Comput Syst Sci & Eng (2018) 1: 41–52
© 2018 CRL Publishing Ltd

International Journal of
Computer Systems
Science & Engineering

Rank-order-correlation-based feature vector context transformation for learning to rank for information retrieval

Jen-Yuan Yeh*

Dept. of Operation, Visitor Service, Collection and Information Management, National Museum of Natural Science, Taichung 40453, Taiwan
E-mail: jenyuan@mail.nmns.edu.tw

As a crucial task in information retrieval, ranking defines the preferential order among the retrieved documents for a given query. Supervised learning has recently been dedicated to automatically learning ranking models by incorporating various models into one effective model. This paper proposes a novel supervised learning method, in which instances are represented as bags of contexts of features, instead of bags of features. The method applies rank-order correlations to measure the correlation relationships between features. The feature vectors of instances, i.e., the 1st-order raw feature vectors, are then mapped into the feature correlation space via projection to derive the context-level feature vectors, i.e., the 2nd-order context feature vectors. As for ranking model learning, Ranking SVM is employed with the 2nd-order context feature vectors as the input. The proposed method is evaluated using the LETOR benchmark datasets and is found to perform well with competitive results. The results suggest that the learning method benefits from the rank-order-correlation-based feature vector context transformation.

Keywords: Learning to rank; feature correlation extraction; 1st-order raw feature vector representation; 2nd-order context feature vector representation; ranking model; ranking prediction; LETOR

1. INTRODUCTION

In information retrieval (IR), ranking is a crucial task that defines the preferential order among the retrieved documents for a given query. Traditional IR has adopted empirical ranking models, such as the Boolean model, the vector space model, and the probabilistic model, which are designed in *unsupervised* manners [1]. In practice, the aforementioned models usually suffer high costs for parameter tuning, and sometimes overfitting occurs, especially when the models are carefully tuned to fit particular needs [22]. Nowadays, as many IR results are increasingly accompanied by relevance judgments, e.g., query and click-through logs collected in search engines, *supervised* learning methods, referred to as the learning to rank (LTR) methods, have been devoted to automatically learning ranking models (e.g., [5, 6, 12, 14, 18, 45, 53]). In general, supervised learning allows the automatic tuning of parameters and the incorporation of various models into a singular one with high effectiveness.[1]

Figure 1 presents the general paradigm of learning to rank for IR. The learning process consists of two phases, namely, *training* and *test*. First, the training phase is introduced. Given a query collection, i.e., $Q = \{q_1, q_2, \ldots, q_{|Q|}\}$, and a document set, i.e., $D = \{d_1, d_2, \ldots, d_{|D|}\}$, a training instance is a query-document pair, i.e., $(q_i, d_j) \in Q \times D$, upon which a relevance judgment indicating the relationship between q_i and d_j is assigned by a labeler. The relevance judgment can be (1) a class label, e.g., relevant or non-relevant; (2) a rating, e.g., a 3-star rating scaling

*Corresponding Author. Tel.: +886 4 2322 6940x728; fax: +886 4 2323 2621

[1]Learning to rank is defined as having the following properties [22]: (1) the promising capability of combining a large number of features (i.e., ranking models); and (2) the capability of automatic learning based on the training data.

from 0 to 2 for non-relevant, possibly relevant, and definitely relevant; (3) an order, e.g., k, meaning that d_j is ranked at the k-th position in the ordering of documents when q_i is considered; or (4) a score, e.g., $sim(q_i, d_j)$, specifying the degree of relevance between q_i and d_j. For each instance, i.e., (q_i, d_j), a feature extractor produces a vector of features that describes the match between q_i and d_j. Such features can be classical IR models (e.g., term frequency, inverse document frequency, and Okapi BM25 [32]) or newly developed models (e.g., HostRank [51], Feature Propagation [29, 36], and Topical PageRank [26]). The inputs to the learning algorithm comprise training instances, their feature vectors, and the corresponding relevance judgments. The output is a ranking model, f, where $f(q_i, d_j)$ is supposed to give the "true" relevance judgment for q_i and d_j. The learning algorithm attempts to learn a ranking model, such that a performance measure, e.g., classification accuracy, error rate, and Mean Average Precision (MAP) [1], with respect to the output relevance judgments can be optimized. In the test phase, the ranking model is applied to judge the relevance between each document d_i in D and a new query q.

The feature vector model,[2] a.k.a. the bag-of-features model, is widely used for representing instances. The assumptions behind the model include [2]: (1) the independence relationship between features (i.e., each feature is *a priori* independent from the others); (2) the flatness of the feature values (i.e., no hierarchy among the values); and (3) the certainty of the observations (i.e., only one value for each feature). However, empirical observations have found that features, i.e., the ranking models in this study, are not always independent. For example, TF-IDF [1] and Okapi BM25 [32] are considered somewhat correlated since both are designed based on term frequency and inverse document frequency. In such cases, the feature vector model neglects the correlations between features and treats the features as independent coordinate axes.

This paper proposes to model instances as bags of contexts of features, instead of bags of features. The contexts are extracted from the feature correlation space that is built with rank-order correlations to capture the correlation relationships between features. The feature vectors of instances (hereafter called *the 1st-order raw feature vectors*) are mapped into the feature correlation space via projection for deriving the context-level feature vectors (hereafter called *the 2nd-order context feature vectors*). The 2nd-order context feature vectors inherently take into account the correlation relationships between features and are believed to be capable of conquering the limitations for the 1st-order raw feature vectors. A new learning method, which extends Figure 1 by incorporating the 2nd-order context feature vectors and the state-of-the-art learning algorithm, Ranking SVM [18], is also developed.

The rest of this paper is structured as follows. Section 2 presents a brief review of the literature. Section 3 describes the technical details of the proposed method. Section 4 provides the experimental results, and Section 5 concludes this paper and points out possible directions for further research.

[2]Each dimension in the feature vector model corresponds to a ranking (or retrieval) model.

2. RELATED WORK

Previous studies of learning to rank fall into three categories [22]: (1) the point-wise approach; (2) the pair-wise approach; and (3) the list-wise approach.

In the *point-wise* approaches, each training instance is associated with a class or rating. The learning process finds a model that maps instances into classes or rates close to their *true* values. The point-wise approach can be further divided into three subcategories, namely, regression-based (e.g., [11]), classification-based (e.g., [25] and McRank [21]), and ordinal regression-based (e.g., [16], [37] and Pranking [12]). A typical example is Pranking [12], which trains a perceptron model to directly maintain a totally-ordered set via projections. Another one is McRank [21], which defines a 5-star rating and casts the ranking problem as multiple classification in accordance with an observation that perfect classifications lead to perfect DCG (Discounted Cumulative Gain) [17] scores. The classification model is learned via gradient boosting.

The *pair-wise* approaches take pairs of objects and their relative preferences as training instances and learn to classify each object pair as correctly-ranked or incorrectly-ranked. Most existing methods are pair-wise approaches, e.g., Ranking SVM [18], RankBoost [14], and RankNet [5]. Ranking SVM employs support vector machines (SVM) to classify object pairs in consideration of large margin rank boundaries. Both RankBoost and QBrank [57] conduct boosting to find a combined ranking, which minimizes the number of mis-ordered pairs of objects. RankNet defines cross entropy as a probabilistic cost function on object pairs and uses neural network to optimize the cost function. LambdaRank [4] also employs neural network but uses gradient based on NDCG (Normalized Discounted Cumulative Gain) [17] scores smoothed by the RankNet loss (also see LambdaMART [3], the boosted tree version of LambdaRank). FRank [43] adopts Fidelity to measure loss of ranking and uses a generalized additive model to minimize the Fidelity loss. SemiRankSVM [27] extends Ranking SVM by a graph-based regularized algorithm to learn a ranking function that minimizes the least squares ranking loss.

Finally, the *list-wise* approaches use a list of ranked objects as training instances and learn to predict the list of objects. There are two sub-categories which are, respectively, based on the direct optimization of IR evaluation measures (e.g., SoftRank [42], AdaRank [50], and $\text{SVM}^{\triangle}_{\text{map}}$ [56]) and the minimization of listwise ranking losses (e.g., ListNet [6], and ListMLE [48]). Examples are briefed as follows. AdaRank [50], a learning algorithm within the framework of boosting, repeatedly constructs "weak rankers" and finally linearly combines the weak rankers to make ranking predictions. $\text{SVM}^{\triangle}_{\text{map}}$ [56] is a SVM-based learning algorithm that efficiently finds a globally optimal solution to a straightforward relaxation of MAP [1]. ListNet [6] introduces a probabilistic-based list-wise loss function and adopts neural network and gradient descent to train a list prediction model. In ListMLE [48], the likelihood loss is employed as the surrogate for the IR evaluation measures.

More examples are described as follows: [25] treats IR as binary classification of relevance and explores the applicability of discriminative classifiers to solve the problem. [18] takes pairs of documents and their relative preferences derived from

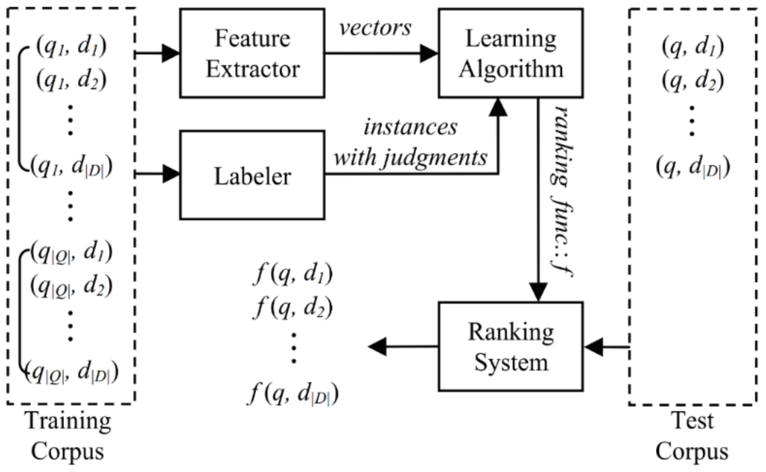

Figure 1 The general paradigm of learning to rank for IR [52].

click-through data (i.e., the log of links that users click on in the presented ranking) as training instances and applies Ranking SVM for learning better retrieval functions. [7] modifies the "Hinge Loss" function in Ranking SVM to consider two essential factors for IR: (1) to have high accuracy on the top-ranked documents, and (2) to avoid training a biased model towards queries with many relevant documents. [49] uses Ranking SVM to address definition search, where the retrieved definitional excerpts of a term are ranked according to their likelihood of being good definitions. [54] extends SVM selecting sampling techniques in classification for learning to rank. [24] proposes a multiple nested ranker approach to re-rank the top scoring documents of the result list, in which RankNet is applied to learn a new ranking at each iteration. RankCosine [30] uses cosine similarity between the ranking list and the ground truth as a query-level loss function. RV-SVM [55] develops the 1-norm Ranking SVM, which is based on 1-norm objective function, for faster training using much less support vectors than the standard Ranking SVM.

Other research directions, which are receiving increasing attention in recent years, include [8]: *online learning to rank* for quickly learning the best re-ranking of the top position of the original ranked list based on real-time user click feedback (e.g., [9, 10, 34, 35]); *large-scale learning to rank* which leverages both the learning theory and computational theory for ranking when facing large-scale training data (e.g., [31, 41, 47]); *learning to rank for diversity* that optimizes not only for relevancy, but also for diversity (i.e., for minimum redundancy) by taking into account document similarity and ranking context (e.g., [33, 38]); and *robust learning to rank* which optimizes the tradeoffs between model effectiveness and robustness for real-world retrieval scenarios (e.g., [20, 46]).

3. THE PROPOSED METHOD

Figure 2 gives an overview of the proposed method, which is essentially an extension of Figure 1. Newly added modules are marked in gray. *Feature Correlation* stores the correlation relationships between features in a matrix, in which the relationships are measured by rank-order correlation coefficients. *Vector Transformation* considers the feature correlation matrix as an intermediary context space for transformation and derives the 2nd-order context feature vectors by projecting the 1st-order raw feature vectors into the space. Last, *Learning Algorithm: Ranking SVM* takes the 2nd-order context feature vectors as the input to train a *linear* binary classifier by Ranking SVM [18] for judging, as regards a particular query, the binary ordering relations between documents.

The details of the proposed method are elaborated in the following subsections. The symbols used are denoted as follows:

- $\boldsymbol{F}$: the feature set. $F = \{f_1, \ldots, f_k\}$, $|F| = k$;
- $\boldsymbol{Q}$: the query set. $Q = \{q_1, \ldots, q_m\}$, $|Q| = m$;
- $\boldsymbol{D}$: the document set. $D = \{d_1, \ldots, d_n\}$, $|D| = n$;
- $\boldsymbol{D(q_i)}$: the retrieved document set for q_i. $D(q_i) = \{d_{i,1}, \ldots, d_{i,n_i}\}$, $D(q_i) \subseteq D$, $|D(q_i)| = n_i$;
- $\boldsymbol{f}$: the ranking model $f(\cdot)$. $f(q, d)$ indicates the relevance judgment for document d with respect to query q.

3.1 Relevance labeling

The relevance labeling annotates instances with proper relevance judgments, which play the role of answers (or observations) that guide the learning algorithm to train an effective ranking model. The labeling scheme in this paper is n-star rating for different levels of relevance. In terms of the 3-star rating, for example, the relevance judgments are quantified as 0 for not relevant, 1 for possibly relevant, and 2 for definitely relevant.

3.2 Feature vector extraction and context transformation

The three following steps are carried out in the process: (1) 1st-order raw feature vector extraction; (2) feature correlation extraction; and (3) 2nd-order context feature vector transformation.

(1) 1st-order raw feature vector extraction

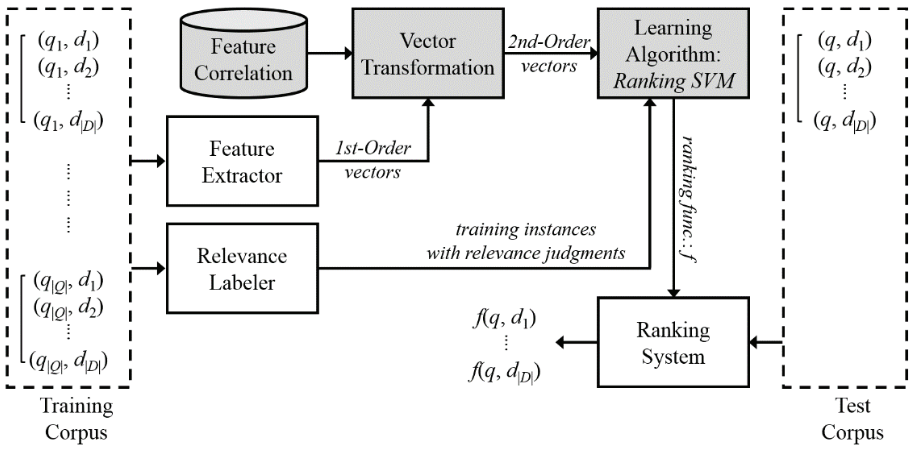

Figure 2 Overview of the proposed method.

With the feature set F comprising a $|F|$-dimensional vector space, each query-document pair $(q_i, d_{i,j})$ is represented as a vector of numerical features:

$$\vec{d}_{i,j} =< w_{i,j,1}, \ldots, w_{i,j,k} >, \tag{1}$$

where $w_{i,j,k} = \text{f\_val}(f_k, q_i, d_{i,j})$ describes the match, regarding f_k, between q_i and $d_{i,j}$. The function, f_val, is a feature extraction function, defined as:

$$\text{f\_val}(\cdot):\{(f_k, q_i, d_{i,j}) | f_k \in F,\ q_i \in Q, d_{i,j} \in D(q_i)\} \rightarrow [0, 1]. \tag{2}$$

Here, $\vec{d}_{i,j}$ is named "*1st-order raw feature vector*," to be distinguishable from "*2nd-order context feature vector*," as will be explained later.

(2) Feature correlation extraction

For query q_i, a document sequence $\check{R}_{i,k}$ can be established by ordering documents in $D(q_i)$ in accordance with their feature values, calculated by Eq. (2), when f_k is considered.

$$\check{R}_{i,k}(D(q_i)) = [r_{i,k}(d_{i,j_l}) \succ \cdots \succ r_{i,k}(d_{i,j_{n_i}})], \tag{3}$$

where $r_{i,k}$ is an order function that maps each document in $D(q_i)$ to an appropriate position in $\check{R}_{i,k}$. For instance, $r_{i,k}(d_{i,j_l}) = l$ implies that d_{i,j_l} is at the l-th position in $\check{R}_{i,k}$. $r_{i,k}(d_{i,p}) \succ r_{i,k}(d_{i,t})$ is satisfied if and only if p and t exist and $\text{f\_val}(f_k, q_i, d_{i,p}) \geq \text{f\_val}(f_k, q_i, d_{i,t})$; in other words, when $\text{f\_val}(f_k, q_i, d_{i,p}) \geq \text{f\_val}(f_k, q_i, d_{i,t})$, $r_{i,k}(d_{i,p}) \succ r_{i,k}(d_{i,t})$ suggests that $d_{i,p}$ is ranked at a higher position in the list than $d_{i,t}$.

Given q_i and $D(q_i)$, k document sequences $\{\check{R}_{i,1}(D(q_i)), \ldots, \check{R}_{i,k}(D(q_i))\}$ can be produced. It becomes possible to assess the correlation between two sequences using rank-order correlation coefficients, such as Pearson's r [28], Spearman's *rho* (ρ) [40], and Kendall's tau (τ) [19]. Here, the correlations between sequences are referred to as the correlations between features with respect to the given query. To get an overall feature correlation in consideration of all queries in Q, a simple strategy is utilized to compute the correlation between f_i and f_j, i.e., $c_{i,j}$, based on the rule of macro-correlation, as presented below:

$$c_{i,j} = \frac{1}{|Q|} \sum_{t}^{|Q|} F\_RCorr(\check{R}_{t,i}(D(q_t)), \check{R}_{t,j}(D(q_t))), \tag{4}$$

where *F_RCorr* is a rank-order correlation coefficient function to measure the rank-order correlation between two document sequences $\check{R}_{t,i}(D(q_t))$ and $\check{R}_{t,j}(D(q_t))$. *F_RCorr* is formulated as:

$$F\_RCorr(\cdot) : \{(\check{R}_{t,i}(D(q_t)), \check{R}_{t,j}(D(q_t))) | q_t \in Q \text{ and } f_i, f_j \in F\} \rightarrow [0, 1]. \tag{5}$$

Finally, a feature correlation matrix, C, is generated by Eq. (6):

$$C = \begin{array}{c} \\ f_1 \\ \vdots \\ f_k \end{array} \begin{array}{c} \begin{array}{ccc} f_1 & \cdots & f_k \end{array} \\ \begin{pmatrix} c_{1,1} & \cdots & c_{1,k} \\ \vdots & \ddots & \vdots \\ c_{k,1} & \cdots & c_{k,k} \end{pmatrix} \end{array} \tag{6}$$

It is recalled that, by Eq. (4), $c_{i,j} = c_{j,i}$ holds. The matrix C is evidently a symmetric matrix so that Eq. (7) is satisfied.

$$C_i = \hat{C}_i, \text{ i.e., } \forall p, \quad c_{i,p} = \hat{c}_{i,p}, \tag{7}$$

where $C_i =< c_{i,1}, \ldots, c_{i,k} >$ is the i-th row vector of C, and $\hat{C}_i =< \hat{c}_{i,1}, \ldots, \hat{c}_{i,k} >$ is the i-th column vector of C. Notably, the row vector (or the column vector) provides a mathematical formalization for f_i, as shown:[3]

$$\begin{aligned} \vec{f_i} &= C_i =< c_{i,1}, \ldots, c_{i,k} > \\ &= \hat{C}_i =< \hat{c}_{i,1}, \ldots, \hat{c}_{i,k} > . \end{aligned} \tag{8}$$

Table 1 lists the rank-order correlation coefficients used for *F_RCorr*. For example, in terms of Kendall's *tau* (τ), values of the coefficient range from -1 (i.e., 100% negative association, or perfect inversion) to $+1$ (i.e., 100% positive association, or perfect agreement). A value of 0 indicates the absence of association. Considering that correlation coefficients may have a different range of values (e.g., Kendall's *tau* (τ) versus Kendall tau distance), the comparison between them in such cases may lead to insignificant contrast. Therefore, each coefficient value is further normalized into a common range of [0, 1] for fair comparisons, where 0 represents no association and $+1$ represents perfect agreement.

[3]Methods of feature clustering and feature selection can be developed based on the context formalizations of features.

Table 1 The rank-order correlation coefficients used for *F_RCorr*.

ID	Rank-order correlation coefficient	Range of values before normalization	Normalization method	Range of values after normalization
G	Goodman and Kruskal's Gamma (G) [15]	$[-1,+1]$ where -1: perfect inversion; 0: no association; and $+1$: perfect agreement.	Set values in $[-1,0)$ to 0.	$[0, +1]$ where 0: no association; and $+1$: perfect agreement.
K1	Kendall's *tau* (τ) [19]			
P	Pearson's r [28]			
SD	Somer's d [39]			
SR	Spearman's *rho* (ρ) [40]			
K2	Kendall tau distance [19]	$[0, +1]$ where 0: perfect agreement; and $+1$: perfect inversion.	1. Set value x to a new value $1-x$ 2. Set values in $[0, 0.5)$ to 0 3. Normalize values in $[0.5, 1]$ to $[0, 1]$ by min-max normalization.	$[0, +1]$ where 0: no association; and $+1$: perfect agreement.

In consideration of one rank-order correlation coefficient used for *F_RCorr*, there are in total $|Q|$ feature correlation values for f_i and f_j, according to Eq. (5). In the implementation, the distribution of $|Q|$ feature correlation values is found to be not concentrated, which might cause the distortion of $c_{i,j}$. Thus, Box-and-Whisker Plot [44], a.k.a. Boxplot, is employed to filter out potential outliers. The outlier detection process is detailed as follows. First, all feature correlation values are arranged in sequence from low to high. Then, the value of the 25th percentile and the value of the 75th percentile are defined as Q_1 and Q_3, respectively. The interquartile range, $\text{IQR} = Q_3 - Q_1$, is computed. Finally, $c_{i,j}$ is calculated as the mean of feature correlation values in range of $[Q_1 - 1.5 \times \text{IQR}, Q_3 + 1.5 \times \text{IQR}]$. Note that feature correlation values outside the range are treated as outliers and are discarded during the computation of Eq. (4).

(3) 2nd-order context feature vector transformation.

In contrast to the 1st-order raw feature vector introduced in Eq. (1), i.e., $\vec{d}_{i,j} =< w_{i,j,1}, \ldots, w_{i,j,k} >$, the 2nd-order context feature vector of $\vec{d}_{i,j}$ is specified as:

$$\vec{d}^{(2)}_{i,j} =< w^{(2)}_{i,j,1}, \ldots, w^{(2)}_{i,j,k} >, \tag{9}$$

where $w^{(2)}_{i,j,k}$ is transformed from $w_{i,j,k}$ based on the feature correlation matrix C.

The proposed 2nd-order context transformation method, named as *Latent Semantic Analysis Based*, applies latent semantic analysis (LSA) [13, 23] to project $\vec{d}_{i,j}$ into a space with latent semantic dimensions that is derived from the feature correlation matrix C. The transformation process consists of three steps which are in sequence *singular value decomposition, dimensionality reduction*, and *folding-in*. Firstly, singular value decomposition (SVD) is performed on the feature correlation matrix C. The SVD of C is defined as $C = USV^T$, where U is a $k \times k$ column-orthonormal matrix of left singular vectors in columns; S is a matrix with singular values $(s_1, \ldots, s_k)$ sorted in descending order in diagonal and zeros elsewhere; and V is a $k \times k$ orthonormal matrix of right singular vectors in columns. Suppose that the rank of C is p, S satisfies $s_1 \geq s_2 \geq \ldots \geq s_p > s_{p+1} = \cdots = s_k = 0$. Dimensionality reduction follows to keep only $z (z < p)$ singular values of S for obtaining a $z \times z$ matrix C_z. Note that C_z is an approximation of C, i.e., $C_z = U_z S_z V_z^T \approx C$, in which S_z represents the latent semantic structure derived from C. Finally, folding-in folds $\vec{d}_{i,j}$ into the latent semantic space S_z to obtain $\vec{d}^{(2)}_{i,j}$ by Eq. (10):

$$\vec{d}^{(2)}_{i,j} = \vec{d}^{T}_{i,j} U_z S_z^{-1}, \tag{10}$$

It is worth noting that by dimensionality reduction related features are mapped onto the same dimensions of the reduced space S_z and unrelated features are mapped onto different dimensions. This operation reflects a grouping of features into z linearly-independent base vectors, i.e., the contexts of features in this study. It can be said that the dimensions of the reduced space correspond to the axes of greatest variation [23]. Thus, folding $\vec{d}_{i,j}$ into the latent semantic space S_z means to represent $\vec{d}_{i,j}$ by these base vectors.

3.3 Ranking model learning and ranking prediction

To learn the ranking model, Ranking SVM [18] is employed[4] since previous studies have already demonstrated its feasibility and effectiveness. The 2nd-order context feature vectors of query-document pairs are viewed as instances. Pairs of instances and their relative preferences are inputted into Ranking SVM. The algorithm targets binary ordering relations between documents with respect to a query and tries to learn a model that can minimize the number of discordant pairs based on the observed parts of the target ranking. In the implementation, SVM^{rank}, a toolkit for efficiently training Ranking SVMs, is used.[5]

The output ranking model is a linear binary classifier, which is capable of determining whether a pair of documents is in concordant order. As for ranking prediction, given a new query q and its retrieved document set $D(q)$, the output of the ranking model comprises binary ordering relations between documents, according to which the final ordering of documents in $D(q)$ can be established.

[4]The learning algorithm inside the proposed method is not limited to Ranking SVM. Other learning algorithms that represent query-document pairs as vectors of features can be integrated. This issue is left for future work.

[5]Available at http://www.cs.cornell.edu/people/tj/svm_light/svm_rank.html. Note that SVM^{rank} learns an unbiased classification rule using linear kernel.

4. EXPERIMENTS

This section describes the datasets and evaluation measures, and reports the preliminary experimental results.

4.1 The LETOR benchmark datasets

The LETOR 3.0 and LETOR 4.0 benchmark datasets[6] are used to evaluate the effectiveness of the proposed method. The LETOR datasets are created as query-document pairs, each containing a feature vector and its corresponding relevance judgment. The 5-fold partitions are provided for cross-validation. In each fold, three subsets are used for learning, one subset for validation, and the other one for testing. The proposed method is tested on the TD2003, TD2004, and MQ2008 datasets, the statistics of which are listed in Table 2. Table 3 provides an illustration of the sample data, with each row standing for a query-document pair.

4.2 Evaluation measures

The standard P@n, NDCG@n, and MAP measures are used in the evaluation.

(1) Precision at position n (P@n) [1]

For a given query, its precision of the top n results of the ranking list is defined as:

$$\text{P@}n = \frac{\text{\# of relevant documents in top } n \text{ results}}{n}. \tag{11}$$

Note that, when computing P@n, a document with the relevance judgment of either *definitely* or *possibly relevant* is regarded as a document relevant to the given query. The mean P@n is reported by averaging the P@n values of all queries.

(2) Mean Average Precision (MAP) [1]

For a given query, its average precision, AP, is computed by Eq. (12), where N is the number of retrieved documents and $rel(n)$ is either 1 or 0, indicating whether the n-th document is relevant to the query or not. The MAP is obtained as the mean average precision over a set of queries.

$$AP = \frac{\sum_{n=1}^{N} \text{P@}n \times rel(n)}{\text{\# of relevant documents for the query}}. \tag{12}$$

(3) Normalized Discounted Cumulative Gain (NDCG) [17]

For a query, the NDCG of its ranking list at position n is calculated by:

$$\text{NDCG@}n = Z_n \sum_{j=1}^{n} \begin{cases} 2^{r(j)} - 1, & j = 1 \\ \frac{2^{r(j)}-1}{\log(j)}, & j > 1 \end{cases}, \tag{13}$$

where $r(j)$ is the rating of the j-th document in the list, and the normalization constant Z_n is set so that the perfect list receives an NDCG of 1. The $r(j)$ is set to the relevance judgment, i.e., 2 when the j-th document is definitely relevant to the query, 1 when the j-th document is possibly relevant to the query, and 0 when the j-th document is irrelevant to the query. The mean NDCG@n is reported by averaging the NDCG@n values of all queries.

[6]Available at http://research.microsoft.com/en-us/um/beijing/projects/letor/default.aspx.

4.3 Experimental settings

In the experiments, 5-fold cross-validation is conducted and the average score is reported. For each fold, the training set is first used to learn a ranking model. The feature correlation matrix (see Eq. (6)) is also built using the training set. The validation set is used for tuning model parameters, and the ranking model is then applied on the testing set. The standard LETOR evaluation tools are used in order to avoid differences in the evaluation results caused by different implementations of the evaluation measures.

Two types of parameters need to be determined, namely, the z value for dimensionality reduction in the latent-semantic-analysis-based 2nd-order context transformation method, and the SVM^{rank} learning-specific options. For the z value, a naïve method that sets z with a reduction ratio is adopted. As an example, supposing that the rank of the feature correlation matrix C is p, z is set to $0.2 \times p$ when a reduction ratio of 20% is considered. The possible ratio takes the values (10%, 20%, ..., 90%). For each fold, the ratio with the best MAP performance on the validation set is selected and its performance on the testing set is reported. The SVM^{rank} learning-specific options are set with "−c <C> −e 0.001 −l 1," where <C>, the trade-off between training error and margin, takes the values (0.00001, 0.00002, 0.00005, 0.0001, 0.0002, 0.0005, 0.001, 0.002, 0.005, 0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1, 2, 5, 10). Similarly, for each fold, the <C> value with the best MAP performance on the validation set is selected and its performance on the testing set is reported.

4.4 Results

The evaluation results are given in Tables 4–12. The baseline, RankSVM-Struct [7] [18], takes the 1st-order raw feature vectors as the input. Various models of the proposed method are examined and indicated in forms of {G, K1, P, SD, SR, K2}-L, where the previous part denotes the rank-order correlation coefficient used for *F_RCorr* (see Table 1) and the later part specifies the proposed 2nd-order context transformation method, i.e., Latent Semantic Analysis Based (see Section 3.2). For example, the model K1-L employs "Kendall's *tau* (τ)" for measuring the feature correlations and "Latent Semantic Analysis Based" for performing the 2nd-order context transformation. The values in the parentheses suggest the relative improvements of the proposed method when being compared with RankSVM-Struct. Last, in each column, the best performance is given in bold.

Tables 4–6 list the results on TD2003. It can be seen that the proposed models significantly outperform RankSVM-Struct in terms of P@[1, 3] and NDCG@[1, 3]. As for P@[5, 10] and NDCG@[5, 10], some of the proposed models generate

[7]The literature has denoted SVM^{rank}, which is a particular implementation of Ranking SVM, as RankSVM-Struct.

Table 2 Statistics of the evaluation datasets.

	TD2003	TD2004	MQ2008
No. of queries	50	75	784
No. of documents	47,240	70,097	14,384
No. of query-document pairs	49,058	74,146	15,211
No. of features	64	64	46
Possible relevance judgments	0: not relevant; and 1: relevant	0: not relevant; and 1: relevant	2: definitely relevant; 1: possibly relevant; and 0: not relevant
Avg. no. of instances in the training set	29,435	44,488	9,127
Avg. no. of instances in the validation set	9,812	14,829	3,042
Avg. no. of instances in the testing set	9,812	14,829	3,042

Table 3 Sample data excerpted from the MQ2008 dataset.

Relevance label	Query	f_1	f_2	...	f_{46}	Note
2	qid:10032	1:0.056537	2:0.000000	...	46:0.076923	#doc: GX029-35-5894638
0	qid:10032	1:0.279152	2:0.000000	...	46:1.000000	#doc: GX030-77-6315042
0	qid:10032	1:0.130742	2:0.000000	...	46:1.000000	#doc: GX140-98-13566007
1	qid:10032	1:0.593640	2:1.000000	...	46:0.000000	#doc: GX256-43-0740276
...	...	...	...	...	...	...

worse results than RankSVM-Struct; for instance, G-L has decreases of 8.7% and 2.22%, regarding P@5 and NDCG@5, respectively. When considering MAP, Table 6 shows that the proposed models are superior to RankSVM-Struct, except that G-L and P-L have slight decreases of 0.26% and 0.22%, respectively. The maximum and minimum increases of improvement are 6.3% (for SD-L) and 2.8% (for SR-L). The average MAP of the proposed models is 0.2785, indicating an increase of 2.65%, compared with RankSVMStruct. In Table 6, the MAP scores of several representative baselines, including ListNet [6], AdaRank [50] (in two versions, namely, AdaRank-NDCG and AdaRank-MAP), and RankBoost [14], are also provided. Evidently, the proposed method performs well with competitive results. The best model, SD-L, for example, outperforms ListNet, AdaRank-MAP, AdaRank-NDCG, and RankBoost with increases of 4.76%, 26.33%, 21.79%, and 26.82%, respectively.0.

Tables 7-9 list the results on TD2004. For all measures, P-L is observed as having significant improvements when compared with RankSVM-Struct. Both SR-L and K2-L have satisfying improvements when P@[1, 3] and NDCG@[1, 3] are considered. Other models, namely, G-L, K1-L, and SD-L, have worse results than RankSVM-Struct. As for MAP, Table 9 indicates that the proposed models are superior to RankSVM-Struct. The maximum and minimum increases of improvement are 6.01% (for P-L) and 2.41% (for K1-L), respectively. The average MAP of the proposed models is 0.2286, implying an increase of 4.11%, compared to RankSVM-Struct. Again, the proposed method is found to be competitive in comparison to other baselines. For instance, the best model, P-L, outperforms ListNet, AdaRank-MAP, and AdaRank-NDCG with increases of 4.35%, 6.35%, and 20.25%, respectively. Unfortunately, none of the proposed models could beat RankBoost.

Tables 10–12 list the results on MQ2008. The proposed models are observed as having better performance than RankSVM-Struct, as regards P@1 and NDCG@1. Some statistics are as follows. The increases of improvement for K1-L are 4.47% for P@1 and 4.08% for NDCG@1. As for P@[3, 5, 10] and NDCG@[3, 5, 10], the proposed models do not work well as expected. Some models obtain slight increases of improvement, while others perform with worse results. Regarding MAP, Table 12 suggests that although the proposed models outperform RankSVM-Struct, the improvements are not significant enough, except for K2-L, which has an increase of 1.21%. The average MAP of the proposed models is 0.4725, denoting an increase of 0.62%, compared to RankSVM-Struct. However, none of the proposed models perform better than the other baselines.

Table 13 lists the upper-bound results of the proposed models on MQ2008. The results are obtained by the following steps. First, ranking models are trained with all possible combinations of parameters. For each fold, 171 ($9 \times 19 = 171$; 9 values for z and 19 values for <C> in Section 4.3) ranking models are produced. Second, all the models are evaluated using the testing set and the best model is picked.[8] Finally, the scores of the best model are reported. Table 13 shows that with the proper parameters, the proposed models can perform better than RankSVM-Struct with significant increases of improvement. It is conjectured that the testing sets and the validation sets in MQ2008 have diverse properties. In such a case, the use of the validation set fails to select a good model for the testing set, which might explain why the proposed method leads to insignificant improvements compared to RankSVM-Struct when it is evaluated on MQ2008 (see Table 12).

Overall, the proposed method behaves differently on differ-

[8] Since the picked model is with the best parameters that are directly optimized using the testing set, the results in Table 13 are regarded as the upper-bound results.

Table 4 P@[1, 3, 5, 10] on TD2003.

Model	P@1	P@3	P@5	P@10
RankSVM-Struct	0.3400	0.2867	**0.2760**	0.1860
G-L	**0.4200** (+23.53%)	0.3067 (+6.98%)	0.2520 (−8.70%)	0.1920 (+3.23%)
K1-L	0.4000 (+17.65%)	**0.3133** (+9.28%)	0.2480 (−10.14%)	0.1960 (+5.38%)
P-L	0.4000 (+17.65%)	0.2867 (0.00%)	0.2360 (−14.49%)	0.1780 (−4.30%)
SD-L	0.3800 (+11.76%)	0.3000 (+4.64%)	**0.2760** (0.00%)	0.1980 (+6.45%)
SR-L	0.4000 (+17.65%)	0.3000 (+4.64%)	0.2520 (−8.70%)	0.1860 (0.00%)
K2-L	0.3570 (+5.00%)	0.3015 (+5.16%)	0.2380 (−13.77%)	**0.2023** (+8.76%)

Table 5 NDCG@[1, 3, 5, 10] on TD2003.

Model	NDCG@1	NDCG@3	NDCG@5	NDCG@10
RankSVM-Struct	0.3400	0.3430	0.3654	0.3467
G-L	**0.4200** (+23.53%)	0.3652 (+6.47%)	0.3573 (−2.22%)	0.3554 (+2.51%)
K1-L	0.4000 (+17.65%)	0.3700 (+7.87%)	0.3517 (−3.75%)	0.3594 (+3.66%)
P-L	0.4000 (+17.65%)	0.3499 (+2.01%)	0.3381 (−7.47%)	0.3360 (−3.09%)
SD-L	0.3800 (+11.76%)	**0.3703** (+7.96%)	**0.3784** (+3.56%)	**0.3698** (+6.66%)
SR-L	0.4000 (+17.65%)	0.3628 (+5.77%)	0.3577 (−2.11%)	0.3539 (+2.08%)
K2-L	0.3570 (+5.00%)	0.3650 (+6.41%)	0.3489 (−4.52%)	0.3692 (+6.49%)

Table 6 MAP on TD2003.

Model	MAP
RankSVM-Struct	0.2713
G-L	0.2706 (−0.26%)
K1-L	0.2812 (+3.65%)
P-L	0.2707 (−0.22%)
SD-L	**0.2884** (+6.30%)
SR-L	0.2789 (+2.80%)
K2-L	0.2811 (+3.61%)
Avg. MAP of the proposed models	0.2785 (+2.65%)
ListNet	0.2753
AdaRank-MAP	0.2283
AdaRank-NDCG	0.2368
RankBoost	0.2274

Table 7 P@[1, 3, 5, 10] on TD2004.

Model	P@1	P@3	P@5	P@10
RankSVM-Struct	0.3467	0.3333	0.2960	0.2560
G-L	0.2933 (−15.40%)	0.3244 (−2.67%)	0.2880 (−2.70%)	0.2573 (+0.51%)
K1-L	0.3200 (−7.70%)	0.3067 (−7.98%)	0.2747 (−7.20%)	0.2560 (0.00%)
P-L	0.3600 (+3.84%)	**0.3556** (+6.69%)	**0.3040** (+2.70%)	**0.2627** (+2.62%)
SD-L	0.3067 (−11.54%)	0.3156 (−5.31%)	0.2907 (−1.79%)	0.2600 (+1.56%)
SR-L	0.4000 (+15.37%)	0.3333 (0.00%)	0.2907 (−1.79%)	0.2533 (−1.05%)
K2-L	**0.4133** (+19.21%)	0.3467 (+4.02%)	0.2933 (−0.91%)	0.2560 (0.00%)

Table 8 NDCG@[1, 3, 5, 10] on TD2004.

Model	NDCG@1	NDCG@3	NDCG@5	NDCG@10
RankSVM-Struct	0.3467	0.3371	0.3192	0.3090
G-L	0.2933 (−15.40%)	0.3232 (−4.12%)	0.3059 (−4.17%)	0.3042 (−1.55%)
K1-L	0.3200 (−7.70%)	0.3123 (−7.36%)	0.2970 (−6.95%)	0.3003 (−2.82%)
P-L	0.3600 (+3.84%)	**0.3624** (+7.51%)	**0.3336** (+4.51%)	**0.3207** (+3.79%)
SD-L	0.3067 (−11.54%)	0.3131 (−7.12%)	0.3033 (−4.98%)	0.3032 (−1.88%)
SR-L	0.4000 (+15.37%)	0.3389 (+0.53%)	0.3160 (−1.00%)	0.3078 (−0.39%)
K2-L	**0.4133** (+19.21%)	0.3579 (+6.17%)	0.3260 (+2.13%)	0.3150 (+1.94%)

Table 9 MAP on TD2004.

Model	MAP
RankSVM-Struct	0.2196
G-L	0.2252 (+2.55%)
K1-L	0.2249 (+2.41%)
P-L	**0.2328** (+6.01%)
SD-L	0.2259 (+2.87%)
SR-L	0.2303 (+4.87%)
K2-L	0.2326 (+5.92%)
Avg. MAP of the proposed models	0.2286 (+4.11%)
ListNet	0.2231
AdaRank-MAP	0.2189
AdaRank-NDCG	0.1936
RankBoost	0.2614

Table 10 P@[1, 3, 5, 10] on MQ2008.

Model	P@1	P@3	P@5	P@10
RankSVM-Struct	0.4273	0.3903	**0.3474**	**0.2491**
G-L	0.4298 (+0.59%)	0.3839 (−1.64%)	0.3469 (−0.14%)	0.2476 (−0.60%)
K1-L	**0.4464** (+4.47%)	0.3890 (−0.33%)	0.3449 (−0.72%)	0.2472 (−0.76%)
P-L	0.4336 (+1.47%)	0.3810 (−2.38%)	0.3472 (−0.06%)	0.2451 (−1.61%)
SD-L	0.4299 (+0.61%)	**0.3907** (+0.10%)	0.3454 (−0.58%)	0.2484 (−0.28%)
SR-L	0.4388 (+2.69%)	0.3890 (−0.33%)	0.3436 (−1.09%)	0.2454 (−1.49%)
K2-L	0.4451 (+4.17%)	0.3822 (−2.08%)	0.3436 (−1.09%)	0.2477 (−0.56%)

Table 11 NDCG@[1, 3, 5, 10] on MQ2008.

Model	NDCG@1	NDCG@3	NDCG@5	NDCG@10
RankSVM-Struct	0.3627	0.4286	0.4695	0.2279
G-L	0.3618 (−0.25%)	0.4277 (−0.21%)	0.4729 (+0.72%)	0.2269 (−0.44%)
K1-L	**0.3775** (+4.08%)	0.4304 (+0.42%)	0.4719 (+0.51%)	0.2261 (−0.79%)
P-L	0.3622 (−0.14%)	0.4242 (−1.03%)	0.4729 (+0.72%)	0.2248 (−1.36%)
SD-L	0.3635 (+0.22%)	0.4316 (+0.70%)	**0.4736** (+0.87%)	**0.2283** (+0.18%)
SR-L	0.3665 (+1.05%)	**0.4331** (+1.05%)	0.4719 (+0.51%)	0.2254 (−1.10%)
K2-L	0.3771 (+3.97%)	0.4255 (−0.72%)	0.4712 (+0.36%)	0.2263 (−0.70%)

Table 12 MAP on MQ2008.

Model	MAP
RankSVM-Struct	0.4696
G-L	0.4707 (+0.23%)
K1-L	0.4739 (+0.92%)
P-L	0.4705 (+0.19%)
SD-L	0.4713 (+0.36%)
SR-L	0.4734 (+0.81%)
K2-L	**0.4753** (+1.21%)
Avg. MAP of the proposed models	0.4725 (+0.62%)
ListNet	0.4775
AdaRank-MAP	0.4764
AdaRank-NDCG	0.4824
RankBoost	0.4775

ent datasets. Similar phenomena happen to the other baselines. With regard to MAP, the proposed method is found to be superior to RankSVM-Struct with significant increases of improvement on TD2003 and TD2004 and a slight increase of improvement on MQ2008 (see Table 6, Table 9, and Table 12). The results suggest that the learning method benefits from the rank-order-correlation-based feature vector context transformation, which attempts to represent instances as vectors of contexts that take

Table 13 Upper-bound performance of P@1, NDCG@1, and MAP on MQ2008.

Model	P@1	NDCG@1	MAP
RankSVM-Struct	0.4273	0.3627	0.4696
G-L	0.4502 (+5.36%)	0.3814 (+5.16%)	0.4778 (+1.75%)
K1-L	0.4477 (+4.77%)	0.3813 (+5.13%)	0.4765 (+1.47%)
P-L	0.4438 (+3.86%)	0.3716 (+2.45%)	0.4760 (+1.36%)
SD-L	0.4464 (+4.47%)	0.3733 (+2.92%)	0.4765 (+1.47%)
SR-L	0.4553 (+6.55%)	0.3814 (+5.16%)	0.4779 (+1.77%)
K2-L	**0.4617** (+8.05%)	**0.3894** (+7.36%)	**0.4813** (+2.49%)

into account the correlation relationships between features. The proposed method is also observed as having good performance in terms of P@1 and NDCG@1, implying that the ranking model trained by the proposed method tends to rank the relevant document in the first place of the retrieved list. Furthermore, according to MAP, we can rank the proposed models in sequence as SD-L > K1-L > K2-L > SR-L > P-L > G-L for TD2003, P-L > K2-L > SR-L > SD-L > G-L > K1-L for TD2004, and K2-L > K1-L > SR-L > SD-L > G-L > P-L for MQ2008. Then, a final ranking of the proposed models can be built as K2-L > SD-L > K1-L = SR-L > P-L > G-L, according to the average rank-order of every model.

5. CONCLUSION AND FUTURE WORK

In this paper, a novel supervised learning method for learning to rank is developed. The method proposes to model instances as bags of contexts of features, instead of as bags of features, i.e., vectors of features, which most supervised learning methods adopt. It applies rank-order correlations to measure the correlation relationships between features. The feature vectors of instances, i.e., the 1st-order raw feature vectors, are then mapped into the feature correlation space via projection to derive the context-level feature vectors, i.e., the 2nd-order context feature vectors. The following six rank-order correlation coefficients are considered for feature correlation extraction (see Table 1): Goodman and Kruskal's Gamma (G), Kendall's *tau* (τ), Pearson's r, Somer's d, Spearman's *rho* (ρ), and Kendall tau distance. One 2nd-order context transformation method is proposed, i.e., Latent Semantic Analysis Based (see Section 3.2), which produces the 2nd-order context feature vector by directly folding the 1st-order raw feature vector into the latent semantic space of the feature correlation matrix. In terms of ranking model learning, Ranking SVM is employed with the 2nd-order context feature vectors as the input. The proposed method is evaluated using the LETOR benchmark datasets and is found to perform well with competitive results. The results suggest that the learning method benefits from the rank-order-correlation-based feature vector context transformation.

Future work will continue to investigate the effectiveness of the proposed method by introducing other rank-order correlation coefficients for feature correlation extraction and different projection techniques for the 2nd-order context transformation. Another interesting objective is to test other learning algorithms by incorporating the 2nd-order context feature vectors. Lastly, considering that a row vector (or a column vector) of the feature correlation matrix provides a context-level mathematical formalization for feature f_i, it would be beneficial to design feature selection or feature clustering technologies for detecting redundant features based on the context formalizations of features.

Acknowledgements

This work was supported by the Ministry of Science and Technology, Taiwan (Grant Num.: MOST 102-2218-E-178-001 and MOST 103-2221-E-178-001).

REFERENCES

1. Baeza-Yates, R., & Ribeiro-Neto, B. (2011). Modern Information Retrieval: The Concepts and Technology behind Search (2nd ed.). Boston, MA: Addison-Wesley Professional.
2. Basili, R., Pazienza, M.T., & Zanzotto, F.M. (2003). Exploiting the Feature Vector Model for Learning Linguistic Representations of Relational Concepts. Proceedings of ECML/PKDD-2003 Workshop on Adaptive Text Extraction and Mining (ATEM 2003), Cavtat-Dubrovnik, Croatia, pp. 2–9.
3. Burges, C.J.C. (2010). From RankNet to LambdaRank to LambdaMART: An Overview. Microsoft Research Technical Report (MSR-TR-2010-82). Available at https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/MSR-TR-2010-82.pdf.
4. Burges, C.J.C., Ragno, R., & Le, Q.V. (2007). Learning to Rank with Nonsmooth Cost Functions. In P.B. Scholkopf, J.C. Platt, & T. Hoffman (Eds.), Advances in Neural Information Processing Systems 19 (pp. 193–200). Cambridge, MA: MIT Press.
5. Burges, C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N., & Hullender, G. (2005). Learning to Rank Using Gradient Descent. Proceedings of the 22nd International Conference on Machine Learning (ICML 2005), Bonn, Germany, pp. 89–96.
6. Cao, Z., Qin, T., Liu, T.-Y., Tsai, M.-F., & Li, H. (2007). Learning to Rank: From Pairwise Approach to Listwise Approach. Proceedings of the 24th International Conference on Machine Learning (ICML 2007), Corvallis, OR, pp. 129–136.
7. Cao, Y., Xu, J., Liu, T.-Y., Li, H., Huang, Y., & Hon, H.-W. (2006). Adaptive Ranking SVM to Document Retrieval. Proceedings of the 29th Annual International ACM SIGIR Conference on Research

and Development in Information Retrieval (SIGIR 2006), Seattle, WA, pp. 186–193.

8. Chapelle, O., Chang, Y., & Liu, T.-Y. (2011). Future Directions in Learning to Rank. Proceedings of the Learning to Rank Challenge, Haifa, Israel, pp. 91–100.
9. Chaudhuri, S. (2016). Learning to Rank: Online Learning, Statistical Theory and Applications. PhD Thesis in Statistics, University of Michigan, Ann Arbor, MI.
10. Chen, Y., & Hofmann, K. (2015). Online Learning to Rank: Absolute vs. Relative. Proceedings of the 24th International Conference on World Wide Web (WWW 2015), Florence, Italy, pp. 19–20.
11. Cossock, D., & Zhang, T. (2006). Subset Ranking Using Regression. Proceedings of the 19th Annual Conference on Learning Theory (COLT 2006), Pittsburgh, PA, pp. 605–619.
12. Crammer, K., & Singer, Y. (2002). Pranking with Ranking. In T.G. Dietterich, S. Becker, & Z. Ghahramani (Eds.), Advances in Neural Information Processing Systems 14 (pp. 641–647). Cambridge, MA: MIT Press.
13. Deerwester, S., Dumais, S.T., Furnas, G.W., Landauer, T.K., & Harshman, R. (1990). Indexing by Latent Semantic Analysis. Journal of the American Society for Information Science, 41(6) 391–407.
14. Freund, Y., Iyer, R., Schapire, R.E., & Singer, Y. (2003). An Efficient Boosting Algorithm for Combining Preferences. Journal of Machine Learning Research, 4(Nov) 933–969.
15. Goodman, L.A., & Kruskal, W.H. (1954). Measures of Association for Cross Classification. Journal of the American Statistical Association, 49(268) 732–764.
16. Herbrich, R., Graepel, T., & Obermayer, K. (2000). Large Margin Rank Boundaries for Ordinal Regression. In A.J. Smola, P.L. Bartlett, B. Scholkopf, & D. Schuurmans (Eds.), Advances in Large Margin Classifiers, (pp. 115–132). Cambridge, MA: MIT Press.
17. Jarvelin, K., & Kekalainen, J. (2002). Cumulated Gain-Based Evaluation of IR Techniques. ACM Transactions on Information Systems, 20(4) 422–446.
18. Joachims, T. (2002). Optimizing Search Engines Using Clickthrough Data. Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2002), Edmonton, AB, Canada, pp. 133–142.
19. Kendall, M.G. (1938). A New Measure of Rank Correlation. Biometrika, 30(1–2) 81–93.
20. Li, J., Liu, G., Yan, C., & Jiang, C. (2016). Robust Learning to Rank Based on Portfolio Theory and AMOSA Algorithm. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 47(6), 1007–1018.
21. Li, P., Burges, C.J.C., & Wu, Q. (2008). McRank: Learning to Rank Using Multiple Classification and Gradient Boosting. In J.C. Platt, D. Koller, Y. Singer, & S.T. Roweis (Eds.), Advances in Neural Information Processing Systems 20 (pp. 897–904). Red Hook, NY: Curran Associates, Inc.
22. Liu, T.-Y. (2011). Learning to Rank for Information Retrieval. Heidelberg, Germany: Springer.
23. Manning, C.D., & Schutze, H. (1999). Foundations of Statistical Natural Language Processing. Cambridge, MA: MIT Press.
24. Matveeva, I., Burges, C., Burkard, T., Laucius, A., & Wong, L. (2006). High Accuracy Retrieval with Multiple Nested Ranker. Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2006), Seattle, WA, pp. 437–444.
25. Nallapati, R. (2004). Discriminative Models for Information Retrieval. Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2004), Sheffield, United Kingdom, pp. 64–71.
26. Nie, L., Davison, B.D., & Qi, X. (2006). Topical Link Analysis for Web Search. Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2006), Seattle, WA, pp. 91–98.
27. Pan, Z., You, X., Chen, H., Tao, D., & Pang, B. (2013). Generalization Performance of Magnitude-Preserving Semi-Supervised Ranking with Graph-Based Regularization. Information Sciences, 221 284–296.
28. Pearson, K. (1895). Note on Regression and Inheritance in the Case of Two Parents. Proceedings of the Royal Society of London, 58 240–242.
29. Qin, T., Liu, T.-Y., Zhang, X.-D., Chen, Z., & Ma, W.-Y. (2005). A Study of Relevance Propagation for Web Search. Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2005), Salvador, Brazil, pp. 408–415.
30. Qin, T., Zhang, X.-D., Tsai, M.-F., Wang, D.-S., Liu, T.-Y., & Li, H. (2008). Query-Level Loss Functions for Information Retrieval. Information Processing & Management, 44(2) 838–855.
31. Raykar, V.C., Duraiswami, R., & Krishnapuram, B. (2008). A Fast Algorithm for Learning a Ranking Function from Large-Scale Data Sets. IEEE Transactions on Pattern Analysis and Machine Intelligence, 30(7) 1158–1170.
32. Robertson, S.E. (1997). Overview of the Okapi Projects. Journal of Documentation, 53(1) 3–7.
33. Santos, R.L.T., Macdonald, C., & Ounis, I. (2011). On the Suitability of Diversity Metrics for Learning-to-Rank for Diversity. Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2011), Beijing, China, pp. 1185–1186.
34. Schuth, A., Hofmann, K., Whiteson, S., & de Rijke, M. (2013). Lerot: An Online Learning to Rank Framework. Proceedings of the 2013 Workshop on Living Labs for Information Retrieval Evolution (LivingLab 2013), San Francisco, CA, pp. 23–26.
35. Schuth, A., Oosterhuis, H., Whiteson, S., & de Rijke, M. (2016). Multileave Gradient Descent for Fast Online Learning to Rank. Proceedings of the 9th ACM International Conference on Web Search and Data Mining (WSDM 2016), San Francisco, CA, pp. 457–466.
36. Shakery, A., & Zhai, C.X. (2003). Relevance Propagation for Topic Distillation UIUC TREC 2003 Web Track Experiments. Proceedings of the 12th Text Retrieval Conference (TREC 2003), Gaithersburg, MD, pp. 673–677.
37. Shashua, A., & Levin, A. (2003). Ranking with Large Margin Principle: Two Approaches. In S. Becker, S. Thrun, & K. Obermayer (Eds.), Advances in Neural Information Processing Systems 15 (pp. 961–968). Cambridge, MA: MIT Press.
38. Slivkins, A., Radlinski, F., & Gollapudi, S. (2013). Ranked Bandits in Metric Spaces: Learning Diverse Rankings over Large Document Collections. Journal of Machine Learning Research, 14(Feb) 399–436.
39. Somers, R.H. (1962). A New Asymmetric Measure of Association for Ordinal Variables. American Sociological Review, 27(6) 799–811.
40. Spearman, C. (1904). The Proof and Measurement of Association between Two Things. American Journal of Psychology, 15(1) 72–101.
41. Tax, N. (2014). Scaling Learning to Rank to Big Data: Using MapReduce to Parallelise Learning to Rank. Master Thesis in Computer Science, University of Twente, Almere, Netherlands.
42. Taylor, M., Guiver, J., Robertson, S., & Minka, T. (2008). SoftRank: Optimizing Non-Smooth Rank Metrics. Proceedings of the 2008 International Conference on Web Search and Data Mining (WSDM 2008), Palo Alto, CA, pp. 77–86.
43. Tsai, M.-F., Liu, T.-Y., Qin, T., Chen, H.-H., & Ma, W.-Y. (2007).

FRank: A Ranking Method with Fidelity Loss. Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2007), Amsterdam, Netherlands, pp. 383–390.

44. Tukey, J.W. (1977). Exploratory Data Analysis. Cambridge, United Kingdom: Pearson.
45. Wang, X., Bendersky, M., Metzler, D., & Najork, M. (2016). Learning to Rank with Selection Bias in Personal Search. Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2016), Pisa, Italy, pp. 115–124.
46. Wang, L., Bennett, P.N., & Collins-Thompson, K. (2012). Robust Ranking Models via Risk-Sensitive Optimization. Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2012), Portland, OR, pp. 761–770.
47. Wang, L., Lin, J., Metzler, D., & Han, J. (2014). Learning to Efficiently Rank on Big Data. Proceedings of the 23rd International Conference on World Wide Web (WWW 2014), Seoul, Korea, pp. 209–210.
48. Xia, F., Liu, T.-Y., Wang, J., Zhang, W., & Li, H. (2008). Listwise Approach to Learning to Rank: Theory and Algorithm. Proceedings of the 25th International Conference on Machine Learning (ICML 2008), Helsinki, Finland, pp. 1192–1199.
49. Xu, J., Cao, Y., Li, H., & Zhao, M. (2005). Ranking Definitions with Supervised Learning Methods. Proceedings of the 14th International Conference World Wide Web (WWW 2005), Chiba, Japan, pp. 811–819.
50. Xu, J., & Li, H. (2007). AdaRank: A Boosting Algorithm for Information Retrieval. Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2007), Amsterdam, Netherlands, pp. 391-398.
51. Xue, G.-R., Yang, Q., Zeng, H.-J., Yu, Y., & Chen, Z. (2005). Exploiting the Hierarchical Structure for Link Analysis. Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2005), Salvador, Brazil, pp. 186–193.
52. Yeh, J.-Y., Lin, J.-Y., Ke, H.-R., & Yang, W.-P. (2007). Learning to Rank for Information Retrieval Using Genetic Programming. Proceedings of SIGIR 2007 Workshop on Learning to Rank for Information Retrieval (LR4IR 2007), Amsterdam, Netherlands, pp. 41–48.
53. Yin, D., Hu, Y., Tang, J., Daly, T., Zhou, M., Ouyang, H., Chen, J., Kang, C., Deng, H., Nobata, C., Langlois, J.-M., & Chang, Y. (2016). Ranking Relevance in Yahoo Search. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2016), San Francisco, CA, pp. 323–332.
54. Yu, H. (2005). SVM Selective Sampling for Ranking with Application to Data Retrieval. Proceedings of the 11th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2005), Chicago, IL, pp. 354–363.
55. Yu, H., Kim, J., Kim, Y., Hwang, S., & Lee, Y.H. (2012). An Efficient Method for Learning Nonlinear Ranking SVM Functions. Information Sciences, 209 37–48.
56. Yue, Y., Finley, T., Radlinski, F., & Joachims, T. (2007). A Support Vector Method for Optimizing Average Precision. Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2007), Amsterdam, Netherlands, pp. 217–278.
57. Zheng, Z., Zha, H., Zhang, T., Chapelle, O., Chen, K., & Sun, G. (2008). A General Boosting Method and its Application to Learning Ranking Functions for Web Search. In J.C. Platt, D. Koller, Y. Singer, & S.T. Roweis (Eds.), Advances in Neural Information Processing Systems 20 (pp. 1697–1704). Red Hook, NY: Curran Associates, Inc.