Expert Systems With Applications 133 (2019) 75–85

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

FGFIREM: A feature generation framework based on information retrieval evaluation measures

Yuan Lin[a], Bo Xu[b,*], Hongfei Lin[b,*], Kan Xu[b], Ping Zhang[b]

[a] WISE Lab, School of Public Administration and Law, Dalian University of Technology, Dalian, China
[b] Faculty of Electronic Information and Electrical Engineering, Dalian University of Technology, Dalian, China

A R T I C L E I N F O

Article history:
Received 18 October 2018
Revised 7 May 2019
Accepted 7 May 2019
Available online 8 May 2019

Keywords:
Learning to rank
Feature generation
Machine learning
Information retrieval

A B S T R A C T

Learning to rank has become one of the most popular research areas in recent years. A series of learning to rank algorithms have been proposed to improve the ranking performance. In this work, we propose three learning to rank algorithms by directly optimizing evaluation measures based on the AdaRank algorithms. We name the three algorithms as AdaRank-ERR, AdaRank-MRR and AdaRank-Q, which optimize three evaluation measures, Expected Reciprocal Rank (ERR), Mean Reciprocal Rank (MRR), and Q-measure (Q), based on AdaRank, respectively. Furthermore, we propose a novel feature generation framework FG-FIREM to enhance the ranking performance. The framework generates effective document ranking features based on the ranking scores assigned by the proposed algorithms, and enriches the original feature space of learning to rank using the generated features for improving the ranking performance. We evaluate the proposed framework on three datasets from LETOR3.0 and the web dataset MSLR-WEB10K. The experimental results demonstrate that our framework can effectively improve the ranking performance.

1. Introduction

Ranking is one of the most important components in information retrieval (IR) systems. To produce effective ranking lists, learning to rank, as a series of supervised ranking methods, has been widely used. Learning to rank utilizes machine-learning technologies to construct ranking models that solve ranking problems in information retrieval for document ranking. In the process of ranking model construction, the training data involve a set of queries and retrieved documents with respect to each query. These documents are represented as feature vectors, in which features are extracted based on the similarity of each query-document pair. IR systems are designed to learn a ranking model based on the training data. The learned model is then fed into the testing process to score each document for a given query, and sort the documents by the relevance scores in a descending order. The final ranking list of documents is then provided to the user who submits the query as the search results. To accurately evaluate the ranking performance of different ranking models, various IR evaluation measures have been proposed and applied to evaluate the effectiveness of document retrieval, including the frequently-used evaluation measures Mean Average Precision (MAP) and Normalized Discounted Cumulative Gain (NDCG) (Manning, Raghavan, & Schütze, 2008), and the newly developed evaluation measures Expected Reciprocal Rank (ERR) (Chapelle, Metzler, & Zhang, 2009) and Q-measure (Q) (Sakai, 2004a). However, it remains a great challenge on how to improve the ranking performance in terms of these new evaluation measures.

To improve the ranking performance, more than 80 ranking algorithms have been proposed to improve the ranking performance (Tax, Bockting, & Hiemstra, 2015), such as RankNet, ListNet, RankBoost, AdaRank, SVMMAP and LambdaMART (Asadi & Lin, 2013; Burges, Shaked, & Renshaw, 2005; Cao, Qin, & Liu, 2007; Duh & Fujin, 2012; Freund & Schapire, 1997; Fuhr, 1989; Joachims, 2002; Li, Burges, & Wu, 2007; Lin, Lin, Xu, Abraham,& Liu, 2015; Song, Ng, Leung, & Fang, 2014; & Ma, 2012; Xu & Li, 2007; Yue, Finley, & Radlinski, 2007). Meanwhile, some studies have focused on the generation of effective document features to improve the ranking performance. For example, Amini, Truong, and Goutte (2008) proposed a novel strategy to assign relevance labels to the unlabeled instances based on the labeled instances, and used these instances to extend the feature space of the training data. Duh and Kirchhoff (2008) exploited the kernel-based Principal Component Analysis (kernel-based PCA) to extract effective features to enrich the training data. Lin, Lin, Yang, and Su (2009) used Singular Value Decomposition (SVD) to extract effective feature vectors from the unlabeled data set (the training and test sets) for enhanced ranking

* Corresponding authors.
E-mail addresses: zhlin@dlut.edu.cn (Y. Lin), xubo@dlut.edu.cn (B. Xu), hflin@dlut.edu.cn (H. Lin), xukan@dlut.edu.cn (K. Xu), pingzhang@mail.dlut.edu.cn (P. Zhang).

https://doi.org/10.1016/j.eswa.2019.05.004
0957-4174/© 2019 Elsevier Ltd. All rights reserved.

models. Lin, Lin, Xu, and Sun (2013) used the smoothing methods of language models for generating new feature vectors based on multiple parameters. These studies have investigated extended features to improve the ranking models and achieved enhanced ranking performance, which inspire us to generate more effective document features for constructing more powerful ranking models.

In this paper, we propose a novel feature generation framework for the optimization of learning to rank. The framework contains two components. The first component involves three new AdaRank-based ranking algorithms for the improvement of the ranking performance, and the second component extends the feature space of learning to rank based on the proposed three AdaRank-based ranking algorithms. The AdaRank (Xu & Li, 2007) ranking algorithm employs the classical machine learning method AdaBoost (Freund & Schapire, 1997) as the core technology, and merges IR evaluation measures into its loss function for the listwise ranking optimization. Previous experimental results on the LETOR3.0 datasets (Qin, Liu, & Xu, 2007) have demonstrated that the listwise AdaRank algorithm can yield better ranking performance than the pointwise and the pairwise algorithms.

The original AdaRank algorithm is designed for optimizing two IR evaluation measures, Mean Average Precision (MAP) and Normalized Discounted Cumulative Gain (NDCG). In this paper, we extend the original AdaRank algorithm to optimize three other evaluation measures, namely Expected Reciprocal Rank (ERR), Mean Reciprocal Rank (MRR) and Q-measure. We denote these three methods as AdaRank-ERR, AdaRank-MRR and AdaRank-Q, respectively. The three evaluation measures are newly developed for ranking performance evaluation in different learning to rank tasks. The ERR evaluation measure is based on the cascade model, which considers both the ranking positions of documents and the dependency relationship among documents in the ranking list. The MRR evaluation measure counts the average reciprocal ranks of results for all the queries, where ‘rank’ refers to the rank position of the first relevant document in the ranking list for each query. The Q-measure is a graded-relevance version of average precision, which inherits both the reliability of average precision and the multi-grade relevance capability of Average Weighted Precision (AWP) (Kando, Kuriyama, & Yoshioka, 2001; Sakai, 2004b). More details about these evaluation measures will be introduced in the IR evaluation measure section.

Directly optimizing above-mentioned different evaluation measures using AdaRank can produce diverse ranking lists of documents for the same query, which may contribute to the improvement of retrieval performance. Therefore, we propose to treat the ranking scores obtained from different ranking lists as new features to enhance the learned ranking models. Based on this idea, we construct 11 new features based on the proposed three AdaRank-based algorithms, and examine the effectiveness of these features from two respects: training ranking models based solely on the generated features and based on the combination of the generated features and the original features from the benchmark learning to rank datasets. We perform several experiments on these two feature sets to test whether our framework can generate effective ranking features. We evaluate our framework on three LETOR3.0 benchmark datasets (OHSUMED, TD2003 and TD2004) (Qin et al., 2007) and a large web dataset MSLR-WEB10K (http://research.microsoft.com/en-us/projects/mslr/). The experimental results demonstrate that our framework can significantly improve the ranking performance.

The main contributions of this paper are summarized as follows. (1) We propose three learning to rank algorithms, AdaRank-ERR, AdaRank-MRR and AdaRank-Q, to improve the AdaRank algorithm. The proposed algorithms can directly optimize three evaluation measures: ERR, MRR and Q-measure. We test the ranking performance of these algorithms in our experiments. (2) We treat the ranking scores generated by the proposed algorithms as new features, and propose a feature generation framework FGFIREM. The framework can generate effective ranking features and improve the ranking performance. (3) We enrich the feature spaces for learning to rank based on AdaRank-ERR, AdaRank-MRR and AdaRank-Q in FGFIREM, and combine the generated features with the original features for enhanced ranking models. The experimental results demonstrate that FGFIREM can significantly improve the ranking performance.

2. Related work

In recent years, learning to rank has become a popular research area in information retrieval. Many learning to rank algorithms have been proposed to improve ranking performance. These algorithms generally can be divided into three categories: the pointwise approach, the pairwise approach and the listwise approach (Liu, 2009). These three categories of approaches deal with a single document, a pair of documents and a list of documents, respectively.

The input space of the pointwise approach includes the feature vector of each single document and the output space contains the predicted relevance degree of each single document. Therefore, the pointwise ranking algorithms can be viewed as the regression-based algorithms or the classification-based algorithms. The representative algorithms include the Polynomial Regression Function method (Fuhr, 1989) and McRank (Li et al., 2007). The main disadvantage of the pointwise approach lies in that it does not consider the relative ranking positions of documents. To further incorporate the relative ranking positions, the pairwise approach has been proposed by considering the preference order of each document pair. The pairwise approach treats the pairwise preferences of document pairs as the input instances. The output space is the relative relevance degree of each document pair. The representative algorithms include RankNet (Burges et al., 2005), Ranking SVM (Joachims, 2002), Frank (Tsai, Liu, Qin, Chen, & Ma, 2007) and RankBoost (Freund, Iyer, Schapire, & Singer, 2003).

Furthermore, the listwise approach treats the entire ranking list of documents with respect to the same query as the input instance. The listwise output space is either the relevance degrees of all documents with respect to the query or the ranking list of the documents. The listwise approach can be divided into two subcategories. The first sub-category measures the difference between the permutation yielded by the ranking model and the ground truth permutation. Two representative methods belonging to this sub-category are RankNet (Burges et al., 2005), ListNet (Cao et al., 2007; Xu, Liu, Lu, Li, & Ma, 2008) and ListMLE (Xia, Liu, Wang, Zhang, & Li, 2008). The second sub-category of the listwise approach defines the ranking loss function based on the bound of IR evaluation measures. In this sub-category, the representative algorithms include the SVM-based methods SVMMAP (Yue et al., 2007), SVMNDCG (Chakrabarti, Khanna, & Sawant, 2008) and SVMERR (Zhang, Lin, Lin, & Wu, 2011), and other popular algorithms, AdaRank (Xu & Li, 2007), LambdaRank (Burges, Ragno, & Le, 2006) and LambdaMART (Donmez, Svore, & Burges, 2009; Yilmaz & Robertson, 2010).

2.1. Motivation of this work

Existing studies on learning to rank have indicated that document features are crucial in constructing ranking models. Effective document features contribute more to the improvement of ranking performance. Traditional ranking features are mainly based on textual statistics such as term frequency and inverse document frequency. Since there exist a limited number of textual statistics, the ranking performance may be further enhanced with more powerful features. To this end, we intend to develop a feature generation framework for learning to rank, which seeks to automatically generate effective ranking features based on a hand-crafted original feature set from benchmark learning to rank datasets. The generated features are obtained from certain effective ranking models. We propose three learning to rank algorithms based on the AdaRank algorithm, and use them to generate effective document features for enriching the feature space. We combine the generated features and the original features to learn more effective ranking models for enhanced ranking performance. Experimental results show that our approach can consistently improve retrieval performance on the test queries.

3. IR evaluation measures

3.1. Mean reciprocal rank

The Mean Reciprocal Rank (MRR) evaluation measure is a statistic measure for evaluating the ranking performance on a sample of given queries. The reciprocal rank of the document list for a certain query is the multiplicative inverse of the rank of the first relevant document: 1 for the first place, 1/2 for the second place, 1/3 for the third place and so on. The mean reciprocal rank is the average of the reciprocal ranks of all the queries, which is defined as follows.

$$MRR = \frac{1}{|Q|}\sum_{i=1}^{|Q|}\frac{1}{rank_i} \tag{1}$$

where $rank_i$ represents the rank position of the first relevant document for the i th query. Q is the query set, and $|Q|$ is the number of queries in Q. MRR counts the rank positions of the first relevant documents for each query. Since the first relevant document can always satisfy the information needs of search users to the utmost extent, MRR has been widely used to evaluate the ranking performance for various information retrieval systems.

3.2. Expected reciprocal rank

The Expected Reciprocal Rank (ERR) evaluation measure is based on the cascade model. The cascade model stems from the position-based model. The position-based model assumes that the selected documents by users are only determined by the ranking positions of the documents. However, the assumption is not always held in real retrieval scenarios, because the browsing behavior of a user can be affected by multiple factors. To deal with the problem, the cascade model was proposed by assuming that users view the retrieval results from the top position to the bottom position, and at each position, the information needs of the users can be satisfied with a certain probability.

Based on the cascade model, the ERR measure supposes that the document at the i th position of the document ranking list can satisfy the information needs of the user with a certain probability R_i. Therefore, the likelihood that the user has been satisfied at the r-th position can be formulized as follows.

$$\prod_{i=1}^{r-1}(1-R_i)R_r \tag{2}$$

The equation means that the user is not satisfied with the top *r-1* documents and is satisfied with the r-th document. The probability R can be represented as a function of document relevance degree, such as Normalized Discounted Cumulative Gain (NDCG). The probability R based on NDCG can be defined as follows.

$$R(g) = \frac{2^g-1}{2^{g_{\max}}}, \qquad g \in \{0, \ldots, g_{\max}\} \tag{3}$$

where g represents the relevance degree and g_{max} represents the relevance degree of the most relevant document. Since the rank positions of documents can largely affect the overall retrieval performance, a utility function φ is then integrated into the ERR measure, which is similar to the discount function used in NDCG. We define the utility function as $\varphi(r) = 1/r$ in our study. Finally, the ERR measure can be formulized as follows.

$$ERR = \sum_{r=1}^{N}\frac{1}{r}\prod_{i=1}^{r-1}(1-R_i)R_r \tag{4}$$

where N is the number of documents in the ranking list. Existing studies have demonstrated that ERR can more accurately evaluate the ranking performance by considering the discount of ranking compared with the position-based measures.

3.3. Q-measure

The Q-measure (Q) evaluation metric inherits both the reliability of Average Precision (AP) and the multi-grade relevance capability of Average Weighted Precision (AWP) (Sakai, 2004b). In this section, we first introduce these two basic evaluation measures, AP and AWP. The AP measure has been widely used in IR tasks, which considers the positions of relevance documents on the ranking list, and accumulates the precision at different cutoff positions of relevant documents. A formal definition of AP is as follows.

$$AP = \frac{\sum_{r=1}^{N}(P@r \cdot rel(r))}{n_q} \tag{5}$$

where n_q is the number of the relevant documents with respect to the query q, r is the rank position, N is the number of retrieved documents, *rel*() is a binary indication function reflecting the relevance of the document at the r-th position.

The AWP measure was designed for the ranking performance evaluation with multi-grade relevance. Let $J(r)$ be the relevance degree of the document at the r-th position. The relevance degrees for learning to rank tasks are three-fold: definitely relevant ($J(r)$=2), partially relevant ($J(r)$=1) and irrelevant ($J(r)$=0). The gain at the r-th position can be calculated using a function $g(r)$ = gain ($J(r)$). If the document at the r-th position is irrelevant, we set $g(r) = 0$. In our method, we set the function *gain*() as a linear function, namely g(r)=αJ(r), where α is a constant. The cumulative gain at the r-th position can be formulated as follows.

$$\begin{aligned} cg(r) &= g(r) + cg(r-1)r > 1 \\ cg(1) &= g(1)r = 1 \end{aligned} \tag{6}$$

Let $cg^*(r)$ be the cumulative gain at the *r-th* position in an ideal ranking list. The AWP measure is then defined as follows.

$$\text{AWP} = \frac{1}{R}\sum_{1\le r\le l} J(r)\frac{cg(r)}{cg*(r)} \tag{7}$$

where R represents the number of relevant documents. The AWP measure can be treated as an extension of the AP measure. However, it still suffers from a serious problem. Namely, if the number of relevant documents R and the idea cumulative gain $cg^*(r)$ are set as constants, AWP will not distinguish the difference between the following two systems: System A that sorts a relevant document at the r-th position ($r<R$) and System B that sorts a relevant document at the r'-th position ($r'>R$). The AP measure is free from this problem. To solve this problem, Q-measure is proposed by introducing the notion of bonus gain $bg(r)$ at the r-th position (Sakai, 2004a). In the paper, we follow the definition of Q-measure according to the work (Sakai & Song, 2011). This version of Q-measure defines $bg(r) = g(r) + 1$ if $g(r) > 0$, otherwise $bg(r) = 0$. Then, the cumulative bonus gain at the r-th position is formulized as follows.

$$cbg(r) = bg(r) + cbg(r-1) r > 1$$
$$cbg(1) = bg(1) r = 1 \tag{8}$$

where $bg(r)$ equals to $cg(r)$ $+C(r)$. $C(r)$ is the number of relevant documents from the first position to the r-th position in the ranking list. $C(r)$ is defined as follows.

$$C(r) = \sum_{k=1}^{r} J(k) \tag{9}$$

Based on the above definition, Q-measure at the l-th cutoff can be formulized as follows.

$$Q@l = \frac{1}{\min(l, R)} \sum_{r=1}^{l} J(r) \frac{C(r) + \beta cg(r)}{r + \beta cg*(r)} \tag{10}$$

The value of Q-measure is equal to 1 when the ranked list is in an ideal order. The parameter β ($\beta \geq 0$) is a persistence parameter for Q-measure. If $\beta = 0$, Q-measure is reduced to the binary average precision. In this paper, we set $\beta = 1$ according to the paper (Sakai & Song, 2011).

4. Methodology

4.1. Feature generation framework

In this section, we introduce the proposed **F**eature **G**eneration **F**ramework based on **I**nformation **R**etrieval **E**valuation **M**easures (FGFIREM) in detail. We first propose three learning to rank algorithms, and then use these algorithms to generate effective ranking features for constructing ranking models. Finally, we use the learned ranking models based on the generated features by FGFIREM to test the ranking performance. To help understand our framework FGFIREM, we list the notations and definitions in our method in Table 1.

Algorithm 1 presents the workflow of the FGFIREM framework. There are two components in this framework: the feature generation component and the ranking model learning component. The feature generation component involves three proposed learning to rank algorithms based on the AdaRank algorithm. We treat these three AdaRank-based algorithms as the feature generation algorithm FG_A, and learn ranking models using FG_A on the training set S. The ranking scores obtained from the learned ranking modes are then used as the generated new feature sets F_S and F_T for the training set S and the test set T, respectively. The dimension of the generated feature set depends on the number of FG_A. Finally, the original feature set O_S is combined with the generated feature set F_S as the new training set E_S. Similarly, the new test set E_T is generated. The ranking model learning component adopts the learning to rank algorithm R_A to learn new ranking models based on F_S and E_S, respectively. The ranking performance of the learned models is evaluated on the corresponding test set F_T and E_T.

In this framework, the feature generation algorithms FG_A and the ranking algorithm R_A can be the same or different. Our preliminary experimental results indicate that using the same learning to rank algorithms for FG_A and R_A can produce better ranking performance. This may be due to the fact that the same algorithm can make the most of the generated features to improve the ranking performance continuously during the process of the feature generation and the model training. In our framework, we adopt the proposed AdaRank-based ranking algorithms as FG_A and also adopt the same algorithms as R_A for model training.

4.2. Directly optimizing IR evaluation measure based on AdaRank

In this section, we first provide a brief introduction on AdaRank, and then elaborate on the proposed three ranking algorithms. Furthermore, we describe the feature generation process FG_A in detail, and discuss the application of the feature generation algorithms to the benchmark learning to rank datasets from LETOR3.0 and MSLR_WEB10K.

Learning to rank is a supervised learning process that requires a training set for model construction. A typical training set for learning to rank consists of n queries Q= $\{q_1, \ldots, q_n\}$. Each query q_i corresponds to a set of documents d_i=$\{d_{i1}, d_{i2}, \ldots, d_{i,\ nq(i)}\}$, which have been assigned a set of ground truth labels (relevance degrees) y_i=$\{y_{i1}, y_{i2}, \ldots, y_{i,\ nq\ (i)}\}$ in advance. $nq(i)$ represents the number of documents in the set d_i. The whole training set can be denoted as $S = \{q_i, d_i, y_i\}_{i=1}^{n}$.

AdaRank based on AdaBoost uses information retrieval evaluation measures to iteratively update the distribution on the training queries, and determines the combination coefficient α_t at each round of iteration. The input of AdaRank comprises the above-mentioned training set S. The parameters of AdaRank include the given evaluation measure M and the number of iterations T. At each iteration, one weak ranker h_t ($t = 1, \ldots, T$) is learned. The weak rankers in all the iterations are linearly combined to form a strong ranker that is regarded as the final ranking model. The flow of AdaRank is shown in Algorithm 2, in which equal weights ($1/n$) are assigned as the initialization on the data distribution $D_1(i)$ for each query i.

In each round, the algorithm generates a weak ranker h_t and the corresponding combination coefficient α_t based on the current distribution. The algorithm can increase the weight of a certain weak ranker h_t if the ranker yields accurate rankings of the documents for the queries. The distribution on queries are updated at the end of each iteration to focus more on the queries whose corresponding documents are not well ranked by h_t in the next

Table 1
Notations for the FGFIREM framework.

Notations	Definitions
FGFIREM	Feature generation framework based on information retrieval evaluation measures
S	Training set
T	Test set
FG_A	Feature generation algorithm
R_A	Ranking algorithm (Learning to rank algorithm)
O_S	Original features of training set
O_T	Original features of test set
F_S	New feature set for training set
F_T	New feature set for test set
E_S	Training set with the original features and the new features
E_T	Test set with the original features and the new feature
AdaRank-M	Ranking model based on the original features that directly optimizes the measure M using AdaRank
AdaRank-M*	Ranking model based on the generated features that directly optimizes the measure M using AdaRank
AdaRank-M**	Ranking model based on the combination of the original features and the generated features that directly optimizes the measure M using AdaRank

Table 2
The collection of the feature generation algorithms FG_A.

AdaRank-NDCG3	AdaRank-NDCG5	AdaRank-NDCG10
AdaRank-ERR3	AdaRank-ERR5	AdaRank-ERR10
AdaRank-Q3	AdaRank-Q5	AdaRank-Q10
AdaRank-MAP	AdaRank-MRR	

iteration. In our implementation, we use a single feature vector as the weak ranker, and use the linear combination of the learned weak rankers h_1, ..., h_t with weights α_1, ..., α_t to update the distribution D_{t+1}.

In feature generation, we employ five evaluation measures: MAP, NDCG, MRR, ERR and Q-measure to define the feature generation algorithms FG_A. Different evaluation measures evaluate the ranking lists of documents from different aspects. Therefore, different evaluation scores may jointly contribute to enriching the feature space of learning to rank. In existing studies (Xu & Li, 2007), MAP and NDCG have been used for model optimization by the AdaRank ranking algorithm, which are denoted as AdaRank-MAP and AdaRank-NDCG. For the rest of this paper, we use 'AdaRank-M' to represent the ranking algorithm that directly optimizes the evaluation measure M using the AdaRank ranking algorithm. In our study, we further extend the AdaRank algorithm to directly optimize MRR, ERR and Q-measure.

We select these three evaluation measures, because they have exhibited powerful capability in IR evaluation and widely used to evaluate the performance of IR-related tasks. Specifically, MRR has been commonly used in the evaluation of question answering systems. ERR employs the cascade model to define a gain function, which has been used for the evaluation on learning to rank challenge organized by Yahoo in 2010 (Chapelle & Chang, 2011). Q-measure, as a graded version of Average Precision (AP), yields better evaluation results than the AP-based measures. Based on these evaluation measures, we design eleven feature generation algorithms that are listed in Table 2.

In Table 2, AdaRank-M N represents directly optimizing the evaluation measure M at the cutoff position N based on AdaRank. The values of N includes 3, 5 and 10, which are widely used in the evaluation of ranking performance. Take NDCG@3 as an example. NDCG@3 measures the NDCG value at the top-3 ranked documents. Top ranked documents are important indicators for evaluating the ranking performance in IR evaluations. Besides, MAP and MRR are not cutoff-based evaluation measures. MAP measures the mean average precision for all the relevant documents at their corresponding cutoffs, and MRR counts the position of the first ranked relevant document in the list. Therefore, MAP and MRR don't need the parameter N in the evaluation process. In our experiments, we apply these feature generation algorithms to the benchmark learning to rank datasets to generate the new features. The detailed generation process of these new features are described in the following section.

4.3. Learning based on feature generation framework on benchmark datasets

In this section, we describe the application of the proposed feature generation algorithms on the benchmark datasets, such as LETOR3.0 and MSLR_WEB10K, for learning to rank. The generated features are integrated with the original features for learning the ranking models.

The benchmark datasets for learning to rank contain five folds. Each fold consists of three files, 'test.txt', 'valid.txt' and 'train.txt', as the test, validation and training sets. The training set is used to learn the ranking models, the validation set is used to tune the parameters, and the test set is used to evaluate the ranking performance of the learned models. The reported experimental results are the averaged performance over all the five test sets. We generate new features for these three sets based on the above-mentioned eleven feature generation algorithms. The ranking models for feature generation are trained on each training set S with these algorithms. The learned ranking models are used to predict the relevant scores on the documents with respect to the query in the test, validation and training sets. We use the scores to generate eleven dimensional features to enrich the original training, validation and test sets. The new feature sets for training and testing are denoted as F_S and F_T, respectively. We examine the effectiveness of the generated new features for ranking model training in our experiments. To further enhance the learned models, we intend to combine the generated features with the original features in the benchmark datasets and used the combined feature set to learn more effective ranking models. We believe that the combined feature set will contribute much to the improvement of ranking performance. Take the OHSUMED dataset from LETOR3.0 as an example, the dataset contains 45 dimensional features. We generate 11 new features to enrich the feature space. The combined feature set totally contains 56 dimensional features for model training.

We adopt two ranking algorithms, AdaRank-ERR10 and AdaRank-Q10, to learn the final ranking models in our experiments. The ranking models are trained based on both the generated feature set and the combined feature set. We select these two learning algorithms, because relatively good ranking performance can be achieved by them on the validation sets among all the algorithms in Table 2.

5. Experiments and analysis

5.1. Corpora

We perform a series of experiments on four benchmark datasets from LETOR3.0 (Qin et al., 2007) and MSLR-WEB10K (http://research.microsoft.com/en-us/projects/mslr/) released by Microsoft Research Asia. These datasets are both evenly divided into five folds for cross validation. LETOR3.0 consists of three datasets: OHSUMED, TD2003, and TD2004. The OHSUMED collection, as a subset of MEDLINE (a database on medical publications), consists of 106 queries, 11,303 irrelevant documents, 4837 relevant documents, and 45 dimensional features. The relevance degrees of documents in OHSUMED fall into three levels: definitely relevant, partially relevant, or irrelevant. The TD2003 dataset contains 50 queries, 48,655 irrelevant documents, and 516 relevant documents. The TD2004 dataset consists of 75 queries, 73,726 irrelevant documents, and 444 relevant documents. Both the TD2003 dataset and the TD2004 dataset contain 64 features. The MSLR-WEB10K, as a set of web data, contains 10,000 queries. Queries and Uniform Resource Locators (URLs) are characterized by identifiers. Feature vectors are extracted for query-URL pairs, and annotated with relevance judgment labels. The relevance judgments are obtained from the query log of the commercial web search engine, Microsoft Bing. The relevance labels take 5 values from 0 (irrelevant) to 4 (perfectly relevant). MSLR-WEB10K contains 136 dimensional feature vectors. The statistics of the datasets are shown in Table 3.

5.2. Experimental settings

We perform three groups of experiments on these benchmark datasets. The first group of experiments is designed to test the ranking performance of the proposed ranking algorithms without considering the feature generation process. The second group of experiments seeks to evaluate the effectiveness of the generated features based on the FG_A algorithms. The third group of experiments is designed to examine the effectiveness of the combined

Table 3
Statistics of the datasets.

Dataset	Number of queries	Dimension of features	Scale of relevance
OHSUMED from LETOR3.0	106	45	3
TD2003 from LETOR3.0	50	64	3
TD2004 from LETOR3.0	75	64	3
MSLR-WEB10K	10,000	136	5

feature set including the original and the generated features. Two proposed ranking algorithms, AdaRank-ERR10 and AdaRank-Q10, are used for training the ranking models. The original, generated and combined feature sets are compared in terms of the ranking performance by the learned models. Moreover, all the experimental results are compared with the state-of-the-art ranking algorithms, including AdaRank-MAP, AdaRank-NDCG, SVMMAP, ListNet and LambdaMART. These algorithms have been proved to be the most effective in learning to rank tasks (Wu, Burges, Svore, & Gao, 2010). We use the standard divisions of the training, test and validation sets in the used datasets for five-fold cross validation. The training set is used to learn the ranking models, the validation set is used to tune the parameters, and the test set is used to evaluate the ranking performance of the learned models. We report the average performance on the five test sets in all the folds in our experiments.

Table 4
The ERR@10 and Q@10 values for FG_A algorithms.

ERR@10 values	OHSUMED	TD2003	TD2004
SVMPMAP	0.5244	0.2659	0.3172
ListNet	0.5615	0.2554	0.3342
AdaRank-MAP	0.5759	0.2918	0.3706
AdaRank-NDCG	0.5645	0.2803	0.3517
AdaRank-ERR10	0.5686^{*+}	0.2804^{*+}	0.3537^{*+}
AdaRank-MRR	0.5390^{*}	0.2736^{*+}	0.3333^{*}
AdaRank-Q10	0.5553^{*}	0.3153^{*+}	0.3442^{*+}
Q@10 values	**OHSUMED**	**TD2003**	**TD2004**
SVMPMAP	0.3245	0.1869	0.1859
ListNet	0.3544	0.1879	0.1641
AdaRank-MAP	0.3636	0.2336	0.2126
AdaRank-NDCG	0.3628	0.1930	0.1867
AdaRank-ERR10	0.3672^{*+}	0.2203^{*+}	0.1843^{+}
AdaRank-MRR	0.3415^{*}	0.2192^{*+}	0.1820^{+}
AdaRank-Q10	0.3566^{*+}	0.2431^{*+}	0.1794^{+}

5.3. Evaluation on the proposed three ranking algorithms

In this section, we present the ranking performance of the proposed ranking algorithms, AdaRank-ERR, AdaRank-MRR, AdaRank-Q compared with the baseline ranking algorithms. The compared baseline algorithms include the original AdaRank algorithms (AdaRank-MAP, AdaRank-NDCG) and the state-of-the-art ranking algorithms SVMMAP and ListNet. We choose the cutoff position of AdaRank-ERR and AdaRank-Q as 10 for model optimization, because relatively good performance can be achieved under this setting. Retrieval performance is evaluated in terms of MAP and NDCG. The comparisons of retrieval performance on three datasets from LETOR3.0 are illustrated in Figs. 1–3.

From these figures, we observe that both AdaRank-Q10 and AdaRank-ERR10 outperform AdaRank-MAP and AdaRank- NDCG on the OHSUMED and TD2003 datasets. AdaRank-MRR only outperforms AdaRank-MAP and AdaRank-NDCG on the TD2003 dataset; the proposed algorithms yield slightly worse performance than AdaRank-MAP and AdaRank-NDCG in terms of MAP, NDCG@5 and NDCG@10 on the TD2004 dataset. Comparing with SVMMAP and ListNet, AdaRank-Q10 achieves the best MAP and NDCG@5 values on OHSUMED and the best NDCG values on TD2003, but not performs very well on TD2004. On the OHSUMED dataset, AdaRank-ERR10 also yields better performance than SVMMAP and ListNet. AdaRank-MRR achieves better performance on TD2003 and TD2004 than OHSUMED.

To further evaluate the ranking performance of different AdaRank-based algorithms, we report the ranking performance in terms of ERR@10 and Q@10 in Table 4, where the superscript '*' indicates significant improvements of our methods over SVMMAP,

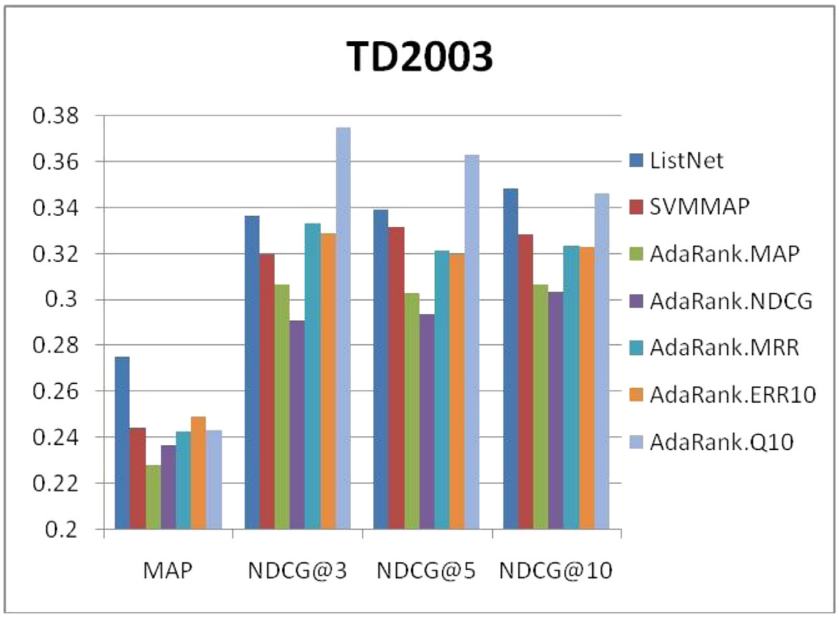

Fig. 1. The MAP and NDCG@K values for the proposed ranking algorithms on TD2003 dataset.

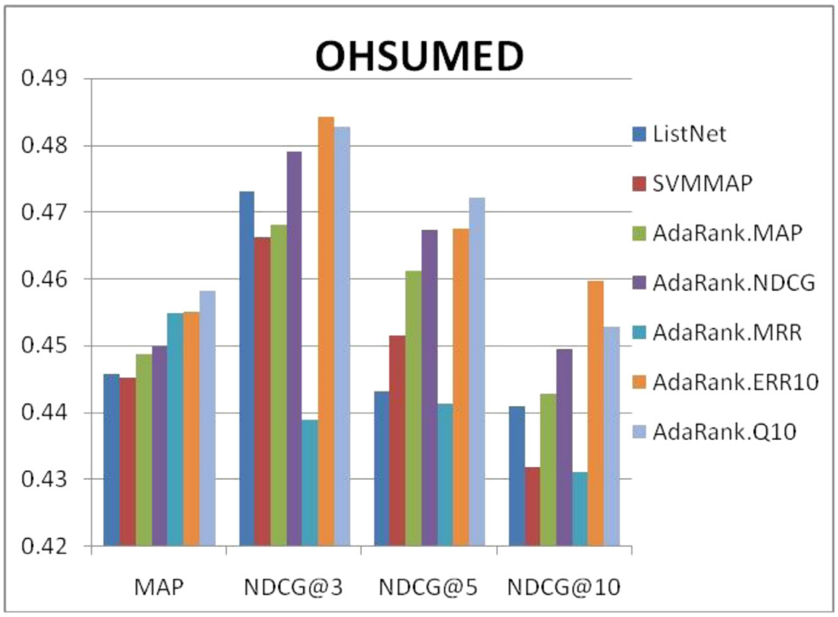

Fig. 2. The MAP and NDCG@K values for the proposed ranking algorithms on OHSUMED dataset.

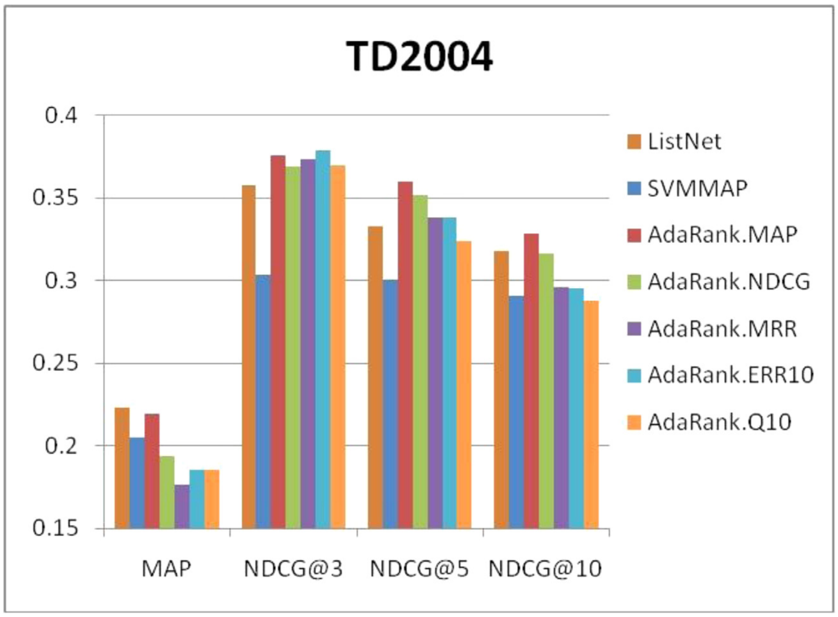

Fig. 3. The MAP and NDCG@K values for the proposed ranking algorithms on TD2004 dataset.

and the superscript '+' indicates significant improvements of our methods over ListNet.

The experimental results in the table show that AdaRank-ERR10 performs better than AdaRank-NDCG in terms of ERR@10 on all the datasets evaluation measure. AdaRank-MAP achieves the best ERR value on the OHSUMED and the TD2004 datasets. AdaRank-Q10 achieves the best performance on the TD2003 dataset. For the Q@10 evaluation measure, AdaRank-ERR10 also achieves the best performance on OHSUMED and outperforms AdaRank-NDCG on TD2003. AdaRank-Q10 and AdaRank-MAP achieve the best performance on TD2003 and TD2004, respectively.

In general, we conclude that the proposed algorithms AdaRank-ERR and AdaRank-Q are more effective in improving the ranking performance than the compared baseline models, particularly on the OHSUMED and TD2003 datasets. These experimental results demonstrate the effectiveness of the proposed AdaRank-based algorithms by directly optimizing evaluation measures. Our algorithms achieve slightly better performance than AdaRank-MAP and AdaRank-NDCG, which indicates that directly optimizing ERR and Q-measure contribute more to the ranking performance. Since the increase in the improvement is still limited, we seek to further enhance the ranking effectiveness by generating new features for model construction. In the next section, we report the experimental results of the ranking models trained on the generated feature sets.

5.4. Evaluations on the generated feature sets

In this section, we seek to answer the question that whether the new generated features can be employed to improve the

Table 5
Evaluation results for the FGFIREM on four public learning to rank datasets.

Ranking performance on OHSUMED dataset	MAP	NDCG@3	NDCG@5	NDCG@10	ERR@10	Q@10
SVMMAP (Yue et al., 2007)	0.4453	0.4663	0.4516	0.4319	0.5244	0.3245
ListNet (Cao et al., 2007)	0.4457	0.4732	0.4432	0.4410	0.5615	0.3544
AdaRank-MAP (Xu & Li, 2007)	0.4487	0.4682	0.4613	0.4429	0.5759	0.3636
AdaRank-NDCG (Xu & Li, 2007)	0.4498	0.4790	0.4673	0.4496	0.5645	0.3628
AdaRank-ERR10	0.4550^{*+}	0.4843^{*+}	0.4676^{*+}	0.4597^{*+}	0.5686^{*+}	0.3672^{*+}
AdaRank-Q10	0.4583^{*+}	0.4829^{*+}	0.4722^{*+}	0.4529^{*+}	0.5553*	0.3566^{*+}
AdaRank-ERR10*	0.4587^{*+}	0.4751^{*+}	0.4653^{*+}	0.4512^{*+}	0.5502*	0.3571^{*+}
AdaRank-Q10*	0.4571^{*+}	0.4708*	0.4607^{*+}	0.4472^{*+}	0.5475*	0.3568^{*+}
AdaRank-ERR10**	0.4575^{*+}	0.4705*	0.4590^{*+}	0.4485^{*+}	0.5531*	0.3564^{*+}
AdaRank-Q10**	0.4579^{*+}	0.4889^{*+}	0.4724^{*+}	0.4576^{*+}	0.5639^{*+}	0.3646^{*+}

Ranking performance on TD2003 dataset	MAP	NDCG@3	NDCG@5	NDCG@10	ERR@10	Q@10
SVMMAP (Yue et al., 2007)	0.2445	0.3199	0.3318	0.3282	0.2659	0.1869
ListNet (Cao et al., 2007)	0.2753	0.3365	0.3393	0.3484	0.2554	0.1879
AdaRank-MAP (Xu & Li, 2007)	0.2283	0.3067	0.3029	0.3069	0.2918	0.2336
AdaRank-NDCG (Xu & Li, 2007)	0.2368	0.2908	0.2939	0.3036	0.2803	0.1930
AdaRank-ERR10	0.2493*	0.3286*	0.3199	0.3231	0.2804^{*+}	0.2203^{*+}
AdaRank-Q10	0.2432	0.3748^{*+}	0.3628^{*+}	0.3461*	0.3153^{*+}	0.2431^{*+}
AdaRank-ERR10*	0.2681*	0.3729^{*+}	0.3563^{*+}	0.3512^{*+}	0.3158^{*+}	0.2485^{*+}
AdaRank-Q10*	0.2766*	0.3857^{*+}	0.3802^{*+}	0.3643^{*+}	0.3181^{*+}	0.2581^{*+}
AdaRank-ERR10**	0.2811^{*+}	0.4155^{*+}	0.4012^{*+}	0.3841^{*+}	0.3368^{*+}	0.2724^{*+}
AdaRank-Q10**	0.2838^{*+}	0.4003^{*+}	0.3928^{*+}	0.3769^{*+}	0.3386^{*+}	0.2723^{*+}

Ranking performance on TD2004 dataset	MAP	NDCG@3	NDCG@5	NDCG@10	ERR@10	Q@10
SVMMAP (Yue et al., 2007)	0.2049	0.3035	0.3007	0.2907	0.3172	0.1859
ListNet (Cao et al., 2007)	0.2231	0.3573	0.3325	0.3175	0.3342	0.1641
AdaRank-MAP (Xu & Li, 2007)	0.2189	0.3757	0.3602	0.3285	0.3706	0.2126
AdaRank-NDCG (Xu & Li, 2007)	0.1936	0.3688	0.3514	0.3163	0.3517	0.1867
AdaRank-ERR10	0.1854	0.3786^{*+}	0.3383^{*+}	0.2948*	0.3537^{*+}	0.1843^{+}
AdaRank-Q10	0.1855	0.3696^{*+}	0.3234*	0.2876	0.3442^{*+}	0.1794^{+}
AdaRank-ERR10*	0.2368^{*+}	0.4166^{*+}	0.3764^{*+}	0.3397^{*+}	0.3836^{*+}	0.2189^{*+}
AdaRank-Q10*	0.2366^{*+}	0.4134^{*+}	0.3759^{*+}	0.3377^{*+}	0.3830^{*+}	0.2175^{*+}
AdaRank-ERR10**	0.2386^{*+}	0.4184^{*+}	0.3866^{*+}	0.3487^{*+}	0.3826^{*+}	0.2248^{*+}
AdaRank-Q10**	0.2401^{*+}	0.4115^{*+}	0.3775^{*+}	0.3577^{*+}	0.3815^{*+}	0.2323^{*+}

Ranking performance on MSLR-WEB10K dataset	MAP	NDCG@3	NDCG@5	NDCG@10	ERR@10	Q@10
LambdaMART (Wu et al, 2010)	0.5546	0.5675	0.5524	0.5324	0.3524	0.3961
AdaRank-MAP	0.5927	0.3753	0.3768	0.3879	0.4734	0.3404
AdaRank-NDCG	0.5910	0.3834	0.3832	0.3907	0.4844	0.3419
AdaRank-ERR10	0.5909*	0.3826	0.3817	0.3905	0.4835*	0.3414
AdaRank-Q10	0.5931*	0.3821	0.3835	0.3940	0.4831*	0.3458
AdaRank-ERR10*	0.5939*	0.3838	0.3847	0.3950	0.4852*	0.3463
AdaRank-Q10*	0.5941*	0.3846	0.3854	0.3953	0.4857*	0.3468
AdaRank-ERR10**	0.5945*	0.3843	0.3851	0.3951	0.4850*	0.3474
AdaRank-Q10**	0.5941*	0.3841	0.3846	0.3952	0.4845*	0.3468

ranking performance. We use the proposed framework FGFIREM to generate new features based on the feature generation algorithms in Table 2. The two best-performed algorithms, AdaRank-ERR10 and AdaRank-Q10, from the previous experiments are adopted as the final learning algorithms to train ranking models on the generated feature sets. We denote the experimental results generated by these two learning algorithms as AdaRank-ERR10* and AdaRank-Q10*, respectively. It is worth noting that the two components in our framework, feature generation and ranking model training, can jointly contribute to improving ranking performance. To distinguish the contribution made by these two components, we also compare the results by AdaRank-ERR10* and AdaRank-Q10* on the generated feature sets with the results by AdaRank-ERR10 and AdaRank-Q10 on the original feature sets.

Table 5 shows the comparisons of results on the feature generation models, AdaRank-ERR10, AdaRank-Q10 and other learning to rank algorithms. For the three datasets from LETOR3.0, we first compare our models with the published evaluation results in terms of MAP and NDCG. Since there are no published evaluation results by ERR@k and Q@k on these datasets, we adopt the RankLib (http://www.cs.umass.edu/~vdang/ranklib.html) toolkit to obtain the evaluation results for the ListNet algorithm, and the svmmap

Algorithm 1
The workflow of the FGFIREM framework.

```
Input: Training set S; Test set T; original feature set O_S;
Input: Learning to rank algorithm(s) FG_A;
Input: Learning to rank algorithm R_A;
Component 1. Feature generation
    1: Learn() on S with FG_A
    2: Score() on S and T, generate new feature set F_S and F_T
    3: Combine(O_S, F_S)=> E_S, Combine(T, F_T)=>E_T
End component 1.
Component 2. Ranking model learning
    1: Learn() on F_S, E_S with R_A
    2: Score() on F_T, E_T with learned model
    3: Eval() on F_T, E_T
End component 2.
Output: Evaluation results.
```

software package (http://projects.yisongyue.com/svmmap/) to obtain the evaluation results for the SVMMAP algorithm by ERR@k and Q@k. For the MSLR-WEB10K dataset, only one published results based on the LambdaMART algorithm are available, which is obtained using the jforests toolkit (Ganjisaffar, Caruana, & Lopes, 2011). Therefore, we only compare with the results of

Algorithm 2
The AdaRank-M algorithm.

```
Input: $S = \{q_i, d_i, y_i\}_{i=1}^{n}$ the number of iterations $T$, the evaluation metric $M$
Initialize the data distribution $D_1(i)=1/n$
For $t=1, \dots, T$
    Train a weak ranker $h_t$ based on the distribution $Dt$ on training set $S$.
    Choose $\alpha_t$ $\alpha_t = \frac{1}{2} \cdot \ln \frac{\sum_{i=1}^{n} D_t(i)(1+M(h_t, d_i, y_i))}{\sum_{i=1}^{n} D_t(i)(1-M(h_t, d_i, y_i))}$
    Update $D_{t+1}$
    $D_{t+1}(i) = \frac{\exp(-M(\sum_{z=1}^{t} \alpha_z h_z, d_i, y_i))}{\sum_{j=1}^{n} \exp(-M(\sum_{z=1}^{t} \alpha_z h_z, d_j, y_j))}$
End for
Output : $\sum_{t=1}^{T} \alpha_t h_t$
```

LambdaMART on the MSLR-WEB10K dataset. We directly compare the evaluation results from the existing studies instead of reproducing the algorithms in our experiments. Since we didn't find published evaluation results using LambdaMART on LETOR, we do not compare our method with the algorithm on datasets from LETOR. We compare the results using statistical significant test, i.e., two-tailed paired Student's t tests with 95% confidential level ($p < 0.05$), where the superscript '*' indicates significant improvements of our methods over SVMMAP or LambdaMART on different datasets, and the superscript '+' indicates significant improvements of our methods over ListNet.

On the OHSUMED dataset, AdaRank-ERR10* outperforms ListNet ($p = 0.033$) and achieves the best MAP value. AdaRank-Q10 also obtains a good ranking performance in terms of MAP and NDCG@5, and AdaRank-ERR10 yields the highest NDCG@3, NDCG@10 and Q@10 values. All the four proposed algorithms outperform other baseline models in terms of MAP. These results indicate that the proposed AdaRank-based algorithms and our feature generation framework can consistently improve the ranking performance. Meanwhile, we observe that AdaRank-Q10 outperforms AdaRank-Q10*, which indicates that the ranking models solely based on the generated features yields slightly worse results than the model based on the original features. This finding may be because the scale of the OHSUMED dataset is relatively smaller than the other used datasets, which limits the extent of improvement to a certain degree.

On the TD2003 dataset, we observe that all the four proposed algorithms outperform other baseline models by MAP. AdaRank-Q10* outperforms ListNet ($p = 0.007$) and SVMMAP ($p < 0.05$), achieving the best ranking performance in terms of all the six evaluation measures. The improvement in the average ranking performance is about 13.8% over the ListNet algorithm and 5.2% over the proposed AdaRank-Q10 algorithm. AdaRank-ERR10* also outperforms other state-of-the-art learning to rank algorithms in terms of most evaluation measures, which achieves 11.1% improvement compared with the proposed AdaRank-ERR10 algorithm. These results show that the proposed FGFIREM framework can generate effective features for the improvement of the ranking performance.

On the TD2004 dataset, AdaRank-ERR10* achieves the best ranking performance in terms of all the six evaluation measures, which significantly outperforms ListNet ($p = 0.001$) and produces 13.7% improvement over the AdaRank-ERR10 algorithm. AdaRank-Q10* yields comparable ranking performance with AdaRank-ERR10*, achieving the second best ranking performance among all the compared models. AdaRank-Q10* improves the ranking performance by 13.6% compared with ListNet. The improvements of the feature generation algorithms AdaRank-Q10* and AdaRank-ERR10* indicate the effectiveness of the proposed FGFIREM framework.

On the new MSLR-WEB10K dataset, AdaRank-Q10* significantly outperforms AdaRank-Q10 ($p = 0.008$) and AdaRank-MAP ($p = 0.017$), achieving the best performance in terms of MAP and ERR@10 among all the baseline models. The LambdaMART algorithm, as the best-performed model on this dataset, achieves the best performance in terms of other evaluation measures. AdaRank-ERR10* also yields a good performance on this dataset. These results indicate that our AdaRank-based feature generation framework can achieve the state-of-the-art ranking performance.

From the experimental results in Table 5, we observe that both AdaRank-ERR10* and AdaRank-Q10* perform well, particularly on the TD2003 and TD2004 datasets. The improvements of our framework over baselines on TD2003, TD2004 and MSLR-WEB10K are consistent. AdaRank-Q10* and AdaRank-ERR10* generally yield better performances than AdaRank-Q10 and AdaRank-ERR10. The ranking performances of AdaRank-ERR10* and AdaRank-Q10* are comparable to each other. These findings indicate that the generated features by our feature generation framework FGFIREM are effective in improving the ranking performance. Since both the generated features and original features contribute to the ranking performance based on the learned ranking models, we believe these two feature sets can capture different aspects of the ranking information in model training. Therefore, we combine them to form a merged feature set for model training, and examine the effectiveness of the learned models in the section below.

5.5. Evaluations on the combined feature sets

In this section, we seek to answer the question that whether the combined feature set with both the generated features and the original features can further improve the ranking performance. We conduct experiments based on the combined feature sets on the datasets from LETOR3.0 and MSLR-WEB10K.

We also employ AdaRank-ERR10 and AdaRank-Q10 as our learning algorithms to train ranking models. We denote the results obtained from the combined feature sets by these two learning algorithms as AdaRank-ERR10** and AdaRank-Q10**. The experimental results are also shown in Table 5.

From Table 5, we observe that AdaRank-ERR10* achieves the highest MAP value on the OHSUMED dataset. AdaRank-Q10** yields the best ranking performance in terms of the other five evaluation measures except MAP. The t-test results show that AdaRank-Q10** significantly outperforms AdaRank-Q10* ($p = 0.004$), but AdaRank-ERR10** does not outperform AdaRank-ERR10*. The results suggest that the ranking performance based on the combined features (**) does not always higher than that based only on the generated features on the OHSUMED dataset. On TD2003 and TD2004 datasets, we observe that both AdaRank-Q10** and AdaRank-ERR10** outperform AdaRank-Q10* and AdaRank-ERR10*. We observe a similar tendency on the MSLR-WEB10K dataset, on which AdaRank-Q10** does not outperform AdaRank-Q10*, but AdaRank-ERR** significantly outperforms AdaRank-ERR10* ($p = 0.035$), although the improvement seems smaller than those on the other datasets. We attribute the reasons for less improvement on the MSLR-WEB10K dataset to the heterogeneity of the queries in the dataset. MSLR-WEB10K consists of a large scale of heterogeneous queries from web search logs, which is different from the homogeneous queries on the other used datasets. Therefore, the improvements on the dataset may be limited to a certain extent.

The experimental results in the table indicate that the combined features can improve the ranking performance in most circumstances. Combined with the previous results, we conclude that our feature generation framework FGFIREM can generate effective features and improve the ranking performance. However, the improvement in ranking performance based on the combined features does not always higher than that based only on the generated features. The experimental results show that the combined features can improve the ranking performance, but the extent of improvement is not remarkable. We provide further analysis and discussion of the proposed framework in the next section for future optimization of our methods.

5.6. Experimental analysis

In this section, we further analyze the experimental results and provide the possible reasons for the improvement of ranking performance. In the above sections, we present three groups of experiments. The first group of experiments focuses on the first component of our FGFIREM framework, which extends the AdaRank algorithm to three other evaluation measures, MRR, ERR and Q-measure. From the experimental results, we conclude that the extended AdaRank algorithm does improve ranking performance in most used datasets. A possible reason for this finding lies that AdaRank belongs to the listwise approach, which generates the ranking model through a linear combination of weak rankers. ERR and Q-measure also evaluate the ranking performance from a listwise perspective. Specifically, ERR based on the cascade model considers the dependency relationship among documents in the ranking list, and Q-measure, as a graded-relevance version of average precision, inherits both the reliability of average precision and the multi-grade relevance capability of average weighted precision. Therefore, employing the AdaRank algorithm to optimizing these evaluation measures yields a better ranking performance.

The second group of experiments utilizes the AdaRank-ERR10 and AdaRank-Q10 as the learning algorithms to train ranking models on the generated feature sets by FGFIREM. The experimental results indicates that the generated features indeed help improve the ranking performance. We attribute the improvement to the good property of the boosting mechinism of AdaRank, which uses weak rankers to consistently boosting the ranking performance. Our framework also learns from the boosting idea to generate new features by directly optimizing five evaluation measures using AdaRank. The experimental results show that the generated features can improve the ranking performance effectively.

The third group of experiments are designed to examine the effectiveness of the combined feature sets that include the original features and the new generated features. The experimental results show that the combined features contribute much to improving the ranking performance in most circumstances. This is because the combined features can simultaneously capture the ranking information through both the orignal and the generated features for model construction, therefore enhancing the ranking performance

From the experimental results, we can conclude that our FGFIREM framework improves the ranking performance over the state-of-the-art learning to rank algorithms in terms of most used evaluation measures. Besides, we further analyze the time complexity based on the workflow of FGFIREM. The time complexity of our framework depends on the boosting-based feature generation algorithms. The time complexity of our framework is $O(G\cdot(F+T)\cdot n\cdot m\log m)$ in the process of feature generation, where G denotes the number of feature generation algorithms, F denotes the number of feature dimensions, T denotes the iterations, n denotes the number of queries in training data, and m is the maximum number of documents for all the queries in training data. The experimental results indicate that the ranking performance over the combined features is not always higher than that over the generated features by FGFIREM, and meanwhile learning ranking models on the combined feature sets needs more memory and time. Therefore, the real IR systems using the proposed framework should balance the relationship between effectiveness and efficiency for more improvement in the overall performance.

6. Conclusions and future work

In this paper, we focus on the problem that whether the new generated features can help improve ranking performance for learning to rank. We propose a feature generation framework FGFIREM to tackle the problem. FGFIREM utilizes learning to rank algorithms to generate better features from the training data for model construction. Specifically, we first propose three ranking algorithms based on the AdaRank algorithm by directly optimizing MRR, ERR and Q-measure. We then use the proposed algorithms to generate effective ranking features, and learn ranking models based on both the generated feature sets and the combined feature sets, respectively. We evaluate our framework on four benchmark datasets from LETOR3.0 and MSLR-WEB10K. The experimental results demonstrate that FGFIREM can generate better features and improve the ranking performance. The proposed FGFIREM framework based on the generated features and the original features provides an approach to enrich the feature space of learning to rank. The proposed algorithms AdaRank-ERR, AdaRank-MRR and AdaRank-Q together with the feature generation framework can offer both the new idea to improve the other ranking approaches and the effective features to train ranking models. Our future work will be carried out to generate better features for other learning to rank algorithms to improve the ranking performance.

Conflict of interest statement

The authors declared that they have no conflicts of interest to this work. We declare that we do not have any commercial or associative interest that represents a conflict of interest in connection with the work submitted.

Credit authorship contribution statement

Yuan Lin: Conceptualization, Methodology, Writing - original draft. **Bo Xu:** Writing - original draft, Writing - review & editing, Supervision. **Hongfei Lin:** Funding acquisition, Supervision. **Kan Xu:** Formal analysis, Investigation, Validation. **Ping Zhang:** Writing - original draft, Data curation, Validation.

Acknowledgments

This work is partially supported by grant from the Natural Science Foundation of China (no. 61602078, 61632011, 61572102, 61572098), the Ministry of Education Humanities and Social Science Project (no. 19YJCZH199, 16YJCZH12), the Fundamental Research Funds for the Central Universities(no. DUT18ZD102, DUT19RW117, DUT19JC45), China Postdoctoral Science Foundation (no. 2018M641691).

References

Amini, M. R., Truong, T. V., & Goutte, C. (2008). A boosting algorithm for learning bipartite ranking functions with partially labeled data. In *ACM Special Interest Group on Information Retrieval* (pp. 99–106).

Asadi, N., & Lin, J. (2013). Training efficient tree-based models for document ranking. In *Proceedings of the 25th European Conference on Advances in Information Retrieval: 7814* (pp. 146–157).

Burges, C., Ragno, R., & Le, Q. V. (2006). Learning to rank with non-smooth cost functions. In *NIPS'06: Proceedings of 19th Advances in Neural Information Processing System* (pp. 193–200).

Burges, C., Shaked, T., & Renshaw, E. (2005). Learning to rank using gradient descent. In *ICML'05: Proceedings of the 22th International Conference on Machine Learning* (pp. 89–96).

Cao, Z., Qin, T., & Liu, T. Y. (2007). Learning to rank: from pairwise approach to listwise approach. In *ICML'07: Proceedings of the 24th International Conference* (pp. 129–136).

Chakrabarti, S., Khanna, R., & Sawant, U. (2008). Structured learning for non-smooth ranking losses. In *KDD'08: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 88–96).

Chapelle, O., & Chang, Y. (2011). Yahoo! learning to rank challenge overview. In *Journal of Machine Learning Research - Proceedings Track: 14* (p. 1,24).

Chapelle, O., Metzler, D., & Zhang, Y. (2009). Expected reciprocal rank for graded relevance. In *CIKM'09: Proceedings of the 18th ACM Conference on Information and Knowledge Management* (pp. 621–630).

Donmez, P., Svore, K. M., & Burges, C. J. (2009). On the local optimality of LambdaRank. In *SIGIR'09: Proceedings of the 32th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 460–467).

Duh, K., & Kirchhoff, K. (2008). Learning to rank with partially-labeled data. In *ACM Special Interest Group on Information Retrieval* (pp. 251–258).

Duh, K., & Fujin, A. (2012). Flexible sample selection strategies for transfer learning in ranking. *Information Processing & Management, 48*(3), 502–512.

Freund, Y., & Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences, 55*(1), 119–139.

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

Fuhr, N. (1989). Optimum polynomial retrieval functions based on the probability ranking principle. *ACM Transactions on Information Systems, 7*, 183–204.

Ganjisaffar, Y., Caruana, R., & Lopes, C. V. (2011). Bagging gradient-boosted trees for high precision, low variance ranking models. In *SIGIR'11: Proceedings of the 34st annual international ACM SIGIR conference on Research and development in information retrieval* (pp. 85–94).

Joachims, T. (2002). Optimizing search engines using clickthrough data. In *KDD'02: Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 133–142).

Kando, N., Kuriyama, K., & Yoshioka, M. (2001). Information retrieval system evaluation using multi-grade relevance judgments—discussion on averageable single-numbered measures (in Japanese). In *IPSJ SIG Notes, FI–63–12* (pp. 105–112).

Li, P., Burges, C., & Wu, Q. (2007). McRank: learning to rank using multiple classification and gradient boosting. In *NIPS'07: Proceedings of the 21th Annual Conference on Neural Information Processing Systems* (pp. 845–852).

Lin, Y., Lin, H. F., Yang, Z. H., & Su, S. (2009). A boosting approach for learning to rank using SVD with partially labeled data. In *AIRS'09: Proceedings of 5th Asia Information Retrieval Symposium (AIRS), LNCS 5839* (pp. 330–338). Springer.

Lin, Y., Lin, H. F., Xu, K., & Sun, X. (2013). Learning to Rank using Smoothing Methods for Language modeling. *Journal of the American Society for Information Science and Technology, 64*(4), 818–828.

Lin, Y., Lin, H. F., Xu, K., Abraham, A., & Liu, H. (2015). Group-enhanced ranking. *Neurocomputing, 150*(SI), 99–105.

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

Manning, C. D., Raghavan, P., & Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press.

Qin, T., Liu, T. Y., & Xu, J. (2007). LETOR: benchmark collection for research on learning to rank for information retrieval. In *LR4IR'07: SIGIR 2007 Workshop on Learning to Rank for Information Retrieval* (pp. 3–10).

Sakai, T. (2004a). New performance metrics based on multigrade relevance: their application to question answering. In *NTCIR-4: Proceedings of the fourth NTCIR Workshop on Research in Information Access Technologies Information Retrieval* Question Answering and Summarization.

Sakai, T. (2004b). Ranking the NTCIR systems based on multigrade relevance. In *Proceedings of Asia information retrieval symposium 2004* (pp. 170–177).

Sakai, T., & Song, R. (2011). Evaluating diversified search results using per-intent graded relevance. In *Proceedings of The 34th Annual ACM SIGIR Conference* (pp. 1043–1052).

Song, Y., Ng, W., Leung, K. W. T., & Fang, Q. (2014). SFP-Rank: significant frequent pattern analysis for effective ranking. *Knowledge and Information Systems, 43*(3), 1–25.

Sun, J., Wang, S., Gao, B. J., & Ma, J. (2012). Learning to rank for hybrid recommendation. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management* (pp. 2239–2242). ACM.

Tsai, M. F., Liu, T. Y., Qin, T., Chen, H. H., & Ma, W. Y. (2007). Frank: a ranking method with fidelity loss. In *SIGIR'07: Proceedings of the 30st annual international ACM SIGIR conference on Research and development in information retrieval* (pp. 383–390).

Tax, N., Bockting, S., & Hiemstra, D. (2015). A cross-benchmark comparison of 87 learning to rank methods. *Information Processing & Management, 51*(6), 757–772.

Wu, Q., Burges, C., Svore, K., & Gao, J. (2010). Adapting boosting for information retrieval measures. *Information Retrieval., 13*(3), 254–270.

Xia, F., Liu, T. Y., Wang, J., Zhang, W., & Li, H. (2008). Listwise approach to learning to rank—theorem and algorithm. In *ICML'08: Proceedings of the 25th Annual International Conference on Machine Learning* (pp. 1192–1199).

Xu, J., & Li, H. (2007). AdaRank: a boosting algorithm for information retrieval. In *SIGIR'07: Proceedings of the 30st annual international ACM SIGIR conference on Research and development in information retrieval* (pp. 391–398).

Xu, J., Liu, T. Y., Lu, M., Li, H., & Ma, W. Y. (2008). Directly optimizing IR evaluation measures in learning to rank. In *SIGIR'08: Proceedings of the 31th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 107–114).

Yilmaz, & Robertson (2010). On the choice of effectiveness measures for learning to rank. *Information Retrieval, 13*(3), 271–290.

Yue, Y. S., Finley, T., & Radlinski, F. (2007). 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* (pp. 271–278).

Zhang, P., Lin, H. F., Lin, Y., & Wu, J. J. (2011). Learning to rank by optimizing expected reciprocal rank. In *AIRS'11: Proceedings of 7th Asia Information Retrieval Symposium (AIRS), LNCS 7097* (pp. 93–102). Springer.