Noname manuscript No.
(will be inserted by the editor)

Linear Feature Extraction for Ranking

Gaurav Pandey · Zhaochun Ren · Shuaiqiang Wang · Jari Veijalainen · Maarten de Rijke

Received: date / Accepted: date

Abstract We address the feature extraction problem for document ranking in information retrieval. We then propose LifeRank, a Linear feature extraction algorithm for Ranking. In LifeRank, we regard each document collection for ranking as a matrix, referred to as the *original matrix*. We optimize a *transformation matrix*, so that a new matrix (dataset) can be generated as the product of the original matrix and a transformation matrix. The transformation matrix projects high-dimensional document vectors into lower dimensions. Theoretically, there could be very large transformation matrices, each leading to a new generated matrix. In LifeRank, we produce a transformation matrix so that the generated new matrix can match the learning to rank problem. Extensive experiments on benchmark datasets show the performance gains of LifeRank in comparison with state-of-the-art feature selection algorithms.

Keywords Feature extraction · Dimension reduction · Learning to rank · Information retrieval

Gaurav Pandey
University of Jyväskylä
E-mail: gaurav.g.pandey@student.jyu.fi

Zhaochun Ren
JD.com
E-mail: renzhaochun@jd.com

Shuaiqiang Wang
University of Jyväskylä
E-mail: shuaiqiang.wang@jyu.fi

Jari Veijalainen
University of Jyväskylä
E-mail: jari.veijalainen@jyu.fi

Maarten de Rijke
University of Amsterdam
E-mail: derijke@uva.nl

1 Introduction

Document ranking is an essential component of information retrieval systems and web search engines. Recently, machine learning-based ranking techniques, referred to as "learning to rank," have given rise to an active and growing research area, both in the information retrieval and machine learning communities [10, 15, 21, 39, 59]. A large number of learning to rank algorithms have been proposed, which incorporate more and more useful features, aiming to improve the performance of the ranking algorithms [33]. In a supervised setting, they first collect a set of training data, which includes a set of queries, each associated with a list of documents labeled by relevance degrees; with the training dataset, they train a ranking model that can order unseen documents according to their degree of relevance [20]. In this situation, dimension reduction inevitably becomes an important issue [16].

Firstly, dimension reduction can enhance the accuracy for many machine learning problems, including learning to rank. With dimension reduction techniques, a small set of more discriminative and less redundant features can be selected or generated for learning. Thus, better results could be achieved, as overfitting becomes less likely [38]. Also, the generalization ability of machine learning models could depend on the radius of training data points, which may decrease when the number of features decreases [5, 16, 56, 57].

Secondly, large number of features leads to high complexity in most learning to rank algorithms. Therefore, dimension reduction often leads to significant improvements in training and prediction efficiency, while maintaining, or having a limited negative impact on, accuracy. With accuracy being the primary metric, efficiency has also emerged as a crucial issue for evaluating learning to rank algorithms [10, 11, 55]. Training datasets and ranking features continue to expand, so as to obtain more accurate models. Furthermore, as a consequence of the dynamic character of the Web, ranking models need to be re-learned repeatedly, and the interval between re-learning procedures decreases sharply [33]. With dimension reduction techniques, fewer features are used, resulting in more efficient training and prediction.

Generally, there are two types of dimension reduction algorithms: feature *selection* and feature *extraction*. The former aims to select a subset of the original features for learning, while the latter attempts to generate a small set of new features from the original features [5, 35, 58]. Recently, feature selection for ranking has been investigated intensively [16, 17, 26, 29, 37, 40, 60]. To the best of our knowledge, the advantages of feature *extraction* have not yet been explored in learning to rank.

In this study, we address the *feature extraction problem for learning to rank*. In comparison with feature selection, the feature extraction problem has a much larger search space. For example, given n original features, feature selection selects a subset of features of size k (where $k < n$) for learning. Here, for a particular value of k, the search space of the problem contains $\binom{n}{k}$ possible solutions. The full search space that can include any number of features (i.e., all values of k in range 1 to n), would lead to $2^n - 1$ solutions. In comparison, for linear feature extraction, each extracted feature is a linear combination of original n features. Since the coefficient associated with each original feature can be any real number, the search space becomes infinite. The search space of non-linear feature extraction would be even larger, as it also includes

solutions involving non-linear combinations of features (e.g. polynomial combinations). Hence, with a larger search space, feature extraction has a greater possibility to achieve better performance than feature selection.

To address the problem of linear feature extraction for learning to rank, we propose LifeRank, a Linear feature extraction algorithm for Ranking. LifeRank regards each dataset for training, validation or test as a matrix, referred to as an *original matrix*, where each row vector represents a document with a set of features. With a given original matrix for training $\mathbf{X}$, LifeRank attempts to discover a transformation matrix $\mathbf{T}$, so that a new matrix (dataset) $\mathbf{X}'$ can be generated as the product of the original matrix and a transformation matrix, i.e., $\mathbf{X}' = \mathbf{X}\mathbf{T}$. Thus $\mathbf{T}$ projects high-dimensional document vectors in $\mathbf{X}$ into lower-dimensional ones in $\mathbf{X}'$. Theoretically, there could be a very large number of possible transformation matrices, each leading to a new generated matrix. LifeRank attempts to discover a transformation matrix to transform the original matrix (dataset) into a low-rank one for dimension reduction, on which learning to rank algorithms can achieve optimum results in comparison with other dimension-reduced matrices.

Our problem formulation is similar to principal component analysis (PCA) [23], and thus our algorithm LifeRank can be understood from the perspective of PCA. PCA is one of the most popular dimension reduction techniques in machine learning. When PCA is performed using singular valued decomposition (SVD) [28], the given matrix $\mathbf{X}$ can be approximately decomposed into three low-rank matrices $\mathbf{X} \approx \mathbf{P}\mathbf{\Sigma}\mathbf{Q}^\top$. Here, $\mathbf{\Sigma}$ is composed of the singular values of $\mathbf{X}$, $\mathbf{P}$ and $\mathbf{Q}$ are composed of the left and right singular vectors of $\mathbf{X}$ respectively, and $\mathbf{P}^\top\mathbf{P} = \mathbf{Q}^\top\mathbf{Q} = \mathbf{I}$ is equal to the identity matrix. Thus a new matrix $\mathbf{X}' = \mathbf{P}\mathbf{\Sigma} \approx \mathbf{X}\mathbf{Q}$. However, it should be noted that while PCA calculates $\mathbf{X}'$ as an approximation of $\mathbf{X}$, in LifeRank $\mathbf{X}$ is transformed to $\mathbf{X}'$ using a transformation matrix.

In LifeRank, we formulate the learning to rank task by using a classical pairwise loss function. A pairwise loss function is used because such functions are fundamental, straightforward and intuitive for ranking. Besides, pairwise loss functions are consistent with the assumption that the labels of documents to rank lie in a rank-differentiable probability space [27], and they are upper bounds of measure-based ranking errors [12]. In the generated matrix, the column vectors represent the features. Since optimization over orthogonal features is beneficial to many machine learning problems [48, 49], we utilize the Lagrange multipliers method [1, 4] to impose orthonormality constraints on the column (feature) vectors of the transformed matrix, and then use gradient descent for optimization. With the transformation matrix $\mathbf{T}$, the training, validation and test datasets can be directly generated with matrix product.

Note that (1) LifeRank generalizes feature selection algorithms for the learning to rank task. Feature selection can be regarded as optimizing a transformation matrix $\mathbf{T}$ so that the column vectors of $\mathbf{T}$ meet the orthonormality constraints and each element in $\mathbf{T}$ can only be either 0 or 1. (2) Although some deep learning-based ranking algorithms [47] also aim to generate a set of features for ranking, our problem is completely different: we try to construct our features based on some predesigned ranking features like term frequency (TF) and inverse document frequency (IDF), which have been comprehensively used in conventional learning to rank algorithms like Ranking SVM [9, 21] and RankBoost [15]. Deep learning-based algorithms, however, try to

build features based on word-level features in a corpus that differ substantially from conventional ranking features.

Our main contributions are as follows: (1) We address the feature extraction problem for learning to rank. Feature extraction is a category of comprehensively used dimension reduction techniques in many machine learning problems for performance gains in accuracy and efficiency, but to the best of our knowledge, feature extraction and its advantages have not been explored in learning to rank yet. (2) We propose LifeRank, a linear feature extraction algorithm that generates datasets to be utilized by the learning to rank task. (3) We perform extensive experiments on benchmark datasets and present the performance gains of LifeRank in comparison with the state-of-the-art feature selection algorithms.

The remainder of the paper is organized as follows. Section 2 reviews related work; Section 3 defines the feature extraction problem for ranking; Section 4 proposes LifeRank, a gradient descent-based algorithm. Section 5 introduces our experimental setup. Section 6 reports the experimental results, and Section 7 concludes the paper.

2 Related work

We discuss three types of related work: learning to rank, feature selection for ranking, and feature extraction for ranking.

2.1 Learning to rank for information retrieval

Learning to rank has received increased attention from both the machine learning and information retrieval community. While there is growing interest in *online* learning to rank [46] and in *counterfactual* learning to rank from online data [22], the bulk of the work on learning to rank concerns *offline* learning to rank, where explicit human annotations are used to label query, document pairs. Offline learning to rank is the focus of this paper. Given its effectiveness, many algorithms have been proposed, which mainly fall into three categories [11, 32]: pointwise, pairwise, and listwise.

Pointwise approaches, such as Pranking [14], McRank [31] and Subset Ranking [13], view each document in the training dataset as a learning instance, and utilize a classification or regression technique to predict the relevance categories or numerical/ordinal relevance scores for unlabeled data. Pairwise approaches, such as Ranking SVM [9, 21], RankBoost [15], RankNet [6], FRank [52], LambdaRank [7], and BoltzRank [54], regard a pair of documents as a learning instance, and try to learn a binary classifier that can predict the more relevant document to the given query from each pair of documents. Then the ranked lists of documents can be aggregated based on the pairwise preferences of the documents. Listwise approaches, such as ListNet [10], SVM-MAP [61], NDCGBoost [53], take the entire ranked list of documents as a learning instance, and attempt to construct a ranking model that can directly predict the full rankings of the documents. Recently, some hybrid algorithms have been proposed, such as FocusedRank [39], MixRank [8], targeting improvements in learning accuracy, efficiency, or both. More algorithms are surveyed in [11, 32, 33].

With the incorporation of more and more useful features for performance gains, dimension reduction inevitably becomes an important issue in the ranking problem [16]. With effective dimension reduction techniques, not only the efficiency of the algorithms could be improved, but also accuracy could be enhanced as a result of using more discriminative features with less redundancy and noise. Furthermore, the generalization of the ranking model can also be increased as a result of using fewer features [16].

2.2 Feature selection for ranking

Recently, considerable efforts have been made on feature selection for ranking. Geng et al [16] present GAS, one of the first attempts to incorporate the importance and similarity of features for ranking. In particular, it evaluates the importance of features with ranking metrics like MAP [3] and NDCG [19], and estimates the similarity between features using agreement between rankings, e.g., with Kendall τ correlation coefficient [24]. Then it greedily selects a subset of features with maximum total importance scores and minimum total similarity scores. Metzler [34] proposes a greedy feature selection algorithm to be used within the Markov random field model for information retrieval. The model automatically generates models that are more effective than, or as effective as, models created by carefully selecting the features manually. Pan et al [40] investigate a boosted regression trees-based feature selection algorithm. It evaluates the importance of the features based on boosted trees. Then it selects features by maximizing the discounted importance of the features, where the importance of each feature is discounted by feature similarity. Yu et al [60] propose RankWrapper and RankSelect, two feature weighting and selection algorithms for learning to rank. They utilize ranking distances of nearest data points in order to identify the key features for ranking, demonstrating significant efficiency gains in comparison with GAS.

Gupta and Rosso [17] present a Kullback-Leibler (KL) divergence-based divergence metric, and select a subset of features for ranking based on features' expected divergence over the relevance classes and the importance of features. Lai et al [26] propose a joint convex optimization formulation for minimizing ranking errors while simultaneously conducting feature selection. This optimization formulation provides a flexible framework in which various importance measures and similarity measures of the features can easily be incorporated. Naini and Altingövde [37] adopt three greedy diversification strategies, maximal marginal relevance, MaxSum dispersion and modern portfolio theory, to the problem of feature selection for ranking. Laporte et al [29] propose a general framework for feature selection in learning to rank based on support vector machine (SVM); they investigate both classical convex regularizations (such as $L1$ and weighted $L1$) and non-convex regularization terms (such as log penalty, Minimax Concave Penalty (MCP) and Lp pseudo norm with $p < 1$). Furthermore, they provided an accelerated proximal approach for solving the convex problems and a re-weighted $L1$ scheme to address the non-convex regularizations.

All of these algorithms are meant to address *feature selection* for ranking. To the best of our knowledge, there is no work targeting *feature extraction* for ranking.

2.3 Feature extraction techniques

Feature extraction has been used extensively used in various machine learning scenarios for performance gains in terms of accuracy and efficiency. Given its effectiveness, many approaches have been proposed, which are either linear or non-linear algorithms.

The main linear technique for feature extraction is principal component analysis (PCA) [23], which performs a linear mapping of high-dimensional data into a lower-dimensional space in such a way that the variance of the data in the low-dimensional representation is maximized. Canonical-correlation analysis (CCA) [18] is another popular linear feature extraction algorithm, which attempts to discover linear combinations of the original features that have maximal correlation with each other. In addition, several probabilistic algorithms, including probabilistic PCA [51], probabilistic CCA [2] and probabilistic partial CCA [36], have been proposed, where a set of latent variables are introduced for probabilistically interpreting these models.

Non-linear feature extraction algorithms can combine the original features to generate a set of features in a non-linear way. For example, the locally linear embedding (LLE) method [44] learns the global structure of non-linear manifolds to yield low-dimensional, neighborhood-preserving embeddings of high-dimensional inputs. Isomap [50] is capable of discovering the non-linear degrees of freedom that underly complex natural observations. It can efficiently compute a globally optimal solution and can be guaranteed to converge asymptotically to the true structure. Besides, some kernel techniques have been proposed to transform linear feature extraction algorithms into nonlinear ones. For example, kernel PCA [45] is a non-linear form of principal component analysis (PCA), which can efficiently compute principal components in high-dimensional feature spaces through the use of integral operator kernel functions.

Although feature extraction techniques have been extensively investigated and shown to demonstrate promising performance gains, to the best of our knowledge, they have not been explored yet in the context of the ranking problem.

3 Problem statement

3.1 Learning to rank for information retrieval

Let $\mathcal{X}$ be a collection of documents, each represented by a vector of feature values. In information retrieval systems, given a query q, a list of documents from $\mathcal{X}$ is returned as search results, where the documents are ranked according to their estimated relevance to q. Given a query q, the ground truth, i.e., relevance judgments of documents with respect to q (produced by human experts) is defined as a function $rel : \mathcal{X} \rightarrow \mathbb{N}_0$, where $\mathbb{N}_0$ is the set of natural numbers (including 0).

Let $f : \mathcal{X} \rightarrow \mathbb{R}$ be a ranking function assigning real valued relevance scores to documents. The goodness of ranking functions can be evaluated by a measure s, such as precision at n ($P@n$), mean average precision (MAP) [3], or normalized discounted cumulative gain ($NDCG@n$) [19].

Definition 1 (Learning to rank) *Given a training dataset $\mathcal{X}$ and an evaluation measure s, the problem of learning to rank is to learn a ranking function f from $\mathcal{X}$ such that $s(f)$ is maximized.*

3.2 Dimension reduction for ranking

In learning to rank, each dataset $\mathcal{X}$ can be regarded as a document matrix $\mathbf{X}_{m\times n}$ with m rows (documents) and n columns (features). In particular, $\mathbf{x}_i$ is the i-th row of $\mathbf{X}$, and ${\mathbf{x}_i}^\top$ is a n-dimensional (column) vector that represents a document with n features. Let $g : \mathbb{R}^n \to \mathbb{R}^k$ ($k \leq n$) be a mapping that projects an n-dimensional vector space into a k-dimensional space. Let $L(\cdot)$ be the loss function for the learning to rank task. Our problem is to discover a mapping function g such that the obtained dataset $\mathbf{X}' = g(\mathbf{X})$ minimizes the loss function.

Definition 2 (Dimension reduction for ranking) *Let $\mathbf{X}_{m\times n}$ be a document matrix with m columns and n rows, where each column ${\mathbf{x}_i}^\top$ is a n-dimensional vector, representing a document with n features. Let $\mathcal{G}$ be the set of all possible mapping functions, where each element $g : \mathbb{R}^n \to \mathbb{R}^k$ ($k \leq n$) is used to project an n-dimensional vector space into a k-dimensional space. The dimension reduction for the learning to rank task tries to discover an optimum mapping function $g^* \in \mathcal{G}$ such that:*

$$\arg\min_{g\in\mathcal{G}} L(g(\mathbf{X})), \tag{1}$$

where $L(\cdot)$ is the loss function for the learning to rank task. Then the new dataset can be generated with $g^*(\mathbf{X})$.

In this paper, we consider linear feature extraction for learning to rank as it is the simplest and most straightforward feature extraction technique in machine learning. Here, each generated feature is a linear combination of the original features. It utilizes a transformation matrix $\mathbf{T}$ to achieve the effectiveness of the mapping function, aiming to discover an optimal matrix $\mathbf{T}$ such that the obtained dataset $\mathbf{X}' = \mathbf{XT}$ results in a minimal value of the loss function.

The problem can be understood from the perspective of PCA [23]. Using PCA, the given matrix $\mathbf{X}$ can be approximately decomposed into three lower-rank matrices:

$$\mathbf{X} \approx \mathbf{P\Sigma Q}^\top, \tag{2}$$

where $\mathbf{\Sigma}$ is composed of the singular values of $\mathbf{X}$, $\mathbf{P}$ and $\mathbf{Q}$ are composed of the left and right singular vectors of $\mathbf{X}$ respectively, and $\mathbf{P}^\top\mathbf{P} = \mathbf{Q}^\top\mathbf{Q} = \mathbf{I}$ (the identity matrix). Thus, a new matrix $\mathbf{X}'$ can be generated as follows:

$$\mathbf{X}' = \mathbf{P\Sigma} \approx \mathbf{XQ}. \tag{3}$$

The role of the transformation matrix $\mathbf{T}$ in LifeRank is very similar to the right singular matrix $\mathbf{Q}$ in PCA, where $\mathbf{Q}$ maps the document vectors to another space spanned by the columns of $\mathbf{Q}$ before transforming them through $\mathbf{\Sigma}$ and going back through $\mathbf{P}$. Hence, in LifeRank we consider the orthonormality constraints of $\mathbf{T}$ in our optimization process, i.e., $\mathbf{T}^\top\mathbf{T} = \mathbf{I}$.

Definition 3 (Constrained linear feature extraction for ranking) *Let $\mathbf{X}_{m\times n}$ be a document matrix, where the transpose of each row, i.e., ${\mathbf{x}_i}^\top = \mathbf{d}_i$ is a n-dimensional vector, representing a document with n features. Linear feature extraction for ranking aims to optimize a transformation matrix $\mathbf{T}_{n\times k}$ by solving the following optimization problem, so that a new document matrix $\mathbf{X}'_{m\times k} = \mathbf{X}_{m\times n}\mathbf{T}_{n\times k}$ can be generated, where each document vector $\mathbf{d}_i$ can be projected into k-dimensional vector $\mathbf{d}'_i = \mathbf{T}^\top\mathbf{d}_i$:*

$$\arg\min_{\mathbf{T}} \quad L(\mathbf{XT}) \text{ such that } \mathbf{T}^\top\mathbf{T} = \mathbf{I}, \tag{4}$$

where $L(\cdot)$ is the loss function for the learning to rank task.

Based on the optimized mapping function g, the new dataset can be generated by taking the product of the original matrix and the transformation matrix, i.e., $\mathbf{X}' = \mathbf{XT}$.

We have used the example of PCA to help us explain the mechanism of LifeRank. However, it should be noted that in PCA $\mathbf{X}'$ is calculated as an approximation of $\mathbf{X}$, whereas in LifeRank we generate a transformed representation of the initial matrix, in order to achieve a better ranking performance. Hence, unlike PCA, $\mathbf{X}'$ as computed in Definition 3 is not an approximation of $\mathbf{X}$, but a transformation.

4 The LifeRank algorithm

Given a high-dimensional dataset $\mathcal{X}$, LifeRank generates a new low-dimensional dataset $\mathcal{X}'$ in two phases. In the first phase, LifeRank first preprocesses the training dataset $\mathcal{X}$ into an original matrix $\mathbf{X}$. Then LifeRank optimizes the transformation matrix $\mathbf{T}$ for $\mathbf{X}$ according to the loss function in Equation 4. In the second phase, LifeRank generates low-dimensional training, validation and test matrices with the projection of $\mathbf{T}$. Then LifeRank constructs new datasets based on the low-dimensional data matrices.

4.1 Phase I: Generation of the transformation matrix

In this study, we utilize a classic pairwise learning to rank loss function to implement the function $L(\cdot)$ in Definition 3. Pairwise loss functions are chosen because apart from being relatively simple and straightforward, they are also intuitive choices for ranking. Besides, with the assumption that the labels of documents to rank lie in a rank-differentiable probability space, pairwise loss functions are consistent [27] and provide upper bounds for measure-based ranking errors like NDCG [12]. Thus, minimizing a pairwise loss function will maximize the ranking measures [27].

First of all, the training dataset $\mathcal{X}$ is preprocessed into an original matrix $\mathbf{X}$ and other information $\mathbf{I}_X$ consisting of identities of the documents and queries, relevance labels, etc. Let $D = \{\mathbf{d}_1, \mathbf{d}_2, \ldots, \mathbf{d}_m\}$ be the set of columns (document vectors) in the matrix $\mathbf{X}^\top_{m\times n}$. We regard each pair of vectors $(\mathbf{d}_i, \mathbf{d}_j) \in D \times D$ as an instance, and the label $y_{i,j} \in \{+1, -1\}$ indicates whether the relevance of the i-th document is higher or lower than the j-th document, corresponding to the given query. Let

$\{\mathbf{t}_1, \mathbf{t}_2, \ldots, \mathbf{t}_k\}$ be the column vectors of $\mathbf{T}$. We try to discover a k-dimensional vector of weights $\mathbf{w}$ such that:

$$\begin{aligned} &\arg\min_{\mathbf{T},\mathbf{w},b} \sum_{\forall(\mathbf{d}_i,\mathbf{d}_j),i\neq j} \log\left(1+e^{-y_{i,j}\left(\mathbf{w}^\top\mathbf{T}^\top(\mathbf{d}_i-\mathbf{d}_j)+b\right)}\right)+\frac{\lambda}{2}||\mathbf{w}||^2 \\ &\text{such that } \mathbf{t}_i^\top\mathbf{t}_j=\begin{cases}1, & i=j\\ 0, & i\neq j\end{cases}, \forall i,j=1,2,\ldots,k, \end{aligned} \tag{5}$$

where the first part calculates the log loss of the ranking accuracy, the second part is the $l2$ norm of the parameters for regularization, and λ is the coefficient of the regularization term for trade-off.

We optimize the constrained loss function based on the Lagrange multipliers method [1, 4] in Equation 5. Let

$$\begin{aligned} \mathcal{L}(\mathbf{T},\mathbf{w},b,\mathbf{A}) = &\sum_{\forall(\mathbf{d}_i,\mathbf{d}_j),i\neq j} \log\left(1+e^{-y_{i,j}\left(\mathbf{w}^\top\mathbf{T}^\top(\mathbf{d}_i-\mathbf{d}_j)+b\right)}\right)+ \\ &\frac{\lambda}{2}||\mathbf{w}||^2+\sum_{i,j=1,\ldots,k\wedge i\neq j}\alpha_{i,j}\mathbf{t}_i^\top\mathbf{t}_j+\sum_{i=1}^{k}\alpha_{i,i}\left(1-\mathbf{t}_i^\top\mathbf{t}_i\right), \end{aligned} \tag{6}$$

where $\mathbf{A}$ is a matrix with k columns and k rows, and elements $\alpha_{i,j}$. Then, the optimum $\mathbf{T}$, $\mathbf{w}$ and b for minimizing $\mathcal{L}$ are the exact results of Equation 5.

In Phase I, we utilize gradient descent to generate the training dataset and the transformation matrix. Initially, we assign all 1s to the vector $\mathbf{w}_{k\times 1}$ so that all of the generated features in the ranking model have the same initial weight. We initialize the transformation matrix $\mathbf{T}$ in a random manner, following work on matrix generalization problems like matrix factorization-based collaborative filtering [25]. After initialization, the weight vector $\mathbf{w}$ and the factorized matrix can be updated iteratively with gradient descent until reaching convergence or the maximum number of iterations with the given learning rate. The gradients of the function $\mathcal{L}$ with respect to the variables are calculated as follows:

$$\begin{aligned} \nabla_{\mathbf{w}}\mathcal{L} &= \sum_{\forall(\mathbf{d}_i,\mathbf{d}_j),i\neq j}\frac{-y_{i,j}\mathbf{T}^\top(\mathbf{d}_i-\mathbf{d}_j)}{1+e^{y_{i,j}\left(\mathbf{w}^\top\mathbf{T}^\top(\mathbf{d}_i-\mathbf{d}_j)+b\right)}}+\lambda\mathbf{w} \\ \nabla_{\mathbf{t}_l}\mathcal{L} &= \sum_{\forall(\mathbf{d}_i,\mathbf{d}_j),i\neq j}\frac{-y_{i,j}w_l(\mathbf{d}_i-\mathbf{d}_j)}{1+e^{y_{i,j}\left(\mathbf{w}^\top\mathbf{T}^\top(\mathbf{d}_i-\mathbf{d}_j)+b\right)}}+ \\ &\quad \sum_{i\neq l}(\alpha_{l,i}+\alpha_{i,l})\mathbf{t}_i-2\alpha_{l,l}\mathbf{t}_l, \quad l=1,\ldots,k \\ \frac{\partial\mathcal{L}}{\partial b} &= \sum_{\forall(\mathbf{d}_i,\mathbf{d}_j),i\neq j}\frac{-y_{i,j}}{1+e^{y_{i,j}\left(\mathbf{w}^\top\mathbf{T}^\top(\mathbf{d}_i-\mathbf{d}_j)+b\right)}} \\ \frac{\partial\mathcal{L}}{\partial\alpha_{i,j}} &= \begin{cases}\mathbf{t}_i^\top\mathbf{t}_j, & i\neq j\\ 1-\mathbf{t}_i^\top\mathbf{t}_j, & i=j\end{cases} \quad \forall i,j=1,\ldots,k, \end{aligned} \tag{7}$$

where $\mathbf{t}_1, \mathbf{t}_2, \ldots \mathbf{t}_n$ are the column vectors of $\mathbf{T}$. Since gradient descent generally does not work with Lagrange multipliers, we use the basic differential multiplier method (BDMM) [41] for optimization, where the sign inversion for α in Equation 8 makes the optimization stable. Given a learning rate η, the update formulas of the gradient descent method are:

$$\begin{aligned} \mathbf{w} &\leftarrow \mathbf{w} - \eta \nabla_{\mathbf{w}} \mathcal{L} \\ \mathbf{t}_l &\leftarrow \mathbf{t}_l - \eta \nabla_{\mathbf{t}_l} \mathcal{L}, \text{ for } l = 1, \ldots, k \\ b &\leftarrow b - \eta \frac{\partial \mathcal{L}}{\partial b} \\ \alpha_{i,j} &\leftarrow \alpha_{i,j} + \eta \frac{\partial \mathcal{L}}{\partial \alpha_{i,j}}, \text{ for } i, j = 1, \ldots, k. \end{aligned} \tag{8}$$

4.2 Phase II: Generation of low-rank datasets

In LifeRank, Phase II generates all of the datasets for learning to rank, including the training, validation and test datasets. According to Definition 3, for each original matrix $\mathbf{X}$, the generated matrix $\mathbf{X}'$ can be obtained as a product of the original dataset $\mathbf{X}$ and the transformation matrix $\mathbf{T}$, formally $\mathbf{X}' = \mathbf{X}\mathbf{T}$. Then, the new low-dimensional dataset $\mathcal{X}'$ can be generated by integrating matrix $\mathbf{X}'$ with other information $\mathbf{I}_X$ that was filtered in the preprocessing step in Phase I.

4.3 Pseudocode

The pseudocode of LifeRank as a dimension reduction algorithm for ranking is summarized in Algorithm 1. Given the number of generated features k and a set of standard learning to rank datasets, including a training dataset $\mathcal{X}$, a validation dataset $\mathcal{V}$ and a test dataset $\mathcal{E}$, LifeRank tries to output new low-dimensional datasets $\mathcal{X}'$, $\mathcal{V}'$ and $\mathcal{E}'$ for training, validation and test, respectively, for the learning to rank procedure.

Algorithm 1 implements the two phases of LifeRank: (I) Lines 1–8 generate the transformation matrix $\mathbf{T}$ based on the original training dataset $\mathcal{X}$; (II) Using $\mathbf{T}$, lines 9–10 generate the low-dimensional matrices for training $\mathbf{X}'$, validation $\mathbf{V}'$ and test $\mathbf{E}'$. Then, line 11 constructs the low-dimensional training, validation and test datasets by directly integrating the low-rank matrices and their corresponding information filtered in the preprocessing steps in lines 1 and 9.

4.4 Discussion

In this section, we reveal a connection between the feature selection for ranking problem and the linear feature extraction for ranking problem. In particular, from the perspective of linear transformations of matrices, the feature selection for ranking problem can be defined as in Definition 4.

Algorithm 1: LifeRank: A Linear Feature Extraction Algorithm for Ranking

Input: A training dataset $\mathcal{X}$, a validation dataset $\mathcal{V}$, a test dataset $\mathcal{E}$, the learning rate η, and the number of features k in the set of generated document.
Output: A generated training dataset $\mathcal{X}'$, validation dataset $\mathcal{V}'$, and test dataset $\mathcal{E}'$, each with k features.

// Phase I
$(\mathbf{X}, \mathbf{I}_X) \leftarrow$ Preprocess$(\mathcal{X})$;
$\mathbf{T}, \mathbf{w}, \{\alpha_{i,j}\}_{i,j=1,\ldots,k} \leftarrow$ Initialize$(\mathbf{X}, k)$;
repeat
  $\mathbf{w} \leftarrow \mathbf{w} - \eta \nabla_{\mathbf{w}} \mathcal{L}$;
  $\mathbf{t}_l \leftarrow \mathbf{t}_l - \eta \nabla_{\mathbf{t}_l} \mathcal{L}$, for $l = 1, \ldots, k$;
  $b \leftarrow b - \eta \frac{\partial \mathcal{L}}{\partial b}$;
  $\alpha_{i,j} \leftarrow \alpha_{i,j} + \eta \frac{\partial \mathcal{L}}{\partial \alpha_{i,j}}$, for $i, j = 1, \ldots, k$;
until *Reach convergence or the max iteration*;
// Phase II
$(\mathbf{V}, \mathbf{I}_V), (\mathbf{E}, \mathbf{I}_E) \leftarrow$ Preprocess$(\mathcal{V}, \mathcal{E})$;
$\mathbf{X}' \leftarrow \mathbf{X}\mathbf{T}, \quad \mathbf{V}' \leftarrow \mathbf{V}\mathbf{T}, \quad \mathbf{E}' \leftarrow \mathbf{V}\mathbf{T}$;
$\mathcal{X}', \mathcal{V}', \mathcal{E}' \leftarrow$ GenerateDatasets$(\mathbf{X}', \mathbf{V}', \mathbf{E}', \mathbf{I}_X, \mathbf{I}_V, \mathbf{I}_E)$;

Definition 4 (Feature selection for ranking) *Let $\mathbf{X}_{m\times n}$ be a document matrix, where the transpose of each row ${\mathbf{x}_i}^\top = \mathbf{d}_i$ is an n-dimensional vector, representing a document with n features. Feature selection for ranking aims to optimize a transformation matrix $\mathbf{T}_{n\times k}$ by solving the following optimization problem, so that a new document matrix $\mathbf{X}'_{m\times k} = \mathbf{X}_{m\times n}\mathbf{T}_{n\times k}$ can be generated, where each n-dimensional document vector $\mathbf{d}_i$ can be projected into a k-dimensional vector $\mathbf{d}'_i = \mathbf{T}^\top \mathbf{d}_i$:*

$$\arg\min_{\mathbf{T}} L(g(\mathbf{X}\mathbf{T})) \text{ such that } \begin{cases} \forall t_{i,j} \in \mathbf{T} : t_{i,j} = \{0, 1\} \\ \mathbf{T}^\top \mathbf{T} = \mathbf{I}. \end{cases} \tag{9}$$

Based on the optimized mapping function g, the new low-rank matrix can be generated as a product of the original matrix and the transformation matrix, i.e., $\mathbf{X}' = \mathbf{X}\mathbf{T}$.

The k columns of the transformation $\mathbf{T}$ mentioned in Defintion 4 present the k iterations of the feature selection processes. The constraints in Equation 9 guarantee that there is only one "1" in each column of the transformation matrix $\mathbf{T}$ and the others are all "0," indicating that each feature selection process only selects one feature. The second constraint $\mathbf{T}^\top \mathbf{T} = \mathbf{I}$ guarantees that the position of the unique "1" in each column is different from other columns, which is the index of the selected feature in that step.

Since the elements in the transformation matrix $\mathbf{T}$ can be any real numbers in Definition 3 while they are only either 0 or 1 in Definition 4, Definition 3 generalizes Definition 4, i.e., the problem of linear feature extraction for ranking generalizes the problem of feature selection for ranking. Because of this, linear feature extraction is expected to outperform or be at least as good as any feature selection technique. The linear feature extraction is expected to use more computational resources than feature selection, since former deals with the search space in real numbers and the

latter with binary case. However, this computational overhead is the tradeoff for the higher performance expected to be achieved by the extracted features, when utilized for learning to rank.

5 Experimental setup

5.1 Research questions

We list the research questions that guide the remainder of the paper.

RQ1 What is the performance of LifeRank in generating low-dimensional datasets? Does LifeRank outperform state-of-the-art feature selection algorithms? (See §6.1)

RQ2 Can the importance and redundancy of the features generated by LifeRank outperform those selected by feature selection algorithms? (See §6.2)

RQ3 What is the effect of the orthonormality constraints of the transformation matrix in Equation 4? Does it help enhance the performance of ranking predictions? (See §6.3.)

5.2 Datasets

In this study, we use the MQ2007 and MQ2008 datasets from LETOR 4.0 [42] and OHSUMED from LETOR 3.0 [43] to evaluate our algorithm. The LETOR[1] datasets are commonly used benchmarks in learning to rank. LETOR 4.0 is the latest version, which was released in July 2009. It uses the Gov2 web page collection (∼25M pages) and two query sets from the Million Query track of TREC 2007 and TREC 2008, which are referred to as MQ2007 and MQ2008. We use both MQ2007 and MQ2008 in our experiments. In MQ2007, there are about 1700 queries and about 70,000 query-document pairs, while MQ2008 has 800 queries and about 15,000 query-document pairs for training, validation and testing. In both datasets, each query-document pair has 46 features. We also use the OHSUMED dataset from LETOR 3.0, which was released in December 2008. OHSUMED is extracted from the online medical information database MEDLINE. It contains 106 queries and about 16,000 query-document pairs, where each query-document pair has 45 features.

In all the datasets that we use, relevance of documents with respect to queries is judged at three levels: 2 (definitely relevant), 1 (partially relevant), and 0 (not relevant). In our experiments, we use 5-fold cross validation. In each fold, 60% queries are used for training, 20% for validation and and the remaining 20% for testing. The performance numbers reported are averaged over the five folds.

5.3 Baselines

LifeRank aims to generate low-dimensional datasets for ranking. In this paper, we utilize three baselines to evaluate the datasets generated by our algorithm:

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

Original datasets: We firstly use the original LETOR datasets as our first baseline, on which no selection or generation has been performed.

Datasets generated by GAS: GAS [16] incorporates importance and similarity information of the features into ranking. It greedily selects a subset of features by maximizing the total importance scores meanwhile minimizing the total similarity scores.

Datasets generated by FSMRank: FSMRank [26] trains a feature selection model with machine learning, which can select a subset of features meanwhile minimizing the ranking errors.

We then run Linear Regression [30]-based learning to rank and RankSVM[2] [21] to determine how well these datasets can address the ranking problem. The former makes pointwise predictions on the relevance of the documents by linear regression, which is implemented in the RankLib learning to rank toolkit.[3] The latter predicts pairwise ranking relation between each pair of documents directly by support vector machine (SVM). These are classical pointwise and pairwise learning to rank algorithms, respectively, with which we can clearly demonstrate the effects of dimension reduction.

Since LifeRank uses a linear approach for feature extraction, it is expected to show effectiveness mainly for linear learning-to-rank methods. This is the reason why we have chosen SVMRank and Linear Regression for experimentation.

5.4 Evaluation measures

5.4.1 Measures for ranking

We use two standard ranking accuracy metrics to evaluate the rankings generated by learning to rank algorithms: mean average precision (*MAP*) [3] and normalized discount cumulative gain (*NDCG@n*) [19].

Statistical significance of observed differences between the performance of two runs is tested using a two-tailed paired t-test and is denoted using $^{\blacktriangle}$ (or $^{\blacktriangledown}$) for strong significance for $\alpha = 0.01$; or $^{\triangle}$ (or $^{\triangledown}$) for weak significance for $\alpha = 0.05$.

5.4.2 Measures for features

We consider two metrics to evaluate the quality of the features: importance and redundancy.

The *importance* of each feature can be evaluated by the ranking performance when the feature is used as a ranking model to order the documents. In particular, we use *NDCG@5* for evaluation. Since for calculating these measures, for some features larger values correspond to higher ranks while for others smaller values lead to higher ranks, we utilize the strategy in GAS [16] for evaluation: We order the documents twice in ascending and descending manners respectively, and take the larger score as

[2] https://www.cs.cornell.edu/people/tj/svm_light/svm_rank.html
[3] https://sourceforge.net/p/lemur/wiki/RankLib/

the importance score of the features. Then we calculate the average importance of the features as the importance of the set of features $F = \{f_1, f_2, \ldots, f_k\}$:

$$Imp(F) = \frac{1}{k} \sum_{i=1}^{k} \max \left\{ eva(\mathcal{X}, f_i), eva(\mathcal{X}, -f_i) \right\},$$

where the function $eva(\mathcal{X}, f_i)$ returns the evaluation results of the ranking model f_i on the dataset $\mathcal{X}$.

The *redundancy* of features can be defined as the average similarity between each pair of features. In practice, we regard each feature as a ranking model to order the documents, and then calculate the similarity between each pair of features as the average similarity of their document rankings associated to different queries. Let Q be the set of queries in the given dataset, each associated with a set of documents for ranking. The redundancy of the features $F = \{f_1, f_2, \ldots, f_k\}$ is calculated as follows:

$$Rdd(F) = \frac{2}{k(k-1)} \sum_{f_i, f_j \in F, i>j} \frac{1}{|Q|} \sum_{q \in Q} sim \left(\sigma_i^{(q)}, \sigma_j^{(q)} \right),$$

where $\sigma_i^{(q)}$ is the ranking of the document associated to the query q when the feature f_i is used as the ranking model to order the documents. In this paper, we take the absolute value of Kendall's τ correlation coefficient [24] as the similarity metric for rankings:

$$\tau \left(\sigma_i, \sigma_j \right) = \frac{N_c - N_d}{N_c + N_d}, \tag{10}$$

where N_c and N_d are the numbers of the concordant pairs and discordant pairs respectively between rankings σ_i and σ_j.

The range of $\tau \left(\sigma_i, \sigma_j \right)$ is $[-1, 1]$, where the sign indicates that the correlation between σ_i and σ_j is either positive or negative, and the absolute value indicates the strength of the correlation. Since positive and negative values should not neutralize and we only consider the strength of the correlations, we take the absolute value of Kendall's τ as the similarity metric in the definition of redundancy.

6 Experimental results

6.1 Performance on generated datasets

Tables 1, 2 and 3 list the results obtained in our experiments on the MQ2007, MQ2008 and OHSUMED datasets, respectively. They show the NDCG@1–10 and MAP scores for the RankSVM and Linear Regression learning to rank algorithms on 4 categories of datasets: the original datasets and 3 datasets generated by dimension reduction algorithms including GAS, FSMRank and our LifeRank. For each dimension reduction algorithm, we consider k generated features, with $k = 5$, 10, 15, 20. The results for the original dataset in the tables are independent of the value of k, but are repeated nevertheless for ease of comparison. The values in bold represent the best performance among GAS, FSMSVM and LifeRank.

Table 1: Ranking performance on MQ2007 and selected/generated datasets.
Statistical significance shown for LifeRank against Original, GAS and FSMSVM
Performance for Original is independent of value of k (corresponds to original dataset)

Performance for RankSVM

NDCG	@1	@2	@3	@4	@5	@6	@7	@8	@9	@10	MAP
k = 5											
Original	0.4079	0.4007	0.4009	0.4030	0.4077	0.4129	0.4201	0.4275	0.4336$^{\triangledown}$	0.4391$^{\triangledown}$	0.4615$^{\blacktriangledown}$
GAS	0.3751	0.3807	0.3869$^{\triangle}$	0.3946	0.3980	0.4068	0.4113	0.4177	0.4242	0.4290	**0.4563**
FSMSVM	0.3598$^{\blacktriangle}$	0.3703$^{\blacktriangle}$	0.3773$^{\blacktriangle}$	0.3811$^{\blacktriangle}$	0.3871$^{\blacktriangle}$	0.3941$^{\blacktriangle}$	0.3997$^{\blacktriangle}$	0.4061$^{\blacktriangle}$	0.4129$^{\blacktriangle}$	0.4193$^{\blacktriangle}$	0.4512
LifeRank	**0.3925**	**0.3925**	**0.3975**	**0.4009**	**0.4062**	**0.4118**	**0.4162**	**0.4209**	**0.4256**	**0.4312**	0.4539
k = 10											
Original	0.4079	0.4007	0.4009	0.4030$^{\triangle}$	0.4077$^{\triangle}$	0.4129$^{\blacktriangle}$	0.4201$^{\triangle}$	0.4275	0.4336	0.4391	0.4615
GAS	0.3897	0.3893$^{\blacktriangle}$	0.3914$^{\blacktriangle}$	0.3965$^{\blacktriangle}$	0.4029$^{\blacktriangle}$	0.4098$^{\blacktriangle}$	0.4153$^{\blacktriangle}$	0.4209$^{\blacktriangle}$	0.4272$^{\blacktriangle}$	0.4343$^{\blacktriangle}$	0.4558$^{\triangle}$
FSMSVM	0.3919	0.3917	0.3982$^{\triangle}$	0.4027$^{\triangle}$	0.4079$^{\triangle}$	0.4134$^{\blacktriangle}$	0.4187$^{\blacktriangle}$	0.4239$^{\blacktriangle}$	0.4290$^{\blacktriangle}$	0.4349$^{\blacktriangle}$	0.4593
LifeRank	**0.4037**	**0.4023**	**0.4089**	**0.4117**	**0.4161**	**0.4215**	**0.4264**	**0.4312**	**0.4370**	**0.4423**	**0.4634**
k = 15											
Original	0.4079	0.4007	0.4009	0.4030	0.4077	0.4129	0.4201	0.4275	0.4336	0.4391	0.4615
GAS	0.3942	0.3954	0.3998	0.4023	0.4063$^{\blacktriangle}$	0.4126$^{\triangle}$	0.4195	0.4256$^{\triangle}$	0.4316$^{\blacktriangle}$	0.4372$^{\blacktriangle}$	0.4593$^{\blacktriangle}$
FSMSVM	0.3905	0.3937$^{\triangle}$	0.4005	0.4060	0.4106	0.4144	0.4201	0.4250$^{\triangle}$	0.4309$^{\blacktriangle}$	0.4365$^{\blacktriangle}$	0.4589$^{\triangle}$
LifeRank	**0.3972**	**0.4032**	**0.4061**	**0.4082**	**0.4141**	**0.4194**	**0.4242**	**0.4308**	**0.4376**	**0.4436**	**0.4635**
k = 20											
Original	0.4079	0.4007	0.4009	0.4030$^{\triangle}$	0.4077$^{\triangle}$	0.4129$^{\blacktriangle}$	0.4201$^{\triangle}$	0.4275	0.4336	0.4391	0.4615
GAS	0.4014	0.3967$^{\blacktriangle}$	0.4007	0.4027$^{\blacktriangle}$	0.4088$^{\blacktriangle}$	0.4137$^{\blacktriangle}$	0.4189$^{\blacktriangle}$	0.4257$^{\blacktriangle}$	0.4326$^{\triangle}$	0.4394$^{\triangle}$	0.4601$^{\blacktriangle}$
FSMSVM	0.3882$^{\blacktriangle}$	0.3929$^{\blacktriangle}$	0.4004	0.4060	0.4096	0.4138$^{\triangle}$	0.4203$^{\triangle}$	0.4259	0.4310$^{\triangle}$	0.4368$^{\blacktriangle}$	0.4595$^{\triangle}$
LifeRank	**0.4097**	**0.4077**	**0.4058**	**0.4106**	**0.4153**	**0.4213**	**0.4265**	**0.4311**	**0.4374**	**0.4438**	**0.4640**

Performance for Linear Regression

NDCG	@1	@2	@3	@4	@5	@6	@7	@8	@9	@10	MAP
k = 5											
Original	0.3750	0.3854	0.3882	0.3926	0.3979	0.4034	0.4088	0.4154	0.4208	0.4277	0.4497
GAS	0.3712	0.3751	0.3797$^{\triangle}$	0.3851$^{\triangle}$	0.3894	0.3952	0.4011$^{\triangle}$	0.4071$^{\triangle}$	0.4136$^{\triangle}$	0.4196$^{\triangle}$	0.4462
FSMSVM	0.3554$^{\blacktriangle}$	0.3634$^{\blacktriangle}$	0.3673$^{\blacktriangle}$	0.3741$^{\blacktriangle}$	0.3780$^{\blacktriangle}$	0.3872$^{\blacktriangle}$	0.3929$^{\blacktriangle}$	0.4002$^{\blacktriangle}$	0.4076$^{\blacktriangle}$	0.4145$^{\blacktriangle}$	0.4489
LifeRank	**0.3852**	**0.3874**	**0.3908**	**0.3955**	**0.3962**	**0.4026**	**0.4089**	**0.4149**	**0.4213**	**0.4279**	**0.4507**
k = 10											
Original	0.3750	0.3854	0.3882	0.3926	0.3979	0.4034	0.4088	0.4154	0.4208	0.4277	0.4497
GAS	0.3886	0.3879	0.3881	0.3929	0.3980	0.4031	0.4090	0.4146	0.4198	0.4261	0.4491
FSMSVM	**0.3903**	**0.3951**	**0.3936**	0.3950	0.3991	0.4049	0.4111	0.4166	0.4223	0.4285	0.4494
LifeRank	0.3852	0.3920	0.3926	**0.3976**	**0.4026**	**0.4073**	**0.4146**	**0.4206**	**0.4270**	**0.4312**	**0.4507**
k = 15											
Original	0.3750	0.3854	0.3882	0.3926	0.3979	0.4034	0.4088$^{\triangle}$	0.4154	0.4208$^{\triangle}$	0.4277$^{\triangle}$	0.4497
GAS	0.3767	0.3849	0.3913	0.3954	0.4000	0.4070	0.4133	0.4191	0.4250	0.4315	**0.4528**
FSMSVM	0.3800	0.3825	0.3855	0.3882$^{\blacktriangle}$	0.3922$^{\blacktriangle}$	0.3976$^{\blacktriangle}$	0.4035$^{\blacktriangle}$	0.4091$^{\blacktriangle}$	0.4147$^{\blacktriangle}$	0.4215$^{\blacktriangle}$	0.4441$^{\blacktriangle}$
LifeRank	**0.3842**	**0.3888**	**0.3943**	**0.3984**	**0.4019**	**0.4091**	**0.4161**	**0.4207**	**0.4274**	**0.4336**	0.4513
k = 20											
Original	0.3750	0.3854	0.3882	0.3926	0.3979	0.4034	0.4088	0.4154	0.4208	0.4277	0.4497
GAS	0.3783	**0.3897**	**0.3956**	**0.3990**	**0.4021**	**0.4079**	0.4120	0.4178	**0.4255**	**0.4313**	**0.4521**
FSMSVM	0.3742	0.3867	0.3916	0.3937	0.3970	0.4011	0.4062$^{\triangle}$	0.4132	0.4184	0.4244$^{\triangle}$	0.4483
LifeRank	**0.3828**	0.3894	0.3912	0.3948	0.4014	0.4068	**0.4121**	**0.4184**	0.4243	0.4306	0.4514

Table 2: Ranking performance on MQ2008 and selected/generated datasets.
Statistical significance shown for LifeRank against Original, GAS and FSMSVM
Performance for Original is independent of value of k (corresponds to original dataset)

Performance for RankSVM											
NDCG	@1	@2	@3	@4	@5	@6	@7	@8	@9	@10	MAP
k = 5											
Original	0.3712	0.3933$^{\blacktriangle}$	0.4238$^{\triangle}$	0.4485$^{\triangle}$	0.4672$^{\blacktriangle}$	0.4814$^{\triangle}$	0.4875$^{\triangle}$	0.4531$^{\triangle}$	0.2234	0.2284	0.4714
GAS	0.3678	0.3983$^{\triangle}$	0.4213$^{\blacktriangle}$	0.4492$^{\triangle}$	0.4665$^{\blacktriangle}$	0.4800$^{\blacktriangle}$	0.4862$^{\blacktriangle}$	0.4522$^{\blacktriangle}$	0.2207$^{\blacktriangle}$	0.2246$^{\blacktriangle}$	0.4714
FSMSVM	**0.3780**	0.4126	0.4384	0.4599	0.4761	0.4909	0.4968	0.4616	**0.2284**	0.2326	0.4776
LifeRank	0.3767	**0.4168**	**0.4389**	**0.4606**	**0.4806**	**0.4921**	**0.4976**	**0.4627**	0.2280	**0.2329**	**0.4788**
k = 10											
Original	0.3712	0.3933$^{\blacktriangle}$	0.4238$^{\triangle}$	0.4485$^{\triangle}$	0.4672$^{\blacktriangle}$	0.4814	0.4875$^{\triangle}$	0.4531$^{\blacktriangle}$	0.2234	0.2284$^{\triangle}$	0.4714
GAS	0.3698	0.4015$^{\triangle}$	0.4292	0.4565	0.4732	0.4862	0.4929	0.4551$^{\triangle}$	0.2217$^{\triangle}$	0.2266$^{\blacktriangle}$	0.4776
FSMSVM	0.3759	0.4157	0.4371	0.4589	0.4781	**0.4925**	0.4962	0.4617	0.2276	0.2322	**0.4793**
LifeRank	**0.3763**	**0.4181**	**0.4384**	**0.4612**	**0.4796**	0.4900	**0.4972**	**0.4625**	**0.2281**	**0.2345**	0.4792
k = 15											
Original	0.3712	0.3933$^{\blacktriangle}$	0.4238$^{\blacktriangle}$	0.4485$^{\blacktriangle}$	0.4672$^{\blacktriangle}$	0.4814$^{\blacktriangle}$	0.4875$^{\blacktriangle}$	0.4531$^{\blacktriangle}$	0.2234$^{\blacktriangle}$	0.2284$^{\blacktriangle}$	0.4714$^{\triangle}$
GAS	0.3720	0.3984$^{\triangle}$	0.4320	0.4533$^{\triangle}$	0.4711$^{\triangle}$	0.4851	0.4902$^{\triangle}$	0.4543$^{\blacktriangle}$	0.2223$^{\blacktriangle}$	0.2279$^{\blacktriangle}$	0.4742
FSMSVM	**0.3788**	**0.4165**	0.4351	0.4557	0.4761	0.4905	0.4967	0.4613	0.2265	0.2311	0.4788
LifeRank	0.3771	0.4140	**0.4379**	**0.4622**	**0.4804**	**0.4920**	**0.4972**	**0.4633**	**0.2310**	**0.2349**	**0.4792**
k = 20											
Original	0.3712	0.3933	0.4238$^{\triangle}$	0.4485$^{\blacktriangle}$	0.4672$^{\triangle}$	0.4814$^{\triangle}$	0.4875$^{\blacktriangle}$	0.4531$^{\blacktriangle}$	0.2234$^{\triangle}$	0.2284$^{\triangle}$	0.4714
GAS	0.3656	0.4027	0.4313	0.4527$^{\triangle}$	0.4720	0.4850	0.4921	0.4566	0.2254	0.2298	0.4719
FSMSVM	**0.3737**	**0.4144**	0.4343	0.4588	0.4757	0.4902	0.4946	0.4590	0.2249	0.2302	**0.4754**
LifeRank	0.3712	0.4045	**0.4363**	**0.4619**	**0.4763**	**0.4903**	**0.4971**	**0.4617**	**0.2293**	**0.2338**	0.4751
Performance for Linear Regression											
NDCG	@1	@2	@3	@4	@5	@6	@7	@8	@9	@10	MAP
k = 5											
Original	0.3465	0.3617$^{\blacktriangle}$	0.3961	0.4225	0.4407	0.4558$^{\triangle}$	0.4684$^{\triangle}$	0.4746$^{\blacktriangle}$	0.4806$^{\blacktriangle}$	0.4871$^{\blacktriangle}$	0.4550$^{\blacktriangle}$
GAS	0.3537	0.3691$^{\triangle}$	0.3947	0.4184	0.4390	0.4584	0.4692	0.4776	0.4838	0.4892	0.4630
FSMSVM	0.3541	0.3755$^{\triangle}$	0.4002	**0.4191**	0.4396$^{\triangle}$	0.4563$^{\blacktriangle}$	0.4681$^{\triangle}$	0.4791$^{\triangle}$	0.4842$^{\triangle}$	0.4903$^{\triangle}$	0.4593$^{\triangle}$
LifeRank	**0.3691**	**0.3907**	**0.4089**	0.3955	**0.4491**	**0.4664**	**0.4772**	**0.4864**	**0.4924**	**0.4979**	**0.4685**
k = 10											
Original	0.3465$^{\triangle}$	0.3617$^{\blacktriangle}$	0.3961$^{\triangle}$	0.4225	0.4407$^{\triangle}$	0.4558$^{\triangle}$	0.4684$^{\blacktriangle}$	0.4746$^{\blacktriangle}$	0.4806$^{\blacktriangle}$	0.4871$^{\blacktriangle}$	0.4550$^{\blacktriangle}$
GAS	0.3605	0.3804	0.3990	**0.4282**	0.4490	0.4628	0.4742	0.4840	0.4895	0.4942	0.4669
FSMSVM	0.3512	0.3779	0.4017	0.4213	0.4442	0.4601	0.4702$^{\triangle}$	0.4789$^{\triangle}$	0.4847	0.4910	0.4622
LifeRank	**0.3698**	**0.3857**	**0.4085**	0.3976	**0.4501**	**0.4659**	**0.4791**	**0.4859**	**0.4908**	**0.4970**	**0.4686**
k = 15											
Original	0.3465	0.3617$^{\blacktriangle}$	0.3961	0.4225	0.4407	0.4558$^{\triangle}$	0.4684	0.4746$^{\blacktriangle}$	0.4806$^{\triangle}$	0.4871$^{\triangle}$	0.4550$^{\triangle}$
GAS	**0.3652**	0.3795	**0.4114**	**0.4302**	0.4497	0.4641	0.4757	0.4845	**0.4914**	**0.4964**	**0.4686**
FSMSVM	0.3631	0.3822	0.4053	0.4285	**0.4527**	**0.4669**	**0.4774**	0.4850	0.4901	0.4956	0.4647
LifeRank	0.3618	**0.3842**	0.4032	0.4296	0.4482	0.4657	0.4766	**0.4852**	0.4904	0.4955	0.4662
k = 20											
Original	0.3465$^{\blacktriangle}$	0.3617$^{\blacktriangle}$	0.3961	0.4225	0.4407$^{\triangle}$	0.4558	0.4684	0.4746$^{\triangle}$	0.4806$^{\blacktriangle}$	0.4871$^{\blacktriangle}$	0.4550$^{\blacktriangle}$
GAS	0.3588	0.3801	**0.4100**	**0.4300**	0.4486	0.4650	0.4744	**0.4843**	**0.4906**	**0.4961**	0.4660
FSMSVM	0.3601$^{\triangle}$	0.3803	0.4075	0.4266	0.4525	**0.4668**	**0.4772**	0.4833	0.4889	0.4945	0.4661
LifeRank	**0.3712**	**0.3820**	0.4018	0.3948	**0.4495**	0.4628	0.4748	0.4833	0.4897	0.4956	**0.4662**

Table 3: Ranking performance on OHSUMED and selected/generated datasets.
Statistical significance shown for LifeRank against Original, GAS and FSMSVM
Performance for Original is independent of value of k (corresponds to original dataset)

Performance for RankSVM

NDCG	@1	@2	@3	@4	@5	@6	@7	@8	@9	@10	MAP
					k = 5						
Original	0.5416	0.5076	0.4775	0.4565$^{\triangle}$	0.4439$^{\blacktriangle}$	0.4452	0.4433	0.4405	0.4338$^{\blacktriangle}$	0.4300$^{\blacktriangle}$	0.4345$^{\blacktriangle}$
GAS	0.5332	0.4901	0.4739	0.4630$^{\triangle}$	0.4578$^{\blacktriangle}$	0.4503$^{\triangle}$	0.4432$^{\blacktriangle}$	0.4398$^{\triangle}$	0.4374$^{\blacktriangle}$	0.4340$^{\blacktriangle}$	0.4642$^{\blacktriangledown}$
FSMSVM	**0.5771**$^{\triangledown}$	0.4889	0.4772	0.4749	0.4694$^{\triangle}$	0.4609	0.4622	0.4559	0.4529	0.4518	**0.4728**
LifeRank	0.5170	**0.5000**	**0.5015**	**0.4950**	**0.4905**	**0.4749**	**0.4701**	**0.4628**	**0.4684**	**0.4668**	0.4523
					k = 10						
Original	0.5416	0.5076	0.4775	0.4565$^{\blacktriangle}$	0.4439$^{\blacktriangle}$	0.4452$^{\blacktriangle}$	0.4433$^{\blacktriangle}$	0.4405$^{\blacktriangle}$	0.4338$^{\blacktriangle}$	0.4300$^{\blacktriangle}$	0.4345$^{\blacktriangle}$
GAS	**0.5677**	**0.5390**	0.5088	0.4944	0.4873	0.4673	0.4652	0.4605	0.4549	0.4507$^{\triangle}$	0.4466
FSMSVM	0.5296	0.4866$^{\blacktriangle}$	0.4794$^{\blacktriangle}$	0.4745$^{\blacktriangle}$	0.4636$^{\blacktriangle}$	0.4602$^{\triangle}$	0.4558$^{\blacktriangle}$	0.4531$^{\triangle}$	0.4484$^{\blacktriangle}$	0.4463$^{\blacktriangle}$	0.4459
LifeRank	0.5518	0.5373	**0.5185**	**0.5053**	**0.4910**	**0.4811**	**0.4774**	**0.4699**	**0.4692**	**0.4663**	**0.4505**
					k = 15						
Original	0.5416	0.5076	0.4775	0.4565$^{\triangle}$	0.4439$^{\blacktriangle}$	0.4452$^{\triangle}$	0.4433	0.4405$^{\triangle}$	0.4338$^{\blacktriangle}$	0.4300$^{\blacktriangle}$	0.4345$^{\blacktriangle}$
GAS	0.5771	0.5068	0.4850	0.4713	0.4656	0.4552	0.4524	0.4494	0.4439$^{\triangle}$	0.4419$^{\blacktriangle}$	0.4402$^{\blacktriangle}$
FSMSVM	**0.5834**	0.5317	0.5021	0.4856	0.4723	0.4660	0.4606	0.4557	0.4525	0.4500$^{\triangle}$	0.4452$^{\blacktriangle}$
LifeRank	0.5769	**0.5344**	**0.5065**	**0.4942**	**0.4835**	**0.4713**	**0.4672**	**0.4630**	**0.4632**	**0.4628**	**0.4536**
					k = 20						
Original	0.5416	0.5076	0.4775$^{\triangle}$	0.4565$^{\blacktriangle}$	0.4439$^{\blacktriangle}$	0.4452$^{\blacktriangle}$	0.4433$^{\blacktriangle}$	0.4405$^{\blacktriangle}$	0.4338$^{\blacktriangle}$	0.4300$^{\blacktriangle}$	0.4345$^{\blacktriangle}$
GAS	0.5519	0.5051	0.4838$^{\triangle}$	0.4761$^{\triangle}$	0.4710$^{\blacktriangle}$	0.4520$^{\blacktriangle}$	0.4473$^{\blacktriangle}$	0.4463$^{\blacktriangle}$	0.4401$^{\blacktriangle}$	0.4405$^{\blacktriangle}$	0.4387$^{\blacktriangle}$
FSMSVM	0.5173$^{\triangle}$	0.4848$^{\triangle}$	0.4816$^{\triangle}$	0.4766	0.4642$^{\blacktriangle}$	0.4522$^{\blacktriangle}$	0.4452$^{\blacktriangle}$	0.4411$^{\blacktriangle}$	0.4388$^{\blacktriangle}$	0.4387$^{\blacktriangle}$	0.4441$^{\blacktriangle}$
LifeRank	**0.5805**	**0.5416**	**0.5204**	**0.5029**	**0.4976**	**0.4834**	**0.4765**	**0.4725**	**0.4669**	**0.4656**	**0.4519**

Performance for Linear Regression

NDCG	@1	@2	@3	@4	@5	@6	@7	@8	@9	@10	MAP
					k = 5						
Original	0.4830	0.4800	0.4749	0.4548	0.4483	0.4368	0.4389	0.4343	0.4311	0.4302	0.4333
GAS	0.4762	0.4491	0.4489	0.4466	0.4378	0.4286	0.4275	0.4209	0.4201	0.4202	0.4549
FSMSVM	**0.5271**	**0.4852**	**0.4686**	**0.4589**	**0.4513**	0.4403	0.4342	0.4335	0.4306	0.4292	**0.4655**
LifeRank	0.4941	0.4736	0.4554	0.4586	0.4485	**0.4459**	**0.4425**	**0.4402**	**0.4347**	**0.4341**	0.4599
					k = 10						
Original	0.4830	0.4800	0.4749	0.4548	0.4483	0.4368	0.4389	0.4343	0.4311	0.4302	0.4333
GAS	0.5202	**0.4883**	**0.4680**	0.4597	0.4543	0.4460	0.4415	0.4354	0.4312	0.4275	0.4339
FSMSVM	0.5082	0.4556	0.4502	0.4395	0.4333	0.4245	0.4259	0.4223	0.4216	0.4180	0.4344
LifeRank	**0.5330**	0.4866	**0.4680**	**0.4681**	**0.4601**	**0.4494**	**0.4436**	**0.4423**	**0.4370**	**0.4366**	**0.4358**
					k = 15						
Original	0.4830	0.4800	0.4749	0.4548	0.4483	0.4368	0.4389	0.4343	0.4311	0.4302	0.4333
GAS	0.5105	**0.4941**	0.4791	0.4654	0.4510	0.4412	0.4351	0.4331	0.4303	0.4301	0.4298$^{\triangle}$
FSMSVM	0.4984	0.4611	0.4639	0.4567	0.4485	0.4400	0.4325	0.4321	0.4302	0.4282	**0.4405**
LifeRank	**0.5342**	0.4839	**0.5001**	**0.4781**	**0.4678**	**0.4529**	**0.4437**	**0.4412**	**0.4370**	**0.4353**	0.4399
					k = 20						
Original	0.4830	0.4800	0.4749	0.4548	0.4483	0.4368	0.4389	0.4343	0.4311	0.4302	0.4333
GAS	0.5050	0.4943$^{\triangle}$	0.4813$^{\blacktriangle}$	**0.4706**$^{\blacktriangledown}$	0.4546$^{\blacktriangle}$	**0.4497**$^{\triangledown}$	0.4389$^{\blacktriangle}$	0.4367$^{\blacktriangle}$	0.4324$^{\blacktriangle}$	0.4299$^{\blacktriangle}$	0.4308$^{\blacktriangle}$
FSMSVM	0.4984	0.4706	0.4574	0.4510	0.4489	0.4416	0.4370	0.4365	0.4341	0.4315	**0.4369**
LifeRank	**0.5264**	**0.5021**	**0.4882**	0.4686	**0.4571**	0.4467	**0.4439**	**0.4406**	**0.4359**	**0.434**1	**0.4369**

Overall, from the tables we can see that: (1) The performance of ranking algorithms can be maintained or slightly improved on the datasets generated by dimension reduction techniques. (2) The performance of the ranking algorithms on the datasets generated by LifeRank is higher than those generated by GAS and FSMRank in most cases. Let us now take a closer look.

6.1.1 Performance of RankSVM

For RankSVM, we can see that LifeRank clearly shows improvements over the original datasets for all the three benchmarks (MQ2007, MQ2008 and OHSUMED) in terms of NDCG@1–10 as well as MAP. The only exception is MQ2007 for $k = 5$, where the performance of LifeRank as well as the other generated datasets does not beat the original dataset. We can also see from the tables that LifeRank clearly outperforms other generated datasets (GAS and FSMSVM) on NDCG@1–10 for all the benchmarks and all values of k.

In terms of MAP, LifeRank outperforms the other generated datasets in most cases. The few exceptions include the case for MQ2007, when GAS has a higher MAP for $k = 5$. For MQ2008, FSMSVM attains slightly higher MAP score than LifeRank for $k = 10$ and $k = 20$, but these differences are not significant. Also, for OHSUMED when $k = 5$, the MAP score attained by LifeRank is lower than FSMSVM and GAS, but it is still an improvement over the original dataset.

6.1.2 Performance of Linear Regression

Also in the case of Linear Regression, for all three benchmarks (MQ2007, MQ2008 and OHSUMED) LifeRank clearly shows improvements over the original datasets in terms of NDCG@1–10 as well as MAP. The only exception is MQ2007 for $k = 5$, where the original dataset performs better than LifeRank as well as the other generated datasets.

On NDCG@1–10, for MQ2007 LifeRank gives the best performance for all values of k, except for $k = 20$, where GAS gives the best performance. For MQ2008, LifeRank gives the best performances for $k = 5$ and $k = 10$ on NDCG@1–10. However, for $k = 15$ and $k = 20$, there is mixed performance where all GAS, FSMSVM and LifeRank give best performances in certain cases. For, OHSUMED, LifeRank gives the best performance on NDCG@1–10 in most cases.

In terms of MAP, LifeRank gives the best performance for MQ2007 for $k = 5$ and $k = 10$, whereas for $k = 15$ and $k = 20$ the best performance is given by GAS. For MQ2008, LifeRank outperforms others for all values of k, except for $k = 15$ where the best performance is given by GAS. Moreover, for OHSUMED, FSMSVM outperforms the others for $k = 5$ and $k = 15$, while LifeRank gives the best performance for $k = 10$. In case of $k = 20$, there is a tie between LifeRank and FSMSVM.

6.1.3 Statistical Significance Overview

In Tables 1, 2 and 3, markups are provided to denote the statistical significance between LifeRank and the following baselines: original dataset, GAS and FSMSVM. It should be noted that the original dataset is independent of the values of k, but is repeated in the table to indicate statistical significance between it and datasets generated by LifeRank for different values of k.

It can be observed from Table 1 that for MQ2007 in the case of RankSVM, there is strong to weak significance between LifeRank and the baselines in most cases across the metrics, while there is no significance shown against original for $k = 15$. Moreover, for $k = 5$, significance is shown against original and GAS in few cases. For Linear Regression, there is strong significance shown against FSMSVM for $k = 5$ and $k = 15$, though there is not much significance shown for $k = 10$ and $k = 20$. Also, weak significance is shown against GAS in few cases for $k = 5$ and against original for $k = 15$.

Table 2 for MQ2008 shows no statistically significant differences between LifeRank and FSMSVM for RankSVM. There is weak to strong statistical significance for LifeRank against original dataset for most cases and against GAS mainly for $k = 5, 10$ and 15. For Linear Regression, LifeRank shows weak to strong statistical significance against original in most cases, GAS in no cases and FSMSVM in few cases. Moreover, Table 3 for OHSUMED shows statistical significance for RankSVM in many cases against the baselines, whereas there is statistical significance observed for Liner Regression for few cases. The comparative lack of statistical significance seen for MQ2008 and OHSUMED can most probably be attributed to the relatively small size of these datasets.

6.2 Quality of the generated features

Table 4 lists the quality scores of the features from four datasets: the original datasets and the three datasets generated by GAS, FSMRank and LifeRank, respectively, for $k = 10$.

Table 4: Performance of the generated features.

Datasets	MQ2007		MQ2008		OHSUMED	
	Imp	*Rdd*	*Imp*	*Rdd*	*Imp*	*Rdd*
Original	0.2671	0.4833	0.3297	0.5318	0.3763	0.5592
GAS	0.2643	**0.3242**	0.3235	**0.3308**	0.3603	**0.3904**
FSMRank	0.3005	0.4706	0.3723	0.5276	0.4170	0.5412
LifeRank	**0.3214**	0.4606	**0.4095**	0.5758	**0.4422**	0.8881

From the table we see that: (1) GAS can significantly reduce the redundancy of the features. The redundancy of the features selected by GAS is the lowest among the four datasets. However, the importance of the features selected by GAS is also lowest and even slightly lower than that of the original datasets. (2) FSMRank can improve

the importance of the features while reducing their redundancy, but the differences in terms redundancy are subtle. (3) LifeRank can sharply improve the importance of the features. The importance of the features generated by LifeRank is highest among the four datasets. Besides, the redundancy of the features can also be slightly reduced by LifeRank for MQ2007 but deteriorated for MQ2008 and OHSUMED. Worse redundancy for LifeRank in comparison with GAS and FSMSVM could be because of the reason that, while these baselines are feature selection methods, for LifeRank each extracted feature is a linear combination of the original features. Moreover, it can be observed that for the larger dataset MQ2007, redundancy for LifeRank is comparable to the baselines, and even better than FSMSVM. However, for smaller dataset MQ2008, the redundancy is worse than the baselines. For the smallest dataset OHSUMED, it is worse than the baselines by a greater difference.

6.3 Effect of the Orthonormality Constraints

To confirm that the orthonormality constraints used in LifeRank do indeed contribute to performance gains, we re-generated the datasets for the benchmarks MQ2007, MQ2008 and OHSUMED using LifeRank for $k = 10$, but this time without the incorporation of the constraints in its algorithm in Phase I (see Algorithm 1, line 1–8). Table 5 shows the comparison of performances of ranking algorithms, for datasets generated by LifeRank and LifeRank without orthonormality constraints (represented by LifeRank^{NO}). Moreover, markups are presented in the table to denote to the statistical significance between LifeRank and LifeRank^{NO}.

From the results in Table 5 we see that the datasets generated by LifeRank show significant improvements in performance over the datasets generated by LifeRank^{NO} for both learning to rank algorithms, RankSVM and Linear Regression. Performance gains can be observed on all three benchmarks and across all performance measures (NDCG@1–10 and MAP). Hence, these results show that the usage of orthonormality constraints is beneficial in the LifeRank algorithm. Also, strong statistical significance between LifeRank and LifeRank^{NO} can be observed for all three benchmarks for RankSVM as well as Linear Regression, across all performance measures, except for a small number of cases where weak or no statistical significance is seen.

7 Conclusion

In this paper, we have addressed the feature extraction problem for learning to rank, and have proposed LifeRank, a linear feature extraction algorithm for ranking. LifeRank regards each dataset for ranking as a matrix, referred to as the *original matrix*. We then optimize a *transformation matrix* by minimizing a classic pairwise learning to rank loss function, so that we can discover the optimal one that matches the ranking task. Then a new matrix (dataset) can be generated by the product of original matrix and transformation matrix. Extensive experiments on benchmark datasets show the performance gains of LifeRank in comparison with the state-of-the-art algorithms.

The performance of LifeRank has been evaluated for RankSVM and Linear Regression. In future work, its benefits for other learning to rank algorithms could be

Table 5: Effect of orthonormality constraints on datasets for $k = 10$.
Statistical significance shown for LifeRank against LifeRank^{NO}

Performance for RankSVM											
NDCG	@1	@2	@3	@4	@5	@6	@7	@8	@9	@10	MAP
MQ2007											
LifeRank	**0.4037**$^{\blacktriangle}$	**0.4023**$^{\blacktriangle}$	**0.4089**$^{\blacktriangle}$	**0.4117**$^{\blacktriangle}$	**0.4161**$^{\blacktriangle}$	**0.4215**$^{\blacktriangle}$	**0.4264**$^{\blacktriangle}$	**0.4312**$^{\blacktriangle}$	**0.4370**$^{\blacktriangle}$	**0.4423**$^{\blacktriangle}$	**0.4634**$^{\blacktriangle}$
LifeRank^{NO}	0.3808	0.3881	0.3923	0.3939	0.3996	0.4044	0.4081	0.4148	0.4198	0.4269	0.4528
MQ2008											
LifeRank	**0.3763**	**0.4181**$^{\blacktriangle}$	**0.4384**$^{\triangle}$	**0.4612**$^{\blacktriangle}$	**0.4796**$^{\blacktriangle}$	**0.4900**$^{\triangle}$	**0.4972**$^{\triangle}$	**0.4625**$^{\triangle}$	**0.2281**$^{\blacktriangle}$	**0.2345**$^{\blacktriangle}$	**0.4792**$^{\triangle}$
LifeRank^{NO}	0.3618	0.3987	0.4264	0.4429	0.4657	0.4807	0.4887	0.4552	0.2199	0.2236	0.4704
OHSUMED											
LifeRank	0.5518	**0.5373**$^{\triangle}$	**0.5185**$^{\blacktriangle}$	**0.5053**$^{\blacktriangle}$	**0.4910**$^{\blacktriangle}$	**0.4811**$^{\triangle}$	**0.4774**$^{\blacktriangle}$	**0.4699**$^{\blacktriangle}$	**0.4692**$^{\blacktriangle}$	**0.4663**$^{\blacktriangle}$	**0.4505**$^{\blacktriangle}$
LifeRank^{NO}	**0.5703**	0.4900	0.4707	0.4673	0.4595	0.4522	0.4450	0.4399	0.4360	0.4312	0.4404
Performance for Linear Regression											
NDCG	@1	@2	@3	@4	@5	@6	@7	@8	@9	@10	MAP
MQ2007											
LifeRank	**0.3852**$^{\blacktriangle}$	**0.3920**$^{\blacktriangle}$	**0.3926**$^{\blacktriangle}$	**0.3976**$^{\blacktriangle}$	**0.4026**$^{\blacktriangle}$	**0.4073**$^{\blacktriangle}$	**0.4146**$^{\blacktriangle}$	**0.4206**$^{\blacktriangle}$	**0.4270**$^{\blacktriangle}$	**0.4312**$^{\blacktriangle}$	**0.4507**$^{\blacktriangle}$
LifeRank^{NO}	0.3584	0.3691	0.3729	0.3764	0.3816	0.3862	0.3918	0.3980	0.4034	0.4084	0.4338
MQ2008											
LifeRank	**0.3698**$^{\blacktriangle}$	**0.3857**$^{\blacktriangle}$	**0.4085**$^{\blacktriangle}$	**0.3976**$^{\blacktriangle}$	**0.4501**$^{\blacktriangle}$	**0.4659**$^{\blacktriangle}$	**0.4791**$^{\blacktriangle}$	**0.4859**$^{\blacktriangle}$	**0.4908**$^{\blacktriangle}$	**0.4970**$^{\blacktriangle}$	**0.4686**$^{\blacktriangle}$
LifeRank^{NO}	0.3295	0.3519	0.3689	0.3934	0.4144	0.4337	0.4446	0.4550	0.4615	0.4682	0.4346
OHSUMED											
LifeRank	**0.5330**	**0.4866**	**0.4680**$^{\triangle}$	**0.4681**$^{\blacktriangle}$	**0.4601**$^{\blacktriangle}$	**0.4494**$^{\blacktriangle}$	**0.4436**$^{\blacktriangle}$	**0.4423**$^{\blacktriangle}$	**0.4370**$^{\blacktriangle}$	**0.4366**$^{\blacktriangle}$	**0.4358**$^{\blacktriangle}$
LifeRank^{NO}	0.4571	0.4383	0.4111	0.4014	0.3915	0.3844	0.3784	0.3710	0.3689	0.3632	0.3963

analysed. Moreover, nonlinear feature extraction techniques like some kernel tricks could be incorporated in LifeRank to further improve its performance. Besides, we plan to try more learning to rank loss functions like some state-of-the-art listwise loss functions for performance gains of our algorithm. In addition, we believe it would be interesting to establish theoretical results on dimension reduction for ranking, including feature extraction and feature selection-based algorithms, especially concerning retrieval performance.

Acknowledgements We would like to thank our anonymous reviewers for valuable comments and suggestions.

This research was supported by Ahold Delhaize, Amsterdam Data Science, the Bloomberg Research Grant program, the Dutch national program COMMIT, Elsevier, the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement nr 312827 (VOX-Pol), the Google Faculty Research Award program, the Microsoft Research Ph.D. program, the Netherlands Institute for Sound and Vision, the Netherlands Organization for Scientific Research (NWO) under project nrs CI-14-25, 652.-002.001, 612.001.551, 652.001.003, and Yandex.

All content represents the opinion of the author(s), which is not necessarily shared or endorsed by their respective employers and/or sponsors.

References

1. Arfken GB (2013) Mathematical Methods for Physicists. Academic Press
2. Bach FR, Jordan MI (2005) A probabilistic interpretation of canonical correlation analysis. Tech. Rep. 688, Department of Statistics, University of California, Berkeley
3. Baeza-Yates R, Ribeiro-Neto B (1999) Modern Information Retrieval. Addison Wesley
4. Bertsekas DP (1999) Nonlinear Programming. Athena Scientific
5. Blum AL, Langley P (1997) Selection of relevant features and examples in machine learning. Artif Intell 97(1):245–271
6. Burges C, Shaked T, Renshaw E, Lazier A, Deeds M, Hamilton N, Hullender G (2005) Learning to rank using gradient descent. In: ICML, pp 89–96
7. Burges CJ, Ragno R, Le QV (2007) Learning to rank with nonsmooth cost functions. In: NIPS, pp 193–200
8. Busa-Fekete R, Kégl B, Éltetö T, Szarvas G (2013) Tune and mix: Learning to rank using ensembles of calibrated multi-class classifiers. Machine Learning 93(2-3):261–292
9. Cao Y, Xu J, Liu TY, Li H, Huang Y, Hon HW (2006) Adapting ranking SVM to document retrieval. In: SIGIR, pp 186–193
10. Cao Z, Qin T, Liu TY, Tsai MF, Li H (2007) Learning to rank: From pairwise approach to listwise approach. In: ICML, pp 129–136
11. Chapelle O, Chang Y, Liu TY (2011) Future directions in learning to rank. Journal of Machine Learning Research 14:91–100
12. Chen W, Liu TY, Lan Y, Ma ZM, Li H (2009) Ranking measures and loss functions in learning to rank. In: NIPS, pp 315–323
13. Cossock D, Zhang T (2008) Statistical analysis of Bayes optimal subset ranking. IEEE Transactions on Information Theory 54(11):5140–5154
14. Crammer K, Singer Y (2001) Pranking with ranking. In: NIPS, pp 641–647
15. Freund Y, Iyer R, Schapire RE, Singer Y (2003) An efficient boosting algorithm for combining preferences. Journal of Machine Learning Research 4(1):933–969
16. Geng X, Liu T, Qin T, Li H (2007) Feature selection for ranking. In: SIGIR, pp 407–414
17. Gupta P, Rosso P (2012) Expected divergence based feature selection for learning to rank. In: COLING, pp 431–440
18. Hardoon DR, Szedmak S, Shawe-Taylor J (2004) Canonical correlation analysis: An overview with application to learning methods. Neural Computation 16(12):2639–2664
19. Järvelin K, Kekäläinen J (2002) Cumulated gain-based evaluation of IR techniques. ACM Transactions on Information Systems 20(4):422–446
20. Joachims T, Li H, Liu TY, Zhai C (2007) Learning to rank for information retrieval (LR4IR 2007). SIGIR Forum 41(2):58–62
21. Joachims T, Finley T, Yu CNJ (2009) Cutting-plane training of structural SVMs. Machine Learning 77(1):27–59
22. Joachims T, Swaminathan A, de Rijke M (2018) Deep learning with logged bandit feedback. In: ICLR 2018

23. Jolliffe I (2002) Principal Component Analysis. Springer Series in Statistics, Springer
24. Kendall MG (1948) Rank Correlation Methods. C. Griffin
25. Koren Y, Bell R, Volinsky C (2009) Matrix factorization techniques for recommender systems. Computer 42(8):30–37
26. Lai H, Pan Y, Tang Y, Yu R (2013) FSMRank: Feature selection algorithm for learning to rank. IEEE Transactions on Neural Networks and Learning Systems 24(6):940–952
27. Lan Y, Guo J, Cheng X, Liu TY (2012) Statistical consistency of ranking methods in a rank-differentiable probability space. In: NIPS, pp 1232–1240
28. Lange K (2010) Singular value decomposition. In: Numerical Analysis for Statisticians, Springer, pp 129–142
29. Laporte L, Flamary R, Canu S, Déjean S, Mothe J (2014) Nonconvex regularizations for feature selection in ranking with sparse SVM. IEEE Transactions on Neural Networks and Learning Systems 25(6):1118–1130
30. Lawson C, Hanson R (1995) Solving Least Square Problems, Classics in Applied Mathematics, vol 15. SIAM, Philadelphia
31. Li P, Wu Q, Burges CJ (2007) McRank: Learning to rank using multiple classification and gradient boosting. In: NIPS, pp 897–904
32. Liu TY (2009) Learning to rank for information retrieval. Foundations and Trends in Information Retrieval 3(3):225–331
33. Liu TY (2011) Learning to Rank for Information Retrieval. Springer Science & Business Media
34. Metzler DA (2007) Automatic feature selection in the markov random field model for information retrieval. In: CIKM, ACM, pp 253–262
35. Motoda H, Liu H (2002) Feature selection, extraction and construction. Communication of IICM (Institute of Information and Computing Machinery, Taiwan) 5:67–72
36. Mukuta Y, Harada T (2014) Probabilistic partial canonical correlation analysis. In: ICML, pp 1449–1457
37. Naini KD, Altingövde IS (2014) Exploiting result diversification methods for feature selection in learning to rank. In: ECIR, Springer, pp 455–461
38. Ng AY (2004) Feature selection, L1 vs. L2 regularization, and rotational invariance. In: ICML, pp 78–82
39. Niu S, Guo J, Lan Y, Cheng X (2012) Top-K learning to rank: Labeling, ranking and evaluation. In: SIGIR, pp 751–760
40. Pan F, Converse T, Ahn D, Salvetti F, Donato G (2009) Feature selection for ranking using boosted trees. In: CIKM, ACM, pp 2025–2028
41. Platt JC, Barr AH (1988) Constrained differential optimization for neural networks. Tech. Rep. TR-88-17, Department of Computer Science, California Institute of Technology
42. Qin T, Liu TY (2013) Introducing LETOR 4.0 datasets. arXiv 1306.2597
43. Qin T, Liu TY, Xu J, Li H (2010) LETOR: A benchmark collection for research on learning to rank for information retrieval. Information Retrieval 13(4):346–374

44. Roweis ST, Saul LK (2000) Nonlinear dimensionality reduction by locally linear embedding. Science 290:2323–2326
45. Schölkopf B, Smola A, Müller KR (1998) Nonlinear component analysis as a kernel eigenvalue problem. Neural Computation 10(5):1299–1319
46. Schuth A, Oosterhuis H, Whiteson S, de Rijke M (2016) Multileave gradient descent for fast online learning to rank. In: WSDM 2016: The 9th International Conference on Web Search and Data Mining, ACM, pp 457–466
47. Severyn A, Moschitti A (2015) Learning to rank short text pairs with convolutional deep neural networks. In: SIGIR, ACM, pp 373–382
48. Shalit U, Chechik G (2014) Coordinate-descent for learning orthogonal matrices through Givens rotations. In: ICML, pp 548–556
49. Shivanna R, Bhattacharyya C (2014) Learning on graphs using orthonormal representation is statistically consistent. In: NIPS, pp 3635–3643
50. Tenenbaum JB, de Silva V, Langford JC (2000) A global geometric framework for nonlinear dimensionality reduction. Science 290(5500):2319–2323
51. Tipping ME, Bishop CM (1999) Probabilistic principal component analysis. Journal of the Royal Statistical Society, Series B 61(3):611–622
52. Tsai MF, Liu TY, Qin T, Chen HH, Ma WY (2007) FRank: A ranking method with fidelity loss. In: SIGIR, ACM, pp 383–390
53. Valizadegan H, Jin R, Zhang R, Mao J (2009) Learning to rank by optimizing NDCG measure. In: NIPS, pp 1883–1891
54. Volkovs M, Zemel RS (2009) Boltzrank: Learning to maximize expected ranking gain. In: ICML, pp 1089–1096
55. Wang S, Wu Y, Gao BJ, Wang K, Lauw HW, Ma J (2015) A cooperative coevolution framework for parallel learning to rank. IEEE Transactions on Knowledge and Data Engineering 27(12):3152–3165
56. Weston J, Mukherjee S, Chapelle O, Pontil M, Poggio T, Vapnik V (2000) Feature selection for SVMs. In: NIPS, pp 668–674
57. Wolf L, Bileschi S (2005) Combining variable selection with dimensionality reduction. In: CVPR, pp 801–806
58. Wyse N, Dubes R, Jain A (1980) A critical evaluation of intrinsic dimensionality algorithms. In: Gelsema E, Kanal L (eds) Pattern Recognition in Practice. Proc. workshop Amsterdam, May 1980, North-Holland, pp 415–425
59. Xu J, Li H (2007) AdaRank: A boosting algorithm for information retrieval. In: SIGIR, ACM, pp 391–398
60. Yu H, Oh J, Han W (2009) Efficient feature weighting methods for ranking. In: CIKM, ACM, pp 1157–1166
61. Yue Y, Finley T, Radlinski F, Joachims T (2007) A support vector method for optimizing average precision. In: SIGIR, ACM, pp 271–278