 5 years in the field of learning to rank. In fact about 100 of such papers have been listed on the LETOR website.[1]

Before 2007 there was no publicly available dataset to compare learning to rank algorithms. The results reported in papers were often on proprietary datasets (Burges et al., 2005; Zheng et al., 2008) and were thus not reproducible. This hampered the research on learning to rank since algorithms could not be easily compared. The release of the LETOR benchmark (Qin et al., 2010)[2] in 2007 was a formidable boost for the development of learning to rank algorithms because researchers were able for the first time to compare their algorithms on the same benchmark datasets.

Unfortunately, the sizes of the datasets in LETOR are several orders of magnitude smaller than the ones used by search engine companies, as indicated in table 1. In particular, the limited number of queries – ranging from 50 to 106 – precludes any statistically significant difference between the algorithms to be reached. Also the number of features and relevance levels is lower than those found in commercial search engines. This became problematic

1. http://research.microsoft.com/en-us/um/beijing/projects/letor/paper.aspx
2. This paper refers to the third version of LETOR.

Table 1: Characteristics of publicly available datasets for learning to rank: number of queries, documents, relevance levels, features and year of release. The size of the 6 datasets for the '.gov' collection in LETOR have been added together. Even though this collection has a fairly large number of documents, only 2000 of them are relevant.

	Queries	Doc.	Rel.	Feat.	Year
LETOR 3.0 – Gov	575	568 k	2	64	2008
LETOR 3.0 – Ohsumed	106	16 k	3	45	2008
LETOR 4.0	2,476	85 k	3	46	2009
Yandex	20,267	213 k	5	245	2009
Yahoo!	36,251	883 k	5	700	2010
Microsoft	31,531	3,771 k	5	136	2010

because several researchers started to notice that the conclusions drawn from experimentation on LETOR datasets can be quite different than the ones from large real datasets. For instance, Taylor et al. (2008) reports in the table 1 of their paper that, on TREC data, SoftRank yields a massive 15.8% NDCG@10 improvement over mean squared error optimization; but on their internal web search data, both methods give similar results.

This observation prompted us to publicly release some of the datasets used internally at Yahoo!. We will detail in section 4 how these datasets were constructed and assembled. It is noteworthy that around the same time, the Russian search engine Yandex also released one of their internal dataset for a competition.[3] More recently, Microsoft announced the release of editorial judgments used to train the Bing ranking function along with 136 features widely used in the research community.[4]

4. Datasets

Two datasets were released for the challenge, corresponding to two different countries. The first one, named SET 1, originates from the US, while the second one, SET 2, is from an Asian country. The reason for releasing two datasets will become clear in the next section; one of the track of the challenge was indeed a transfer learning track. Both datasets are in fact a subset of the entire training set used internally to train the ranking functions of the Yahoo! search engine. Table 2 shows some statistics of these datasets.

The queries, urls and features descriptions were not disclosed, only the feature values were. There were two reasons of competitive nature for doing so:

1. Feature engineering is a critical component of any learning to rank system. For this reason, search engine companies rarely disclose the features they use. Releasing the queries and urls would lead to a risk of reverse engineering of our features.

3. Yandex, Internet Mathematics, 2009. Available at http://imat2009.yandex.ru/en/datasets.
4. Microsoft Learning to Rank Datasets, 2010. Available at http://research.microsoft.com/mslr.

Table 2: Statistics of the two datasets released for the challenge.

	SET 1			SET 2		
	Train	Valid.	Test	Train	Valid.	Test
Queries	19,944	2,994	6,983	1,266	1,266	3,798
Dococuments	473,134	71,083	165,660	34,815	34,881	103,174
Features		519			596	

2. Our editorial judgments are a valuable asset, and along with queries and urls, it could be used to train a ranking model. We would thus give a competitive advantage to potential competitors by allowing them to use our editorial judgments to train their model.

4.1. Dataset construction

This section details how queries, documents and judgments were collected.

Queries The queries were randomly sampled from the query logs of the Yahoo! search engine. Each query was given a unique identifier. A frequent query is likely to be sampled several times from the query logs, and each of these replicates has a different identifier. This was done to ensure that the query distribution in these datasets follows the same distribution as in the query logs: frequent queries have effectively more weight.

Documents The documents are selected using the so-called *pooling* strategy as adopted by TREC (Harman, 1995). The top 5 documents from different systems (different internal ranking functions as well as external search engines) are retrieved and merged. This process is typically repeated at each update of the dataset. A description of this incremental process is given in algorithm 1.

An histogram of the number of documents associated with each query can be found in figure 1. The average number of document per query is 24, but some queries have more than 100 documents. These queries are typically the most difficult ones: for difficult queries, the overlap in documents retrieved by the different engines is small and the documents change over time; these two factors explain that algorithm 1 produces a large number of documents for this type of queries.

Judgments The relevance of each document to the query has been judged by a professional editor who could give one of the 5 relevance labels of set (1). Each of these relevance labels is then converted to an integer ranging from 0 (for `bad`) to 4 (for `perfect`). There are some specific guidelines given to the editors instructing them how to perform these relevance judgments. The main purpose of these guidelines is to reduce the amount of disagreement across editors.

Table 3 shows the distribution of the relevance labels. It can be seen that there are very few `perfect`. This is because, according to the guidelines, a `perfect` is only given to the destination page of a navigational query.

Algorithm 1 High-level description of the dataset construction process.

```
Q = ∅                                                    Set of queries
J = ∅                         Dictionary of (query,document), judgment
for T = 1, ..., T do           Create J incrementally over T time steps
  Q = Q ∪ {new random queries}
  for all q ∈ Q do
    for all e ∈ E do                         E is a set of search engines
      for k = 1, ..., k_max do                          Typically k_max = 5
        u = k^th document for query q according to engine e.
        if (q, u) ∉ J then
          Get judgment j for (q, u)
          J(q, u) = j
        end if
      end for
    end for
  end for
end for
```

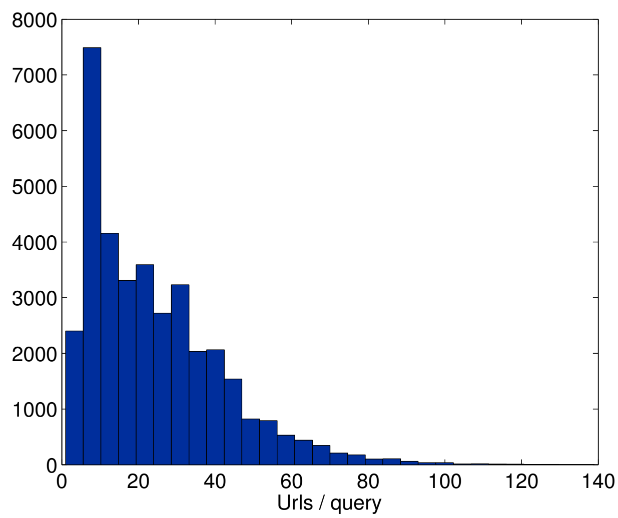


Figure 1: Distribution, over both sets, of the number of documents per query.

Table 3: Distribution of relevance labels.

Grade	Label	SET 1	SET 2
`Perfect`	4	1.67%	1.89%
`Excellent`	3	3.88%	7.67%
`Good`	2	22.30%	28.55%
`Fair`	1	50.22%	35.80%
`Bad`	0	21.92%	26.09%

4.2. Features

We now give an overview of the features released in these datasets. We cannot give specifics of how these features are computed, but instead give a high-level description, organized by feature type. Before going further, several general remarks need to be made. First, the features are described at the document level, but a lot of them have counterparts at the host level. This is to enable generalization for documents for which we have little information. Second, count features are often normalized in a sensible way. For instance, in addition to counting the number of clicks on a document, we would also compute the *click-through rate* (CTR), that is the ratio of the number of clicks to the number of impressions. Finally, some base features are often aggregated or combined into a new composite feature.

The features can be divided in the main following categories.

Web graph This type of features tries to determine the quality or the popularity of a document based on its connectivity in the web graph. Simple features are functions of the number of inlinks and outlinks while more complex ones involve some kind of propagation on the graph. A famous example is PageRank (Page et al., 1999). Other features include distance or propagation of a score from known good or bad documents (Gyöngyi et al., 2004; Joshi et al., 2007).

Document statistics These features compute some basic statistics of the document such as the number of words in various fields. This category also includes characteristics of the url, for instance the number of slashes.

Document classifier Various classifiers are applied to the document, such as spam, adult, language, main topic, quality, type of page (e.g. navigational destination vs informational). In case of a binary classifier, the feature value is the real-valued output of the classifier. In case of multiples classes, there is one feature per class.

Query Features which help in characterizing the query type: number of terms, frequency of the query and of its terms, click-through rate of the query. There are also *result set features*, that are computed as an average of other features over the top documents retrieved by a previous ranking function. For example, the average adult score of the top documents retrieved for a query is a good indicator of whether the query is an adult one or not.

Text match The most important type of features is of course the textual similarity between the query and the document; this is the largest category of features. The basic features are computed from different sections of the document (title, body, abstract, keywords) as well as from the anchor text and the url. These features are then aggregated to form new composite features. The match score can be as simple as a count or can be more complex such as BM25 (Robertson and Zaragoza, 2009). Counts can be the number of occurrences in the document, the number of missing query terms or the number of extra terms (i.e. not in the query). Some basic features are defined over the query terms, while some others are arithmetic functions (min, max, or average) of them. Finally, there are also *proximity* features which try to quantify how far in the document are the query terms (the closer the better) (Metzler and Croft, 2005).

Topical matching This type of feature tries to go beyond similarity at the word level and compute similarity at the topic level. This can for instance been done by classifying both the query and the document in a large topical taxonomy. In the context of contextual advertising, details can be found in (Broder et al., 2007).

Click These features try to incorporate the user feedback, most importantly the clicked results (Agichtein et al., 2006). They are derived either from the search or the toolbar logs. For a given query and document, different click probabilities can be computed: probability of click, first click, last click, long dwell time click or only click. Also of interest is the probability of skip (not clicked, but a document below is). If the given query is rare, these clicks features can be computed using similar, but more frequent queries. The average dwell time can be used as an indication of the landing page quality. The geographic similarity of the users clicking on a page is a useful feature to determine its localness. Finally, for a given host, the entropy of the click distribution over queries is an indication of its specificity.

External references For certain documents, some meta-information, such as Delicious tags, is available and can be use to refine the text matching features. Also documents from specific domains have additional information which can be used to evaluate the quality of the page: for instance, the rating of an answer in Yahoo! Answers documents.

Time For time sensitive queries, the freshness of a page is important. There are several features which measure the age of a document as well as the one of its inlinks and outlinks. More information on such features can be found in (Dong et al., 2010, Section 5.1).

The datasets contain thus different types of features: binary, count, and continuous ones. The categorical features have been converted to binary ones. Figure 2 hints at how many such features there are: it shows an histogram of the number of different values a features takes. There are for instance 48 binary features (taking only two values).

The features that we released are the result of a feature selection step in which the most predictive features for ranking are kept. They are typically part of the ranking system used in production. As a consequence, the datasets we are releasing are very realistic because they are used in production. The downside is that we cannot reveal the features semantics. The dataset recently released by Microsoft is different in that respect because the features description is given; they are some commonly used features in the research community and none of them is proprietary.

4.3. Final processing

Each set has been randomly split into training, validation and test subsets. As explained in the next section, the validation set is used to give immediate feedback to the participants after a submission. In order to prevent participants from optimizing on the validation set, we purposely kept it relatively small. The training part of set 2 is also quite small in order to be able to see the benefits from transfer learning. Sizes of the different subsets can be found in table 2.

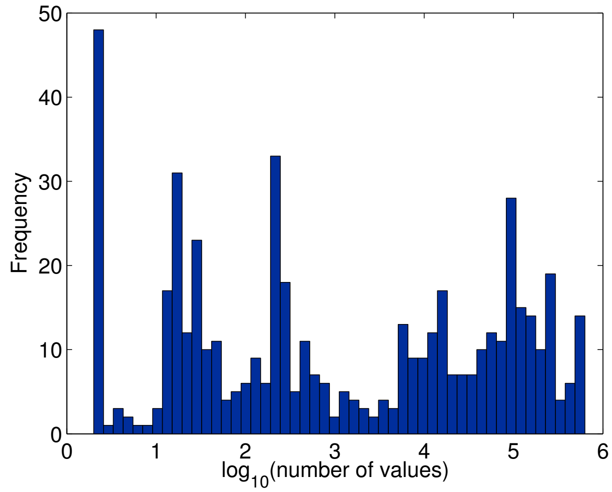


Figure 2: Number of different values for a feature. The x-axis is the number of different values and the y-axis is the number of features falling into the corresponding bin.

Table 4: Data format

```
<line> .=. <relevance> qid:<qid> <feature>:<value> ... <feature>:<value>
<relevance> .=. 0 | 1 | 2 | 3 | 4
<qid> .=. <positive integer>
<feature> .=. <positive integer>
<value> .=. <float>
```

The features are not the same on both sets: some of them are defined on SET 1 or SET 2 only, while 415 features are defined on both sets. The total number of features is 700. When a feature is undefined for a set, its value is 0. All the features have been normalized to be in the [0,1] range through the inverse cumulative distribution:

$$\tilde{x}_i := \frac{1}{n-1}|\{j,\ x_j < x_i\}|,$$

where $x_1, \ldots, x_n$ are the original values for a given feature and $\tilde{x}_i$ is the new value for the i-th example. This transformation is done simultaneously on the training, validation and test subsets of SET 1 and SET 2.

The format of the data is the same as in the SVM-light software[5] and is described in table 4. The full datasets (including validation and test labels) are available under the WEBSCOPE program at http://webscope.sandbox.yahoo.com. Note that unlike other WEBSCOPE datasets, this one is *not* restricted to academic researchers.

5. http://svmlight.joachims.org

Table 5: Performance of the 3 baselines methods on the validation and test sets of SET 1: BM25F-SD is a text match feature, RankSVM is linear pairwise learning to rank method and GBDT is a non-linear regression technique.

	Validation		Test	
	ERR	NDCG	ERR	NDCG
BM25F-SD	0.42598	0.73231	0.42853	0.73214
RankSVM	0.43109	0.75156	0.43680	0.75924
GBDT	0.45625	0.78608	0.46201	0.79013

4.4. Baselines

We report in table 5 the performance on SET 1 of 3 baseline methods:

BM25F-SD A high level description of this powerful text match feature is given in (Broder et al., 2010). It is a combination of BM25F (Robertson et al., 2004) and Metzler's sequential dependence (SD) model (Metzler and Croft, 2005), which provides an effective framework for term proximity matching. This feature has index 637 in the datasets and is one of the most predictive features for relevance.

RankSVM We considered the linear version of this pairwise ranking algorithm first introduced in (Herbrich et al., 2000). There are several efficient implementation of this algorithm that are publicly available (Joachims, 2006; Chapelle and Keerthi, 2010). We used the former code available at `http://www.cs.cornell.edu/People/tj/svm_light/svm_rank.html`.

GBDT Gradient Boosted Decision Tree (Friedman, 2002) is a simple yet very effective method for learning non-linear functions. It performs standard regression on the targets and is thus not specific to ranking. The targets were obtained by mapping the labels through the equation R used in the definition of the ERR metric (2). Publicly available packages exist (Ridgeway, 2007), but we used our internal implementation (Ye et al., 2009).

The parameters of RankSVM and GBDT were selected by a crude search on the validation set: for RankSVM, C was set to 200; and for GBDT, the parameters were set as follows: shrinkage rate = 0.05; sampling rate = 0.5; number of nodes = 20; number of trees = 2400.

5. Challenge

We first review the rules and organization of the challenge and then give some statistics about the participation and submissions.

5.1. Rules

The challenge ran from March 1st 2010 until May 31 2010. The official website was http://learningtorankchallenge.yahoo.com. The challenge was divided into two tracks, and competitors could compete in one or both of them.

1. A standard learning to rank track, using only SET 1.
2. A *transfer learning* track, where the goal is to leverage the training set from SET 1 to build a better ranking function on SET 2.

The reason for having a transfer learning track is that it has recently attracted a lot of interest (Chen et al., 2009; Gao et al., 2009; Long et al., 2009; Chapelle et al., 2011b), especially for learning web search ranking function across different countries. It is indeed expensive to collect large training sets for each individual country. We refer here to this task as *transfer learning*, but we could also have named it with the closely related concepts of *multi-task learning* or *domain adaption*.

5.1.1. Metrics

Submissions were evaluated using two criteria: the Normalized Discounted Cumulative Gain (NDCG) (Jarvelin and Kekalainen, 2002) and the Expected Reciprocal Rank (ERR) (Chapelle et al., 2009). NDCG is a popular metric for relevance judgments. Following (Burges et al., 2005), it became usual to assign exponentially high weight to highly relevant documents. We used the same formula in the challenge:

$$\mathrm{NDCG} = \frac{\mathrm{DCG}}{\text{Ideal DCG}} \quad \text{and} \quad \mathrm{DCG} = \sum_{i=1}^{\min(10,n)} \frac{2^{y_i} - 1}{\log_2(1+i)}$$

ERR is a novel metric based on the cascade user model (Craswell et al., 2008) described in algorithm 2. It is defined as the expected reciprocal rank at which the user will stop his search under this model. The resulting formula is:

$$\begin{aligned} \mathrm{ERR} &= \sum_{i=1}^{n} \frac{1}{i} P(\text{user stops at } i) \\ &= \sum_{i=1}^{n} \frac{1}{i} R(y_i) \prod_{j=1}^{i-1} (1 - R(y_j)) \quad \text{with} \quad R(y) := \frac{2^y - 1}{16} \end{aligned} \tag{2}$$

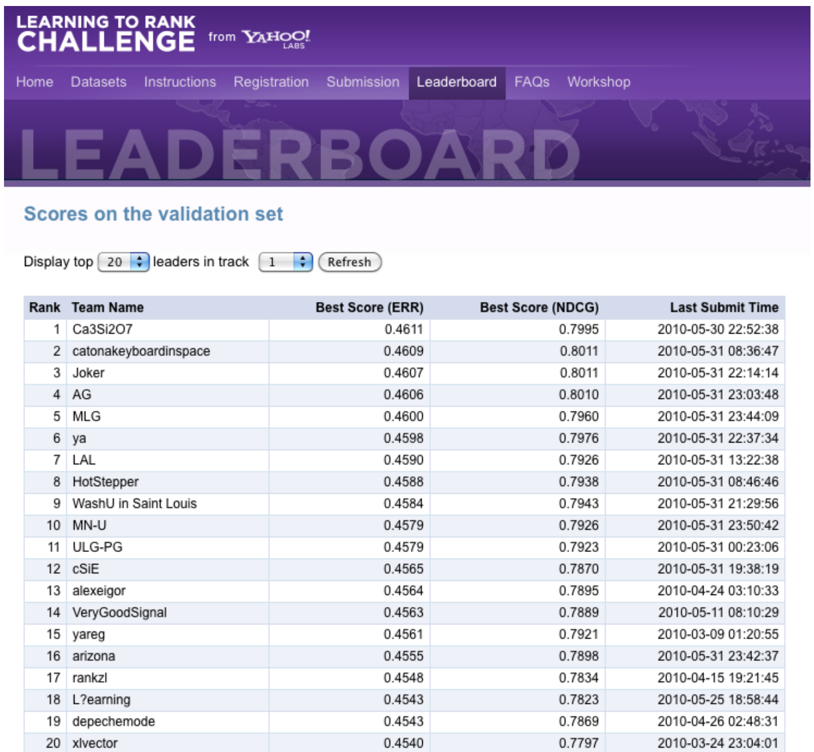
The ERR metric is very similar to the *pFound* metric used by Yandex (Segalovich, 2010). In fact pFound is identical to the ERR variant described in (Chapelle et al., 2009, Section 7.2).

Both metrics – NCDG and ERR – are defined at the query level. The score of a submission according to a metric is the arithmetic mean of this metric over the queries in the corresponding set. The NDCG scores were only provided for informational purposes: in order to determine the winners, the submissions were ranked according to their ERR score.

Algorithm 2 The cascade user model

Require: $R_1, \ldots, R_{10}$ the *relevance* of the 10 documents on the result page.

1: $i = 1$
2: User examines position i.
3: **if** random(0,1) $\leq R_i$ **then**
4:     User is satisfied with the i-th document and stops.
5: **else**
6:     $i \leftarrow i + 1$; go to 2
7: **end if**



Rank	Team Name	Best Score (ERR)	Best Score (NDCG)	Last Submit Time
1	Ca3Si2O7	0.4611	0.7995	2010-05-30 22:52:38
2	catonakeyboardinspace	0.4609	0.8011	2010-05-31 08:36:47
3	Joker	0.4607	0.8011	2010-05-31 22:14:14
4	AG	0.4606	0.8010	2010-05-31 23:03:48
5	MLG	0.4600	0.7960	2010-05-31 23:44:09
6	ya	0.4598	0.7976	2010-05-31 22:37:34
7	LAL	0.4590	0.7926	2010-05-31 13:22:38
8	HotStepper	0.4588	0.7938	2010-05-31 08:46:46
9	WashU in Saint Louis	0.4584	0.7943	2010-05-31 21:29:56
10	MN-U	0.4579	0.7926	2010-05-31 23:50:42
11	ULG-PG	0.4579	0.7923	2010-05-31 00:23:06
12	cSiE	0.4565	0.7870	2010-05-31 19:38:19
13	alexeigor	0.4564	0.7895	2010-04-24 03:10:33
14	VeryGoodSignal	0.4563	0.7889	2010-05-11 08:10:29
15	yareg	0.4561	0.7921	2010-03-09 01:20:55
16	arizona	0.4555	0.7898	2010-05-31 23:42:37
17	rankzl	0.4548	0.7834	2010-04-15 19:21:45
18	L?earning	0.4543	0.7823	2010-05-25 18:58:44
19	depechemode	0.4543	0.7869	2010-04-26 02:48:31
20	xlvector	0.4540	0.7797	2010-03-24 23:04:01

Figure 3: Leaderboard at the end of the challenge

5.1.2. Submissions

Competitors were required to submit the predicted ranks of the documents on the validation and test sets. They were getting immediate feedback – ERR and NDCG scores – on the validation set. A leaderboard was showing up to the top 100 teams. Figure 3 is a picture of the leaderboard on track 1 at the end of the challenge. Getting feedback on a validation set has now become standard in machine learning challenges: it enables competitors to gauge how well they are doing and it makes the challenge more fun and interactive. The results on the test set were not disclosed until the end of the challenge and were used to rank the winners.

Table 6: Participation: number of teams as a function of their number of submissions.

All	1055
Submissions ≥ 1	383
Submissions ≥ 2	307
Submissions ≥ 10	104

Participants were allowed to upload multiple submissions over the course of the challenge, but not more than one submission every 8 hours. This was to prevent server overload as well as guessing of the validation set labels. However, only one submission – the so-called *primary submission* – counted for determining the winners of the challenge. At any time during the challenge, the competitors had the opportunity to select, among all their submissions, the one they wished to have judged as their primary submission.

5.1.3. Prizes

At the end of the challenge, the primary entries from all participants were ranked in decreasing order of their respective ERR scores. The top four competitors of each track received the following cash prizes, summing up to $30,000.

Place	Prize
1st	$8,000
2nd	$4,000
3rd	$2,000
4th	$1,000

5.2. Participation

Competitors could enter the challenge as an individual or as part of a team, but they were not allowed to be part of multiples teams. This was to ensure fairness in the challenge and prevent a competitor from increasing his chance of winning by creating multiple teams. However some people tried to bypass this rule by creating different accounts – up to 200 – and registered a team under each account. We thus manually clean the challenge database and did our best to delete these fraudulent competitors. The statistics presented below are after this initial cleaning step.

There were 1294 participants and 1055 teams registered. This implies that most teams were single participants, but the largest team was fairly large with 7 individuals. The degree of involvement in the challenge differed greatly between teams. Table 6 lists the number of teams as a function their number of submissions. The numbers indicate that 64% of the teams registered in the challenge did not submit any entry. These people were probably merely interested in obtaining the datasets. The datasets were indeed only available to registered competitors. But as indicated at the end of section 4, the datasets are now publicly available.

Track 1 turned out to be more popular than track 2: there were 3711 submissions in track 1 coming from 363 teams, while there were only 1025 submissions from 121 teams

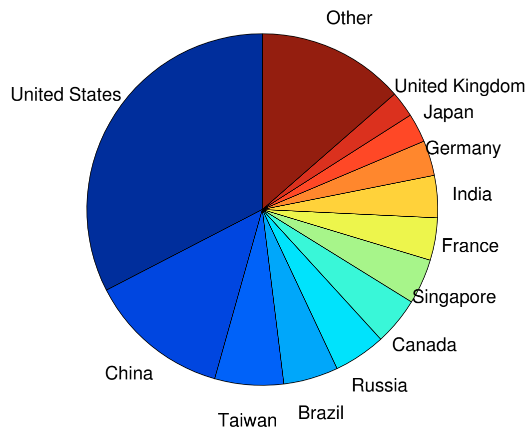


Figure 4: Breakdown of the number of page views by country of origin.

in track 2. This is probably because track 2 required more specialized algorithms and in contrast track 1 was more accessible.

We were also interested in quantifying the diversity in the geographic origin of the participants. We did not have directly this information for the competitors, but we estimated it by analyzing the traffic of the website. During the 3 months that the challenge lasted, there were 113,000 page views from 97 countries, and the distribution across countries is plotted in figure 4. We were pleased to see that the challenge drew interest all around the world.

When registering, there was an optional *affiliation* field. Based on this field and the email address of the competitors, we tried to estimate the proportion of academic and industrial competitors. For 55% of the competitors, we were not able to determine their affiliation – because the affiliation field is empty and the domain of the email address is uninformative – but among other competitors, 78% had an academic affiliation while 22% worked in industry. It is not too surprising to have a large majority of participants from academia because industrial researchers have typically less time to devote to challenges. However 22% is a substantial proportion and it shows that ranking is a relevant problem in the industrial world.

Finally, figure 5 shows the number of submissions received each day of the challenge. This number was stable around 50 submissions during most of the challenge, but not unexpectedly, the number of submissions soared in the last couple of days.

5.3. Analysis of the submissions

Metrics Since we reported two metrics in this challenge, a natural question is to know to what extent these two metrics correlate. As can be seen in figure 6, there is a relatively high correlation between them: the Kendall τ correlation coefficient, computed across the track 1 submissions, is 0.89. However, when zooming on the best submissions (right hand side of

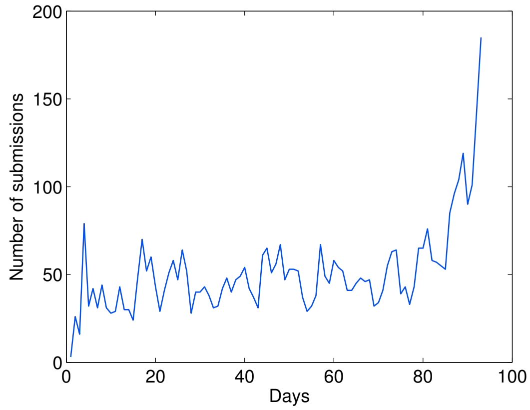


Figure 5: Number of submissions per day (both tracks).

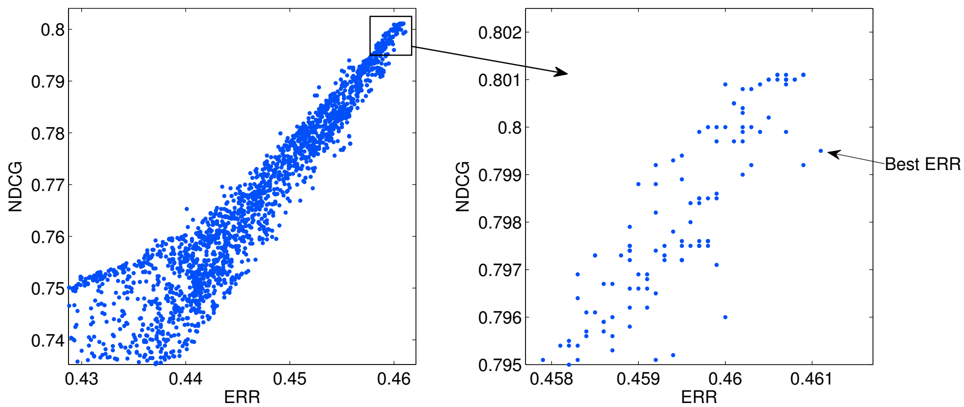


Figure 6: Scatter plot of the the NDCG and ERR scores on the validation set of track 1.

the figure), it appears that there are some substantial differences between both metrics; in particular the best submissions according to NDCG and ERR are not the same.

Distribution of scores The distribution of ERR scores on both tracks is shown in figure 7. Since the data was given in the SVM-Light format, it is not too surprising that most competitors tried the ranking version of that software at first. This correspond to the the "Linear RankSVM" bump in the figure. Note that these distributions have in fact quite a heavy tail on the left (not shown), corresponding to random or erroneous submissions.

Evolution of the scores The evolution of the test score of the best competitors is shown in figure 8. The winner team on track 1 (Ca3Si2O7) entered the challenge rather late, but had a strong finish and was able to secure the first place couple of days before the end of the challenge. Most teams did not overfit during the challenge, meaning that a better score on the validation set often meant a better score on the test set. The only exception seems to be the team WashU in track 2 which had a drop on the test set at around the 83rd day.

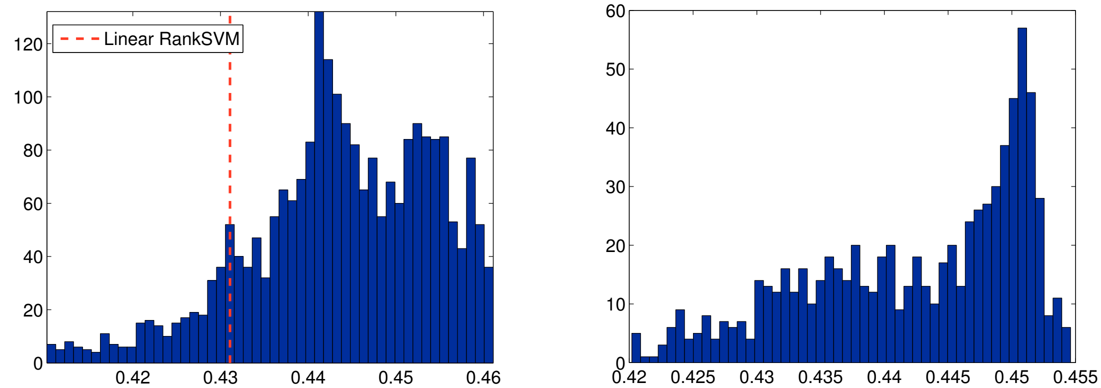


Figure 7: Distribution of the ERR scores on the validation sets of track 1 (left) and track 2 (right).

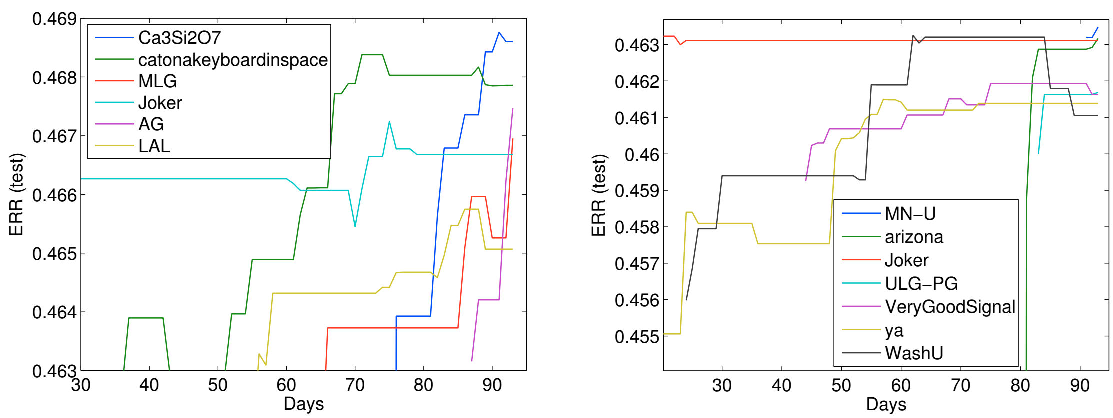


Figure 8: Evolution of the test score of the best submission (according to the validation score) on track 1 (left) and track 2 (right).

Table 7: Winners of the challenge along with the ERR score of their primary submission.

Track 1

1	C. Burges, K. Svore, O. Dekel, Q. Wu, P. Bennett, A. Pastusiak and J. Platt (Microsoft Research)	0.46861
2	E. Gottschalk (Activision Blizzard) and D. Vogel (Data Mining Solutions)	0.46786
3	M. Parakhin (Microsoft) – *Prize declined*	0.46695
4	D. Pavlov and C. Brunk (Yandex Labs)	0.46678
5	D. Sorokina (Yandex Labs)	0.46616

Track 2

1	I. Kuralenok (Yandex)	0.46348
2	P. Li (Cornell University)	0.46317
3	D. Pavlov and C. Brunk (Yandex Labs)	0.46311
4	P. Geurts (University of Liège)	0.46169

6. Outcome

6.1. Winners

As explained above, we ranked each team based on the ERR score of their primary submission on the test set. The winners are listed in table 7. A condition for winning the prize was to prepare a presentation describing the method used for the challenge. The winners were then invited to give this presentation at an ICML workshop in Haifa on June 23rd, 2010. The 3rd competitor in track 1 declined the prize and as a result the 4th and 5th teams were promoted of one rank.

The profile of the winners is quite diverse: several of them work for search engine companies (Microsoft and Yandex); P. Li and P. Guerts are academic researchers; E. Gottschalk and D. Vogel are actively participating in machine learning and data mining challenges such as the KDD cups. It is worth noting that the leader of the winning team of track 1, Chris Burges, is the author of the first web search learning to rank paper (Burges et al., 2005).

The ERR scores in table 7 of the top competitors are very close. We thus performed a paired t-test between the primary submissions of each of the 5 top competitors. The resulting p-values are listed in table 8. It can be seen that most of the differences are not statistically significant. This is even more pronounced on track 2 where the test set was smaller than the one of track 1. The closeness of the scores does not necessarily imply that the predictions are similar. In fact an oracle ensemble method, choosing for each query the best ranking among the top 10 primary submissions results, would have an ERR score of 0.4972 in track 1. This number is substantially higher than those listed in table 8 and indicates that the predictions from the competitors are different enough and that there is probably still some room for improvement.

Table 8: p-values from a paired t-test between the top 5 primary submissions on track 1 (left) and track 2 (right)

	Pos 2	Pos 3	Pos 4	Pos 5
Pos 1	0.262	0.007	0.007	0.0002
Pos 2		0.221	0.123	0.019
Pos 3			0.802	0.254
Pos 4				0.325

	Pos 2	Pos 3	Pos 4	Pos 5
Pos 1	0.712	0.659	0.069	0.045
Pos 2		0.941	0.111	0.075
Pos 3			0.057	0.084
Pos 4				0.956

6.2. Methods used

The similarity between the methods used by the winners is striking: all of them used decision trees and ensemble methods.

Burges et al. (2011) used a linear combination of 12 ranking models, 8 of which were LambdaMART (Burges, 2010) boosted tree models, 2 of which were LambdaRank neural nets, and 2 of which were logistic regression models. While LambdaRank was originally instantiated using neural nets, LambdaMART implements the same ideas using the boosted-tree style MART algorithm, which itself may be viewed as a gradient descent algorithm. Four of the LambdaMART rankers (and one of the nets) were trained using the ERR measure, and four (and the other net) were trained using NDCG. Extended training sets were also generated by randomly deleting feature vectors for each query. Various approaches were explored to linearly combine the 12 rankers, but simply adding the normalized model scores worked as well as the other approaches.

Eric Gottschalk and David Vogel first processed the datasets to create new normalized features. The original and derived features were then used as inputs into a random forest procedure. Multiple random forests were then created with different parameters used in training process. The out-of-bag estimates from the random forests were then used in a linear regression to ensemble the forests together. For the final submission, this ensemble was blended with a gradient boosting machine trained on a transformed version of the dependent variable.

Dmitry Pavlov and Cliff Brunk tested a machine learning approach for regression based on the idea of combining bagging and boosting called BagBoo (Gorodilov et al., 2010). The model borrows its high accuracy potential from Friedman's gradient boosting, and high efficiency and scalability through parallelism from Breiman's bagging. It often achieves better accuracies than bagging or boosting alone. For the transfer learning track, they combined the datasets in a way that puts 7 times higher weight on set 2.

Daria Sorokina also used the idea of combining bagging and boosting in an algorithm called Additive Groves (Sorokina et al., 2007).

Igor Kuralenok proposed a novel pairwise method called YetiRank (Gulin et al., 2011) that modifies Friedman's gradient boosting method in the gradient computation part. It also takes uncertainty in human judgements into account.

Ping Li recently proposed Robust LogitBoost (Li, 2010) to provide a numerically stable implementation of the highly influential LogitBoost algorithm (Friedman et al., 2000), for classifications. Unlike the widely-used MART algorithm, (robust) LogitBoost use both the first and second-order derivatives of the loss function in the tree-splitting criterion. The five-level ranking problem was viewed as a set of four binary classification problems. The predicted class probabilities were then mapped to a relevance score as in (Li et al., 2008). For transfer learning, classifiers were learned on each set and a linear combination of the class probabilities from both sets was used.

Geurts and Louppe (2011) experimented with several tree-based ensemble methods, including bagging, random forests, and extremely randomized trees (Geurts et al., 2006), several (pointwise) classification and regression-based coding of the relevance label, and several ranking aggregation schemes. The best result on the first track was obtained with the extremely randomized trees in a standard regression setting. On the second transfer learning track, the best entry was obtained using extremely randomized regression trees built only on the SET 2 data. While several attempts at combining both sets were somewhat successful when cross-validated on the training set, the improvements were slight and actually not confirmed on the validation set.

Finally the proceedings of the challenge include contributions from two teams that were not among the winning teams, but still performed very well. Busa-Fekete et al. (2011) used decision trees within a multi-class version of AdaBoost while Mohan et al. (2011) tried various combinations of boosted decision trees and random forests for the transfer learning track.

6.3. Lessons

The Yahoo! Learning to Rank Challenge was overall very successful, with much higher participation than anticipated. We designed the rules and organized the challenge by taking into account the experience gained from previous machine learning challenges. As for learning to rank, there are two main lessons we learned from this challenge.

First, we used to believe that more advanced ranking algorithms could largely improve the relevance. However, the results clearly demonstrate that the solutions to ranking problem are quite mature. Comparing the best solution of ensemble learning with the baseline of regression model (GBDT), the relevance gap is rather small. The good performance of simple regression based techniques appears to be at odds with most publications on learning to rank. There are two possible explanations for this. One of them is that some of the “improvements” reported in papers are due to chance. A recent paper (Blanco and Zaragoza, 2011) analyzes this kind of random discoveries on small datasets. The other explanation has to do with the class of functions. In general, the choice of the loss function is all the more critical as the class of function is small, resulting in underfitting; figure 9 illustrates that point in classification. But when the class of functions is sufficiently large and underfitting is not an issue anymore, the choice of the loss function is of secondary importance. Most learning to rank papers consider a linear function space for the sake of simplicity. This space of functions is probably too limited and the above reasoning explains that substantial gains can be obtained by designing a loss function specifically tuned for ranking. But with ensemble of decision trees, the modeling complexity is large enough and squared loss optimization is sufficient. A theoretical analysis of the link between squared loss and DCG can be found in (Cossock and Zhang, 2008)

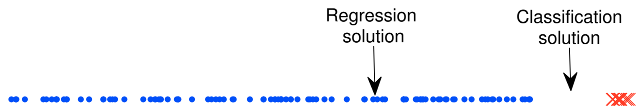


Figure 9: One dimensional example illustrating the effect of the loss function. The goal is to classify the red crosses against the blue dots. The solution of linear regression (with targets ± 1) is bad, but changing the squared to loss to any classification loss yields the desired solution. Alternatively, non-linear regression also solves the problem; here is a decision stump suffices.

A second lesson from this challenge is that the benefits from transfer learning seem limited. None of the competitors were able to clearly outperform the baseline consisting in learning from the local data only. In order to provide a good benchmark data for transfer learning research, we should either release another dataset which has more similarities with the US data, or reduce the total amount of local training data.

7. Conclusion

Research on learning to rank is heavily dependent on a reliable benchmark dataset. We believe that the datasets released for the Yahoo! Learning to Rank Challenge help the research community to develop and evaluate state-of-the-art ranking algorithms in a reliable and realistic way.

The results of the challenge clearly showed that nonlinear models such as trees and ensemble learning methods are powerful techniques. It was also surprising to notice that the relevance difference among the top winners is very small, suggesting that the existing solutions to ranking problem are quite mature and that the research on learning to rank should now go beyond the traditional setting that this challenge considered. This is the reason why we suggest, in the afterword of these proceedings, some future research directions for learning to rank (Chapelle et al., 2011a).

Acknowledgments

We are very grateful to Jamie Lockwood, Kim Capps-Tanaka and Alex Harte for their help in organizing and launching this challenge; and to George Levchenko and Prasenjit Sarkar for the web site design and engineering. Finally special thanks to Isabelle Guyon for her valuable feedback on the organization and rules of the challenge.

References

E. Agichtein, Er. Brill, and S. Dumais. Improving web search ranking by incorporating user behavior information. In *SIGIR '06: Proceedings of the 29th annual international conference on Research and development in information retrieval*, pages 19–26, 2006.

R. Blanco and H. Zaragoza. Beware of relatively large but meaningless improvements. Technical Report YL-2011-001, Yahoo! Research, 2011.

A. Broder, M. Fontoura, V. Josifovski, and L. Riedel. A semantic approach to contextual advertising. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, 2007.

A. Broder, E. Gabrilovich, V. Josifovski, G. Mavromatis, D. Metzler, and J. Wang. Exploiting site-level information to improve web search. In *CIKM '10: Proceedings of the 19th ACM Conference on Information and Knowledge Management:Proceedings of the ACM Conference on Information and Knowledge Management*, 2010.

C. Burges. From RankNet to LambdaRank to LambdaMART: An overview. Technical Report MSR-TR-2010-82, Microsoft Research, 2010.

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*, 2005.

C. Burges, K. Svore, P. Bennett, A. Pastusiak, and Q. Wu. Learning to rank using an ensemble of lambda-gradient models. *JMLR Workshop and Conference Proceedings*, 14: 25–35, 2011.

C. J. Burges, Quoc V. Le, and R. Ragno. Learning to rank with nonsmooth cost functions. In Schölkopf, J. Platt, and T. Hofmann, editors, *Advances in Neural Information Processing Systems 19*, 2007.

R. Busa-Fekete, B. Kégl, T. Éltető, and G. Szarvas. Ranking by calibrated AdaBoost. *JMLR Workshop and Conference Proceedings*, 14:37–48, 2011.

Y. Cao, J. Xu, T. Y. Liu, H. Li, Y. Huang, and H. W. Hon. Adapting ranking SVM to document retrieval. In *SIGIR*, 2006.

Z. Cao, T. Qin, T-Y. Liu, M-F. Tsai, and H. Li. Learning to rank: from pairwise approach to listwise approach. In *International Conference on Machine Learning*, 2007.

S. Chakrabarti, R. Khanna, U. Sawant, and C. Bhattacharyya. Structured learning for non-smooth ranking losses. In *International Conference on Knowledge Discovery and Data Mining (KDD)*, 2008.

O. Chapelle and S. S. Keerthi. Efficient algorithms for ranking with SVMs. *Information Retrieval*, 13(3):201–215, 2010.

O. Chapelle, Q. Le, and A. Smola. Large margin optimization of ranking measures. In *NIPS workshop on Machine Learning for Web Search*, 2007.

O. Chapelle, D. Metlzer, Y. Zhang, and P. Grinspan. Expected reciprocal rank for graded relevance. In *CIKM '09: Proceedings of the 18th ACM Conference on Information and Knowledge Management*, 2009.

O. Chapelle, Y. Chang, and T.-Y. Liu. Future directions in learning to rank. *JMLR Workshop and Conference Proceedings*, 14:91–100, 2011a.

O. Chapelle, P. Shivaswamy, S. Vadrevu, K. Weinberger, Y. Zhang, and B. Tseng. Boosted multi-task learning. *Machine Learning Journal*, 2011b. To appear.

D. Chen, Y. Xiong, J. Yan, G.-R. Xue, G. Wang, and Z. Chen. Knowledge transfer for cross domain learning to rank. *Information Retrieval*, 13(3):236–253, 2009.

D. Cossock and T. Zhang. Statistical analysis of bayes optimal subset ranking. *IEEE Transactions on Information Theory*, 54(11):5140–5154, 2008.

N. Craswell, O. Zoeter, M. Taylor, and B. Ramsey. An experimental comparison of click position-bias models. In *WSDM '08: Proceedings of the international conference on Web search and web data mining*, pages 87–94. ACM, 2008.

A. Dong, Y. Chang, Z. Zheng, G. Mishne, J. Bai, R. Zhang, K. Buchner, C. Liao, and F. Diaz. Towards recency ranking in web search. In *WSDM '10: Proceedings of the third ACM international conference on Web search and data mining*, pages 11–20, 2010.

Y. Freund, R. Iyer, R. E. Schapire, and Y. Singer. An efficient boosting algorithm for combining preferences. *Journal of Machine Learning Research*, 4:933–969, 2003.

J. Friedman. Stochastic gradient boosting. *Computational Statistics and Data Analysis*, 38 (4):367–378, 2002.

J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting. *Annals of Statistics*, 28(2):337–407, 2000.

J. Gao, Q. Wu, C. Burges, K. Svore, Y. Su, N. Khan, S. Shah, and H. Zhou. Model adaptation via model interpolation and boosting for web search ranking. In *EMNLP*, pages 505–513, 2009.

P. Geurts and G. Louppe. Learning to rank with extremely randomized trees. *JMLR Workshop and Conference Proceedings*, 14:49–61, 2011.

P. Geurts, D. Ernst, and L. Wehenkel. Extremely randomized trees. *Machine Learning*, 36 (1):3–42, 2006.

A. Gorodilov, C. Brunk, and D. Pavlov. BagBoo: A scalable hybrid model based on bagging and boosting. In *CIKM '10: Proceedings of the 19th ACM Conference on Information and Knowledge Management*, 2010.

A. Gulin, I. Kuralenok, and D. Pavlov. Winning the transfer learning track of yahoo!'s learning to rank challenge with YetiRank. *JMLR Workshop and Conference Proceedings*, 14:63–76, 2011.

Z. Gyöngyi, H. Garcia-Molina, and J. Pedersen. Combating web spam with trustrank. In *VLDB '04: Proceedings of the Thirtieth international conference on Very large data bases*, pages 576–587. VLDB Endowment, 2004. ISBN 0-12-088469-0.

D. Harman. Overview of the second text retrieval conference (TREC-2). *Information Processing & Management*, 31(3):271–289, 1995.

R. Herbrich, T. Graepel, and K. Obermayer. Large margin rank boundaries for ordinal regression. In Smola, Bartlett, Schoelkopf, and Schuurmans, editors, *Advances in Large Margin Classifiers*. MIT Press, Cambridge, MA, 2000.

K. Jarvelin and J. Kekalainen. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems*, 20(4):422–446, 2002.

T. Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD)*. ACM, 2002.

T. Joachims. Training linear SVMs in linear time. In *KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 217–226, 2006.

A. Joshi, R. Kumar, B. Reed, and A. Tomkins. Anchor-based proximity measures. In *Proceedings of the 16th international conference on World Wide Web*, page 1132. ACM, 2007.

P. Li. Robust LogitBoost and Adaptive Base Class (ABC) LogitBoost. In *Uncertainty on Artificial Intelligence*, 2010.

P. Li, C. Burges, and Q. Wu. McRank: Learning to rank using multiple classification and gradient boosting. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 897–904. MIT Press, Cambridge, MA, 2008.

T. Y. Liu. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, 3(3):225–331, 2009.

B. Long, S. Lamkhede, S. Vadrevu, Y. Zhang, and B. Tseng. A risk minimization framework for domain adaptation. In *Proceeding of the 18th ACM conference on Information and knowledge management*, pages 1347–1356. ACM, 2009.

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*, 2005.

A. Mohan, Z. Chen, and K. Weinberger. Web-search ranking with initialized gradient boosted regression trees. *JMLR Workshop and Conference Proceedings*, 14:77–89, 2011.

L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab, November 1999. URL http://ilpubs.stanford.edu:8090/422/.

T. Qin, T.Y. Liu, J. Xu, and H. Li. LETOR: A benchmark collection for research on learning to rank for information retrieval. *Information Retrieval*, 13(4):346–374, 2010.

G. Ridgeway. *Generalized Boosted Models: A guide to the gbm package*. http://cran.r-project.org/web/packages/gbm/vignettes/gbm.pdf, 2007.

S. Robertson and H. Zaragoza. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4):333–389, 2009.

S. Robertson, H. Zaragoza, and M. Taylor. Simple BM25 extension to multiple weighted fields. In *CIKM '04: Proceedings of the thirteenth ACM international conference on Information and knowledge management*, pages 42–49, 2004.

S. E. Robertson and S. Walker. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, 1994.

I. Segalovich. Machine learning in search quality at yandex. Presentation at the industry track of the 33rd Annual ACM SIGIR Conference, 2010. download.yandex.ru/company/presentation/yandex-sigir.ppt.

D. Sorokina, R. Caruana, and M. Riedewald. Additive groves of regression trees. *European Conference on Machine Learning*, pages 323–334, 2007.

M. Taylor, J. Guiver, S. Robertson, and T. Minka. SoftRank: optimizing non-smooth rank metrics. In *WSDM '08: Proceedings of the international conference on Web search and web data mining*, pages 77–86. ACM, 2008.

F. Xia, T.Y. Liu, J. Wang, W. Zhang, and H. Li. Listwise approach to learning to rank: theory and algorithm. In *Proceedings of the 25th international conference on Machine learning*, pages 1192–1199, 2008.

J. Xu and H. Li. Adarank: a boosting algorithm for information retrieval. In *International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2007.

J. Ye, J.-H. Chow, J. Chen, and Z. Zheng. Stochastic gradient boosted distributed decision trees. In *CIKM '09: Proceeding of the 18th ACM conference on Information and knowledge management*, pages 2061–2064, 2009.

Y. Yue, T. Finley, F. Radlinski, and T. Joachims. A support vector method for optimizing average precision. In *SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 271–278. ACM, 2007.

Z. Zheng, H. Zha, T. Zhang, O. Chapelle, K. Chen, and G. Sun. A general boosting method and its application to learning ranking functions for web search. In *Advances in Neural Information Processing Systems 20*, pages 1697–1704. MIT Press, 2008.