Hindawi
Computational Intelligence and Neuroscience
Volume 2017, Article ID 4629534, 9 pages
https://doi.org/10.1155/2017/4629534

Research Article

Ranking Support Vector Machine with Kernel Approximation

Kai Chen,[1] Rongchun Li,[1] Yong Dou,[1] Zhengfa Liang,[2] and Qi Lv[1]

[1]*National Laboratory for Parallel and Distributed Processing, National University of Defense Technology, Changsha, China*
[2]*College of Computer, National University of Defense Technology, Changsha, China*

Correspondence should be addressed to Rongchun Li; rongchunli@nudt.edu.cn

Received 18 November 2016; Revised 12 January 2017; Accepted 18 January 2017; Published 13 February 2017

Academic Editor: Carlos M. Travieso-González

Learning to rank algorithm has become important in recent years due to its successful application in information retrieval, recommender system, and computational biology, and so forth. Ranking support vector machine (RankSVM) is one of the state-of-art ranking models and has been favorably used. Nonlinear RankSVM (RankSVM with nonlinear kernels) can give higher accuracy than linear RankSVM (RankSVM with a linear kernel) for complex nonlinear ranking problem. However, the learning methods for nonlinear RankSVM are still time-consuming because of the calculation of kernel matrix. In this paper, we propose a fast ranking algorithm based on kernel approximation to avoid computing the kernel matrix. We explore two types of kernel approximation methods, namely, the Nyström method and random Fourier features. Primal truncated Newton method is used to optimize the pairwise L2-loss (squared Hinge-loss) objective function of the ranking model after the nonlinear kernel approximation. Experimental results demonstrate that our proposed method gets a much faster training speed than kernel RankSVM and achieves comparable or better performance over state-of-the-art ranking algorithms.

1. Introduction

Learning to rank is an important research area in machine learning. It has attracted the interests of many researchers because of its growing application in areas like information retrieval systems [1], recommender systems [2, 3], machine translation, and computational biology [4]. For example, in document retrieval domain, a ranking model is trained based on the training data of some queries. Each query contains a group of corresponding retrieved documents and their relevance levels labeled by humans. When a new query arrives for prediction, the trained model is used to rank the corresponding retrieved documents for the query.

Many types of machine learning algorithms have been proposed for the ranking problem. Among them, RankSVM [5], which is extended from the basic support vector machine (SVM) [6], is one of the commonly used methods. The basic idea of RankSVM is transforming the ranking problem into pairwise classification problem. The early implementation of RanSVM [7] was slow because the explicit pairwise transformation led a large number of the training samples. In order to accelerate the training process, [8] proposed a primal Newton method algorithm to solve the linear RankSVM-struct problem without the need of explicit pairwise transformation. And [9] proposed the RankSVM based on the structured output learning framework.

As with the SVM, kernel trick can be used to generalize the linear ranking problem to nonlinear case for RankSVM [7, 9]. Kernel RankSVM can give higher accuracy than the linear RankSVM for complex nonlinear ranking problem [10]. The nonlinear kernel can map the original features into some high-dimensional space where the nonlinear problem can be ranked linearly. However, the training time of kernel RankSVM dramatically grows as the training data set increases in size. The computational complexity is at least quadratic in the number of training examples because of the calculation of kernel matrix. Kernel approximation is an efficient way to solve the above problem. It can avoid computing kernel matrix by explicitly generating a vector representation of data that approximates the kernel similarity between any two data points.

The approximation methods can be classified into two categories: the Nyström method [11, 12] and random Fourier

features [13, 14]. The Nyström method approximates the kernel matrix by a low rank matrix. The random Fourier features method approximates the shift-invariant kernel based on Fourier transformation of nonnegative measure [15]. In this paper, we use the kernel approximation method to solve the problem of lengthy training time of kernel RankSVM.

To the best of our knowledge, this is the first work using the kernel approximation method to solve the learning to rank problem. We use two types of approximation methods, namely, the Nyström method or random Fourier features, to map the features into high-dimensional space. After the approximation mapping, primal truncated Newton method is used to optimize pairwise L2-loss (squared Hinge-loss) function of the RankSVM model. Experimental results demonstrate that our proposed method can achieve high performance and fast training speed than the kernel RankSVM. Compared to state-of-the-art ranking algorithms, our proposed method can also get comparable or better performance. Matlab code for our algorithm is available online (https://github.com/KaenChan/rank-kernel-appr).

2. Background and Related Works

In this section, we present the background and related works of learning to rank algorithm and RankSVM.

2.1. Learning to Rank Algorithms. Learning to rank algorithms can be classified into three categories: pointwise approach, pairwise approach, and list-wise approach.

(i) Pointwise: it transforms the ranking problem into regression or classification on single objects. Then existing regression or classification algorithms are directly applied to model the labels of single objects. This approach includes McRank [16] and OC SVM [17].

(ii) Pairwise: it transforms the ranking problem into regression or classification on object pairs. It can model the preferences within the object pairs. This approach includes RankSVM [5] and RankBoost [18].

(iii) List-wise: it takes ranking lists as instances in both learning and prediction and can optimize the list-wise loss function directly. This approach includes ListNet [19], AdaRank [20], BoltzRank [21], and SVM MAP [22].

In this paper, we focus on the pairwise ranking algorithm based on SVM.

2.2. Linear RankSVM. Linear RankSVM is a commonly used pairwise ranking algorithm [5]. For the web search problem with n queries and a set of documents of each query, features $\mathbf{x}_i \in \mathbb{R}^d$ are extracted from the query-document pair (q_i, doc_i) and label $y_i \in \mathbb{Z}$ is the relevance level of the doc_i to the query q_i. Thus, the training data is a set of label-query-instance tuples $(y_i, q_i, \mathbf{x}_i)$. Let $\mathscr{P}$ denote the set of preference pairs. If $(i, j) \in \mathscr{P}$, doc_i and doc_j are in the same query

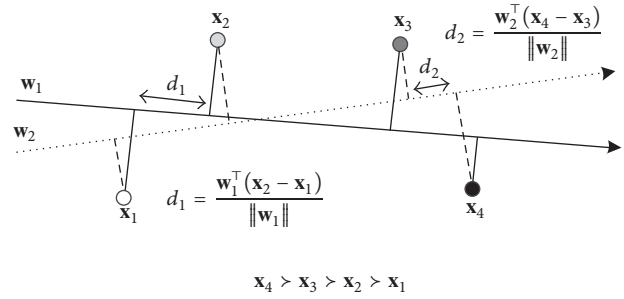

Figure 1: Margin-maximization for linear RankSVM. Four data points have the preference $\mathbf{x}_4 \succ \mathbf{x}_3 \succ \mathbf{x}_2 \succ \mathbf{x}_1$ and can be linearly ranked. d_1 and d_2 are the marginal distances for $\mathbf{w}_1$ and $\mathbf{w}_2$.

($q_i = q_j$) and doc_i is preferred over doc_j ($y_i > y_j$). The goal of linear RankSVM is to get a ranking function

$$f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} \tag{1}$$

such that $\forall (i, j) \in \mathscr{P}$, $f(\mathbf{x}_i) > f(\mathbf{x}_j) = \mathbf{w}^\top \mathbf{x}_i > \mathbf{w}^\top \mathbf{x}_j$, and $\mathbf{w} \in \mathbb{R}^d$.

RankSVM has a good generalization due to the margin-maximization property. According to [27], the margin is defined as the closest distance between two data points when the data points project to the ranking vector $\mathbf{w}$:

$$d = \min \frac{\mathbf{w}^\top \left(\mathbf{x}_i - \mathbf{x}_j\right)}{\|\mathbf{w}\|}, \quad \forall (i, j) \in \mathscr{P}. \tag{2}$$

Maximizing the margin is good because data point pairs with small margins represent very uncertain ranking decisions. RankSVM can guarantee to find a ranking vector $\mathbf{w}$ with the maximum margin [27]. Figure 1 shows the margin-maximization of four data points for linear RankSVM. The weights of two linear ranking, namely, $\mathbf{w}_1$ and $\mathbf{w}_2$, can both rank the four data correctly. But $\mathbf{w}_1$ generalizes better than $\mathbf{w}_2$ because the margin d_1 of $\mathbf{w}_1$ is larger than the margin d_2 of $\mathbf{w}_2$.

For L1-loss (Hinge-loss) linear RankSVM [5], the objective loss function is

$$\frac{1}{2}\|\mathbf{w}\|^2 + C \sum_{(i,j)\in\mathscr{P}} \max\left(0, 1 - \mathbf{w}^\top \left(\mathbf{x}_i - \mathbf{x}_j\right)\right), \tag{3}$$

where C is the regularization parameter. Equation (3) can be solved by standard SVM classification on pairwise difference vectors $(\mathbf{x}_i - \mathbf{x}_j)$. But this method is very slow because of the large size of $\mathscr{P}$.

In [8], an efficient algorithm was proposed to solve the L2-loss (squared Hinge-loss) linear RankSVM problem

$$\frac{1}{2}\|\mathbf{w}\|^2 + C \sum_{(i,j)\in\mathscr{P}} \max\left(0, 1 - \mathbf{w}^\top \left(\mathbf{x}_i - \mathbf{x}_j\right)\right)^2. \tag{4}$$

They used a $p \times n$ sparse matrix $\mathbf{A}$ to obtain the pairwise difference training sample $(\mathbf{x}_i - \mathbf{x}_j)$ implicitly ($p = |\mathscr{P}|$). If $(i, j) \in \mathscr{P}$, there exists a number k such that $\mathbf{A}_{ki} = 1$ and

$\mathbf{A}_{jk} = -1$ and the rest is 0. Let $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n]^\top$. Equation (4) can be written as

$$\frac{1}{2}\|\mathbf{w}\|^2 + C\left(\mathbf{1} - \mathbf{AXw}\right)^\top \mathbf{D}\left(\mathbf{1} - \mathbf{AXw}\right), \tag{5}$$

where $\mathbf{D}$ is a $p \times p$ diagonal matrix with $D_{(i,j)(i,j)} = 1$ if $1 - \mathbf{w}^\top(\mathbf{x}_i - \mathbf{x}_j) > 0$ and 0 otherwise. Then, (5) is optimized by primal truncated Newton method in $\mathcal{O}(nd + p)$.

2.3. Kernel RankSVM. The key of kernel method is that if kernel function κ is positive definite, there exists a mapping ϕ into the reproducing kernel Hilbert spaces (RKHS), such that

$$\kappa\left(\mathbf{x}, \mathbf{x}'\right) = \left\langle \phi\left(\mathbf{x}\right), \phi\left(\mathbf{x}'\right)\right\rangle, \tag{6}$$

where $\langle\cdot,\cdot\rangle$ denotes the inner product. The advantage of the kernel method is that the mapping ϕ never has to be calculated explicitly.

For L1-loss RankSVM, the objective loss function with the kernel mapping ϕ has the form [7]

$$\frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{(i,j)\in\mathscr{P}} \max\left(0, 1 - \mathbf{w}^\top\left(\phi\left(\mathbf{x}_i\right) - \phi\left(\mathbf{x}_j\right)\right)\right). \tag{7}$$

The primal problem of (7) can be transformed to the dual problem using the Lagrange multipliers.

$$\begin{aligned} \max_{\alpha}: \quad & \sum_{ij}\alpha_{ij} - \sum_{(i,j)\in P}\sum_{(u,v)\in P}\alpha_{ij}\alpha_{uv}Q_{ij,uv} \\ \text{s.t.:} \quad & 0 \le \alpha_{ij} \le C, \quad \forall\left(i,j\right) \in \mathscr{P}, \end{aligned} \tag{8}$$

where each Langrage multiplier α_{ij} corresponds to the pair index (i, j) in $\mathscr{P}$ and

$$\begin{aligned} Q_{ij,uv} &= \left(\phi\left(\mathbf{x}_i\right) - \phi\left(\mathbf{x}_j\right)\right)\left(\phi\left(\mathbf{x}_u\right) - \phi\left(\mathbf{x}_v\right)\right) \\ &= \kappa\left(\mathbf{x}_i, \mathbf{x}_u\right) + \kappa\left(\mathbf{x}_j, \mathbf{x}_v\right) - \kappa\left(\mathbf{x}_i, \mathbf{x}_v\right) - \kappa\left(\mathbf{x}_j, \mathbf{x}_u\right). \end{aligned} \tag{9}$$

Solving the kernel RankSVM is a large quadratic programming problem. Instead of directly computing the matrix $\mathbf{Q}$, we can save the cost by $\mathbf{A}$ in (5).

$$\mathbf{Q} = \mathbf{AKA}^\top, \quad \text{where } \mathbf{K}_{ij} = \kappa\left(\mathbf{x}_i, \mathbf{x}_j\right). \tag{10}$$

The ranking function of the kernel RankSVM has the form

$$f\left(\mathbf{x}\right) = \sum_{(i,j)\in\mathscr{P}} \alpha_{ij}\left(\kappa\left(\mathbf{x}_i, \mathbf{x}\right) - \kappa\left(\mathbf{x}_j, \mathbf{x}\right)\right). \tag{11}$$

The computation of $\mathbf{Q}$ requires $\mathcal{O}(n^2)$ kernel evaluations. It is difficult to scale to large kernel RankSVM by solving (8).

Several works have been proposed to accelerate the training speed of kernel RankSVM, such as 1-slack structural method [9], representer theorem reformulation [27], and pairwise problem reformulation [10]. However, these methods are still slow for large-scale ranking problem because the computational cost is at least quadratic in the number of training examples.

3. RankSVM with Kernel Approximation

3.1. A Unified Model. The drawback of kernel RankSVM is that it needs to store many kernel values $\kappa(\mathbf{x}_i, \mathbf{x}_j)$ during optimization. Moreover, $\kappa(\mathbf{x}_i, \mathbf{x})$ needs to be computed for new data $\mathbf{x}$ during the prediction, possibly for many vector $\mathbf{x}_i$. This problem can be solved by approximating the kernel mapping explicitly:

$$\kappa\left(\mathbf{x}, \mathbf{x}'\right) \approx \left\langle \widetilde{\phi}\left(\mathbf{x}\right), \widetilde{\phi}\left(\mathbf{x}'\right)\right\rangle, \tag{12}$$

where $\widetilde{\phi}$ is the mapping of kernel approximation. The original feature $\mathbf{x}$ can be mapped into the approximated Hilbert space by $\widetilde{\phi}$. The objective function of RankSVM with the kernel approximation can be written as

$$\frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{(i,j)\in P} \ell\left(\mathbf{w}^\top\widetilde{\phi}\left(\mathbf{x}_i\right) - \mathbf{w}^\top\widetilde{\phi}\left(\mathbf{x}_j\right)\right), \tag{13}$$

where ℓ is a loss function for SVM, such as $\ell(t) = \max(0, 1-t)$ for L1-loss SVM and $\ell(t) = \max(0, 1-t)^2$ for L2-loss SVM. The problems of (13) can be solved using linear RankSVM after the approximation mapping. The kernel never needs to be calculated during the training process. Moreover, the weights $\mathbf{w}$ can be computed directly without the need of storing any training sample. For new data $\mathbf{x}$, the ranking function is

$$f\left(\mathbf{x}\right) = \widetilde{\phi}\left(\mathbf{x}\right)^\top \mathbf{w}. \tag{14}$$

Our proposed method mainly includes mapping process and ranking process.

(i) Mapping process: the kernel approximation is used to map the original data into high dimensional space. We use two kinds of kernel approximation methods, namely, the Nyström method and random Fourier features, which will be discussed in Section 3.2.

(ii) Ranking process: the linear RankSVM is used to train a ranking model. We use the L2-loss RankSVM because of its high accuracy and fast training speed. The optimization procedure will be described in Section 3.3.

The Nyström method is data dependent and the random Fourier features method is data independent [28]. The Nyström method can usually get a better approximation than random Fourier features, whereas the Nyström method is slightly slower than the random Fourier features. Additionally, in the ranking process, we can replace the L2-loss RankSVM with any other linear ranking algorithms, such as ListNet [19] and FRank [23].

3.2. Kernel Approximation

3.2.1. Nyström Method. Nyström method gets a low-rank approximation of kernel matrix $\mathbf{K} = [\kappa(\mathbf{x}_i, \mathbf{x}_j)]_{n\times n}$ by uniformly sampling $m \ll n$ examples from $\mathbf{X}$, denoted by

Require: A, X
Ensure: $\widetilde{\phi}(\cdot)$
(1) Uniformly sample m rows from $\mathbf{X}$, denoted by $\mathbf{X}_{\text{sub}}$;
(2) $[\widehat{\mathbf{x}}_1, \ldots, \widehat{\mathbf{x}}_m] = \mathbf{X}_{\text{sub}}^\top$;
(3) $\mathbf{W} = [\kappa(\widehat{\mathbf{x}}_i, \widehat{\mathbf{x}}_j)]_{m\times m}$;
(4) Perform SVD on $\mathbf{W}$ to get $\mathbf{U}\Sigma\mathbf{U}^\top$;
(5) $\widetilde{\phi}(\mathbf{x}) = \Sigma_k^{-1/2}\mathbf{U}_k^\top[\kappa(\mathbf{x}, \widehat{\mathbf{x}}_1), \ldots, \kappa(\mathbf{x}, \widehat{\mathbf{x}}_m)]^\top$

Algorithm 1: Nyström method.

$\widehat{\mathbf{x}}_1, \ldots, \widehat{\mathbf{x}}_m$. Let $\mathbf{C} = [\kappa(\mathbf{x}_i, \widehat{\mathbf{x}}_j)]_{n\times m}$ and $\mathbf{W} = [\kappa(\widehat{\mathbf{x}}_i, \widehat{\mathbf{x}}_j)]_{m\times m}$. The rows and columns of $\mathbf{C}$ and $\mathbf{K}$ can be rearranged as

$$\mathbf{C} = \begin{bmatrix} \mathbf{W} \\ \mathbf{K}_{21} \end{bmatrix}, \quad \mathbf{K} = \begin{bmatrix} \mathbf{W} & \mathbf{K}_{21}^\top \\ \mathbf{K}_{21} & \mathbf{K}_{22} \end{bmatrix}, \tag{15}$$

where $\mathbf{K}_{21} \in \mathbb{R}^{(n-m)\times m}$ and $\mathbf{K}_{22} \in \mathbb{R}^{(n-m)\times(n-m)}$. Then the rank-$k$ approximation matrix $\widetilde{\mathbf{K}}$ of $\mathbf{K}$ can be calculated as [11]

$$\widetilde{\mathbf{K}} = \mathbf{C}\mathbf{W}_k^+\mathbf{C}^\top \approx \mathbf{K}, \tag{16}$$

where $\mathbf{W}_k^+$ is the pseudo-inverse of $\mathbf{W}_k$ and $\mathbf{W}_k$ is the best k-rank approximation of $\mathbf{W}$. The solution of $\mathbf{W}_k$ can be obtained by singular value decomposition (SVD) of $\mathbf{W}$, $\mathbf{W} = \mathbf{U}\mathbf{\Sigma}\mathbf{U}^\top$, where $\mathbf{U}$ is an orthonormal matrix and $\mathbf{\Sigma} = \text{diag}(\sigma_1, \sigma_2, \ldots, \sigma_m)$ is the diagonal matrix with $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_m \geq 0$. The solution of $\mathbf{W}_k^+$ can be obtained as

$$\mathbf{W}_k^+ = \mathbf{U}_k\mathbf{\Sigma}_k^{-1}\mathbf{U}_k^\top, \tag{17}$$

where $\mathbf{U}_k$ is the first k columns of $\mathbf{U}$ and $\mathbf{\Sigma}_k = \text{diag}(\sigma_1, \ldots, \sigma_k)$. Thus, the nonlinear feature mapping of Nyström method can be written as [28]

$$\widetilde{\phi}(\mathbf{x}) = \Sigma_k^{-1/2}\mathbf{U}_k^\top\left[\kappa\left(\mathbf{x}, \widehat{\mathbf{x}}_1\right), \ldots, \kappa\left(\mathbf{x}, \widehat{\mathbf{x}}_m\right)\right]^\top. \tag{18}$$

The algorithm of the Nyström method is described in Algorithm 1. The total time complexity of the approximation of n samples is $\mathcal{O}(nmk + m^3)$. The approximation error of the Nyström method is $\mathcal{O}(m^{-1/2})$ [11].

3.2.2. Random Fourier Features. Random Fourier features is an efficient feature transformation method for kernel matrix approximation by calculating the inner product of relatively low dimensional mappings.

When kernel $\kappa(\mathbf{x}, \mathbf{y})$ is shift-invariant, continuous, and positive-definite, the Fourier transform of the kernel can be written as

$$\kappa\left(\mathbf{x}, \mathbf{y}\right) = \int p\left(\omega\right)\exp^{j\omega^\top(\mathbf{x}-\mathbf{y})}d\omega, \tag{19}$$

where $p(\omega)$ is a probability density function and $\omega \in \mathbb{R}^d$. According to Bochner's theorem [15], the kernel can be approximated as

$$\kappa\left(\mathbf{x}, \mathbf{y}\right) = \mathbb{E}_\omega\left[\widetilde{\phi}_\omega\left(\mathbf{x}\right)^\top\widetilde{\phi}_\omega\left(\mathbf{y}\right)\right], \tag{20}$$

Require: A, X
Ensure: $\widetilde{\phi}(\cdot)$
(1) Compute the Fourier transform p of kernel;
(2) Sample $\omega_1, \ldots, \omega_m$ from the distribution $p(\omega)$
(3) Uniformly sample $b_1, \ldots, b_m$ from $[0, 2\pi]$;
(4) $\widetilde{\phi}(\mathbf{x}) = \sqrt{2/m}[\cos(\omega_1^\top\mathbf{x} + b_1), \ldots, \cos(\omega_m^\top\mathbf{x} + b_m)]^\top$

Algorithm 2: Random Fourier features.

where ω is sampled from $p(\omega)$. Since $p(\omega)$ and $\kappa(\mathbf{x}, \mathbf{y})$ are real, $\widetilde{\phi}_\omega(\mathbf{x}) = \sqrt{2}\cos(\omega^\top\mathbf{x} + b)$ where b is drawn uniformly from $[0, 2\pi]$ [13]. The expectation in (20) can be approximated by the mean over m Fourier components as

$$\widetilde{\phi}\left(\mathbf{x}\right) = \sqrt{\frac{2}{m}}\left[\cos\left(\omega_1^\top\mathbf{x} + b_1\right), \ldots, \cos\left(\omega_m^\top\mathbf{x} + b_m\right)\right]^\top, \tag{21}$$

where $\omega_i \in \mathbb{R}^d$ is sampled from the distribution $p(\omega)$ and $b_i \in \mathbb{R}$ is uniformly sampled from $[0, 2\pi]$. The algorithm is described in Algorithm 2. The total time complexity of the approximation of n samples is $\mathcal{O}(nmd)$. The approximation error of the Nyström method is $\mathcal{O}(n^{-1/2} + m^{-1/2})$ [14].

3.3. Ranking Optimization. In this section, we solve the L2-loss (squared Hinge-loss) ranking problem of (13) after the kernel approximation mapping of training data

$$\frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{(i,j)\in\mathcal{P}}\max\left(0, 1 - \mathbf{w}^\top\left(\widetilde{\phi}\left(\mathbf{x}_i\right) - \widetilde{\phi}\left(\mathbf{x}_j\right)\right)\right)^2. \tag{22}$$

Similar as (5), the loss function can be rewritten as

$$\frac{1}{2}\|\mathbf{w}\|^2 + C\left(\mathbf{1} - \mathbf{A}\widetilde{\mathbf{\Phi}}\mathbf{w}\right)^\top\mathbf{D}\left(\mathbf{1} - \mathbf{A}\widetilde{\mathbf{\Phi}}\mathbf{w}\right), \tag{23}$$

where $\widetilde{\mathbf{\Phi}} = [\widetilde{\phi}(\mathbf{x}_1), \ldots, \widetilde{\phi}(\mathbf{x}_n)]^\top$. The gradient and the generalized Hessian matrices of (23) are

$$\mathbf{g} = \mathbf{w} + 2C\widetilde{\mathbf{\Phi}}^\top\mathbf{A}^\top\mathbf{D}\left(\mathbf{A}\widetilde{\mathbf{\Phi}}\mathbf{w} - \mathbf{1}\right), \quad \mathbf{H} = \mathbf{I} + 2C\widetilde{\mathbf{\Phi}}^\top\mathbf{A}^\top\mathbf{D}\mathbf{A}\widetilde{\mathbf{\Phi}}, \tag{24}$$

where $\mathbf{I}$ is the identity matrix. The Hessian matrix does not need to be computed explicitly using truncated Newton method [8]. The Newton step $\mathbf{H}^{-1}\mathbf{g}$ can be approximately computed using linear conjugate gradient (CG). The main computation of linear CG method is the Hessian-vector multiplication $\mathbf{Hs}$ for some vector $\mathbf{s}$

$$\mathbf{Hs} = \mathbf{s} + 2C\widetilde{\mathbf{\Phi}}^\top\mathbf{A}^\top\mathbf{D}\mathbf{A}\widetilde{\mathbf{\Phi}}\mathbf{s}. \tag{25}$$

Assuming that the embedding space $\widetilde{\phi}$ has m dimensions, the total complexity of this method is $\mathcal{O}(nm + p)$ where $p = |\mathcal{P}|$. The main step of our proposed algorithm is described in Algorithm 3. We calculate the approximation embedding $\widetilde{\phi}$ using the Nyström method or random Fourier features in line (1). Then $\widetilde{\phi}$ is applied to all training samples in line (2). The linear RankSVM model with primal truncated Newton method is applied in the embedding space in line (3)–(11).

Require: $\mathbf{X}$, $\mathbf{A}$, C, t
Ensure: $\mathbf{w}$
(1) Calculate the approximation embedding $\widetilde{\phi}$ using the Nyström method or random Fourier features;
(2) Apply $\widetilde{\phi}$ to training samples, $\widetilde{\mathbf{\Phi}} = [\widetilde{\phi}(\mathbf{x}_1), \ldots, \widetilde{\phi}(\mathbf{x}_n)]^\top$;
(3) **repeat**
(4) $\quad \mathbf{D} = \mathbf{A}\widetilde{\mathbf{\Phi}}\mathbf{w} > 1$;
(5) $\quad \mathbf{g} = \mathbf{w} + 2C\widetilde{\mathbf{\Phi}}^\top \mathbf{A}^\top \mathbf{D}\ (\mathbf{A}\widetilde{\mathbf{\Phi}}\mathbf{w} - \mathbf{1})$;
(6) $\quad$ // *Solve* $\delta\mathbf{w} = \mathbf{H}^{-1}\mathbf{g}$ *by linear CG*
(7) $\quad$ **repeat**
(8) $\quad\quad$ Update based on the computation of Hessian-vector multiplication, $\mathbf{Hs} = \mathbf{s} + 2C\widetilde{\mathbf{\Phi}}^\top \mathbf{A}^\top \mathbf{DA}\widetilde{\mathbf{\Phi}}\mathbf{s}$, for some vector $\mathbf{s}$;
(9) $\quad$ **until** Convergence
(10) $\quad \mathbf{w} \leftarrow \mathbf{w} - t\delta\mathbf{w}$;
(11) **until** Convergence of the Newton step

Algorithm 3: RankSVM with kernel approximation.

Table 1: Information of used LETOR data sets. Q-D Pairs denote Query-Document Pairs. Each pair of the data sets has relevance labels in 0 (nonrelevant), 1 (possibly relevant), and 2 (relevant).

Data set	Queries	Q-D Pairs	Features	Relevance
TD2004	75	74,146	64	{0, 1}
OHSUMED	106	16,140	45	{0, 1, 2}
MQ2007	1692	69,623	46	{0, 1, 2}

4. Experiments

4.1. Experimental Settings. We use three data sets from LETOR (http://research.microsoft.com/en-us/um/beijing/projects/letor), namely, OHSUMED, MQ2007, and MQ2008, to validate our proposed ranking algorithm. The examples of the data sets are extracted from the information retrieval data collections. These data sets are often used for evaluating new learning to rank algorithms. Table 1 lists the properties of the data sets. Mean average precision (MAP) [29] and normalized discounted cumulative gain (NDCG) [30] are chosen as the evaluation metrics on the performance of the ranking models.

We compare our proposed method with linear and kernel RankSVM as follows:

(i) *RankSVM-Primal* [8]: it is discussed in Section 2.1 by solving the primal problem of linear L2-loss RankSVM (http://olivier.chapelle.cc/primal/).

(ii) *RankSVM-Struct* [9]: it solves an equivalent 1-slack structural SVM problem with linear kernel (http://www.cs.cornell.edu/People/tj/svm_light/svm_rank.html).

(iii) *RankSVM-TRON* [10]: it solves the linear or kernel ranking SVM problem by trust region Newton method (https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/).

(iv) *RankNyström*: our proposed RankSVM with the Nyström kernel approximation.

(v) *RankRandomFourier*: our proposed RankSVM with the random Fourier features kernel approximation.

The hyperparameters of the algorithms are selected by grid search. The regularization parameter C of each algorithm is chosen from $[2^{-12}, 2^{-11}, \ldots, 2^6]$. For kernel RankSVM and our approximation methods, the parameter γ of RBF kernel is chosen from $[2^{-12}, 2^{-11}, \ldots, 2^2]$. For MQ2007 dataset, the number of sampling for kernel approximation m is set to 2000, whereas $m = 500$ for the other datasets. All experiments are conducted on a high performance server with 2.0 GHz 16-cores CPU and 64 GB of memory.

4.2. Comparison of the Nyström Method and Random Fourier Features. Figure 2 shows the performance comparison of RankSVM with the Nyström method and random Fourier features on MQ2007 dataset. We take the linear RankSVM algorithm, RankSVM-Primal, as the baseline method, which is plotted as dotted line. The remaining two lines represent RankNyström and RankRandomFourier, respectively. In the beginning, the performances of kernel approximate methods are worse than linear RankSVM. But along with the increase of m (the number of sampling of approximation), both of the kernel approximate methods can outperform the linear RankSVM. We also observe that RankNyström gets better results than RankRandomFourier when m is small and the two methods obtain similar results when $m = 2000$.

4.3. Comparison with Linear and Kernel RankSVM. In this part, we compare our proposed kernel approximation ranking algorithms to other linear and kernel RankSVM algorithms. We take $N = 2000$ for the kernel approximation. Table 2 gives the results of different RankSVM algorithms on the first fold of MQ2007 dataset. The linear RankSVM algorithms use less training time, but their MeanNDCG values are lower than the values of the kernel RankSVM algorithms. Our kernel approximation methods obtain better performance than the kernel RankSVM-TRON with much faster training speed in this dataset. The training time of our kernel approximation methods is about ten seconds, whereas the training time of the kernel RankSVM-TRON is more than 13 hours. The result of random Fourier features is slightly better than the RankNyström method. Moreover,

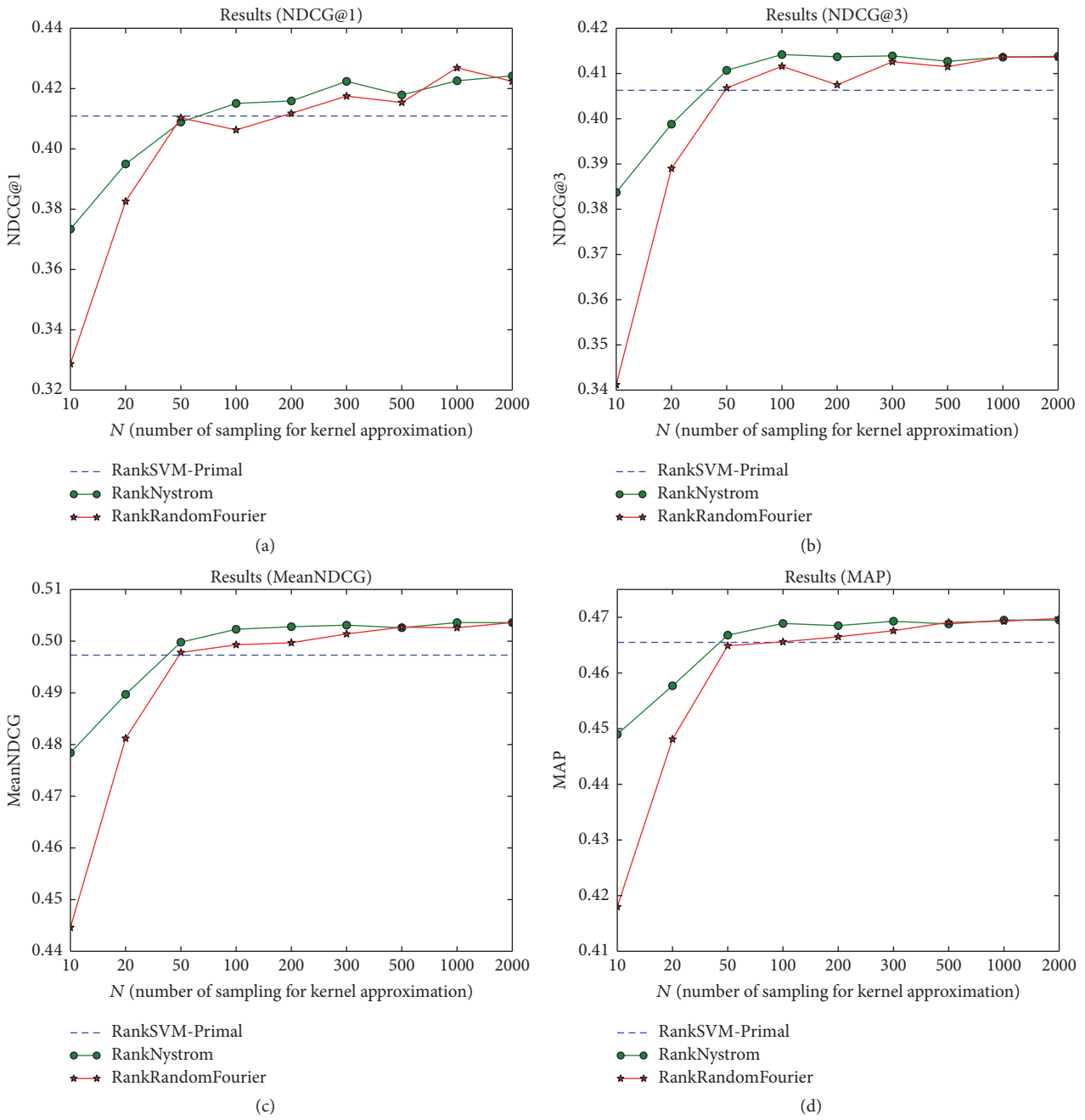

Figure 2: Performance comparison of RankSVM with the Nyström method and random Fourier features on MQ2007 dataset. (a) NDCG@1; (b) NDCG@3; (c) MeanNDCG; (d) MAP.

Table 2: Results of different RankSVM algorithms on the first fold of MQ2007 dataset. We take $m = 2000$ for the kernel approximation method.

Algorithm	Type	Loss	C	g	Mean-NDCG	Time (s)
RankSVM-TRON	linear	L1	2^{-5}	—	0.5265	1.9
RankSVM-Struct	linear	L1	2^{-1}	—	0.5268	2.2
RankSVM-Primal	linear	L2	2^{-10}	—	0.5270	1.2
RankSVM-TRON	RBF	L1	2^{-2}	2^{-5}	0.5310	47463.5
RankNystöm	RBF	L2	2^{-2}	2^{-5}	0.5330	10.9
RankRandomFourier	RBF	L2	2^{-2}	2^{-5}	**0.5336**	16.1

Table 3: Performance comparison on TD2004 data set.

	NDCG@1	NDCG@3	NDCG@5	P@1	P@3	MAP
AdaRank-MAP [20]	0.4133	0.4017	0.3932	0.4133	0.3422	0.3308
AdaRank-NDCG [20]	0.3600	0.3838	0.3769	0.3600	0.3289	0.2986
FRank [23]	0.4400	**0.4479**	0.4362	0.4400	0.3867	0.3809
ListNet [19]	0.4400	0.4371	0.4209	0.4400	0.4000	0.3721
RankBoost [18]	0.4800	0.4640	0.4368	0.4800	**0.4044**	0.3835
RankSVM-Struct [9]	0.4400	0.4092	0.3935	0.4400	0.3511	0.3505
RankSVM-Primal [8]	0.4666	0.4468	0.4277	0.4666	0.4000	0.3793
RankNyström	**0.4933**	0.4348	0.4254	**0.4933**	0.3911	0.3899
RankRandomFourier	**0.4933**	0.4422	**0.4265**	**0.4933**	0.4000	**0.3924**

Table 4: Performance comparison on OHSUMED data set.

	NDCG@1	NDCG@3	NDCG@5	P@1	P@3	MAP
RankSVM-Struct [9]	0.5515	0.4850	0.4729	0.6338	0.5898	0.4478
ListNet [19]	0.5326	0.4732	0.4432	0.6524	0.6016	0.4457
AdaRank-MAP [20]	0.5388	0.4682	0.4613	0.6338	0.5895	0.4487
AdaRank-NDCG [20]	0.5330	0.4790	0.4673	0.6719	0.5984	**0.4498**
RankBoost [18]	0.4632	0.4555	0.4494	0.5576	0.5609	0.4411
RankRLS [24]	0.5490	0.4770	0.4530	0.6440	0.5860	0.4470
RankSVM-Primal [8]	0.5645	0.5004	0.4782	0.6710	**0.6112**	0.4439
RankNyström	**0.5730**	0.4874	0.4780	**0.6801**	0.5890	0.4473
RankRandomFourier	0.5728	**0.4965**	**0.4804**	**0.6801**	0.5983	0.4472

Table 5: Performance comparison on MQ2007 data set.

	NDCG@1	NDCG@3	MeanNDCG	P@1	P@3	MAP
RankSVM-Struct [9]	0.4096	0.4063	0.4966	0.4746	0.4315	0.4645
ListNet [19]	0.4002	0.4091	0.4988	0.4640	0.4334	0.4652
AdaRank-MAP [20]	0.3821	0.3984	0.4891	0.4392	0.4230	0.4577
AdaRank-NDCG [20]	0.3876	0.4044	0.4914	0.4475	0.4305	0.4602
RankBoost [18]	0.4134	0.4072	0.5003	0.4823	0.4348	0.4662
LambdaMART [25]	0.4147	0.4119	0.5011	—	—	0.4660
BL-MART [25]	0.4200	**0.4224**	**0.5093**	—	—	**0.4730**
CRR [26]	—	—	0.5000	—	—	0.4660
RankSVM-Primal [8]	0.4109	0.4063	0.4973	0.4747	0.4317	0.4655
RankNyström	**0.4242**	0.4138	0.5036	**0.4888**	**0.4394**	0.4695
RankRandomFourier	0.4224	0.4136	0.5036	0.4871	0.4386	0.4698

the L2-loss RankSVM can get better performance than the L1-loss RankSVM on this dataset. The MeanNDCG of RankSVM-Primal (linear) is slightly higher than RankSVM-TRON (linear). The kernel approximation methods get better MeanNDCG than RankSVM-TRON with RBF kernel.

4.4. Comparison with State-of-the-Art. In this part, we compare our proposed algorithm with the state-of-the-art ranking algorithms. Most of the results of the comparison algorithms come from the baselines of LETOR. The remaining results come from the papers of the algorithms. The hyperparameters C and γ of our proposed kernel approximation RankSVM are selected by grid search as in Section 4.1.

Table 3 provides the comparison of testing NDCG and MAP results of different ranking algorithms on the TD2004 dataset. The number of sampling for kernel approximation m is set to 500. We can observe that the kernel approximation ranking methods can achieve the best performances on 3 terms of all the 6 metrics. Also, the results of RankNyström and RankRandomFourier are similar.

Table 4 provides the performance comparison on the OHSUMED dataset. m is set to 500. We once observe that RankRandomFourier achieves the best performances on 3 metrics of all the 6 metrics. RankNyström gets the best results on 2 metrics.

Table 5 provides the comparison of results on the MQ2007 dataset. m is set to 2000. We observe that RankNyström

obtains the best scores on 3 metrics on MQ2007 dataset. BL-MART also achieves the best scores on 3 metrics. However, BL-MART trains 10,000 LambdaMART and creates bagged model by randomly selecting a subset of these models, whereas our proposed RankNyström algorithm only trains one model.

5. Conclusions

In this paper, we propose a fast RankSVM algorithm with kernel approximation to solve the problem of lengthy training time of kernel RankSVM. First, we proposed a unified model for kernel approximation RankSVM. Approximation method is used to avoid computing kernel matrix by explicitly approximating the kernel similarity between any two data points. Then, two types of methods, namely, the Nyströem method and random Fourier features, are explored to approximate the kernel matrix. Also, the primal truncated Newton method is used to optimize the L2-loss (squared Hinge-loss) objective function of the ranking model. Experimental results indicate that our proposed method requires much less computational cost than kernel RankSVM and achieves comparable or better performance over state-of-the-art ranking algorithms. In the future, we plan to use more efficient kernel approximation and ranking models for large-scale ranking problems.

Competing Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This work was mainly supported by Natural Science Foundation of China (61125201, 61303070, U1435219).

References

[1] T.-Y. Liu, "Learning to rank for information retrieval," *Foundations and Trends in Information Retrieval*, vol. 3, no. 3, pp. 225–231, 2009.

[2] Y. Lv, T. Moon, P. Kolari, Z. Zheng, X. Wang, and Y. Chang, "Learning to model relatedness for news recommendation," in *Proceedings of the 20th International Conference on World Wide Web (WWW '11)*, pp. 57–66, ACM, April 2011.

[3] Y. Yu, H. Wang, G. Yin, and T. Wang, "Reviewer recommendation for pull-requests in GitHub: what can we learn from code review and bug assignment?" *Information and Software Technology*, vol. 74, pp. 204–218, 2016.

[4] K. Duh and K. Kirchhoff, "Learning to rank with partially-labeled data," in *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 251–258, ACM, Singapore, July 2008.

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

[6] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*, pp. 144–152, Pittsburgh, Pa, USA, July 1992.

[7] T. Joachims, "Optimizing search engines using clickthrough data," in *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 133–142, Edmonton, Canada, 2002.

[8] O. Chapelle and S. S. Keerthi, "Efficient algorithms for ranking with SVMs," *Information Retrieval*, vol. 13, no. 3, pp. 201–215, 2010.

[9] T. Joachims, "Training linear SVMs in linear time," in *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 217–226, ACM, 2006.

[10] T.-M. Kuo, C.-P. Lee, and C.-J. Lin, "Large-scale kernel rankSVM," in *Proceedings of the SIAM International Conference on Data Mining (SDM '14)*, pp. 812–820, 2014.

[11] P. Drineas and M. W. Mahoney, "On the Nyström method for approximating a gram matrix for improved kernel-based learning," *Journal of Machine Learning Research*, vol. 6, pp. 2153–2175, 2005.

[12] C. K. I. Williams and M. Seeger, "Using the Nyström method to speed up Kernel machines," in *Advances in Neural Information Processing Systems 13*, pp. 682–688, MIT Press, Cambridge, Mass, USA, 2001.

[13] A. Rahimi and B. Recht, "Random features for large-scale kernel machines," in *Advances in Neural Information Processing Systems 20*, pp. 1177–1184, Curran Associates, Newry, UK, 2007.

[14] A. Rahimi and B. Recht, "Weighted sums of random kitchen sinks: replacing minimization with randomization in learning," in *Advances in Neural Information Processing Systems*, pp. 1313–1320, 2008.

[15] W. Rudin, *Fourier Analysis on Groups*, Springer, New York, NY, USA, 2003.

[16] P. Li, Q. Wu, and C. J. Burges, "McRank: learning to rank using multiple classification and gradient boosting," in *Advances in Neural Information Processing Systems*, pp. 897–904, MIT Press, 2007.

[17] A. Shashua and A. Levin, "Ranking with large margin principle: two approaches," in *Advances in Neural Information Processing Systems*, pp. 937–944, 2002.

[18] Y. Freund, R. Iyer, R. E. Schapire, and Y. Singer, "An efficient boosting algorithm for combining preferences," *The Journal of Machine Learning Research*, vol. 4, no. 6, pp. 933–969, 2003.

[19] Z. Cao, T. Qin, T.-Y. Liu, M.-F. Tsai, and H. Li, "Learning to rank: from pairwise approach to listwise approach," in *Proceedings of the 24th International Conference on Machine Learning (ICML '07)*, pp. 129–136, ACM, June 2007.

[20] J. Xu and H. Li, "AdaRank: a boosting algorithm for information retrieval," in *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '07)*, pp. 391–398, ACM, July 2007.

[21] M. N. Volkovs and R. S. Zemel, "BoltzRank: learning to maximize expected ranking gain," in *Proceedings of the 26th International Conference on Machine Learning (ICML '09)*, pp. 1089–1096, ACM, June 2009.

[22] Y. Yue, T. Finley, F. Radlinski, and T. Joachims, "A support vector method for optimizing average precision," in *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '07)*, pp. 271–278, ACM, July 2007.

[23] M. F. Tsai, T. Y. Liu, T. Qin, H. H. Chen, and W. Y. Ma, "FRank: a ranking method with fidelity loss," in *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '07)*, pp. 383–390, Amsterdam, the Netherlands, July 2007.

[24] T. Pahikkala, E. Tsivtsivadze, A. Airola, J. Boberg, and T. Salakoski, "Learning to rank with pairwise regularized least-squares," in *Proceedings of the SIGIR Workshop on Learning to Rank for Information Retrieval*, vol. 80, pp. 27–33, 2007.

[25] Y. Ganjisaffar, R. Caruana, and C. V. Lopes, "Bagging gradient-boosted trees for high precision, low variance ranking models," in *Proceedings of the 34th ACM SIGIR International Conference on Research and Development in Information Retrieval (SIGIR '11)*, pp. 85–94, ACM, Beijing, China, July 2011.

[26] D. Sculley, "Combined regression and ranking," in *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 979–988, ACM, Washington, DC, USA, 2010.

[27] H. Yu and S. Kim, "SVM tutorial—classification, regression and ranking," in *Handbook of Natural Computing*, pp. 479–506, Springer, Berlin, Germany, 2012.

[28] T. Yang, Y. F. Li, M. Mahdavi, R. Jin, and Z. H. Zhou, "Nyström method vs random Fourier features: a theoretical and empirical comparison," in *Advances in Neural Information Processing Systems*, pp. 485–493, MIT Press, 2012.

[29] R. Baeza-Yates and B. Ribeiro-Neto, *Modern Information Retrieval*, vol. 463, ACM Press, New York, NY, USA, 1999.

[30] K. Järvelin and J. Kekäläinen, "IR evaluation methods for retrieving highly relevant documents," in *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 41–48, ACM, Athens, Greece, July 2000.