Is learning to rank effective for Web search?

Min Zhang, Da Kuang, Guichun Hua, Yiqun Liu, Shaoping Ma
State Key Laboratory of Intelligent Technology and Systems,
Tsinghua National Laboratory for Information Science and Technology,
Department of Computer Science and Technology, Tsinghua University, Bejing 100084, China.
+86-10-62792595
z-m@tsinghua.edu.cn

ABSTRACT
LETOR, the benchmark collection for learning to rank, helps make comparative study on different approaches in experimental research. Since the collection is constructed mainly based on TREC datasets, queries and documents in LETOR differ from true Web search scenario on some aspects, such as its incomplete link information, limited documents' domain, and lack of user click information. Hence the observations derived by the collection could be different from that in real Web environment. This paper empirically studies the effectiveness of the state-of-art learning to rank algorithms, especially in Web search scenario. Besides LETOR, a Web search collection is constructed based on the search log of a commercial search engine. Five approaches have been studied, including linear regression, RankBoost, ListNet, top k optimization of ListMLE, and SVM-MAP. Comparative study has been made among algorithms and across different datasets. Furthermore, the effects of learning to rank algorithms are compared with that of content-based and link-based ranking features. Essential differences have been observed and analyzed in the paper in terms of the effectiveness and stability of the algorithms and the feature selection. We believe this study will help the Web search community for better knowledge about whether and to what extent learning to rank algorithms are effective in real applications.

Categories and Subject Descriptors
H.3.3 [**Information Search and Retrieval**]: Retrieval Model

General Terms
Algorithms, Performance, Experimentation, Security, Human Factors, Standardization, Languages, Theory, Legal Aspects, Verification.

Keywords
learning to rank; Web search; LETOR; feature selection.

1. INTRODUCTION
How to integrate multiple features into the final model is an essential problem in information retrieval. Learning to rank focuses on using machine learning algorithms for better ranking, In previous work, many learning to rank algorithms have been proposed. And rich study of loss function has been made. To help researchers perform comparative study on different approaches and features, a benchmark collection LETOR has been built by Microsoft Research Aisa (MSRA), which is constructed mainly based on TREC collection. This collection has provided a good test bed for learning to rank comparative study and has been used by more and more researchers (e.g. [4][10][13][14]).

Rank integration is also an important issue in Web search. Generally a search engine could have hundreds or even thousands of features and parameters to tune. To find the optimized final ranking function is not a trivial task. Therefore learning to rank is also expected to be quite helpful in Web search scenario. TREC data is a good test bed for experimental information retrieval; however it is different from the real Web search environment in some non-ignorable aspect. First the documents collected are restricted to the .gov domain, the characteristics of which are quite different from the .com domain which is the dominant part in the Web. Second, the links in the collection are partial and the link graph built based on the data is observed to be incomplete. Hence in past TREC experiments, most link-based analysis approaches were less useful, which definitely differs from the reality in Web search. Furthermore, the queries were manually generated in 2003 and 2004, which were five or six years ago. Therefore they are most probably different from what users want today in search engines. Finally, the collection is lacking of click information, which is an essential part of user behavior analysis in current Web search application. Considering the above issues, using TREC-based LETOR datasets is not suitable for the true Web search scenario related comparative study, although it is still informative. Hence in this work, we constructed a Web search dataset as complement of LETOR for learning to rank comparative empirical study. Queries in the Web search dataset are sampled from two months' search log in a commercial search engine. The relevance judgment is annotated by traditional pooling technology, where the top 100 results given by three dominant search engines are used in the construction of the pool. Features of content, link analysis and click through information are extracted for each document with respect to each query.

The comparative study is made in this paper in several aspects. First is the empirical comparison on five classical learning to rank

SIGIR'09 workshop LR4IR'09, July 19–23, 2009, Boston, US.

[1] Supported by the Chinese National Key Foundation Research & Development Plan (2004CB318108), Natural Science Foundation (60621062, 60503064, 60736044) and National 863 High Technology Project (2006AA01Z141).

algorithms in LETOR and the Web search dataset respectively. Second is the relative performance comparison on the two different test collections. Third is the comparative study on Web search scenario, such as the impact of general features, web search and usage specific information. Essential differences have been observed and analyzed. To the best of our knowledge, it is the first study to make such kind of comparative empirical study on learning to rank algorithms for Web search, and we believe the observations will lead to some progress in adopting learning to rank research to real application.

The rest of the paper is constructed as follows. We first give a brief introduction to the related work in section 2. Section 3 describes the learning to rank algorithms we studied in this work. Then we present the empirical comparative study on section 4, including methodology, experimental settings such as the construction of the Web search dataset etc, comparative experimental results and analyses. Conclusions and the future work are finally addressed in section 5.

2. RELATED WORK

The state-of-the-art learning to rank methods can be classified into three categories: *pointwise*, *pairwise* and *listwise*. In *pointwise* methods, such as Pranking with ranking [6] and the linear regression which is also popularly used in general result integration, the relevance score is calculated based on the features of a single document with respect to a query. Such algorithm try to find the best fit for each document in terms of relevance. In *pairwise* methods, such as Ranking SVM [1], RankBoost [3][14], they take document pairs w.r.t. a query as instances and use machine learning methods to train the models on the pairs. The optimization object is to find the best document pair preferences. In *listwise* methods, such as ListNet [2], listMLE [13] and SVM-MAP [12][15], they take document lists w.r.t. a query as instances to train ranking models. The best ranked list is the final goal of learning.

In this paper, we do not prefer to propose new learning to rank algorithms, but focus on a comprehensive comparative study of the state-of-the-art approaches in Web search scenario. At least one algorithm in each category is selected according to the learning success and popularity reported in previous work. Hence linear regression for pointwise, RankBoost for pairwise, and ListNet, ListMLE and SVM-MAP for listwise approaches are studied. Since linear regression is not specific to learning to rank study and is familiar to all the researchers, redundant description is not necessary in this paper. A detailed introduction on the rest four algorithms is given in the next section.

Out of the learning to rank test collections, currently LETOR is the only one benchmark dataset, which is constructed by MSRA [7][8]. Three versions of LETOR datasets released in early 2007, late 2007 and late 2008, respectively. Detailed information can be found on its website[2]. Qin et al discussed on how to make LETOR more useful and reliable [10]. Minka et al pointed out the selection bias in the LETOR datasets, including the bias on the sampled document, etc [9].

On LETOR website, baseline results of the state-of-the-art learning to rank algorithms are given, but no insight comparative analysis has been made on the effects of the approaches. These baseline results are also helpful to verify the correctness and reliability of the implementation of the algorithms in this paper.

To the best of our knowledge, there is no comparative study on learning to rank approaches in true Web search scenario. The comparative study on learning to rank algorithms for Web search in this paper will help the information retrieval research and industry communities better understand whether and to what extent learning to rank approaches are effective in real applications.

[2] http://research.microsoft.com/en-us/um/beijing/projects/letor/

3. LEARNING TO RANK ALGORITHMS

3.1 Model settings

In this section, we describe the formal model setting for the algorithms. All the following approaches take the same setting here unless specified otherwise.

Define:

- The set of queries $Q = \{ q^{(1)}, q^{(2)}, \ldots, q^{(m)} \}$.
- $n(i)$ is the num of documents w.r.t $q^{(i)}$.
- Each query is associated with a list of documents $d^{(i)} = \{d_1^{(i)}, d_2^{(i)}, \ldots, d_{n(i)}^{(i)}\}$ w.r.t. $q^{(i)}$.
- A list of ground-truth labels $y^{(i)} = \{y_1^{(i)}, y_2^{(i)}, \ldots, y_{n(i)}^{(i)}\}$ w.r.t. $q^{(i)}$, where $y_r^{(i)}$ is the label of the document ranked at position r in the list $y^{(i)}$.
- And a list of feature vectors $x^{(i)}$ which is created based on $q^{(i)}$ and $d^{(i)}$, $x^{(i)} = \{x_1^{(i)}, x_2^{(i)}, \ldots, x_{n(i)}^{(i)}\}$.

3.2 RankBoost

Freund, Y. et al applied the boosting approach to the field of learning to rank and proposed the RankBoost algorithm [3]. The algorithm belongs to the pairwise branches. It emphasizes the relative ranking order between two documents, and the preference matrix is defined based on the document pairs. Similarly to other boosting approaches, RankBoost trains a weak ranker and update the ranking function in each round of iteration. During the update, the algorithm increases the weights of correctly ranked document pairs and decreases the weights in another case [14]. The details of the algorithm are shown below.

Algorithm 1. RankBoost:
Definition:
χ: Document set

D: Distribution over $\chi \times \chi$, e.g. if x_1 ranks higher than x_0, then $\phi(x_0, x_1) = 1$, and $\phi(x_1, x_0) = -1$.

At last $D(x_0, x_1) = Z * Max\{0, \phi(x_0, x_1)\}$ where Z is the normalization factor which ensures $\sum_{(x_0, x_1) \in \chi \times \chi} D(x_0, x_1) = 1$

Given:
1. The distribution D over $\chi \times \chi$;
2. The number of iteration T.

Initialize: $D_1 = D$.

For t = 1…T :

(1) Train weak learner using distribution D_t.

(2) Get weak ranking $h_t : \chi \to R$.

(3) Choose $\alpha_t \in R$.

(4) Update: $D_{t+1}(x_0, x_1) = \frac{D_t(x_0, x_1)\exp(\alpha_t(h_t(x_0) - h_t(x_1)))}{Z_t}$

where Z_t is a normalization factor (chosen so that D_{t+1} will be a distribution).

Output the final ranking: $H(x) = \sum_{t=1}^{T} \alpha_t h_t(x)$

3.3 ListNet

ListNet is a listwise ranking algorithm. Unlike pairwise approaches which focus on pairs of objects in learning, lists of objects are used as "instances" in listwise approach, and the optima is try to find in the level of the whole list. Cao et al. defines a formulation of permutation probability, and employs Neural Network model and the Cross Entropy loss as the listwise loss function in Gradient Descent [2]. According to previous work, ListNet outperforms most pairwise methods such as RankSVM, RankBoost, because it matches the ranking scenario and trains the model on document list directly [2]. In ListNet, *permutation probability* is defined to represent the likelihood of a permutation (ranking list) given the ranking function. And Top one probability, which equals to the sum of permutation probabilities of permutations in which the object is ranked on the top of the list, is introduced ([2]) to represent the distance (listwise loss function) between the two score lists.

We implement a linear Neural Network in experiment using top one probability. Hence the ListNet algorithm is shown as following Algorithm 2.

Algorithm 2. ListNet

Input: training data $(x^{(1)}, y^{(1)})$, $(x^{(2)}, y^{(2)})$, … , $(x^{(m)}, y^{(m)})$.

Parameter: initialized linear model ω, number of iterations T, learning rate η, and linear mapping parameter α from ground-truth labels to scores.

for t=1, …, T **do**
  for i=1, …, m **do**
    (1) Compute score of each document j with current ω:
    $$f_\omega(x_j^{(i)}) = <\omega, x_j^{(i)}>$$
    (2) Compute gradient:
    $$\Delta\omega = \frac{\sum_{i=1}^{n^{(i)}} x_j^{(i)} \exp\left(f_\omega\left(x_j^{(i)}\right)\right)}{\sum_{j=1}^{n^{(i)}} \exp\left(f_\omega\left(x_j^{(i)}\right)\right)} - \frac{\sum_{j=1}^{n^{(i)}} x_j^{(i)} \exp\left(\alpha \cdot y_j^{(i)}\right)}{\sum_{j=1}^{n^{(i)}} \exp\left(\alpha \cdot y_j^{(i)}\right)}$$
    (3) Update: $\omega = \omega - \eta \cdot \Delta\omega$
    end for
end for
Output Neural Network model ω

3.4 ListMLE-top k probability optimization

ListMLE, as a ranking algorithm, is a variation of ListNet and also belongs to listwise approach. Xia et al. re-examines the statistical properties of loss functions, and introduces a likelihood loss, with better properties in soundness and convexity, as the listwise loss function in gradient descent [13].

We improve the traditional ListMLE algorithm using Top k probability optimization. Then the improved algorithm is described as follows.

Algorithm 3. ListMLE-topk

Input: training data $(x^{(1)}, y^{(1)})$, $(x^{(2)}, y^{(2)})$, … , $(x^{(m)}, y^{(m)})$.
Parameter: initialized linear model ω, tolerance rate ε, learning rate η, and linear mapping parameter α from labels to scores.
repeat
  for i=1, …, m **do**
    (1) Compute score of each document j with current ω:
    $$f_\omega(x_j^{(i)}) = <\omega, x_j^{(i)}>$$
    (2) Compute gradient:
    $$\Delta\omega = \sum_{t=1}^{k}\left(\frac{\sum_{k=t}^{n^{(i)}} x_{y(k)} \cdot \exp\left(f_\omega\left(x_{y(k)}^{(i)}\right)\right)}{\sum_{k=t}^{n^{(i)}} \exp\left(f_\omega\left(x_{y(k)}^{(i)}\right)\right)} - x_{y(t)}^{(i)}\right)$$
    (3) Update: $\omega = \omega - \eta \cdot \Delta\omega$
    end for
    Compute likelihood loss:
    $$L = -\sum_{i=1}^{m} \log \prod_{t=1}^{k} \frac{\exp\left(f_\omega\left(x_{y(t)}^{(i)}\right)\right)}{\sum_{k=t}^{n^{(i)}} \exp\left(f_\omega\left(x_{y(k)}^{(i)}\right)\right)}$$
until change of likelihood loss is below ε times the previous loss
Output Neural Network model ω
(The only difference between ListMLE and ListMLE-topk is: when calculating likelihood loss L, t is iterated from 1 to $n(i)$ in ListMLE, and here t=1 to k in ListMLE-topk.)

3.5 SVM-MAP

SVM-MAP is a structural Support Vector Machine method which optimizes a differentiable upper bound of MAP in the predicted rankings. The optimization is performed on a working set of constraints, which is a finite subset of the infinite constraints in the structural SVM. The most violated constraint is selected and added to the working set $\mathcal{W}$ iteratively, until no constraint is violated in the sense of desired precision ε. [15]

For simplicity, we omit the superscript i in the following symbols information, such as $q^{(i)}$, $d^{(i)}$, $d_r^{(i)}$ etc, w.r.t to the i^{th} query, when there is no ambiguity.

Let C^Q and $C^{\bar{Q}}$ be the set of relevant and irrevelant documents for query q. **Define:**

$$\Delta(d, \hat{d}) = 1 - MAP(rank(d), rank(\hat{d}))$$

$$\Psi(q,d) = \frac{1}{|C^Q| \cdot |C^{\bar{Q}}|} \sum_{i:d_i \in C^Q} \sum_{j:d_j \in C^{\bar{Q}}} y_{ij}(\phi(q,d_i) - \phi(q,d_j))$$

where $d_{ij} = \begin{cases} 1 & \text{if } d_i \text{ is ranked ahead of } d_j \\ 0 & \text{if } d_i \text{ and } d_j \text{ have equal rank} \\ -1 & \text{if } d_j \text{ is ranked ahead of } d_i \end{cases}$

A constraint named $\hat{d}$ is equivalent to:

$$w^T \Psi(q_i, d_i) \geq w^T \Psi(q_i, \hat{d}) + \Delta(d_i, \hat{d}) - \xi_i .$$

Then the algorithm is shown below:

Algorithm 4. SVM-MAP [15]

Input: training data $(x^{(1)}, y^{(1)})$, $(x^{(2)}, y^{(2)})$, … , $(x^{(m)}, y^{(m)})$.
Parameter: tradeoff parameter C, precision ε.
For all i=1, …, n, $\mathcal{W}_i \leftarrow \emptyset$
repeat
  for i=1, …, n **do**

(1) $H(d;w) \equiv \Delta(d_i, d) + w^T \psi(q_i, d) - w^T \psi(q_i, d_i)$

(2) Compute $\hat{d} = \arg\max_{d \in D} H(d;w)$

(3) Compute $\xi_i = \max\{0, \max_{w \in W_i} H(d;w)\}$

(4) **if** $H(\hat{d};w) > \xi_i + \varepsilon$ **then**

$W_i \leftarrow W_i \cup \{\hat{d}\}$

$w \leftarrow optimize \ \min_{w,\xi \geq 0} \frac{1}{2}\|w\|^2 + \frac{c}{n}\sum_{i=1}^{n} \xi_i \ over \ w = \bigcup_i W_i$

end if

end for

until no W_i has changed during iteration

Output w

4. COMPARATIVE EMPIRICAL STUDIES

4.1 Methodology

A collection comes from the web search engine scenario is used, called Web search data (shown as 'WebSE' for short). In this paper, LETOR dataset is being used for learning algorithms baseline study and for learning to rank algorithm verification. And the WebSE dataset is taken for further comparative study on the affects of learning to rank approaches in true Web search scenario. Detailed information about the two datasets is described in the following section 4.2.

In this paper, a total of five algorithms have been studied, namely *linear regression*, *RankBoost*, *ListNet, ListMLE-topk*, and *SVM-MAP*[3]. Linear regression is a kind of pointwise learning to rank methods, and is the one commonly used to combine multiple results. RankBoost is a typical pairwise approach for learning to rank. And the latter three are all listwise approaches. Besides the five ranking algorithms, the results generated by using content-based BM25 feature are also given in the two collections, and in WebSE dataset, there is one more result compared which is that of using PageRank for ranking. Since the PageRank score is computed on the partial link graph in LETOR data which makes the ranking not reliable, the comparison on PageRank in LETOR makes no sense.

The comparative study is made in this paper in several aspects. First is the empirical comparison of different algorithms on the same dataset. Second is the comparative study on Web search scenario, e.g. the impact of Web specific features.

Five-fold cross validation has been implemented in both collections. The k-fold validation in Web search dataset is similar to that in LETOR [7]. Each data set is divided into five parts equally. In each loop, three folds are selected for training, one fold for validation, and one for test. The experimental results shown in this paper are that on test set unless specified otherwise.

4.2 Experimental settings

4.2.1 LETOR benchmark dataset

LETOR is a standard benchmark for learning to rank research which has been constructed by Microsoft Research Asia [7][8]. LETOR is built mainly based on TREC collection, including queries, retrieved documents, and the relevance judgments. Furthermore, 64 features have been extracted based on the top 1000 documents by BM25 for each query.

In this experiment, LETOR 3.0 is adopted, which contains seven datasets, namely named page finding (NP) 2003 and 2004, home page finding (HP) 2003 and 2004, topic distillation (TD) 2003 and 2004, and OHSUMED. Since OHSUMED is the subset of MEDLINE collection and has little correlation with Web search, only the former six datasets have been used in our experiments, and we keep the name of the benchmark collection as LETOR in this paper for simplicity.

4.2.2 Web search dataset

The queries in the Web Search dataset are sampled from search logs by a commercial Chinese search engine. The search log is a faction of the log from January to February, 2009, which contains totally 108,945,644 sessions and 15,585,010 unique queries.

Finally there are totally 614 queries and the corresponding relevance judgment results, covering hot queries (queries with extremely high frequency), common queries (queries with high frequency) and rare queries (queries with low frequency). The principle of constructing query sets is to find a balance between the expressiveness of Web search scenario and the usability of the query for information retrieval research.

Following Figure 1 shows the comparison on statistics of the sample queries for WebSE data and all the queries in log data in terms of unique queries number. Figure 2 gives the logarithmic queries' frequency distribution of the WebSE dataset.

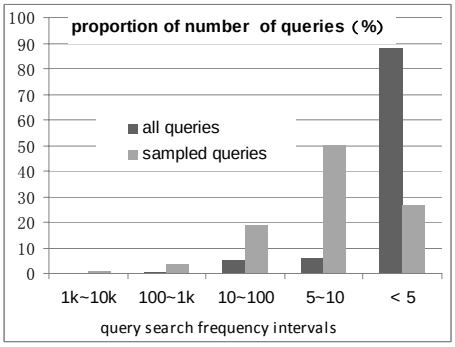

Figure 1 Comparison on the distribution of the number of unique queries (sampled v.s. all queries)

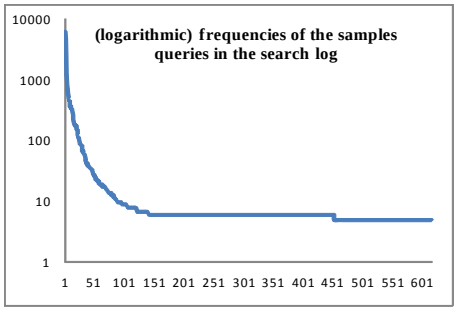

Figure 2 Frequency distribution of the queries in the WebSE

Shown by the comparison on unique queries proportion in Figure 1, the sampled queries for WebSE dataset make a good representation of the original whole query sets, except for the

[3] In our experiments, the online SVM-MAP package is used, which can be found at http://projects.yisongyue.com/svmmap/ .

queries which frequencies are less than 5. Although the number of rare queries (whose frequency is less than 5 in the search log in this paper) is large, complete match of the original query proportion will lead to a strong bias that the rare queries will be over emphasized. Hence less rare queries have been selected in the WebSE dataset. According to figure 2, the frequency distribution of the constructed WebSE queries well fits the knowledge that the volume distribution of Web search queries follows the power law [11].

The results relevance judgment is manually annotated by three skilled laboratories based on the classical pooling techniques in information retrieval evaluation. The candidates in the pool are collected by the three most popular commercial search engines in China with the top 100 returned results lists. And the four-level relevance score have been assigned to each document.

The documents sampling strategy in WebSE dataset is to select the top 200 results according to a simple linear combination of content-based BM25 ranks and PageRank ranks. The introducing of PageRank in the documental samples is based on the Web search application background, in which the link-based ranking plays an important role.

The features extracted from the sampled documents for each query are in two forms. For global features, such as IDF, PageRank etc, the background document collections are used which is the complete crawling result of Chinese Web pages in the Web. The computation of the global features is done with the help of a Chinese commercial search engine. For local features, such as TF, document length etc, is calculated directly from the sampled documents. Furthermore, features by user click information are also implemented in the WebSE dataset. By doing so, the dataset is constructed to simulate the true Web search scenario with the most effort.

4.2.3 Evaluation methods

Three metrics are used in this work to evaluation the performance, i.e. p@n, MAP and NDCG@n, which are widely used in information retrieval research and application.

p@n is the precision at top *n* returned results, which is defined as: $p@n = \frac{1}{n}\sum_{i=1}^{n} rel(d_i)$, where $rel(d_i)$ is binary, which is set to 1 when d_i is relevant to the query.

MAP is the mean average precision for the queries. It is a comprehensive measure which takes both precision and recall into consideration. The definition can be shown as:

$$\text{MAP} = \frac{1}{m}\sum_{j=1}^{m} AP_j = \frac{1}{m}\sum_{j=1}^{m}\frac{1}{R_j}(\sum_{i=1}^{k} rel_j(d_i)\cdot(p@i)_j)\text{,}$$

where m is the total number of queries, R_j is the total number of relevant documents for the jth query, k is the total number of returned documents for the query, $rel_j(d_i)$ and $(p@i)_j$ are the $rel(d_i)$ and $p@i$ scores for the jth query respectively.

NDCG@n is a quite useful measure for evaluate web search and related tasks. It takes into account the position (rank) of the document with different relevance degree in the returned result list, The *NDCG* score at top *n* returned results is defined as [16]:

$$NDCG\ @n = \frac{1}{Z_n} DCG\ @n = \frac{1}{Z_n}\sum_{i=1}^{n}\frac{2^{rel(d_i)}-1}{\log(1+i)}\text{,}$$

where Z_n is the DCG@n score of ideal result ranking list.

4.3 Comparative performance of algorithms on LETOR data

Figure 3 (a) and (b) show the performance comparison on LETOR collection NP/HP tasks and TD tasks respectively. Except for the five algorithms studied in the paper, the result of using content-based BM25 ranking function is also shown in the figure to give more information on the relative effectiveness.

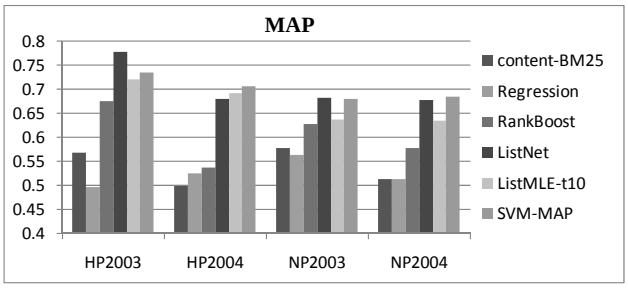

Figure 3 (a) MAP comparison on HP and NP tasks

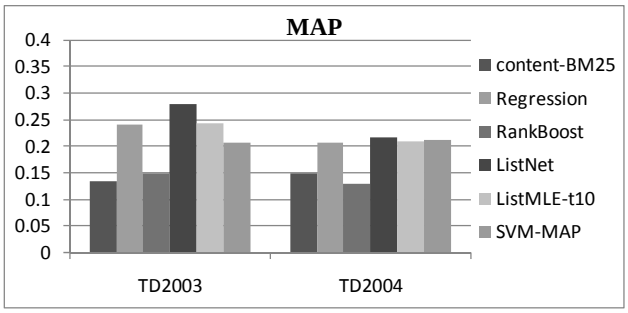

Figure 3 (b) MAP comparison on TD tasks

It is clear from the figure 3 that generally speaking, linear regression does not make improvement compared with content-based retrieval result on HP/NP tasks, but gets better performance on TD tasks. ListNet leads to the best performance in most cases. Next comes the SVM-MAP in terms of performance and stability. ListMLE-topk is not stable. The performances vary from HP/NP tasks and TD tasks, and even changes with the same task in different years. RankBoost are not effective compared with other learning to rank methods. It is only a slightly better than content-based approach and linear regression, if not worse. Hence the listwise algorithms outperform pairwise and pointwise approaches when evaluated with MAP measure.

Since the comparative performance among different approaches on HP2003, HP2004, NP2003, NP2004 are generally consistent, as well as those on TD tasks in 2003 and 2004, we only show results on HP2004 and TD2004 in the following. If changes are observed, more results will be given.

The following Figure 4 shows the performances on HP2004 and TD2004. Looking at homepage finding task, the algorithms integrate into two groups under p@n measure. The first group is composed of SVM-MAP, ListMLE-topk and ListNet; and another includes regression and RankBoost, which get nearly the same performance with content-based BM25. The p@n results on TD2004 are slightly different, where ListMLE-topk gets best

result. SVM-MAP also shows encouraging results which is consistent with the results under MAP. One thing worth of mention is that the p@1 results are chaotic because the metric is too sensitive for informational queries. Hence the performance under p@1 is less reliable.

Figure 5 shows the performances of *NDCG*@n on HP2004 and TD2004 tasks. According to results on HP2004 dataset, ListMLE-topk are the most effective algorithms in terms of NDCG@n. SVM-MAP and ListNet are similar on the encouraging performance. Again Rankboost and linear regression get almost the same results. In TD2004 dataset, ListMLE-topk is no longer the most effective, while linear regression takes the best results instead. RankBoost performs still closely to content-based BM25 ranking, which is not useful for integrating ranks and features.

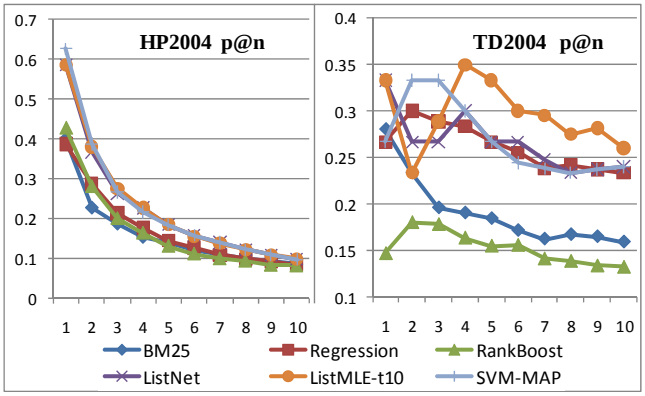

Figure 4 Comparison with p@n on LETOR datasets

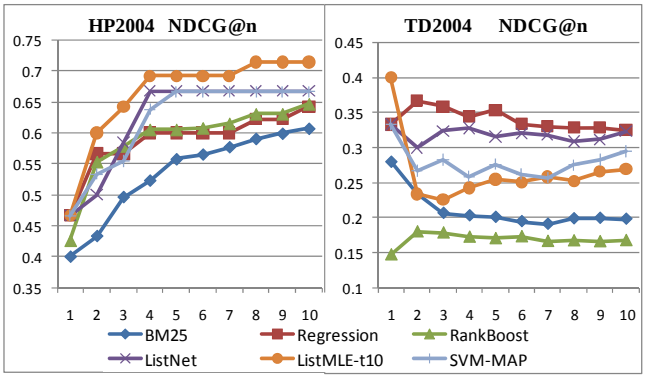

Figure 5 Comparison with NDCG@n on LETOR datasets

Comparing the results under MAP, p@n and NDCG@n measures on LETOR datasets, several observations are drawn:

(1) SVM-MAP is a stable and effective approach in every measure and every task, which is shown to be helpful in learning to rank tasks. (2) ListNet comes next and is also stable in different tasks and measures. (3) ListMLE-topk is good at home/named page finding tasks which represent navigational search goals, but are unstable at topic distillation (TD) task which corresponds to a kind of informational search goals. (4) Generally speaking, RankBoost is not effective, which is only slightly better than content-based BM25 rank in most cases, if not worse. (5) Linear regression does not make improvement compared with content-based retrieval result on home page and named page finding tasks, but get much better performance on Topic distillation tasks.

4.4 Comparative performance of algorithms on Web search data

In comparative study on WebSE dataset, the result based on PageRank is introduced, because such kind of link-based analysis is generally helpful in Web search scenario. As introduced in the section 4.2.2, the top 200 results for each query are sampled according to a simple linear combination of content-based BM25 ranks and PageRank ranks. Based on this document set for each query, two baseline results are given, which are ranked by BM25 scores and PageRank scores respectively. Hence there are totally 7 ranking results to study under MAP, p@n and NDCG@n metrics, i.e. BM25, PageRank, regression, RankBoost, ListNet, ListMLE-topk, and SVM-MAP.

Table 1 shows comparative results on WebSE dataset. The first notable thing is the effect of PageRank, which is much better than the result of content-based BM25 ranking that 59.10% improvement over BM25 result is achieved. The observation is totally different from that on LETOR 3.0 data in which PageRank gives trivial impact and the performances, in which the performance on all the three measures are less than 1/3 of the performance with BM25 ranking results. It verifies the specifications in former sections that LETOR collection differs from the real Web search environment.

On the effects of learning to rank algorithms, ListMLE-topk gets the best result as well as SVM-MAP, both of which achieve more than 86%, 17% and 15% improvement over results based on BM25, PageRank and regression, respectively. ListNet is also effective which is only slightly worse than the two best algorithms. Although the improvement is not as great as the former mentioned algorithms, RankBoost gets better performance than regression.

Table 1 MAP comparison of different algorithms.

BM: BM25, PR: PageRank, RG: linear regression, RB: RankBoost, LN: ListNet, LM: ListMLE-topk, SM: SVM-MAP

	MAP	v.s. BM (+)	v.s. PR (+)	v.s. RG (+)
BM	0.2702	--	--	--
PR	0.4298	59.10%	--	--
RG	0.4354	61.16%	1.29%	--
RB	0.4629	71.33%	7.68%	6.31%
LN	*0.4894*	*81.13%*	*13.84%*	*12.39%*
LM	***0.5052***	***86.99%***	***17.53%***	***16.03%***
SM	***0.5029***	***86.15%***	***17.00%***	***15.50%***

Figure 6 and Figure 7 gives the performance comparison in terms of p@n and NDCG@n on WebSE dataset respectively. The results are consistent with that have been observed with MAP metric. The best performance is achieved by ListMLE-topk and SVM-MAP. Then follows ListNet. And RankBoost is the least effective but still helpful compared with linear regression and PageRank. All of these approaches are much better than the result of content-based BM25 in Web search scenario.

Now it is not difficult to sum up the empirical comparative results of learning to rank algorithms on WebSE datasets. Consistent improvements have been made in all of the five learning to rank approaches, and the relative performance ranks of different approaches are kept stable on all the metrics.

Define '>' means 'the performance is better than' and '~' means 'the performance is similar to', then the drawn is drawn that:

ListMLE-topk ~ SVM-MAP > RankBoost > linear regression ~ PageRank > BM25.

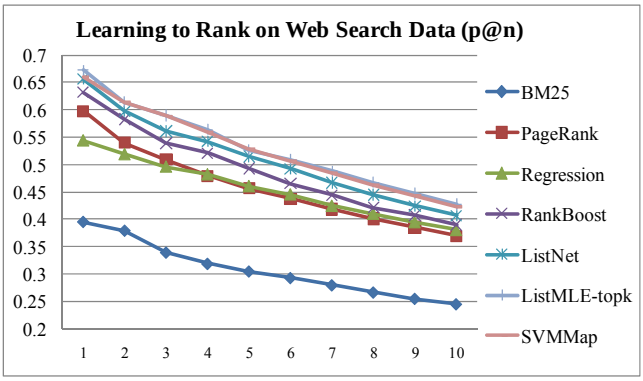

Figure 6 performance comparisons on WebSE with p@n

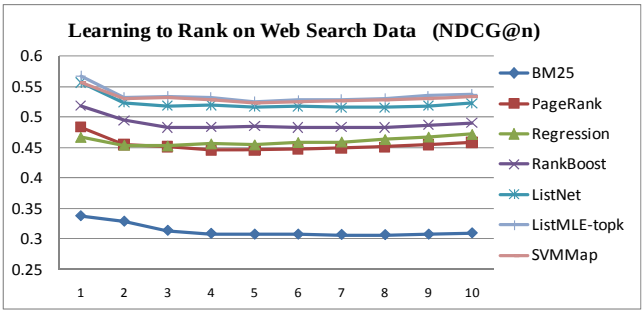

Figure 7 performance comparisons on WebSE with NDCG@n

4.5 Feature analysis

In previous sections, we analyzed that the ListMLE-topk result is not stable in LETOR datasets. In this section, we discuss this problem in terms of features.

After learning procedure is finished, features are selected with different weights which represent the role of features in getting final rank. The correlation between the final features' ranks of the two runs with ListMLE-topk algorithms on LETOR is calculated.

Given $rank_i(f_j)$ is the rank of the feature f_j on the ith fold, we define the rank of the feature f_j as: $rank(f_j)$ = average$\{rank_i(f_j)\}$. Note that $rank_i(f_j)$ is calculated based on the normalized absolute value of the feature's weight on the ith fold.

The surprising result is the average correlation is only 0.6057. It shows that the ListMLE-topk algorithm on LETOR datasets is unstable, since arbitrary two runs of the same algorithm on the same dataset with the same settings will lead to strongly different final feature ranking strategies. The only difference between the two runs is the randomly initialized linear model. Similar analysis is made on ListNet. The encouraging result is the average correlation between final features' ranking strategies by two runs is 0.9349, which shows the stability of the algorithm. Note that in ListNet, the initial linear model is also randomly set. For SVM-MAP, every run with the same settings keeps the same results.

Looking into details of the selected feature by different algorithms on LETOR datasets, it is easy to find that ListNet and SVM-MAP selected similar top 5 features, which differ from that selected by ListMLE-topk greatly. The final generated top 5 features are listed in Table 2.

Table 3 gives the final top-ranked features selected based on WebSE data collection, and the corresponding final ranks of these features selected by different algorithms on LETOR. As shown in the Table 3, the three algorithms, namely ListMLE-topk, ListNet and SVM-MAP, agree with each other on the final top-ranked features on WebSE dataset. It's also consistent with the performance comparison result in Table 1, in which PageRank is shown to be quite effective that only use it achieves similar result with linear regression. But the selected top ranked features distinctly disagree with those selected on LETOR. The main difference lies on the features based on link analysis and user behavior analysis, which cannot be generated on LETOR.

Table 2 Top 5 effective features learnt on LETOR

Rank	ListMLE-topk	ListNet	SVM-MAP
1	IDF on URL	Sitemap based score propagation	Sitemap based term propagation
2	IDF on anchor	Sitemap based term propagation	Sitemap based score propagation
3	IDF on Body	HostRank	LMIR.ABS of whole document
4	LMIR.JM of whole doc	LMIR.ABS of whole doc	Hyperlink base feature propagation: weighted in-link
5	IDF on whole doc	LMIR.JM of whole doc	LMIR.ABS of anchor

Table 3 Top effective features selected on WebSE and their ranks on LETOR with different algorithms

	WebSE, 3 alg. *	LETOR, ListMLE-topk	LETOR, ListNet	LETOR, SVMMAP
PageRank	1	49	37	45
BM25 on whole doc	2	46	11	7
BM25 on anchors	3	56	5	16
User-click based fea.	4	--	--	--

* On WebSE collection, the three algorithms agree with each other, hence only one column is shown for them.

5. Conclusions and future work

In this paper, we made an empirical comparative study on the effectiveness of the state-of-the-art learning to rank algorithms for

Web search. Since the open benchmark collection LETOR have several limitations to be representative of true Web search scenario, a complementary dataset named WebSE is constructed based on the two-month's search log of a Chinese commercial search engine. A total of five algorithms have been studied, including linear regression, the pointwise learning approach, RankBoost which represents pairwise methods, and listwise algorithms ListNet, improved ListMLE with top-k probability optimization, and SVM-MAP. Besides these algorithms, the results of BM25 and PageRank are also adopted for comparison. To the best of our knowledge, such comparative empirical study for true Web search scenario has not been made before.

Several conclusions are drawn as follows.

First, learning to rank algorithms do help in Web search scenario. (1) Consistent improvements have been made in all of the five learning to rank approaches, and the relative performance ranks of different approaches are kept stable on all the metrics;

(2) Define '>' means 'the performance is better than' and '~' means 'the performance is similar to', then the conclusion is drawn that: ListMLE-topk ~ SVM-MAP > RankBoost > linear regression ~ PageRank > BM25;

Second, on the final feature ranking strategies:

(1) ListMLE-topk algorithm on LETOR datasets is unstable, since arbitrary two runs of the same algorithm on the same dataset with the same settings lead to strongly different feature ranking strategies for generate final ranking results;

(2) ListMLE-topk shows a less agreement on feature selection with ListNet and SVM-MAP, while the latter two are better correlated;

(3) For true Web search scenario, learning algorithms agree with each other on final feature ranking. But the selected top ranked features distinctly disagree with those selected on the LETOR test bed.

Finally, for Web search study, LETOR has limitations on link analysis and user behavior features.

In the future, more issues on Web search will be studied. For example, learning to rank based on query type classification or query clustering. More research on feature selection will be made. New features and algorithms are also expected.

6. ACKNOWLEDGMENTS

The algorithms of the ListNet and ListMLE are implemented at Microsoft Research Asia (MSRA) by the second author during the Tsinghua-MSRA computer science research practice course, and many thanks Tao Qin, Tieyan Liu, and Hang Li for their kind supervision. Special thanks to Hao Yu, Zijian Tong and Liyun Ru for their great help on generating Web search collection. And we also would like to thank anonymous reviewers who give many valuable comments for the better representation of the paper.

7. REFERENCES

[1] Y. Cao, J. Xu, T. Liu, H. Li, Y. Huang, and H. Hon, "Adapting ranking SVM to document retrieval", In SIGIR'06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval, pp.186-193, 2006.

[2] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. Learning to rank: from pairwise approach to listwise approach. In ICML '07: Proceedings of the 24th international conference on Machine learning, pages 129–136, New York, NY, USA, 2007. ACM Press.

[3] Y. Freund, R. Iyer, R. E. Schapire, and Y. Singer. An efficient boosting algorithm for combining preferences. J. Mach. Learn. Res., 4:933-969, 2003.

[4] John Guiver, Edward Snelson. Learning to rank with softrank and Gaussian processes. In SIGIR'08: Proceedings of the 31th annual international ACM SIGIR conference on Research and development in information retrieval, 2008.

[5] K., Jarvelin, J., Kekalainen. Cumulated Gain-based evaluation of IR techniques, ACM Transactions on Information System, 2002, 20, 422-446.

[6] K. Crammer and Y. Singer. Pranking with ranking. In NIPS 2002: Neural Information Processing Systems, pages 641-647, 2002.

[7] T.-Y. Liu, J. Xu, T. Qin, W.-Y. Xiong, and H. Li. LETOR: Benchmark dataset for research on learning to rank for information retrieval. In SIGIR'07 Workshop on learning to rank for information retrieval, 2007.

[8] LETOR, http://research.microsoft.com/en-us/um/beijing/projects/letor/index.html.

[9] Tom Minka, Stephen Robertson. Selection bias in the letor datasets. In proceedings of SIGIR 2008 workshop on learning to rank for information retrieval, 2008.

[10] Tao Qin, Tie-Yan Liu, Jun Xu, and Hang Li. How to make letor more useful and reliable, in proceedings of SIGIR 2008 workshop on learning to rank for information retrieval. 2008

[11] A. Spink, D. Wolfram, B. Jansen, and T. Saracevic. Searching the Web: The public and their queries. Journal of the American Society for Information Science and Technology, 53(3):226-234, 2001.

[12] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun, Large Margin Methods for Structured and Interdependent Output Variables, Journal of Machine Learning Research (JMLR), 6(Sep):1453-1484, 2005.

[13] Fen Xia, Tie-Yan Liu, Jue Wang, Wensheng Zhang, and Hang Li. Listwise approach to learning to rank: theory and algorithm. In ICML '08: Proceedings of the 25th international conference on Machine learning, pages 1192–1199, New York, NY, USA, 2008. ACM Press.

[14] Jun Xu and Hang Li. AdaRank: A Boosting Algorithm for Information Retrieval, SIGIR'07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval.

[15] Y. Yue, T. Finley, F. Radlinski and T. Joachims. A Support Vector Method for Optimizing Average Precision, Proceedings of the ACM Conference on Research and Development in Information Retrieval (SIGIR), 2007

[16] K., Jarvelin, J., Kekalainen. Cumulated Gain-based evaluation of IR techniques, ACM Transactions on Information System, 2002, 20, 422-446.