Journal of Machine Learning Research 14 (2013) 2519-2548 Submitted 10/12; Revised 5/13; Published 9/13

Learning Bilinear Model for Matching Queries and Documents

Wei Wu WUWEI@MICROSOFT.COM
Microsoft Research Asia
13F, Building 2
No.5 Danling Street
Beijing, 100080, P. R. China

Zhengdong Lu LU.ZHENGDONG@HUAWEI.COM
Hang Li HANGLI.HL@HUAWEI.COM
Huawei Noah's Ark Lab
Units 525-530, Core Building 2
Hong Kong Science Park
Shatin, Hong Kong

Editor: Charles Elkan

Abstract

The task of matching data from two heterogeneous domains naturally arises in various areas such as web search, collaborative filtering, and drug design. In web search, existing work has designed relevance models to match queries and documents by exploiting either user clicks or content of queries and documents. To the best of our knowledge, however, there has been little work on principled approaches to leveraging both clicks and content to learn a matching model for search. In this paper, we propose a framework for learning to match heterogeneous objects. The framework learns two linear mappings for two objects respectively, and matches them via the dot product of their images after mapping. Moreover, when different regularizations are enforced, the framework renders a rich family of matching models. With orthonormal constraints on mapping functions, the framework subsumes Partial Least Squares (PLS) as a special case. Alternatively, with a ℓ_1+ℓ_2 regularization, we obtain a new model called *Regularized Mapping to Latent Structures* (RMLS). RMLS enjoys many advantages over PLS, including lower time complexity and easy parallelization. To further understand the matching framework, we conduct generalization analysis and apply the result to both PLS and RMLS. We apply the framework to web search and implement both PLS and RMLS using a click-through bipartite with metadata representing features of queries and documents. We test the efficacy and scalability of RMLS and PLS on large scale web search problems. The results show that both PLS and RMLS can significantly outperform baseline methods, while RMLS substantially speeds up the learning process.

Keywords: web search, partial least squares, regularized mapping to latent structures, generalization analysis

1. Introduction

Many tasks in machine learning and data mining can be formalized as matching between objects from two spaces. One particular example is web search, where the retrieved documents are ordered according to their relevance to the given query. The relevance is determined by the matching scores between the query and the documents. It is therefore crucial to accurately calculate the matching

score for any given query-document pair. Similarly, matching between heterogeneous data sources can be found in collaborative filtering, image annotation, drug design, etc.

Existing models in web search use information from different sources to match queries and documents. On one hand, conventional relevance models, including Vector Space Model (VSM) (Salton and McGill, 1986), BM25 (Robertson et al., 1994), and Language Models for Information Retrieval (LMIR) (Ponte and Croft, 1998; Zhai and Lafferty, 2004), match queries and documents based on their content. Specifically, queries and documents are represented as feature vectors in a Euclidean space, and conventional relevance models match them by the dot products of their feature vectors (Xu et al., 2010; Wu et al., 2011). On the other hand, a click-through bipartite graph, which represents users' implicit judgments on query-document relevance, has proven to be a very valuable resource for matching queries and documents. Many methods have been proposed (Craswell and Szummer, 2007; Ma et al., 2008), but they rely only on the structure of the bipartite graph. Existing models rely on either features or the click-through bipartite graph to match queries and documents. Therefore, there is need for a principled approach to learning to match with both features and the click-through bipartite graph. The learnt model must maintain efficacy and scalability (due to the massive scale of web search problems). Moreover, we also would like to understand the generalization ability of matching models.

This paper proposes a general framework for learning to match objects from two spaces. Specifically, the framework learns a linear mapping for each object and map the two objects into a common latent space. After that, the dot product of the images of the two objects is taken as their matching score. The matching model is linear in terms of both objects, and therefore, we actually learn a bilinear model for matching two objects. The types of linear mapping can be further specified by a set of constraints. By limiting the mappings to projections,[1] we obtain a natural generalization to Partial Least Squares (PLS), a classic model in statistics for analyzing the correlation between two variables. More interestingly, when replacing this constraint with regularization constraints based on ℓ_1 and ℓ_2 norms, we get a new learning to match model, called Regularized Mapping to Latent Structures (RMLS). This model allows easy parallelization for learning and fast computation in testing due to the induced sparsity in the mapping matrices. More specifically, RMLS allows pre-computing intermediate parameters, making optimization independent of training instances. Moreover, the pre-computation can be easily distributed across different machines, and therefore can further enhance the efficiency and scalability of RMLS. To further understand this framework, we give a generalization analysis under a hierarchical sampling assumption which is natural in the matching problems encountered in web search. Our results indicate that to obtain a good generalization ability, it is necessary to use a large number of instances for each type of object.

With the framework, we learn a matching model for search by leveraging both click-through and features. Specifically, we implement both PLS and RMLS using a click-through bipartite graph with metadata on the nodes representing features of queries and documents. We take click numbers as a response and learn linear mappings to matching queries and documents, each represented by heterogeneous feature vectors consisting of both key words and click numbers. On a small data set, RMLS and PLS perform comparably well and both of them significantly outperform other baseline methods. RMLS is more efficient than PLS and the advantage becomes more significant under parallelization. RMLS also scales well on large data sets. On a data set with millions of queries and

1. In this paper, by projection, we mean a linear mapping specified by a matrix P, with $P^\top P = \mathbb{I}$.

documents, RMLS can leverage high dimensional features and significantly outperform all other baseline methods.

Our contributions are three-fold: 1) we propose a framework for learning to match heterogeneous data, with PLS and the more scalable RMLS as special cases; 2) generalization analysis of this framework as well as its application to RMLS and PLS; 3) empirical verification of the efficacy and scalability of RMLS and PLS on real-world large-scale web search data.

2. Related Work

Matching pairs of objects with a similarity function defined as a dot product has been researched for quite some time. When the pair of objects are from the same space (i.e., they are homogeneous data), the similarity function becomes positive semi-definite, and the matching problem is essentially finding a good kernel (Cristianini et al., 2001; Lanckriet et al., 2002; Bach et al., 2004; Ong et al., 2005; Micchelli and Pontil, 2005; Bach, 2008; Varma and Babu, 2009; Cortes, 2009). Among existing works, distance metric learning (Jolliffe, 2002; Xing et al., 2003; Schultz and Joachims, 2003; JacobGoldberger and GeoffHinton, 2004; Hertz et al., 2004; Hoi et al., 2006; Yang et al., 2006; Davis et al., 2007; Sugiyama, 2007; Weinberger and Saul, 2009; Ying et al., 2009) is a representative approach of learning similarities (or dissimilarities) for homogeneous data. In distance metric learning, a linear transformation is learnt for mapping objects from the same space into a latent space. In the space, dot product or Euclidean distance is taken as a means to measure similarity or dissimilarity. Recently, learning a similarity function for object pairs from two different spaces has also emerged as a hot research topic (Grangier and Bengio, 2008; Abernethy et al., 2009). Our model belongs to the latter category, but is tailored for web search and tries to solve problems central to that, for example, scalability.

Our model, when applied to web search, is also obviously related to the effort on learning to rank (Herbrich et al., 1999; Crammer and Singer, 2001; Joachims, 2002; Agarwal and Niyogi, 2005; Rudin et al., 2005; Burges et al., 2006; Cao et al., 2006; Xu and Li, 2007; Cao et al., 2007; Liu, 2009; Li, 2011). However, we focus on learning to match queries and documents, while learning to rank has been more concerned with optimizing the ranking model. Clearly the matching score learned with our method can be integrated as a feature for a particular learning to rank model, and therefore our model is in a sense feature learning for learning to rank.

In web search, existing work for matching queries and documents can be roughly categorized into two groups: feature based methods and graph based methods. In the former group, Vector Space Model (VSM) (Salton and McGill, 1986), BM25 (Robertson et al., 1994), and Language Models for Information Retrieval (LMIR) (Ponte and Croft, 1998; Zhai and Lafferty, 2004) make use of features, particularly, n-gram features to calculate query-document matching scores. As pointed out by Xu et al. (2010) as well as Wu et al. (2011), these models perform matching by using the dot product between a query vector and a document vector as a query-document similarity function. In the latter group, graph based methods exploit the structure of a click-through bipartite graph to match query-document pairs. For example, Latent Semantic Indexing (LSI) (Deerwester et al., 1990) can be employed, which uses SVD to project queries and documents in a click-through bipartite graph into a latent space, and calculates query-document matching scores through the dot product of their images in the latent space. Craswell and Szummer (2007) propose adopting a backward random walk process on a click-through bipartite graph to propagate similarity through probabilistic transitions. In this paper, we propose a general framework for matching queries and

documents. The framework can leverage both features and a click-through bipartite graph to learn a matching model. In doing so, we actually combine feature based methods and graph based methods in a principled way.

In information retrieval, some recent work also considers leveraging both user clicks and content of queries and documents. Bai et al. (2009) propose learning a low rank model for ranking documents, which is like matching queries and documents. On the other hand, there are also stark differences between our work and theirs. For example, their work requires a pair-wise input supervision and learns a ranking model using hinge loss, while our work employs a point-wise input and learns a matching model using alignment. Gao et al. (2011) propose combining ideas in semantic representation and statistical machine translation to learn relevance models for web search. Compared with their work, our method is non-probabilistic and can leverage different regularizations, such as the ℓ_1 regularization, to achieve better performance.

The matching problem is also widely studied in collaborative filtering (CF) whose goal can be viewed as matching users and items (Hofmann, 2004; Srebro et al., 2005; Abernethy et al., 2009; Chen et al., 2012). The characteristics of the problems CF attempts to solve, for example, the sampling assumption and the nature of "ratings", are different from the matching problem in web search. We have compared our methods with a state-of-the-art CF model which can handle extra attributes in two domains. The results indicate that our methods are more effective in web search than the existing CF model.

In existing statistical models, Partial Least Squares (PLS) (Rosipal and Krämer, 2006; Schreier, 2008) and Canonical Correlation Analysis (CCA) (Hardoon et al., 2004) are classic tools for capturing correlations of two variables via common latent structures. In this paper, we provide a general matching framework, which subsumes PLS and allows rather scalable implementations.

3. A Framework For Matching Objects From Two Spaces

We first give a general framework for learning to match objects from two heterogeneous spaces. Suppose that there are two spaces $\mathcal{X} \subset \mathbb{R}^{d_x}$ and $\mathcal{Y} \subset \mathbb{R}^{d_y}$. For any $x \in \mathcal{X}$ and $y \in \mathcal{Y}$, there is a response $r \doteq r(x,y) \geqslant 0$ in space $\mathcal{R}$, indicating the actual correlation between object x and object y. For web search, the objects are queries and documents, and the response can be judgment from human labelers or the click number from user logs.

We first describe the hierarchical sampling process for generating any sample triple (x_i, y_{ij}, r_{ij}).

Assumption 1 *First x_i is sampled according to $P(x)$. Then y_{ij} is sampled according to $P(y|x_i)$. After that, there is a response $r_{ij} = r(x_i, y_{ij})$ associated with pair (x_i, y_{ij}).*

We argue that this is an appropriate sampling assumption for web search (Chen et al., 2010), since the selected y_{ij} (in this case, retrieved document) depends heavily on x_i (in this case, query). This dependence is largely rendered by several factors of a search engine, including the indexed pages and the ranking algorithms. Under Assumption 1, we have a sample set $\mathcal{S} = \{(x_i, y_{ij}, r_{ij})\}$, with $1 \leqslant i \leqslant n^x$, and for any given i, $1 \leqslant j \leqslant n_i^y$. Here $\{x_i\}_{i=1}^{n^x}$ are i.i.d. sampled and for a given x_i, $\{y_{ij}\}_{j=1}^{n_i^y}$ are i.i.d. samples conditioned on x_i. Relying on this sampling assumption, we will give the learning to match framework, and later carry out the generalization analysis.

3.1 Model

We intend to find a linear mapping pair (L_x, L_y), so that the corresponding images $L_x^\top x$ and $L_y^\top y$ are in the same d-dimensional latent space $\mathcal{L}$ (with $d \ll \min\{d_x, d_y\}$), and the degree of matching between x and y can be reduced to the dot product in $\mathcal{L}$:

$$\mathsf{match}_{L_x,L_y}(x,y) = x^\top L_x L_y^\top y.$$

Dot product is a popular form of matching in applications like search. In fact, traditional relevance models in search such as VSM (Salton and McGill, 1986), BM25 (Robertson et al., 1994), and LMIR (Ponte and Croft, 1998; Zhai and Lafferty, 2004) are all dot products of a query vector and a document vector, as pointed out by Xu et al. (2010) as well as Wu et al. (2011). Recently, Bai et al. (2009) proposed supervised semantic indexing, which also uses dot product as the measure of query-document similarity. We hope the score defined this way can reflect the actual response. More specifically, we would like to maximize the following expected alignment between this matching score and the response

$$\mathbb{E}_{x,y}\{r(x,y) \cdot \mathsf{match}_{L_x,L_y}(x,y)\} = \mathbb{E}_x \mathbb{E}_{y|x}\{r(x,y) x^\top L_x L_y^\top y\}, \tag{1}$$

which is in the same spirit as the technique used by Cristianini et al. (2001) for kernel learning. The expectation in (1) can be estimated as

$$\frac{1}{n^x} \sum_{i=1}^{n^x} \frac{1}{n_i^y} \sum_{j=1}^{n_i^y} r_{ij} x_i^\top L_x L_y^\top y_{ij}.$$

The learning problem hence boils down to

$$\begin{aligned} \underset{L_x,L_y}{\arg\max} \quad & \frac{1}{n^x} \sum_{i=1}^{n^x} \frac{1}{n_i^y} \sum_{j=1}^{n_i^y} r_{ij} x_i^\top L_x L_y^\top y_{ij}, \\ \text{s.t.} \quad & L_x \in \mathcal{H}_x,\ L_y \in \mathcal{H}_y, \end{aligned} \tag{2}$$

where $\mathcal{H}_x$ and $\mathcal{H}_y$ are hypothesis spaces for L_x and L_y respectively. Since the final matching model is linear in terms of both x and y, Framework (2) actually learns a bilinear model for matching objects from two spaces.

3.2 Special Cases

The matching framework in (2) defines a rather rich family of matching models, with different choices of $\mathcal{H}_x$ and $\mathcal{H}_y$. More specifically, we define $\mathcal{H}_x = \{L_x \mid L_x^\top L_x = \mathbb{I}_{d\times d}\}$ and $\mathcal{H}_y = \{L_y \mid L_y^\top L_y = \mathbb{I}_{d\times d}\}$. In other words, both L_x and L_y are confined to be matrices with orthonormal columns, then the program in (2) becomes a natural extension to Partial Least Squares (PLS) (Rosipal and Krämer, 2006; Schreier, 2008). Like the well-known Canonical Correlation Analysis (CCA) (Hardoon et al., 2004), PLS is also a classic statistical model for capturing the correlation between two variables. In contrast, in (2) we allow an instance in one domain to be associated with multiple instances in the other domain, and argument each association with a weight (response). This extension enables us to model the complex bipartite association relations in matching tasks. To see the connection between

(1) and traditional PLS, we re-write (2) as

$$\begin{aligned} \underset{L_x, L_y}{\arg\max} \quad & \frac{1}{n^x}\sum_{i=1}^{n^x} x_i^\top L_x L_y^\top y_i' = \mathrm{trace}(L_y^\top (\frac{1}{n^x}\sum_{i=1}^{n^x} y_i' x_i^\top) L_x), \\ \text{s.t.} \quad & L_x^\top L_x = L_y^\top L_y = \mathbb{I}_{d\times d}, \end{aligned}$$

where $y_i' = 1/n_i^y \sum_{j=1}^{n_i^y} r_{ij} y_{ij}$. The program is exactly the formulation of PLS as formulated by Schreier (2008) when viewing y_i' a variable.[2]

Interestingly, our framework in (2) also subsumes the Latent Semantic Index (LSI) (Deerwester et al., 1990) used in information retrieval. Specifically, suppose that $\mathcal{X}$ represents document space and $\mathcal{Y}$ represents term space. Response r represents the tf-idf weight of a term y in a document x. Let x and y be the indicator vectors of a document and a term, that is, there is only non-zero element *one* in x and y at the location indexing the corresponding document or term. The objective function in (2) becomes trace $\left(L_y^\top (\sum_{i=1}^{n^x}\sum_{j=1}^{n_i^y} r_{ij} y_{ij} x_i^\top) L_x\right)$ after ignoring n^x and n_i^y, which is exactly the objective for the SVD in LSI assuming the same orthonormal $\mathcal{H}_x$ and $\mathcal{H}_y$ defined for PLS.

The orthonormal constraints in PLS requires SVD of large matrices (Schreier, 2008), rendering it impractical for web scale applications (e.g., millions of objects with millions of features in basic settings). In next section we will consider other choices of $\mathcal{H}_x$ and $\mathcal{H}_y$ for more scalable alternatives.

4. Regularized Mapping to Latent Structures

Heading towards a more scalable matching model, we drop the orthonormal constraints in PLS, and replace them with ℓ_1 norm and ℓ_2 norm based constraints on L_x and L_y. More specifically, we define the following hypothesis spaces

$$\begin{aligned} \mathcal{H}_x &= \{L_x \mid |l_{xu}| \leqslant \lambda_x, \|l_{xu}\| \leqslant \theta_x, u = 1, \ldots, d_x\}, \\ \mathcal{H}_y &= \{L_y \mid |l_{yv}| \leqslant \lambda_y, \|l_{yv}\| \leqslant \theta_y, v = 1, \ldots, d_y\}, \end{aligned}$$

where $|\cdot|$ and $\|\cdot\|$ are respectively the ℓ_1-norm and ℓ_2-norm, l_{xu} and l_{yv} are respectively the u^{th} and v^{th} row of L_x and L_y, $\{\lambda_x, \theta_x, \lambda_y, \theta_y\}$ are parameters. Here the ℓ_1-norm based constraints will induce row-wise sparsity in L_x and L_y. The ℓ_2-norm on rows, in addition to posing further regularization, avoids degenerative solutions (see Appendix A for details). The row-wise sparsity in L_x and L_y in turn yields sparse images in $\mathcal{L}$ with sparse x and y. Indeed, for any $x = [x^{(1)} \ \ldots \ x^{(d_x)}]^\top$, its image in $\mathcal{L}$ is $L_x^\top x = \sum_{u=1}^{d_x} x^{(u)} l_{xu}$. When both x and l_{xu} are sparse, $L_x^\top x$ is the sum of a few sparse vectors, and therefore likely to be sparse itself. A similar scenario holds true for y. In web search, it is usually the case that both x and y are extremely sparse. Sparse mapping matrices and sparse images in latent structures will mitigate the memory pressure and enhance efficiency in both training and testing. With $\mathcal{H}_x$ and $\mathcal{H}_y$ defined above, we have the following program:

$$\begin{aligned} \underset{L_x, L_y}{\arg\max} \quad & \frac{1}{n^x}\sum_{i=1}^{n^x}\frac{1}{n_i^y}\sum_{j=1}^{n_i^y} r_{ij} x_i^\top L_x L_y^\top y_{ij}, \\ s.t. \quad & |l_{xu}| \leqslant \lambda_x, \ \|l_{xu}\| \leqslant \theta_x, \ |l_{yv}| \leqslant \lambda_y, \ \|l_{yv}\| \leqslant \theta_y, \quad 1 \leqslant u \leqslant d_x, \ 1 \leqslant v \leqslant d_y. \end{aligned} \tag{3}$$

The matching model defined in (3) is called *Regularized Mapping to Latent Structures* (RMLS).

2. We can assume that x and y' are centered.

4.1 Optimization

In practice, we instead solve the following penalized variant of (3) for easier optimization

$$\begin{aligned} \underset{L_x,L_y}{\arg\min} \quad & -\frac{1}{n^x}\sum_{i=1}^{n^x}\sum_{j=1}^{n_i^y}\frac{1}{n_i^y}r_{ij}x_i^\top L_x L_y^\top y_{ij} + \beta\sum_{u=1}^{d_x}|l_{xu}| + \gamma\sum_{v=1}^{d_y}|l_{yv}|, \\ s.t. \quad & \|l_{xu}\| \leqslant \theta_x, \|l_{yv}\| \leqslant \theta_y,\ 1 \leqslant u \leqslant d_x, 1 \leqslant v \leqslant d_y, \end{aligned} \tag{4}$$

where $\beta > 0$ and $\gamma > 0$ control the trade-off between the objective and the penalty. We employ the coordinate descent technique to solve problem (4). Since the objective in (4) is not convex, there is no guarantee for convergence to a global minimum.

Specifically, for a fixed L_y, the objective function of problem (4) can be re-written as

$$\sum_{u=1}^{d_x}\left(-(\sum_{i=1}^{n^x}\sum_{j=1}^{n_i^y}\frac{1}{n^x n_i^y}x_i^{(u)}r_{ij}L_y^\top y_{ij})^\top l_{xu} + \beta|l_{xu}|\right).$$

Representing the d-dimensional $\sum_{i=1}^{n^x}\sum_{j=1}^{n_i^y}\frac{1}{n^x n_i^y}x_i^{(u)}r_{ij}L_y^\top y_{ij}$ as $\omega_u = [\omega_u^{(1)}, \omega_u^{(2)}, \ldots, \omega_u^{(d)}]^\top$, the optimal l_{xu} is given by

$$l_{xu}^{(z)*} = C_u \cdot \left(\max(|\omega_u^{(z)}| - \beta, 0)\text{sign}(\omega_u^{(z)})\right),\ 1 \leqslant z \leqslant d, \tag{5}$$

where $l_{xu}^{(z)}$ represents the z^{th} element of l_{xu}. $\text{sign}(\cdot)$ represents the sign function. C_u is a constant that makes $\|l_{xu}^*\| = \theta_x$ if there are nonzero elements in l_{xu}^*, otherwise $C_u = 0$.

Similarly, for a fixed L_x, the objective function of problem (4) can be re-written as

$$\sum_{v=1}^{d_y}\left(-(\sum_{i=1}^{n^x}\sum_{j=1}^{n_i^y}\frac{1}{n^x n_i^y}y_{ij}^{(v)}r_{ij}L_x^\top x_i)^\top l_{yv} + \gamma|l_{yv}|\right).$$

Writing $\sum_{i=1}^{n^x}\sum_{j=1}^{n_i^y}\frac{1}{n^x n_i^y}y_{ij}^{(v)}r_{ij}L_x^\top x_i$ as $\eta_v = [\eta_v^{(1)}, \ldots, \eta_v^{(d)}]^\top$, the optimal l_{yv} is given by

$$l_{yv}^{(z)*} = C_v \cdot \left(\max(|\eta_v^{(z)}| - \gamma, 0)\text{sign}(\eta_v^{(z)})\right),\ 1 \leqslant z \leqslant d, \tag{6}$$

where $l_{yv}^{(z)}$ represents the z^{th} element of l_{yv}. C_v is a constant that makes $||l_{yv}^*|| = \theta_y$ if there are nonzero elements in l_{yv}^*, otherwise $C_v = 0$. Note that $\sum_{i=1}^{n^x}\sum_{j=1}^{n_i^y}\frac{1}{n^x n_i^y}x_i^{(u)}r_{ij}L_y^\top y_{ij} = L_y^\top w_{xu}$, where $w_{xu} = \sum_{i=1}^{n^x}\sum_{j=1}^{n_i^y}\frac{1}{n^x n_i^y}x^{(u)}r_{ij}y_{ij}$ does not rely on the update of L_x and L_y and can be pre-calculated to save time. Similarly we pre-calculate $w_{yv} = \sum_{i=1}^{n^x}\sum_{j=1}^{n_i^y}\frac{1}{n^x n_i^y}y_{ij}^{(v)}r_{ij}x_i$.

The preprocessing is described in Algorithm 1, whose time complexity is $O(d_x N_x \tilde{n}^y c_y + d_y N_y \tilde{n}^x c_x)$, where N_x stands for the average number of nonzeros in all x samples per dimension, N_y is the average number of nonzeros in all y samples per dimension, $\tilde{n}^x$ is the average number of related x samples per y, $\tilde{n}^y$ is the mean of n_i^y, c_x is the average number of nonzeros in each x sample, and c_y is the average number of nonzeros in each y sample.

Algorithm 1 Preprocessing

Input: $\mathcal{S} = \{(x_i, y_{ij}, r_{ij})\}$, $1 \leqslant i \leqslant n^x$, and $1 \leqslant j \leqslant n_i^y$.
for $u = 1 : d_x$
  $w_{xu} \leftarrow \mathbf{0}$
for $v = 1 : d_y$
  $w_{yv} \leftarrow \mathbf{0}$
for $u = 1 : d_x$, $i = 1 : n^x$, $j = 1 : n_i^y$
  $w_{xu} \leftarrow w_{xu} + \frac{1}{n^x n_i^y} x_i^{(u)} r_{ij} y_{ij}$
for $v = 1 : d_y$, $i = 1 : n^x$, $j = 1 : n_i^y$
  $w_{yv} \leftarrow w_{yv} + \frac{1}{n^x n_i^y} y_{ij}^{(v)} r_{ij} x_i$
Output: $\{w_{xu}\}_{u=1}^{d_x}$, $\{w_{yv}\}_{v=1}^{d_y}$.

Algorithm 2 RMLS

Input: $\{w_{xu}\}_{u=1}^{d_x}$, $\{w_{yv}\}_{v=1}^{d_y}$, d, β, γ, θ_x, θ_y.
Initialization: randomly set L_x and L_y as L_x^0 and L_y^0, $t \leftarrow 0$.
While not converged and $t \leqslant T$
  for $u = 1 : d_x$
    calculate ω_u by ${L_y^t}^\top w_{xu}$.
    calculate ${l_{xu}}^*$ using Equation (5).
  update L_x^{t+1}.
  for $v = 1 : d_y$
    calculate η_v by ${L_x^{t+1}}^\top w_{yv}$.
    calculate ${l_{yv}}^*$ using Equation (6).
  update L_y^{t+1}, $t \leftarrow t+1$
Output: L_x^t and L_y^t.

After preprocessing, we take $\{w_{xu}\}_{i=1}^{d_x}$ and $\{w_{yv}\}_{i=1}^{d_y}$ as input and iteratively optimize L_x and L_y, as described in Algorithm 2. Suppose that each w_{xu} has on average W_x nonzeros and each w_{yv} has on average W_y nonzeros, then the average time complexity of Algorithm 2 is $O(d_x W_x d + d_y W_y d)$.

In web search, it is usually the case that queries (x here) and documents (y here) are of high dimension (e.g., $> 10^6$) but extremely sparse. In other words, both c_x and c_y are small despite large d_x and d_y. Moreover, it is quite common that for each x, there are only a few y that have response with it and vice versa, rendering quite small $\tilde{n}^y$ and $\tilde{n}^x$. This situation is easy to understand in the context of web search, since for each query only a small number of documents are retrieved and viewed, and each document can only be retrieved with a few queries and get viewed. Finally, we observed that in practice, N_x and N_y are also small. For example, in web search, with the features extracted from the content of queries and documents, each word only relates to a few queries and documents. In Algorithm 2, when input vectors are sparse, $\{w_{xu}\}_{u=1}^{d_x}$ and $\{w_{yv}\}_{v=1}^{d_y}$ are also sparse, which makes W_x and W_y small. In summary, under sparse input as we often see in web search, RMLS can be implemented fairly efficiently.

4.2 Parallelization

The learning process of RMLS is still quite expensive for web scale data due to high dimensionality of x and y. Parallelization can greatly improve the speed of learning in RMLS, making it scalable enough for massive data sets.

The key in parallelizing Algorithm 1 and Algorithm 2 is that the calculation of different parameters can be executed concurrently. In Algorithm 1 there is no dependency among the calculation of different w_{xu} and w_{yv}, therefore, they can be calculated by multiple processors or multiple computers simultaneously. Similar thing can be said in the update of L_x and L_y in Algorithm 2, since different rows are updated independently. We implement a multicore version for both Algorithm 1 and Algorithm 2. Specifically, suppose that we have K processors. we randomly partition $\{1,2,\ldots,d_x\}$ and $\{1,2,\ldots,d_y\}$ into K subsets. In Algorithm 1, different processors share $\mathcal{S}$ and calculate $\{w_{xu}\}$ and $\{w_{yv}\}$ with indices in their own partition simultaneously. In Algorithm 2, when updating L_x, different processors share the same input and L_y. Rows of L_x with indices in different partitions are updated simultaneously. The same parallelization strategy is used when updating L_y.

5. Generalization Analysis

We conduct generalization analysis for matching framework (2) in this section. We first give a generalization bound for the framework, which relies on the complexity of hypothesis spaces $\mathcal{H}_x$ and $\mathcal{H}_y$. After that, we analyze the complexity of $\mathcal{H}_x$ and $\mathcal{H}_y$ for both RMLS and PLS, and give their specific bounds. The proofs of the theorems are given in Appendix B.

We formally define $D(\mathcal{S})$ as the gap between the expected objective and the empirical objective over all L_x and L_y

$$D(\mathcal{S}) \doteq \sup_{L_x,L_y} \left| \frac{1}{n^x} \sum_{i=1}^{n^x} \frac{1}{n_i^y} \sum_{j=1}^{n_i^y} r_{ij} x_i^\top L_x L_y^\top y_{ij} - \mathbb{E}_{x,y}\left(r(x,y) x^\top L_x L_y^\top y \right) \right|,$$

and bound it. With this bound, given a solution $(\hat{L}_x, \hat{L}_y)$, we can estimate its performance on unseen data (i.e., $\mathbb{E}_{x,y}\left(r(x,y)x^\top \hat{L}_x \hat{L}_y^\top y\right)$) based on its performance on observed samples. For notational simplicity, we define $f_{L_x,L_y}(x,y) \doteq r(x,y) x^\top L_x L_y^\top y$, and further assume

$$\|x\| \leqslant 1, \;\; \|y\| \leqslant 1, \;\; r(x,y) \geqslant 0, \;\; \sup_{x,y} r(x,y) \leqslant R.$$

To characterize the sparsity of inputs, we suppose that the numbers of nonzeros in x and y are bounded by m_x and m_y.

Under Assumption 1, we divide $D(\mathcal{S})$ into two parts:

1. $\sup_{L_x,L_y} \left| \frac{1}{n^x} \sum_{i=1}^{n^x} \left(\frac{1}{n_i^y} \sum_{j=1}^{n_i^y} f_{L_x,L_y}(x_i, y_{ij}) - \mathbb{E}_{y|\{x_i\}} f_{L_x,L_y}(x_i, y) \right) \right|$, denoted as $D_1(\mathcal{S})$,

2. $\sup_{L_x,L_y} \left| \frac{1}{n_x} \sum_{i=1}^{n^x} \mathbb{E}_{y|\{x_i\}} f_{L_x,L_y}(x_i, y) - \mathbb{E}_{x,y} f_{L_x,L_y}(x,y) \right|$, denoted as $D_2(\{x_i\}_{i=1}^{n^x})$.

Clearly $D(\mathcal{S}) \leqslant D_1(\mathcal{S}) + D_2(\{x_i\}_{i=1}^{n^x})$, thus we separately bound $D_1(\mathcal{S})$ and $D_2(\{x_i\}_{i=1}^{n^x})$, and finally obtain the bound for $D(\mathcal{S})$.

We first bound $D_1(\mathcal{S})$. Suppose $\sup_{x,y,L_x,L_y} \|L_x^\top x\| \, \|L_y^\top y\| \leqslant B$, and $\sup_{L_x,L_y} \|\mathrm{vec}(L_x L_y^\top)\| \leqslant C$, where B and C are constants and $\mathrm{vec}(\cdot)$ is the vectorization of a matrix. We have

Theorem 1 *Given an arbitrary small positive number* δ*, with probability at least* $1-\delta$*, the following inequality holds:*

$$D_1(\mathcal{S}) \leqslant \frac{2CR}{\sqrt{n^x n^y}} + \frac{RB\sqrt{2\log\frac{1}{\delta}}}{\sqrt{n^x n^y}},$$

where n^y *represents the harmonic mean of* $\{n_i^y\}_{i=1}^{n^x}$.

Using similar techniques we have

Theorem 2 *Given an arbitrary small positive number* δ*, with probability at least* $1-\delta$*, the following inequality holds:*

$$D_2(\{x_i\}_{i=1}^{n^x}) \leqslant \frac{2CR}{\sqrt{n^x}} + \frac{RB\sqrt{2\log\frac{1}{\delta}}}{\sqrt{n^x}}.$$

Combining Theorem 1 and Theorem 2, we are able to bound $D(\mathcal{S})$:

Theorem 3 *Given an arbitrary small positive number* δ*, with probability at least* $1-2\delta$*, the following inequality holds:*

$$D(\mathcal{S}) \leqslant (2CR + RB\sqrt{2\log\frac{1}{\delta}})(\frac{1}{\sqrt{n^x n^y}} + \frac{1}{\sqrt{n^x}}). \tag{7}$$

Equation (7) gives a general generalization bound for framework (2). Since $n^y = \frac{n^x}{\sum_{i=1}^{n^x} 1/n_i^y}$, the bound tells us that to make the gap between the empirical objective and the expected objective small enough, we not only need large n^x, but also need large n_i^y for *each* x_i, which is consistent with our intuition. The two constants B and C are dependent on the hypothesis spaces $\mathcal{H}_x$ and $\mathcal{H}_y$. Below we will analyze B and C for PLS and RMLS, and give their specific bounds based on (7).

The following two theorems give B and C for PLS and RMLS, and give their specific bounds:

Theorem 4 *Suppose that* $\mathcal{H}_x = \{L_x \mid L_x^\top L_x = \mathbb{I}_{d\times d}\}$ *and* $\mathcal{H}_y = \{L_y \mid L_y^\top L_y = \mathbb{I}_{d\times d}\}$*, then* $B = 1$ *and* $C = \sqrt{d}$*. Thus, the generalization bound for PLS is given by*

$$D(\mathcal{S}) \leqslant (2\sqrt{d}R + R\sqrt{2\log\frac{1}{\delta}})(\frac{1}{\sqrt{n^x n^y}} + \frac{1}{\sqrt{n^x}}).$$

Theorem 5 *Suppose that* $\mathcal{H}_x = \{L_x \mid ||l_{xu}|| \leqslant \lambda_x, ||l_{xu}|| \leqslant \theta_x, 1 \leqslant u \leqslant d_x\}$ *and* $\mathcal{H}_y = \{L_y \mid ||l_{yv}|| \leqslant \lambda_y, ||l_{yv}|| \leqslant \theta_y, 1 \leqslant v \leqslant d_y\}$*. If we suppose that the numbers of nonzero elements in x and y are respectively bounded by* m_x *and* m_y*, then* $B = \sqrt{m_x m_y}\min(d\lambda_x\lambda_y, \theta_x\theta_y)$ *and* $C = \sqrt{d_x d_y}\min(\lambda_x\lambda_y, \theta_x\theta_y)$*. Thus, the generalization bound for RMLS is given by*

$$D(\mathcal{S}) \leqslant (\frac{1}{\sqrt{n^x n^y}} + \frac{1}{\sqrt{n^x}}) \times (2\sqrt{d_x d_y}\min(\lambda_x\lambda_y, \theta_x\theta_y)R + \sqrt{m_x m_y}\min(d\lambda_x\lambda_y, \theta_x\theta_y)R\sqrt{2\log\frac{1}{\delta}}).$$

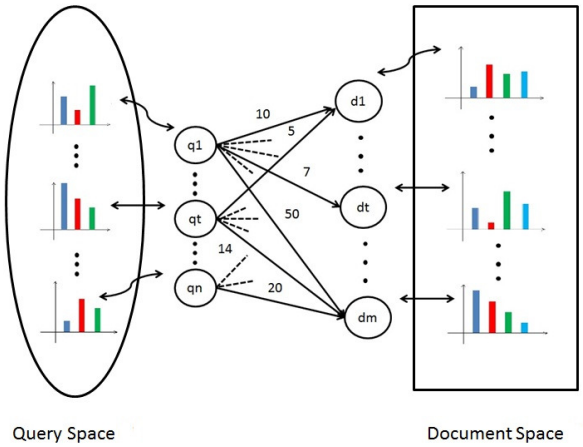

Figure 1: Click-through bipartite graph with metadata on nodes, representing queries and documents in feature spaces and their associations.

6. Experiment

We applied both RMLS and PLS to relevance ranking in web search, where matching models are used to predict relevance. Specifically, we suppose that we have a click-through bipartite with vertices representing queries and documents. The edges between query vertices and document vertices are weighted by number of user click. Besides this, we assume that there exists metadata on the vertices of the graph. The metadata represents features of queries and documents. The features may stand for the content of queries and documents and the clicks of queries and documents on the bipartite graph (Baeza-Yates and Tiberi, 2007), as seen below. Queries and documents are represented as feature vectors in the query space and the document space, respectively. Figure 1 illustrates the relationships.

We implemented the matching framework (2) and its associated RMLS and PLS using the click-through bipartite with metadata. $\mathcal{X}$ and $\mathcal{Y}$ are the query space and the document space respectively. Given a query x and a document y, we treated user click number as the response r. In this case, we actually leveraged both user clicks and features of queries and documents to perform matching. We conducted experiments on a small data set and a large data set with millions of queries and documents.

6.1 Experiment Setup

We collected 1) one week of click-through data and 2) half a year of click-through data from a commercial web search engine. After filtering out noise, there are $94,022$ queries and $111,631$ documents in the one week data set, and $6,372,254$ queries and $4,599,849$ documents in the half year data set. We extracted features from two sources, namely word and clicks. For the word feature, we represented queries and documents as tf-idf vectors (Salton and McGill, 1986) in a word space, where words are extracted from queries, URLs and the titles of documents. There are $101,904$ and $271,561$ unique words in one week data and half year data respectively. For the click feature, we followed (Baeza-Yates and Tiberi, 2007) and took the number of clicks of documents as a feature of queries, and the number of clicks of queries as a feature of documents.

Finally, we concatenated the features from the two sources to create long but extremely sparse feature vectors for both queries and documents. Note that query space and document space have different dimensions and characteristics, and should be treated as heterogeneous domains. Table 1 gives the statistics on the experiment data. Note that the notations in the table are the same as in Section 4.

6.1.1 BASELINE METHODS

We employed three different kinds of baseline methods:

- **Individual feature based model and graph based model:** We employed BM25 (Robertson et al., 1994) as a representative of feature based relevance models, and LSI (Deerwester et al., 1990) and random walk on click-through bipartite graph (Craswell and Szummer, 2007) ("RW" for short) as representatives of graph based relevance models. Particularly, we implemented two versions of LSI in this paper, one on a document-term matrix, denoted as LSI_{dt}, the other one on a query-document matrix with each element representing the click number, denoted as LSI_{qd}.

- **Combination of feature based model and graph based model:** We used models which linearly combine LSI_{qd} and random walk with BM25, denoted as LSI_{qd}+BM25 and RW+BM25, respectively.

- **Other existing models:** Besides the heuristic combination models, we also employed the bilingual topic model (BLTM) proposed by Gao et al. (2011) and supervised semantic indexing (SSI) proposed by Bai et al. (2009) as baseline methods. BLTM is the best performing model in Gao et al. (2011), and it can leverage both the user clicks and the content of queries and documents. SSI employs a hinge loss function to model pairwise preference between objects. It learns a large margin perceptron to map queries and documents into a latent space and measures their similarity in the space. To implement SSI, we additionally collected impression data from the search log. Besides click numbers, the data also contains positions of documents in ranking lists of search engine. We followed the rules proposed by Joachims (2002) to generate preference pairs.

- **Model proposed for collaborative filtering:** Besides the models in information retrieval, we also employed a state of the art model in collaborative filtering as a baseline method[3] (Chen et al., 2012). The model, named SVDFeature, can leverage the same features as RMLS and PLS.

We obtained relevance data consisting of judged query-document pairs from the search engine in a different time period from the click-through data. There are five levels of judgments, including "Perfect", "Excellent", "Good", "Fair", and "Bad". For one week data, we obtained $4,445$ judged queries and each query has on average 11.34 judged documents. For half year data, more judged data was collected. There are $57,514$ judged queries and each query has on average 13.84 judged documents. We randomly split each judged data set and used half of them for tuning model parameters and the other half for model evaluation. In summary, for both data sets, we learned models on the whole click-through data, tuned model parameters on the validation set of relevance data and evaluated model performances on the held-out test set.

3. The model won the 1st place in track 1 of KDD-Cup 2012.

	d_x	d_y	c_x	c_y	$\tilde{n}^y$	$\tilde{n}^x$	N_x	N_y
one week	$2.1 \cdot 10^5$	$2.0 \cdot 10^5$	4.0	5.9	1.74	1.46	1.7	3.4
half year	$4.9 \cdot 10^6$	$6.6 \cdot 10^6$	5.5	8.6	2.92	4.04	7.2	5.9

Table 1: Statistics on query and document features

To evaluate the performances of different methods, we employed Normalized Discounted Cumulative Gain (NDCG) (Jarvelin and Kekalainen, 2000) at positions of 1, 3, and 5 as evaluation measures.

6.2 Parameter Setting

We set the parameters for the methods in the following way. In BM25, the default setting was used. There are two parameters in random walk: the self-transition probability and the number of transition steps. Following the conclusion from Craswell and Szummer (2007), we fixed the self-transition probability as 0.9 and chose the number of transition steps from $\{1,\ldots,10\}$ on the validation set. We found that random walk reaches a "stable" state with just a few steps. In our experiments, after five steps we saw no improvement on the validation data in terms of evaluation measures. Therefore, we set five as the number of transition steps of random walk.

In LSI, SVDFeature, BLTM, SSI, PLS and RMLS, one important parameter is the dimensionality of latent space. We set the parameter in the range of $\{100,200,\ldots,1000\}$. We found that the performance of both PLS and RMLS on the validation data improves with dimensionality. On the other hand, a large dimensionality means more parameters to store in memory ($d \times (d_x + d_y)$) and more training time for each iteration. Therefore, we finally chose 1000 as the dimensionality of latent space for PLS and RMLS. For other baseline methods except SSI, a similar phenomenon was observed. For SSI, we found that its performance on the validation data is not sensitive to the dimensionality of latent space.

Besides dimensionality of latent space, parameters for regularization items (e.g., θ_x, θ_y, β, and γ in Equation (4), learning rates in SSI and SVDFeature, and number of iterations may also affect the final performance and therefore need tuning. Again, we tuned these parameters on the validation data one by one. Particularly, we found that the performance of RMLS is not sensitive to parameters for ℓ_2 norm (i.e., θ_x, θ_y). In addition, we observed that RMLS can quickly reach a good performance after a few iterations (less than 10 loops), and the early stopping also led to good generalization on the test data.

In LSI_{qd}+BM25 and RW+BM25, the combination weights are also parameters. We tuned the combination weights within $\{0.1,0.2,\ldots,0.9\}$ on the validation data.

6.3 Results on One Week Data

We conducted experiments on a workstation with 24 AMD Opteron 6172 processors and 96 GB RAM. We first compared the performance of different methods, with results summarized in Table 2. We can see that RMLS performs comparably well with PLS, and both of them significantly outperform all other baselines ($p < 0.05$ from t-test). Among the baseline methods, the performance of SSI is rather poor. We analyzed the data and found that although pairwise preference can alleviate rank bias, it also misses some important information. For example, we observed that 49.1% of 40,676 pairs that have judgments in our labeled data violate the rules proposed by Joachims (2002).

	NDCG@1	NDCG@3	NDCG@5
RMLS	**0.686**	**0.732**	0.729
PLS	0.676	**0.728**	**0.736**
SVDFeature	0.663	0.720	0.727
BLTM	0.657	0.702	0.701
SSI	0.538	0.621	0.629
RW	0.655	0.704	0.704
RW+BM25	0.671	0.718	0.716
LSI_{qd}	0.588	0.665	0.676
LSI_{qd}+BM25	0.649	0.705	0.706
LSI_{dt}	0.616	0.675	0.680
BM25	0.637	0.690	0.690

Table 2: Relevance ranking result on one week data

Documents ranked higher in these pairs have more click-through rates[4] and better or equally good judgments than documents ranked lower. Those query document associations will be well represented in either PLS or RMLS.

We then compared RMLS with PLS on efficiency. In PLS, the linear mappings are learned through SVD. We implemented an SVD solver using power method (Wegelin, 2000) with C++, and further optimized the data structure for our task. This SVD implementation can handle large data sets on which state-of-the-art SVD tools like SVDLIBC[5] fail. Since the efficiency of algorithms is influenced by implementation strategies, for example, different numbers of iterations or termination criteria, to make a fair comparison, we only report the time cost in the learning of the best performing models. RMLS significantly improves the efficiency of PLS. On a single processor, it takes RMLS $1,380$ seconds to train the model, while the training of PLS needs $945,382$ seconds. The reason is that PLS requires SVD and has a complexity of at least $O(dcd_xd_y + d^2\max(d_x,d_y))$, where c represents the density of the matrix for SVD. Even with a small c, the high dimensionality of input space (i.e., large d_x and d_y) and the quadratic growth with respect to d still make SVD quite expensive. For RMLS, W_x and W_y are quite small with a sparse input (W_x=24.82 W_y= 27.05), and hence the time complexity is nearly linear to $d \cdot \max(d_x,d_y)$. Therefore, RMLS is significantly more efficient than PLS with high dimensional but sparse inputs.

Finally, we examined the time cost of parallelized RMLS on multiple processors, as summarized by Figure 2. Clearly the running time decreases with the number of threads. With 20 threads, RMLS only takes 277 seconds to achieve a comparable performance with PLS.

6.4 Results on Half Year Data

We further tested the performance of RMLS and PLS on a half year data set with millions of queries and documents. On such large scale, SVD-based methods and random walk become infeasible (e.g., taking months to run). SSI is also infeasible because of the huge number of pairs. We therefore implemented RMLS with *full* features and PLS with only word features, and compared them with

4. Click-through rate = number of clicks / number of impressions.
5. SVDLIBC can be found at `http://tedlab.mit.edu/~dr/SVDLIBC/`.

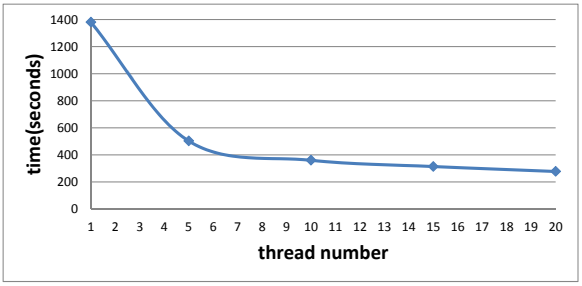

Figure 2: Time cost trend of RMLS under multiple processors.

	NDCG@1	NDCG@3	NDCG@5
RMLS	**0.742**	**0.767**	**0.776**
PLS (word)	0.638	0.666	0.677
SVDFeature	**0.742**	0.746	0.749
BLTM	0.684	0.708	0.717
BM25	0.643	0.663	0.670

Table 3: Relevance ranking result on half year data

BM25, BLTM, and SVDFeature.[6] With only word features ($d_x = d_y = 271,561$), PLS is slow but still feasible.

As shown in Table 3, RMLS outperforms all the baselines ($p < 0.01$ from t-test). We hypothesize that PLS with full features can perform comparably with RMLS, but the high computation complexity of PLS prevents us from testing it. For RMLS, it takes 20,523 seconds to achieve the result using 20 threads. For PLS, although it only uses word features, it takes 1,121,440 seconds to finish learning. In other words, parallelized RMLS can be used to tackle web search problem of real-world scale.

6.5 Discussion

In this section, we investigate the effect of matching models as features in a state of the art learning to rank algorithm and performance of matching models across queries with different numbers of click.

6.5.1 MATCHING MODELS AS FEATURES IN A LEARNING TO RANK ALGORITHM

Results in Table 2 and Table 3 demonstrate the efficacy of RMLS and PLS as individual relevance models. In modern web search, relevance models usually act as features in a learning to rank system. Therefore, a natural question is whether RMLS or PLS can enhance the performance of learn to rank algorithms as an additional feature. To answer this question, we employed RankLib[7] and trained a gradient boosted tree (MART, Friedman, 2001) with the validation data. We conducted experiments in the following three steps: first, we trained a ranker with all baseline methods as features. We denoted it as MART-Baseline. Then, we included PLS and RMLS respectively as an additional

6. SVDFeature is actually *not* based on SVD implementation.
7. RankLib can be found at `http://people.cs.umass.edu/~vdang/ranklib.html`.

	NDCG@1	NDCG@3	NDCG@5
MART-Baseline	0.661	0.737	0.779
MART-PLS	0.683	0.751	0.792
MART-RMLS	0.681	0.750	0.789
MART-All	0.689	0.757	0.797

Table 4: Performance of gradient boosted tree (MART) on one week data

	NDCG@1	NDCG@3	NDCG@5
MART-Baseline	0.708	0.760	0.791
MART-PLS	0.706	0.760	0.792
MART-RMLS	0.757	0.799	0.827
MART-All	0.756	0.798	0.826

Table 5: Performance of gradient boosted tree (MART) on half year data

feature and denoted the new rankers as MART-PLS and MART-RMLS, respectively. Finally, we trained a ranker with all models as features and denoted it as MART-All. Table 4 and Table 5 present the evaluation results.

From Table 4, we conclude that 1) RMLS and PLS significantly improve the performance of MART with baseline methods as features, which demonstrates the efficacy of the proposed framework in relevance ranking; 2) RMLS can be a good alternative to PLS in practice, because MART-PLS and MART-RMLS are comparable in ranking performance but RMLS is more efficient and scalable; 3) There is overlap between the effect of RMLS and PLS in learning to rank, because when including all models as features, the performance of ranker is only slightly improved. Results in Table 5 further demonstrate the advantage of RMLS in relevance ranking. PLS is not capable of leveraging all features of large scale data, and therefore fails to improve the performance of MART. On the other hand, RMLS successfully leverages both word features and click features, and significantly improves the ranking performance of MART.

6.5.2 EVALUATION ACROSS QUERIES WITH DIFFERENT NUMBERS OF CLICKS

In Framework (2), response r is treated as a weight for each object pair (x,y). The framework, when applied to web search, weights each query-document pair with the number of clicks between them. Usually, number of clicks has a large variance among queries, from a few to tens of thousands. An interesting question is therefore how different matching models perform across queries with different numbers of click. To answer this question, we divided queries into different bins based on the total numbers of clicks associated with them over documents. We took four levels: $totalclick \leqslant 10$, $10 < totalclick \leqslant 100$, $100 < totalclick \leqslant 1000$, and $totalclick > 1000$. We separately evaluated matching models on each level. Table 6 and Table 7 show the evaluation results, where @1, @3, and @5 mean NDCG@1, NDCG@3, and NDCG@5, respectively.

From Table 6, we can see that RMLS and PLS beat other baseline methods on queries with moderate and large number of clicks, but lose to RW and RW+BM25 when queries only have relatively few clicks (less than 100). RMLS and PLS use the absolute click number as a weight for each query-document pair. Therefore, in training, head queries may overwhelm those tail queries. We try to mitigate this effect by taking some simple transformations (e.g., logarithm) on click num-

	$totalclick \leqslant 10$			$10 < totalclick \leqslant 100$			$100 < totalclick \leqslant 1000$			$totalclick > 1000$		
	# *queries* = 230			#*queries* = 772			# *queries* = 757			#*queries* = 464		
	@1	@3	@5	@1	@3	@5	@1	@3	@5	@1	@3	@5
RMLS	0.754	**0.795**	**0.767**	0.749	0.791	0.766	**0.679**	**0.744**	**0.755**	0.557	0.612	0.655
PLS	0.704	0.776	0.764	0.706	0.769	0.748	0.650	0.727	**0.754**	**0.655**	**0.661**	**0.695**
SVDFeature	0.648	0.732	0.707	0.684	0.742	0.727	0.647	0.723	0.750	0.576	0.632	0.669
BLTM	0.758	0.787	0.755	0.750	0.795	0.766	0.636	0.700	0.726	0.485	0.557	0.603
SSI	0.633	0.730	0.700	0.607	0.696	0.673	0.523	0.616	0.657	0.403	0.491	0.540
RW	0.769	0.793	0.760	**0.773**	0.809	0.780	0.622	0.709	0.740	0.458	0.527	0.582
RW+BM25	**0.773**	0.786	0.758	0.770	**0.815**	**0.787**	0.654	0.726	0.750	0.485	0.554	0.601
LSI_{qd}	0.631	0.717	0.687	0.634	0.709	0.696	0.584	0.676	0.703	0.496	0.573	0.621
LSI_{qd}+BM25	0.696	0.745	0.719	0.726	0.777	0.759	0.646	0.716	0.736	0.500	0.576	0.619
LSI_{dt}	0.685	0.745	0.730	0.688	0.752	0.727	0.608	0.676	0.705	0.473	0.549	0.596
BM25	0.698	0.746	0.724	0.719	0.772	0.748	0.641	0.701	0.722	0.463	0.544	0.592

Table 6: Evaluation on different query bins on one week data

	$totalclick \leqslant 10$			$10 < totalclick \leqslant 100$			$100 < totalclick \leqslant 1000$			$totalclick > 1000$		
	# *queries* = 704			# *queries* = 5260			# *queries* = 8980			# *queries* = 13813		
	@1	@3	@5	@1	@3	@5	@1	@3	@5	@1	@3	@5
RMLS	**0.804**	**0.801**	**0.792**	**0.785**	**0.794**	**0.796**	**0.780**	**0.795**	**0.797**	0.698	**0.742**	**0.760**
PLS (word)	0.723	0.720	0.708	0.681	0.697	0.703	0.668	0.691	0.697	0.599	0.642	0.660
SVDFeature	0.728	0.730	0.721	0.738	0.734	0.734	0.766	0.765	0.761	**0.728**	**0.740**	0.748
BLTM	0.783	0.775	0.763	0.750	0.757	0.760	0.725	0.745	0.749	0.626	0.669	0.688
BM25	0.747	0.739	0.717	0.710	0.713	0.712	0.690	0.703	0.704	0.582	0.622	0.641

Table 7: Evaluation on different query bins on half year data

bers, but find that simple transformations not only fail to deal with tail query issues but also hurt the performance of RMLS and PLS on head queries. In contrast, BLTM and SSI perform better on tail queries than themselves on head queries. The phenomenon reminds us that introducing large margin into Framework (2) could be a potential approach to solve the problem of RMLS, although after doing so, scalability may become a more serious issue, which we leave to our future work.

SVDFeature suffers from head query effect more seriously than RMLS and PLS, which may stem from its directly fitting similarity function with absolute click numbers.

In Table 7, due to the scalability issue, results of some baseline methods are not available. In spite of this, we can still see that SVDFeature performs consistently with itself on one week data, and we can also guess that the comparisons of RMLS with RW and RW+BM25 may follow the same trends as those on one week data.

7. Conclusion

We have proposed a framework for learning to match heterogeneous data via shared latent structures, and studied its generalization ability under a hierarchical sampling assumption for web search. The framework subsumes Partial Least Squares (PLS) as a special case, and enables us to devise a more scalable algorithm called Regularized Mapping to Latent Structures (RMLS) as another special case. We applied both PLS and RMLS to web search, leveraging a click-through bipartite graph with metadata representing features of queries and documents to learn relevance models. Results on a small data set and a large data set with millions of queries and documents show the promising performance of PLS and RMLS, and particularly demonstrate the advantage of RMLS on scalability.

Appendix A. Degenerative Solution With ℓ_1 Constraints *Only*

Suppose that all the input vectors have only non-negative elements (which is natural in web search), we consider the following learning problem:

$$\begin{aligned} \arg\max_{L_x, L_y} \quad & \frac{1}{n^x}\sum_{i=1}^{n^x}\frac{1}{n_i^y}\sum_{j=1}^{n_i^y} r_{ij} x_i^\top L_x L_y^\top y_{ij}, \\ s.t. \quad & |l_{xu}| \leqslant \lambda_x, \quad 1 \leqslant u \leqslant d_x, \\ & |l_{yv}| \leqslant \lambda_y, \quad 1 \leqslant v \leqslant d_y. \end{aligned} \tag{8}$$

We assert that the optimization problem (8) has a global optimum, and the global optimum can be obtained by letting $L_x = \lambda_x e_x l^\top$ and $L_y = \lambda_y e_y l^\top$, where l is a d dimensional vector satisfying $|l| = 1$ and $\|l\| = 1$, e_x is a d_x dimensional vector with all elements ones and e_y is a d_y dimensional vector with all elements ones. To demonstrate this, first we prove that the objective function (8) can be upper bounded under ℓ_1 constraints:

Proof The objective of problem (8) can be re-written as

$$\begin{aligned} \frac{1}{n^x}\sum_{i=1}^{n^x}\frac{1}{n_i^y}\sum_{j=1}^{n_i^y} r_{ij} x_i^\top L_x L_y^\top y_{ij} &= \sum_{i=1}^{n^x}\sum_{j=1}^{n_i^y}\frac{r_{ij}}{n^x n_i^y}\sum_{u=1}^{d_x}\sum_{v=1}^{d_y}(l_{xu})^\top\left(x_i^{(u)} y_{ij}^{(v)} l_{yv}\right) \\ &= \sum_{u=1}^{d_x}\sum_{v=1}^{d_y}(l_{xu})^\top\left(\sum_{i=1}^{n^x}\sum_{j=1}^{n_i^y}\frac{r_{ij}}{n^x n_i^y} x_i^{(u)} y_{ij}^{(v)} l_{yv}\right). \end{aligned}$$

If we define $a_{uv} = \sum_{i=1}^{n^x}\sum_{j=1}^{n_i^y}\frac{r_{ij}}{n^x n_i^y} x_i^{(u)} y_{ij}^{(v)}$, the objective of problem (8) can be re-written as

$$\sum_{u=1}^{d_x}\sum_{v=1}^{d_y} a_{uv} l_{xu}^\top l_{yv}.$$

Since the input vectors are non-negative, $a_{uv} \geqslant 0, \forall u, v$. Thus, we have

$$\begin{aligned} \sum_{u=1}^{d_x}\sum_{v=1}^{d_y} a_{uv} l_{xu}^\top l_{yv} &\leqslant \sum_{u=1}^{d_x}\sum_{v=1}^{d_y} a_{uv}\left(\sum_{k=1}^{d}|l_{xu}^{(k)}||l_{yv}^{(k)}|\right) \\ &\leqslant \sum_{u=1}^{d_x}\sum_{v=1}^{d_y} a_{uv}\left(\max_{1\leqslant k\leqslant d}(|l_{xu}^{(k)}|)\sum_{k=1}^{d}|l_{yv}^{(k)}|\right). \end{aligned}$$

Since $|l_{xu}| \leqslant \lambda_x$ and $|l_{yv}| \leqslant \lambda_y$, we know that $\max_{1\leqslant k\leqslant d}(|l_{xu}^{(k)}|) \leqslant \lambda_x$, and thus we have

$$\sum_{u=1}^{d_x}\sum_{v=1}^{d_y} a_{uv}\left(\max_{1\leqslant k\leqslant d}(|l_{xu}^{(k)}|)\sum_{k=1}^{d}|l_{yv}^{(k)}|\right) \leqslant \sum_{u=1}^{d_x}\sum_{v=1}^{d_y} a_{uv}\lambda_x\lambda_y.$$

■

With the existence of the upper bound, we can see that if $L_x = \lambda_x e_x l^\top$ and $L_y = \lambda_y e_y l^\top$, the value of the objective (8) is

$$\sum_{u=1}^{d_x}\sum_{v=1}^{d_y} a_{uv} l_{xu}^\top l_{yv} = \sum_{u=1}^{d_x}\sum_{v=1}^{d_y} a_{uv}\lambda_x\lambda_y\|l\|^2 = \sum_{u=1}^{d_x}\sum_{v=1}^{d_y} a_{uv}\lambda_x\lambda_y.$$

Thus, the optimization problem (8) reaches its global optimum when $L_x = \lambda_x e_x l^\top$ and $L_y = \lambda_y e_y l^\top$, which is an undesired degenerative solution.

Appendix B. Proofs of Theorems

We give the proofs of the theorems in Section 5.

B.1 Proof of Theorem 1

Theorem 1 *Given an arbitrary small positive number δ, with probability at least $1-\delta$, the following inequality holds:*

$$D_1(\mathcal{S}) \leqslant \frac{2CR}{\sqrt{n^x n^y}} + \frac{RB\sqrt{2\log\frac{1}{\delta}}}{\sqrt{n^x n^y}},$$

where n^y represents the harmonic mean of $\{n_i^y\}_{i=1}^{n^x}$.

To prove this theorem, we need two lemmas:

Lemma 1 *Given $\varepsilon > 0$, the following inequality holds:*

$$P\left(D_1(\mathcal{S}) - \mathbb{E}_{\{y_{ij}\}|\{x_i\}} D_1(\mathcal{S}) \geqslant \varepsilon | \{x_i\}\right) \leqslant \exp\left(-\frac{\varepsilon^2 n^x n^y}{2R^2B^2}\right).$$

Proof Given $\{x_i\}_{i=1}^{n^x}$, we re-write $D_1(\mathcal{S})$ as $D_1(\{y_{ij}\}|\{x_i\})$. Since $\sup_{x,y,L_x,L_y} ||L_x^\top x||||L_y^\top y|| \leqslant B$ and $r \leqslant R$, $\forall\, u, v$, we have

$$|D_1(\{y_{ij}\}|\{x_i\}) - D_1\left((\{y_{ij}\} - y_{uv}) \bigcup y'_{uv} | \{x_i\}\right)| \leqslant \frac{2RB}{n^x n_u^y}.$$

Given $\{x_i\}_{i=1}^{n^x}$, $\{y_{ij}\}$ are independent. By McDiarmid inequality (Bartlett and Mendelson, 2002), we know

$$\begin{aligned}
P\left(D_1(\mathcal{S}) - \mathbb{E}_{\{y_{ij}\}|\{x_i\}} D_1(\mathcal{S}) \geqslant \varepsilon | \{x_i\}\right) &\leqslant exp\left(-\frac{2\varepsilon^2}{\sum_{i=1}^{n^x}\sum_{j=1}^{n_i^y}\frac{4R^2B^2}{\left(n^x n_i^y\right)^2}}\right) \\
&= exp\left(-\frac{\varepsilon^2}{2R^2B^2\sum_{i=1}^{n^x}\frac{1}{(n^x)^2 n_i^y}}\right) \\
&= exp\left(-\frac{\varepsilon^2 (n^x)^2}{2R^2B^2\sum_{i=1}^{n^x}\frac{1}{n_i^y}}\right) \\
&= exp\left(-\frac{\varepsilon^2 n^x n^y}{2R^2B^2}\right).
\end{aligned}$$

■

Lemma 2

$$\mathbb{E}_{\{y_{ij}\}|\{x_i\}} D_1(\mathcal{S}) \leqslant \frac{2CR}{\sqrt{n^x n^y}}.$$

Proof Define $r(x,y)x^\top L_x L_y^\top y$ as $f_{L_x,L_y}(x,y)$. We have

$$\begin{aligned}\mathbb{E}_{\{y_{ij}\}|\{x_i\}} D_1(\mathcal{S}) &= \mathbb{E}_{\{y_{ij}\}|\{x_i\}} \sup_{L_x,L_y} |\frac{1}{n^x}\sum_{i=1}^{n^x}\frac{1}{n_i^y}\sum_{j=1}^{n_i^y} f_{L_x,L_y}(x_i,y_{ij}) - \frac{1}{n^x}\sum_{i=1}^{n^x}\mathbb{E}_{y|\{x_i\}} f_{L_x,L_y}(x_i,y)| \\ &= \mathbb{E}_{\{y_{ij}\}|\{x_i\}} \sup_{L_x,L_y} |\frac{1}{n^x}\sum_{i=1}^{n^x}\frac{1}{n_i^y}\sum_{j=1}^{n_i^y} f_{L_x,L_y}(x_i,y_{ij}) - \frac{1}{n^x}\sum_{i=1}^{n^x}\frac{1}{n_i^y}\sum_{j=1}^{n_i^y}\mathbb{E}_{\{y'_{ij}\}|\{x_i\}} f_{L_x,L_y}(x_i,y'_{ij})|,\end{aligned}$$

where $\{y'_{ij}\}$ are i.i.d. random variables with $\{y_{ij}\}$.

$$\begin{aligned}&\mathbb{E}_{\{y_{ij}\}|\{x_i\}} \sup_{L_x,L_y} |\frac{1}{n^x}\sum_{i=1}^{n^x}\frac{1}{n_i^y}\sum_{j=1}^{n_i^y} f(x_i,y_{ij}) - \frac{1}{n^x}\sum_{i=1}^{n^x}\frac{1}{n_i^y}\sum_{j=1}^{n_i^y}\mathbb{E}_{\{y'_{ij}\}|\{x_i\}} f(x_i,y'_{ij})| \\ &\leqslant \mathbb{E}_{\{y_{ij}\},\{y'_{ij}\}|\{x_i\}} \sup_{L_x,L_y} \frac{1}{n^x}\sum_{i=1}^{n^x}\frac{1}{n_i^y}\sum_{j=1}^{n_i^y} |f(x_i,y_{ij}) - f(x_i,y'_{ij})| \\ &= \mathbb{E}_{\{y_{ij}\},\{y'_{ij}\},\{\sigma_{ij}\}|\{x_i\}} \sup_{L_x,L_y} \frac{1}{n^x}\sum_{i=1}^{n^x}\frac{1}{n_i^y}\sum_{j=1}^{n_i^y} \sigma_{ij}\left(f(x_i,y_{ij}) - f(x_i,y'_{ij})\right),\end{aligned}$$

where given $\{x_i\}_{i=1}^{n^x}$, $\{\sigma_{ij}\}$ are i.i.d. random variables with $P(\sigma_{ij}=1) = P(\sigma_{ij}=-1) = 0.5$.

$$\mathbb{E}_{\{y_{ij},y'_{ij},\sigma_{ij}\}|\{x_i\}} \sup_{L_x,L_y} \sum_{i=1}^{n^x}\sum_{j=1}^{n_i^y} \frac{\sigma_{ij}\left(f(x_i,y_{ij}) - f(x_i,y'_{ij})\right)}{n^x n_i^y} \leqslant 2\mathbb{E}_{\{y_{ij},\sigma_{ij}\}|\{x_i\}} \sup_{L_x,L_y} |\sum_{i=1}^{n^x}\sum_{j=1}^{n_i^y} \frac{\sigma_{ij} f(x_i,y_{ij})}{n^x n_i^y}|.$$

Note that

$$\sigma_{ij} f(x_i,y_{ij}) = \sigma_{ij} r(x_i,y_{ij}) x_i^\top L_x L_y^\top y_{ij} = \sigma_{ij}\left\langle vec(L_x L_y^\top), r(x_i,y_{ij})\mathrm{vec}(y_{ij}\otimes x_i)\right\rangle,$$

where $y_{ij}\otimes x_i$ represents the tensor of column vectors y_{ij} and x_i, and $\mathrm{vec}(\cdot)$ is the vectorization of a matrix. Thus, we have

$$\begin{aligned}&\sup_{L_x,L_y} |\frac{1}{n^x}\sum_{i=1}^{n^x}\frac{1}{n_i^y}\sum_{j=1}^{n_i^y}\sigma_{ij} f(x_i,y_{ij})| \\ &= \sup_{L_x,L_y} |\langle \mathrm{vec}(L_x L_y^\top), \frac{1}{n^x}\sum_{i=1}^{n^x}\frac{1}{n_i^y}\sum_{j=1}^{n_i^y}\sigma_{ij} r(x_i,y_{ij})\mathrm{vec}(y_{ij}\otimes x_i)\rangle| \\ &\leqslant \sup_{L_x,L_y} ||\mathrm{vec}(L_x L_y^\top)|| \sqrt{\frac{1}{(n^x)^2}\sum_{i=1}^{n^x}\sum_{j=1}^{n^x}\frac{1}{n_i^y n_j^y}\sum_{u=1}^{n_i^y}\sum_{v=1}^{n_j^y}\sigma_{iu}\sigma_{jv} r(x_i,y_{iu}) r(x_j,y_{jv})\langle x_i,x_j\rangle\langle y_{iu},y_{jv}\rangle}.\end{aligned}$$

Since we suppose that $\sup_{L_x,L_y} ||\mathrm{vec}(L_x L_y^\top)|| \leqslant C$, we have

$$
\begin{aligned}
&2\mathbb{E}_{\{y_{ij}\},\{\sigma_{ij}\}|\{x_i\}} \sup_{L_x,L_y} \left| \frac{1}{n^x} \sum_{i=1}^{n^x} \frac{1}{n_i^y} \sum_{j=1}^{n_i^y} \sigma_{ij} f(x_i, y_{ij}) \right| \\
&\leqslant 2C \sqrt{\frac{1}{(n^x)^2} \sum_{i=1}^{n^x} \sum_{j=1}^{n^x} \frac{1}{n_i^y n_j^y} \sum_{u=1}^{n_i^y} \sum_{v=1}^{n_j^y} \mathbb{E}_{\{y_{ij}\},\{\sigma_{ij}\}|\{x_i\}} \left(\sigma_{iu}\sigma_{jv} r(x_i, y_{iu}) r(x_j, y_{jv}) \langle x_i, x_j \rangle \langle y_{iu}, y_{jv} \rangle \right)} \\
&= 2C \sqrt{\frac{1}{(n^x)^2} \sum_{i=1}^{n^x} \frac{1}{(n_i^y)^2} \sum_{j=1}^{n_i^y} \mathbb{E}_{\{y_{ij}\}|\{x_i\}} \left(r^2(x_i, y_{ij}) \langle x_i, x_i \rangle \langle y_{ij}, y_{ij} \rangle \right)} \\
&\leqslant 2C \sqrt{\frac{1}{(n^x)^2} \sum_{i=1}^{n^x} \frac{1}{(n_i^y)^2} n_i^y R^2} \\
&\leqslant 2CR \frac{1}{\sqrt{n^x n^y}}.
\end{aligned}
$$

We obtain the conclusion of the lemma. ■

With Lemma 1 and Lemma 2, we can prove Theorem 1:

Proof By the conclusions of Lemma 1 and Lemma 2, we have

$$
\begin{aligned}
P\left(D_1(\mathcal{S}) - \frac{2CR}{\sqrt{n^x n^y}} \geqslant \varepsilon \right) &= \mathbb{E}_{\{x_i\}} P\left(D_1(\mathcal{S}) - \frac{2CR}{\sqrt{n^x n^y}} \geqslant \varepsilon | \{x_i\} \right) \\
&\leqslant \mathbb{E}_{\{x_i\}} P\left(D_1(\mathcal{S}) - \mathbb{E}_{\{y_{ij}\}|\{x_i\}} D_1(\mathcal{S}) \geqslant \varepsilon | \{x_i\} \right) \\
&\leqslant \exp\left(-\frac{\varepsilon^2 n^x n^y}{2R^2 B^2} \right).
\end{aligned}
$$

Given a small number $\delta > 0$, by letting $\exp\left(-\frac{\varepsilon^2 n^x n^y}{2R^2B^2}\right) = \delta$, we have

$$
\begin{aligned}
-\frac{\varepsilon^2 n^x n^y}{2R^2 B^2} &= \log \delta \\
\varepsilon^2 &= \frac{2R^2 B^2 \log \frac{1}{\delta}}{n^x n^y} \\
\varepsilon &= \frac{RB\sqrt{2\log \frac{1}{\delta}}}{\sqrt{n^x n^y}}.
\end{aligned}
$$

Thus, with probability at least $1 - \delta$,

$$
D_1(\mathcal{S}) \leqslant \frac{2CR}{\sqrt{n^x n^y}} + \frac{RB\sqrt{2\log \frac{1}{\delta}}}{\sqrt{n^x n^y}}
$$

holds true. ■

B.2 Proof of Theorem 2

Theorem 2 *Given an arbitrary small positive number δ, with probability at least $1-\delta$, the following inequality holds:*

$$D_2(\{x_i\}_{i=1}^{n^x}) \leqslant \frac{2CR}{\sqrt{n^x}} + \frac{RB\sqrt{2\log\frac{1}{\delta}}}{\sqrt{n^x}}.$$

To prove Theorem 2, we also need two lemmas:

Lemma 3 *Given $\varepsilon > 0$, the following inequality holds:*

$$P\left(D_2(\{x_i\}_{i=1}^{n^x}) - \mathbb{E}_{\{x_i\}} D_2(\{x_i\}_{i=1}^{n^x}) \geqslant \varepsilon\right) \leqslant \exp\left(-\frac{\varepsilon^2 n^x}{2R^2B^2}\right).$$

Proof Similar to the proof of Lemma 1, since $\sup_{x,y,L_x,L_y} ||L_x^\top x||||L_y^\top y|| \leqslant B$ and $r \leqslant R$, $\forall\ j$, we have

$$|D_2(\{x_i\}_{i=1}^{n^x}) - D_2\left((\{x_i\}_{i=1}^{n^x} - x_j)\bigcup x'_j\right)| \leqslant \frac{2RB}{n^x}.$$

Since $\{x_i\}_{i=1}^{n^x}$ are i.i.d. random variables, by McDiarmid inequality (Bartlett and Mendelson, 2002), we know

$$\begin{aligned} P\left(D_2(\{x_i\}_{i=1}^{n^x}) - \mathbb{E}_{\{x_i\}} D_2(\{x_i\}_{i=1}^{n^x}) \geqslant \varepsilon\right) &\leqslant exp\left(-\frac{2\varepsilon^2}{\sum_{i=1}^{n^x}\frac{4R^2B^2}{(n^x)^2}}\right) \\ &= exp\left(-\frac{\varepsilon^2 n^x}{2R^2B^2}\right). \end{aligned}$$

■

Lemma 4

$$\mathbb{E}_{\{x_i\}} D_2(\{x_i\}_{i=1}^{n^x}) \leqslant \frac{2CR}{\sqrt{n^x}}.$$

Proof Similar to the proof of Lemma 2,

$$\begin{aligned} \mathbb{E}_{\{x_i\}} D_2(\{x_i\}_{i=1}^{n^x}) &= \mathbb{E}_{\{x_i\}} \sup_{L_x,L_y} |\frac{1}{n^x}\sum_{i=1}^{n^x} \mathbb{E}_{y|\{x_i\}} r(x_i,y) x_i^\top L_x L_y^\top y - \mathbb{E}_{x,y} r(x,y) x^\top L_x L_y^\top y| \\ &= \mathbb{E}_{\{x_i\}} \sup_{L_x,L_y} |\frac{1}{n^x}\sum_{i=1}^{n^x} \mathbb{E}_{y|\{x_i\}} r(x_i,y) x_i^\top L_x L_y^\top y - \frac{1}{n^x}\sum_{i=1}^{n^x} \mathbb{E}_{\{x'_i\}}\mathbb{E}_{y|\{x'_i\}} r(x'_i,y) x_i'^\top L_x L_y^\top y| \\ &\leqslant \mathbb{E}_{\{x_i\},\{x'_i\}} \sup_{L_x,L_y} \frac{1}{n^x}\sum_{i=1}^{n^x} |\mathbb{E}_{y|\{x_i\}} r(x_i,y) x_i^\top L_x L_y^\top y - \mathbb{E}_{y|\{x'_i\}} r(x'_i,y) x_i'^\top L_x L_y^\top y| \\ &= \mathbb{E}_{\{x_i\},\{x'_i\},\{\sigma_i\}} \sup_{L_x,L_y} \frac{1}{n^x}\sum_{i=1}^{n^x} \sigma_i \left(\mathbb{E}_{y|\{x^i\}} r(x_i,y) x_i^\top L_x L_y^\top y - \mathbb{E}_{y|\{x'_i\}} r(x'_i,y) x_i'^\top L_x L_y^\top y\right), \end{aligned}$$

where $\{x_i'\}$ are i.i.d. random variables with $\{x_i\}$. $\{\sigma_i\}$ are i.i.d. random variables with $P(\sigma_i = 1) = P(\sigma_i = -1) = 0.5$.

$$\begin{aligned}
&\mathbb{E}_{\{x_i\},\{x_i'\},\{\sigma_i\}} \sup_{L_x,L_y} \frac{1}{n^x} \sum_{i=1}^{n^x} \sigma_i \left(\mathbb{E}_{y|\{x_i\}} r(x_i,y) x_i^\top L_x L_y^\top y - \mathbb{E}_{y|\{x_i'\}} r(x_i',y) x_i'^\top L_x L_y^\top y \right) \\
&\leqslant 2\mathbb{E}_{\{x_i\},\{\sigma_i\}} \sup_{L_x,L_y} |\frac{1}{n^x} \sum_{i=1}^{n^x} \sigma_i \mathbb{E}_{y|\{x_i\}} r(x_i,y) x_i^\top L_x L_y^\top y| \\
&= 2\mathbb{E}_{\{x_i\},\{\sigma_i\}} \sup_{L_x,L_y} |\mathbb{E}_{y|\{x_i\}} \langle \mathrm{vec}(L_x L_y^\top), \frac{1}{n^x} \sum_{i=1}^{n^x} \sigma_i r(x_i,y) \mathrm{vec}(y \otimes x_i) \rangle| \\
&\leqslant 2\mathbb{E}_{\{x_i\},\{\sigma_i\}} \mathbb{E}_{y|\{x_i\}} \sup_{L_x,L_y} ||\mathrm{vec}(L_x L_y^\top)|| \sqrt{\frac{1}{(n^x)^2} \sum_{i=1}^{n^x} \sum_{j=1}^{n^x} \sigma_i \sigma_j r(x_i,y) r(x_j,y) \langle x_i, x_j \rangle \langle y, y \rangle}.
\end{aligned}$$

Since $\sup_{L_x,L_y} ||\mathrm{vec}(L_x L_y^\top)|| \leqslant C$,

$$\begin{aligned}
&2\mathbb{E}_{\{x_i\},\{\sigma_i\}} \mathbb{E}_{y|\{x_i\}} \sup_{L_x,L_y} ||\mathrm{vec}(L_x L_y^\top)|| \sqrt{\frac{1}{(n^x)^2} \sum_{i=1}^{n^x} \sum_{j=1}^{n^x} \sigma_i \sigma_j r(x_i,y) r(x_j,y) \langle x_i, x_j \rangle \langle y, y \rangle} \\
&\leqslant 2C \sqrt{\frac{1}{(n^x)^2} \sum_{i=1}^{n^x} \sum_{j=1}^{n^x} \mathbb{E}_{\{x_i\},\{\sigma_i\},y} \sigma_i \sigma_j r(x_i,y) r(x_j,y) \langle x_i, x_j \rangle \langle y, y \rangle} \\
&\leqslant 2C \sqrt{\frac{1}{(n^x)^2} \sum_{i=1}^{n^x} \mathbb{E}_{\{x_i\},y} r^2(x_i,y) \langle x_i, x_i \rangle \langle y, y \rangle} \\
&\leqslant \frac{2CR}{\sqrt{n^x}}.
\end{aligned}$$

We obtain the conclusion of the lemma. ■

With Lemma 3 and Lemma 4, we can prove Theorem 2:

Proof Combining the conclusions of Lemma 3 and Lemma 4, we have

$$\begin{aligned}
P\left(D_2(\{x_i\}_{i=1}^{n^x}) - \frac{2CR}{\sqrt{n^x}} \geqslant \varepsilon \right) &\leqslant P\left(D_2(\{x_i\}_{i=1}^{n^x}) - \mathbb{E}_{\{x_i\}} D_2(\{x_i\}_{i=1}^{n^x}) \geqslant \varepsilon \right) \\
&\leqslant \exp\left(-\frac{\varepsilon^2 n^x}{2R^2 B^2} \right).
\end{aligned}$$

Given a small number $\delta > 0$, by letting $\exp\left(-\frac{\varepsilon^2 n^x}{2R^2B^2}\right) = \delta$, we have

$$-\frac{\varepsilon^2 n^x}{2R^2B^2} = \log\delta$$
$$\varepsilon^2 = \frac{2R^2B^2\log\frac{1}{\delta}}{n^x}$$
$$\varepsilon = \frac{RB\sqrt{2\log\frac{1}{\delta}}}{\sqrt{n^x}}.$$

Thus, with probability at least $1-\delta$,

$$D_2(\{x_i\}_{i=1}^{n^x}) \leqslant \frac{2CR}{\sqrt{n^x}} + \frac{RB\sqrt{2\log\frac{1}{\delta}}}{\sqrt{n^x}}$$

holds true. ■

B.3 Proof of Theorem 4

Theorem 4 *Suppose that* $\mathcal{H}_x = \{L_x \mid L_x^\top L_x = \mathbb{I}_{d\times d}\}$ *and* $\mathcal{H}_y = \{L_y \mid L_y^\top L_y = \mathbb{I}_{d\times d}\}$, *then* $B = 1$ *and* $C = \sqrt{d}$. *Thus, the generalization bound for PLS is given by*

$$D(\mathcal{S}) \leqslant (2\sqrt{d}R + R\sqrt{2\log\frac{1}{\delta}})(\frac{1}{\sqrt{n^x n^y}} + \frac{1}{\sqrt{n^x}}).$$

Proof First, we analyze B. Remember that B is defined by $\sup_{x,y,L_x,L_y} ||L_x^\top x||||L_y^\top y|| \leqslant B$. Suppose that $L_x = \left[l_x^1, l_x^2, \ldots, l_x^d\right]$ and $L_y = \left[l_y^1, l_y^2, \ldots, l_y^d\right]$, where $\{l_x^k\}_{k=1}^d$ and $\{l_y^k\}_{k=1}^d$ represent the columns of L_x and L_y respectively. Note that

$$||L_x^\top x||^2 = \sum_{k=1}^d \left(x^\top l_x^k\right)^2, ||L_y^\top y||^2 = \sum_{k=1}^d \left(y^\top l_y^k\right)^2.$$

Since $L_x^\top L_x = \mathbb{I}_{d\times d}$ and $L_y^\top L_y = \mathbb{I}_{d\times d}$, we have

$$||L_x^\top x||^2 \leqslant ||x||^2, ||L_y^\top y||^2 \leqslant ||y||^2.$$

Since $||x|| \leqslant 1$ and $||y|| \leqslant 1$, we have

$$\sup_{x,y,L_x,L_y} ||L_x^\top x||||L_y^\top y|| \leqslant 1.$$

Thus, we can choose $B = 1$. Next, we analyze C. C is defined by $\sup_{L_x,L_y} ||\mathrm{vec}(L_x L_y^\top)|| \leqslant C$. It is easy to see that

$$||\mathrm{vec}(L_x L_y^\top)||^2 = \mathrm{trace}(L_y L_x^\top L_x L_y^\top) = \mathrm{trace}(\mathbb{I}_{d\times d}) = d.$$

Thus,

$$\sup_{L_x,L_y} ||\mathrm{vec}(L_x L_y^\top)|| = \sqrt{d}.$$

We can choose C as $\sqrt{d}$. ■

B.4 Proof of Theorem 5

Theorem 5 *Suppose that* $\mathcal{H}_x = \{L_x \mid |l_{xu}| \leqslant \lambda_x, ||l_{xu}|| \leqslant \theta_x, 1 \leqslant u \leqslant d_x\}$ *and* $\mathcal{H}_y = \{L_y \mid |l_{yv}| \leqslant \lambda_y, ||l_{yv}|| \leqslant \theta_y, 1 \leqslant v \leqslant d_y\}$. *If we suppose that the numbers of nonzero elements in x and y are respectively bounded by* m_x *and* m_y, *then* $B = \sqrt{m_x m_y}\min(d\lambda_x\lambda_y, \theta_x\theta_y)$ *and* $C = \sqrt{d_x d_y}\min(\lambda_x\lambda_y, \theta_x\theta_y)$. *Thus, the generalization bound for RMLS is given by*

$$D(\mathcal{S}) \leqslant \left(2\sqrt{d_x d_y}\min(\lambda_x\lambda_y, \theta_x\theta_y)R + \sqrt{m_x m_y}\min(d\lambda_x\lambda_y, \theta_x\theta_y)R\sqrt{2\log\frac{1}{\delta}}\right)\left(\frac{1}{\sqrt{n^x n^y}} + \frac{1}{\sqrt{n^x}}\right).$$

Proof Remember that B is defined by $\sup_{x,y,L_x,L_y} ||L_x^\top x||||L_y^\top y|| \leqslant B$. Since

$$||L_x^\top x||^2 = ||\sum_{u=1}^{d_x} x^{(u)} l_{xu}||^2, ||L_y^\top y||^2 = ||\sum_{v=1}^{d_y} y^{(v)} l_{yv}||^2,$$

where $x = \left[x^{(1)}, x^{(2)}, \ldots, x^{(d_x)}\right]^\top$ and $y = \left[y^{(1)}, y^{(2)}, \ldots, y^{(d_y)}\right]^\top$. $\{l_{xu}\}_{u=1}^{d_x}$ and $\{l_{yv}\}_{v=1}^{d_y}$ represent the rows of L_x and L_y respectively. Suppose that $\forall u,\ l_{xu} = \left[l_{xu}^{(1)}, l_{xu}^{(2)}, \ldots, l_{xu}^{(d)}\right]^\top$ and $\forall v,\ l_{yv} = \left[l_{yv}^{(1)}, l_{yv}^{(2)}, \ldots, l_{yv}^{(d)}\right]^\top$. Since $||x|| \leqslant 1$, $||l_{xu}||^2 \leqslant \theta_x^2$, and $\#\{x^{(u)} \mid x^{(u)} \neq 0\} \leqslant m_x$, we have

$$\begin{aligned}
||L_x^\top x||^2 &= ||\sum_{u=1}^{d_x} x^{(u)} l_{xu}||^2 \\
&= \sum_{k=1}^{d}\left(\sum_{u=1}^{d_x} x^{(u)}\mathbb{1}[x^{(u)} \neq 0] l_{xu}^{(k)}\right)^2 \\
&\leqslant \sum_{k=1}^{d}\left(\sum_{u=1}^{d_x}(x^{(u)})^2\right)\left(\sum_{u=1}^{d_x}\mathbb{1}[x^{(u)} \neq 0](l_{xu}^{(k)})^2\right) \\
&= \left(\sum_{u=1}^{d_x}(x^{(u)})^2\right)\left(\sum_{k=1}^{d}\sum_{u=1}^{d_x}\mathbb{1}[x^{(u)} \neq 0](l_{xu}^{(k)})^2\right) \\
&= ||x||^2\left(\sum_{u=1}^{d_x}\mathbb{1}[x^{(u)} \neq 0]||l_{xu}||^2\right) \\
&\leqslant m_x\theta_x^2.
\end{aligned}$$

Similarly, since $||y|| \leqslant 1$, $||l_{yv}||^2 \leqslant \theta_y^2$, and $\#\{y^{(v)} \mid y^{(v)} \neq 0\} \leqslant m_y$ we have $||L_y^\top y||^2 \leqslant m_y\theta_y^2$. Thus $\sup_{x,y,L_x,L_y} ||L_x^\top x||||L_y^\top y|| \leqslant \sqrt{m_x m_y}\theta_x\theta_y$. On the other hand, it is easy to see that

$$\begin{aligned}
||L_x^\top x||^2 &= ||\sum_{u=1}^{d_x} x^{(u)} l_{xu}||^2 \\
&= \sum_{k=1}^{d} \left(\sum_{u=1}^{d_x} x^{(u)} \mathbb{1}[x^{(u)} \neq 0] l_{xu}^{(k)} \right)^2 \\
&\leqslant \sum_{k=1}^{d} \left(\sum_{u=1}^{d_x} |x^{(u)} \mathbb{1}[x^{(u)} \neq 0] l_{xu}^{(k)}| \right)^2 \\
&\leqslant \sum_{k=1}^{d} \left(\max_{1\leqslant u\leqslant d_x} (|l_{xu}^{(k)}|) \sum_{u=1}^{d_x} \mathbb{1}[x^{(u)} \neq 0] |x^{(u)}| \right)^2 .
\end{aligned}$$

Since $|l_{xu}| = \sum_{k=1}^{d} |l_{xu}^{(k)}| \leqslant \lambda_x$, $\forall\ 1 \leqslant u \leqslant d_x$, thus $\max_{1\leqslant k\leqslant d} \max_{1\leqslant u\leqslant d_x} (|l_{xu}^{(k)}|) \leqslant \lambda_x$. Note that $||x|| \leqslant 1$ and $\#\{x^{(u)} \mid x^{(u)} \neq 0\} \leqslant m_x$, then we have

$$\begin{aligned}
\sum_{k=1}^{d} \left(\max_{1\leqslant u\leqslant d_x} (|l_{xu}^{(k)}|) \sum_{u=1}^{d_x} \mathbb{1}[x^{(u)} \neq 0] |x^{(u)}| \right)^2 &\leqslant \lambda_x^2 \sum_{k=1}^{d} \left(\sum_{u=1}^{d_x} \mathbb{1}[x^{(u)} \neq 0] |x^{(u)}| \right)^2 \\
&\leqslant \lambda_x^2 \sum_{k=1}^{d} \left(\sum_{u=1}^{d_x} (\mathbb{1}[x^{(u)} \neq 0])^2 \right) \left(\sum_{u=1}^{d_x} (x^{(u)})^2 \right) \\
&\leqslant d\lambda_x^2 m_x .
\end{aligned}$$

Similarly, since $||y|| \leqslant 1$, $|l_{yv}| \leqslant \lambda_y$, $\forall\ 1 \leqslant v \leqslant d_y$, and $\#\{y^{(v)} \mid y^{(v)} \neq 0\} \leqslant m_y$, we have $||L_y^\top y||^2 \leqslant d\lambda_y^2 m_y$. Thus, $\sup_{x,y,L_x,L_y} ||L_x^\top x||||L_y^\top y|| \leqslant \sqrt{m_x m_y} d\lambda_x\lambda_y$.

Therefore, we know $\sup_{x,y,L_x,L_y} ||L_x^\top x||||L_y^\top y|| \leqslant \sqrt{m_x m_y}\min(d\lambda_x\lambda_y, \theta_x\theta_y)$, and we can choose B as $\sqrt{m_x m_y}\min(d\lambda_x\lambda_y, \theta_x\theta_y)$.

Next, we analyze C. C is defined by $\sup_{L_x,L_y} ||\mathrm{vec}(L_x L_y^\top)|| \leqslant C$. Since $||l_{xu}|| \leqslant \theta_x$ and $||l_{yv}|| \leqslant \theta_y$, $\forall\ 1 \leqslant u \leqslant d_x$ and $1 \leqslant v \leqslant d_y$, we have

$$\begin{aligned}
||\mathrm{vec}(L_x L_y^\top)||^2 &= \mathrm{trace}(L_y L_x^\top L_x L_y^\top) \\
&= \sum_{u=1}^{d_x}\sum_{v=1}^{d_y} \left(\sum_{k=1}^{d} l_{xu}^{(k)} l_{yv}^{(k)} \right)^2 \\
&\leqslant \sum_{u=1}^{d_x}\sum_{v=1}^{d_y} \left(\sum_{k=1}^{d} \left(l_{xu}^{(k)} \right)^2 \right) \left(\sum_{k=1}^{d} \left(l_{yv}^{(k)} \right)^2 \right) \\
&= \sum_{u=1}^{d_x}\sum_{v=1}^{d_y} ||l_{xu}||^2 ||l_{yv}||^2 \\
&\leqslant d_x d_y \theta_x^2 \theta_y^2 .
\end{aligned}$$

On the other hand, since $|l_{xu}| \leqslant \lambda_x$ and $|l_{yv}| \leqslant \lambda_y$, $\forall\ 1 \leqslant u \leqslant d_x$ and $1 \leqslant v \leqslant d_y$, we have

$$\begin{aligned}
||\mathrm{vec}(L_x L_y^\top)||^2 &= \mathrm{trace}(L_y L_x^\top L_x L_y^\top) \\
&= \sum_{u=1}^{d_x}\sum_{v=1}^{d_y}\left(\sum_{k=1}^{d} l_{xu}^{(k)} l_{yv}^{(k)}\right)^2 \\
&\leqslant \sum_{u=1}^{d_x}\sum_{v=1}^{d_y}\left(\sum_{k=1}^{d} |l_{xu}^{(k)} l_{yv}^{(k)}|\right)^2 \\
&\leqslant \sum_{u=1}^{d_x}\sum_{v=1}^{d_y}\left(\max_{1\leqslant k\leqslant d}(|l_{xu}^{(k)}|)\sum_{k=1}^{d}|l_{yv}^{(k)}|\right)^2 \\
&\leqslant \sum_{u=1}^{d_x}\sum_{v=1}^{d_y}\lambda_x^2\left(\sum_{k=1}^{d}|l_{yv}^{(k)}|\right)^2 \\
&\leqslant d_x d_y \lambda_x^2 \lambda_y^2.
\end{aligned}$$

Therefore, we have $\sup_{L_x, L_y} ||\mathrm{vec}(L_x L_y^\top)|| \leqslant \sqrt{d_x d_y}\min(\lambda_x\lambda_y, \theta_x\theta_y)$, and we can choose C as $\sqrt{d_x d_y}\min(\lambda_x\lambda_y, \theta_x\theta_y)$. ■

References

J. Abernethy, F. Bach, T. Evgeniou, and J.P. Vert. A new approach to collaborative filtering: Operator estimation with spectral regularization. *JMLR '09*, 10:803–826, 2009.

S. Agarwal and P. Niyogi. Stability and generalization of bipartite ranking algorithms. In *COLT'05*, pages 32–47, 2005.

F. Bach. Exploring large feature spaces with hierarchical multiple kernel learning. In *NIPS'08*, pages 105–112, 2008.

F. Bach, G. Lanckriet, and M. Jordan. Multiple kernel learning, conic duality, and the SMO algorithm. In *ICML'04*, 2004.

R. Baeza-Yates and A. Tiberi. Extracting semantic relations from query logs. In *SIGKDD'07*, pages 76–85, 2007.

B. Bai, J. Weston, D. Grangier, R. Collobert, K. Sadamasa, Y. Qi, O. Chapelle, and K. Weinberger. Supervised semantic indexing. In *CIKM'09*, pages 187–196, 2009.

P. L. Bartlett and S. Mendelson. Rademacher and gaussian complexities: Risk bounds and structural results. *JMLR'02*, 3:463–482, 2002.

C. Burges, R. Ragno, and Q. Le. Learning to rank with nonsmooth cost functions. In *NIPS'06*, pages 395–402. 2006.

Y. Cao, J. Xu, T.Y. Liu, H. Li, Y. Huang, and H.W. Hon. Adapting ranking svm to document retrieval. In *SIGIR '06*, pages 186–193, 2006.

Z. Cao, T. Qin, T.Y. Liu, M.F. Tsai, and H. Li. Learning to rank: from pairwise approach to listwise approach. In *ICML '07*, pages 129–136, 2007.

T.Q. Chen, W.N. Zhang, Q.X. Lu, K.L. Chen, Z. Zhao, and Y. Yu. Svdfeature: A toolkit for feature-based collaborative filtering. *JMLR'12*, 13, 2012.

W. Chen, T.Y. Liu, and Z. Ma. Two-layer generalization analysis for ranking using rademacher average. *NIPS'10*, 23:370–378, 2010.

C. Cortes. Invited talk: Can learning kernels help performance? In *ICML'09*, page 161, 2009.

K. Crammer and Y. Singer. Pranking with ranking. In *NIPS'01*, pages 641–647, 2001.

N. Craswell and M. Szummer. Random walks on the click graph. In *SIGIR'07*, pages 239–246, 2007.

N. Cristianini, J. Shawe-taylor, A. Elisseeff, and J. Kandola. On kernel-target alignment. In *NIPS'01*, 2001.

J.V. Davis, B. Kulis, P. Jain, S. Sra, and I.S. Dhillon. Information-theoretic metric learning. In *ICML'07*, page 216, 2007.

S. Deerwester, S. Dumais, G. Furnas, T. Landauer, and R. Harshman. Indexing by latent semantic analysis. *JASIS'90*, 41:391–407, 1990.

J.H. Friedman. Greedy function approximation: a gradient boosting machine. *Ann. Statist*, 29(5): 1189–1232, 2001.

J. Gao, K. Toutanova, and W. Yih. Clickthrough-based latent semantic models for web search. In *SIGIR'11*, pages 675–684, 2011.

D. Grangier and S. Bengio. A discriminative kernel-based model to rank images from text queries. *IEEE Transactions on PAMI*, 30(8):1371–1384, 2008.

D.R. Hardoon, S. Szedmak, and J. Shawe-Taylor. Canonical correlation analysis: An overview with application to learning methods. *Neural Computation*, 16(12):2639–2664, 2004.

R. Herbrich, T. Graepel, and K. Obermayer. Large margin rank boundaries for ordinal regression. *NIPS'99*, pages 115–132, 1999.

T. Hertz, A. Bar-Hillel, and D. Weinshall. Boosting margin based distance functions for clustering. In *ICML'04*, page 50, 2004.

T. Hofmann. Latent semantic models for collaborative filtering. *ACM Trans. Inf. Syst.*, 22:89–115, 2004.

S. CH. Hoi, W. Liu, M.R. Lyu, and W.Y. Ma. Learning distance metrics with contextual constraints for image retrieval. In *CVPR'06*, volume 2, pages 2072–2078, 2006.

S. JacobGoldberger and R. GeoffHinton. Neighbourhood components analysis. *NIPS'04*, 2004.

K. Jarvelin and J. Kekalainen. Ir evaluation methods for retrieving highly relevant documents. In *SIGIR'00*, pages 41–48, 2000.

T. Joachims. Optimizing search engines using clickthrough data. In *KDD '02*, pages 133–142, 2002.

I.T. Jolliffe. *Principal Component Analysis*. Springer verlag, 2002.

G. R. G. Lanckriet, N. Cristianini, P. Bartlett, L. E. Ghaoui, and M. I. Jordan. Learning the kernel matrix with semi-definite programming. In *ICML'02*, pages 323–330, 2002.

H. Li. Learning to rank for information retrieval and natural language processing. *Synthesis Lectures on Human Language Technologies*, 4(1):1–113, 2011.

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

H. Ma, H.X. Yang, I. King, and M. R. Lyu. Learning latent semantic relations from clickthrough data for query suggestion. In *CIKM'08*, pages 709–718, 2008.

C. Micchelli and M. Pontil. Learning the kernel function via regularization. *JMLR'05*, 6:1099–1125, 2005.

C.S. Ong, A.J. Smola, and R.C. Williamson. Learning the kernel with hyperkernels. *JMLR '05*, 6: 1043–1071, 2005.

J.M. Ponte and W.B. Croft. A language modeling approach to information retrieval. In *SIGIR'98*, pages 275–281, 1998.

S. E. Robertson, S. Walker, S. Jones, M. Hancock-Beaulieu, and M. Gatford. Okapi at trec-3. In *TREC*, 1994.

R. Rosipal and N. Krämer. Overview and recent advances in partial least squares. *Subspace, Latent Structure and Feature Selection*, pages 34–51, 2006.

C. Rudin, C. Cortes, M. Mohri, and R. E. Schapire. Margin-based ranking meets boosting in the middle. In *COLT'05*, pages 63–78, 2005.

G. Salton and M. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc., New York, NY, USA, 1986.

P.J. Schreier. A unifying discussion of correlation analysis for complex random vectors. *Signal Processing, IEEE Transactions on*, 56(4):1327–1336, 2008.

M. Schultz and T. Joachims. Learning a distance metric from relative comparisons. *NIPS'03*, 2003.

N. Srebro, J.D.M. Rennie, and T. Jaakkola. Maximum-margin matrix factorization. *NIPS'05*, pages 1329–1336, 2005.

M. Sugiyama. Dimensionality reduction of multimodal labeled data by local fisher discriminant analysis. *JMLR'07*, 8:1027–1061, 2007.

M. Varma and B. R. Babu. More generality in efficient multiple kernel learning. In *ICML'09*, page 134, 2009.

J.A. Wegelin. A survey of partial least squares (pls) methods, with emphasis on the two-block case. *Technical Report, No.371, Seattle: Department of Statistics, Univ. of Wash.*, 2000.

K.Q. Weinberger and L.K. Saul. Distance metric learning for large margin nearest neighbor classification. *JMLR'09*, 10:207–244, 2009.

W. Wu, J. Xu, H. Li, and O. Satoshi. Learning a robust relevance model for search using kernel methods. *JMLR'11*, 12:1429–1458, 2011.

E.P. Xing, A.Y. Ng, M.I. Jordan, and S. Russell. Distance metric learning with application to clustering with side-information. *NIPS'03*, pages 521–528, 2003.

J. Xu and H. Li. Adarank: a boosting algorithm for information retrieval. In *SIGIR' 07*, pages 391–398, 2007.

J. Xu, H. Li, and Z.L. Zhong. Relevance ranking using kernels. In *AIRS '10*, 2010.

L. Yang, R. Jin, R. Sukthankar, and Y. Liu. An efficient algorithm for local distance metric learning. In *AAAI'06*, page 543, 2006.

Y. Ying, K. Huang, and C. Campbell. Sparse Metric Learning via Smooth Optimization. *NIPS'09*, pages 521–528, 2009.

C.X. Zhai and J. Lafferty. A study of smoothing methods for language models applied to information retrieval. *ACM Trans. Inf. Syst.*, 22(2):179–214, 2004.