Regularized Mapping to Latent Structures and Its Application to Web Search

Wei Wu, Zhengdong Lu, Hang Li

May 21, 2012

Abstract

Projection to Latent Structures (PLS), also known as Partial Least Squares, is a method for matching objects from two heterogeneous domains. Although PLS is empirically verified effective for matching queries and documents, its scalability becomes a major hurdle for its application in real-world web search. In this paper, we study a general framework for matching heterogeneous objects, which renders a rich family of matching models when different regularization are enforced, with PLS as a special case. Particularly, with ℓ_1 and ℓ_2 type of regularization on the mapping functions, we obtain the model called *Regularized Mapping to Latent Structures* (RMLS). RMLS enjoys many advantages over PLS, including lower time complexity and easy parallelization. As another contribution, we give a generalization analysis of this matching framework, and apply it to both PLS and RMLS. In experiments, we compare the effectiveness and efficiency of RMLS and PLS on large scale web search problems. The results show that RMLS can achieve equally good performance as PLS for relevance ranking, while significantly speeding up the learning process.

1 Introduction

Many tasks in machine learning and data mining can be formalized as matching between objects from two spaces. For example, in web search, the retrieved documents are ordered according to their relevance to the given query, where 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 (query, document) pair. Similarly, matching between heterogeneous data sources can be found in collaborative filtering, image annotation, drug design, etc.

Canonical Correlation Analysis (CCA) [cf., 12] and Projection to Latent Structures (PLS), also known as Partial Least Squares [cf., 21] are well-known methods for matching objects from two different spaces using a shared latent space. More specifically, in CCA and PLS, the objects in the two spaces are mapped into the same latent space through two *projections*. The matching degree of these two objects is then measured as the *cosine* or the dot product of their images in the latent space. Among the two, PLS is usually preferred for large scale problems, mostly because CCA requires calculating the inverse of matrices and therefore is prohibitively expensive when the dimension of data is very large [12].

Previous work has shown the efficacy of PLS in web search [26] as a relevance model, for solving the term mismatch problem, one of the major challenges in search. For example, if the query is "NY" and the document only contains "New York", then the query and document will not match based on their terms. Conventional relevance models (matching models) such as Vector Space Model (VSM) [22], BM25 [20], and Language Models for Information Retrieval (LMIR) [19, 28] do not function in such cases. PLS provides a solution to deal with the problem, by mapping queries and documents into a latent space with a much lower dimensionality, and carrying out the matching in this space. It is quite natural to assume that queries and documents are heterogeneous, because they are very different in nature, e.g., queries are short while documents are much longer, among other things.

Despite the success of PLS in solving the term mismatch problem, two issues prevent us from applying this latent space idea to matching heterogenous data in real world:

- **Scalability** PLS requires Singular Value Decomposition (SVD), which has high time complexity [18] and hard to parallelize, and therefore does not scale up to massive data sets seen in web search;
- **Generalization Analysis** We do not theoretically understand how well this type of matching model behave on unseen data, due to the lack of generalization analysis on matching models. Moreover, in web search, the usual assumption that pairs of objects from the two spaces are i.i.d. no long holds [6], which calls for new insight into understanding the generalization ability.

This paper attempts to address both problems. More specifically, we first propose a general framework for learning to match objects from two heterogeneous spaces by mapping them into a latent space. The type of mappings can be further specified by a set of constraints. By limiting the mapping to be a projection[1], we recover the PLS. 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. To further understand this framework, we give a generalization analysis of it under a hierarchical sampling assumption which is natural in the matching problems met 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 objects.

Our contributions are three-folds: 1) proposal of a new method named RMLS for matching between heterogeneous data, which is scalable and efficient; 2) generalization analysis of framework for matching and application of it to RMLS and PLS; 3) empirical verification of the efficacy and the efficiency of RMLS on real-world large scale web search data.

2 A Framework For Matching Objects From Two Spaces

We take one step back from PLS and give a more general framework for learning to match objects from two heterogeneous spaces. Later in this section we will show that this framework subsumes PLS as its special cases, and relates to Latent Semantic Index (LSI) in information retrieval.

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 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 [6], 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 particular 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.

2.1 Model

We intend to find a 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 $\mathcal{L}$

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

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

Dot product is a popular form of matching in applications like search. In fact, traditional relevance models in search such as VSM, BM25, and LMIR are all dot products of a query vector and a document vector, as pointed out in [27]. 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

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

which is in the same spirit as the technique used in [9] for kernel learning[2]. This expectation defined above can be estimated from $\mathcal{S}$ as follows

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

The learning problem hence boils down to

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

where $\mathcal{H}_x$ and $\mathcal{H}_y$ are hypothesis spaces for L_x and L_y respectively.

2.2 Special Cases

The matching framework in (1) defines a rather rich family of matching models, with different choices of $\mathcal{H}_x$ and $\mathcal{H}_y$. Most importantly, if both L_x and L_y are confined to be matrices with orthonormal columns, or more formally $\mathcal{H}_x = \{L_x \mid L_x^\top L_x = \mathbb{I}_{d\times d}\}$ and $\mathcal{H}_y = \{L_y \mid L_y^\top L_y = \mathbb{I}_{d\times d}\}$ (where $\mathbb{I}_{d\times d}$ stands for the $d \times d$ identity matrix), the program in (1) becomes PLS [21, 23]. This can be easily seen if we re-write (1) as

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

where $y_i' = \frac{1}{n_i^y} \sum_{j=1}^{n_i^y} r_{ij} y_{ij}$. The program is exactly the formulation of PLS as formulated in [23][3].

Also interestingly, our framework in (1) also subsumes the Latent Semantic Index (LSI) [18] 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 the queries and documents, i.e., there is only non-zero element *one* in x and y at the location indexing the corresponding query or document. The objective function in (1) becomes $\mathrm{trace}\left(L_y^\top (\sum_{i=1}^{n^x} \sum_{j=1}^{n_i^y} r_{ij} y_{ij} x_i^\top) L_x\right)$ 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, although theoretically sound and empirically effective [26], requires SVD of large matrices, 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.

[2]We do not take a cosine as in [9] since the scale problem will be automatically considered in the regularization of the mappings.

[3]We can assume that x and y' are centered.

3 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

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

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 supplementary material 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. Similar thing holds 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:

$$\begin{aligned}\underset{L_x, L_y}{\arg\max} \quad & \frac{1}{n^x} \sum_{i=1}^{n^x} \frac{1}{n_i^y} \sum_{j=1}^{n_i^y} r_{ij} x_i^\top L_x L_y^\top y_{ij} \\ s.t. \quad & |l_{xu}| \leqslant \lambda_x,\ \|l_{xu}\| \leqslant \theta_x, \quad 1 \leqslant u \leqslant d_x \\ & |l_{yv}| \leqslant \lambda_y,\ \|l_{yv}\| \leqslant \theta_y, \quad 1 \leqslant v \leqslant d_y.\end{aligned} \tag{2}$$

The matching model defined in (2) is called *Regularized Mapping to Latent Structures* (RMLS).

3.1 Optimization

In practice, we solve instead the following penalized variant of (2) for easier optimization

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

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 (3). Since the objective in (3) is not convex, there is no guarantee for convergence to a global minimum.

Specifically, for a fixed L_y, the objective function of problem (3) 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

$$l_{xu}^{(z)*} = C_u \cdot \left(\max(|\omega_u^{(z)}| - \beta, 0) \text{sign}(\omega_u^{(z)}) \right),\ 1 \leqslant z \leqslant d, \tag{4}$$

where $l_{xu}^{(z)}$ represents the z^{th} element of l_{xu}. $\text{sign}(\omega_u^{(z)})$ returns 1 if $\omega_u^{(z)} > 0$, returns 0 if $\omega_u^{(z)} = 0$, and returns -1 if $\omega_u^{(z)} < 0$. C_u is a constant that makes $\|l_{xu}^*\| = \theta_x$ if there are nonzero elements in l_{xu}^*, otherwise $C_u = 0$.

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 (4).
  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 (5).
  update L_y^{t+1}, $\ t \leftarrow t + 1$
Output: L_x^t and L_y^t.

Similarly, for a fixed L_x, the objective function of problem (3) 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

$$l_{yv}^{(z)*} = C_v \cdot \left(\max(|\eta_v^{(z)}| - \gamma, 0) sign(\eta_v^{(z)}) \right),\ 1 \leqslant z \leqslant d, \tag{5}$$

where $l_{yv}^{(z)}$ represents the z^{th} element of l_{yv}. C_v is a constant that makes $||l_{yv}^*|| = \theta_{yv}$ 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.

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 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.

3.2 Parallelization

The learning process of RMLS are 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 reason is that in the hypothesis spaces of RMLS, we adopt ℓ_1-norm and ℓ_2-norm based constraints on rows of the mapping matrices, which removes the dependency among different rows. That is in contrast to PLS, where the columns of mapping matrices are forced to be orthogonal to each other, which impedes the concurrent optimization.

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.

4 Generalization Analysis

We will first give a generalization bound for the matching framework in (1), 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 our supplementary material.

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

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

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

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

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 4.1. *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 the similar techniques in the analysis of the bound of $D_1(\mathcal{S})$, we can obtain the bound for $D_2(\{x_i\}_{i=1}^{n^x})$:

Theorem 4.2. *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 4.1 & 4.2, we are able to bound $D(\mathcal{S})$:

Theorem 4.3. *Given an arbitrary small positive number δ, with probability at least $1-2\delta$, the following inequality holds:*

$$D(\mathcal{S}) \leqslant (2CR + RB\sqrt{2\log\frac{1}{\delta}})(\frac{1}{\sqrt{n^x n^y}} + \frac{1}{\sqrt{n^x}}). \tag{6}$$

Equation (6) gives a general generalization bound for framework (1). 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 (6).

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

Theorem 4.4. *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*

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

Theorem 4.5. *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*

$$D(\mathcal{S}) \leqslant (\frac{1}{\sqrt{n^x n^y}} + \frac{1}{\sqrt{n^x}}) \times (2\sqrt{d_x d_y}\min(\lambda_x\lambda_y, \theta_x\theta_y)R + \sqrt{m_x m_y}\min(d\lambda_x\lambda_y, \theta_x\theta_y)R\sqrt{2\log\frac{1}{\delta}}). \tag{8}$$

Note that B and C given in Theorem 4.4 and Theorem 4.5 are tight bounds for $\sup_{x,y,L_x,L_y} ||L_x^\top x||||L_y^\top y||$ and $\sup_{L_x,L_y} ||\mathrm{vec}(L_x L_y^\top)||$. To see this, let us consider an extreme case: $L_x = \theta_x e_x l^\top$ and $L_y = \theta_y e_y l^\top$, where l is a d dimensional vector that satisfies $||l|| = 1$ and $|l| = 1$, e_x and e_y are respectively d_x and d_y dimensional vectors consisting of all ones. We suppose that $\theta_x \leqslant \lambda_x$. $x = \frac{1}{\sqrt{m_x}} g_x$ and $y = \frac{1}{\sqrt{m_y}} g_y$, where g_x and g_y are respectively d_x and d_y dimensional vectors consisting only m_x and m_y nonzero elements as ones. Then, we know $||L_x^\top x||^2 = ||\theta_x l e_x^\top g_x \frac{1}{\sqrt{m_x}}||^2 = m_x\theta_x^2$ and $||L_y^\top y||^2 = ||\theta_y l e_y^\top g_y \frac{1}{\sqrt{m_y}}||^2 = m_y\theta_y^2$. Thus, $\sup_{x,y,L_x,L_y} ||L_x^\top x|| \, ||L_y^\top y|| = \sqrt{m_x m_y}\theta_x\theta_y$. $||\mathrm{vec}(L_x L_y^\top)||^2 = \mathrm{trace}(\theta_x^2\theta_y^2 e_y l^\top l e_x^\top e_x l^\top l e_y^\top) = d_x d_y \theta_x^2\theta_y^2$, thus $\sup_{L_x,L_y} ||\mathrm{vec}(L_x L_y^\top)|| = \sqrt{d_x d_y}\theta_x\theta_y$. On the other hand, in PLS, since $L_x^\top L_x = \mathbb{I}_{d\times d}$ and $L_y^\top L_y = \mathbb{I}_{d\times d}$, $||\mathrm{vec}(L_x L_y^\top)|| = \sqrt{d}$, for all L_x, L_y (see our proof in supplementary material). Given an L_x and an L_y with sparse columns, if we let x be a column of L_x and y be a column of L_y, then $||L_x^\top x|| \, ||L_y^\top y|| = 1$ and $\sup_{x,y,L_x,L_y} ||L_x^\top x|| \, ||L_y^\top y|| = 1$.

5 Experiment

We apply RMLS to a relevance ranking task in web search. In this case, $\mathcal{X}$ and $\mathcal{Y}$ are respectively the query space and the document space. Given a query x and a document y, we treat user click number as the response r. The matching model learned by RMLS is used as a relevance model. We conduct experiments on a small data set and a large data set with millions of queries and documents.

5.1 Experiment Setup

We collected one week click-through data and half year click-through data in two different time period from a commercial web search engine. To filter out noise, we discarded the query-document pairs with click frequency ≤ 3. After that, 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. On average, each query has 1.74 clicked documents (i.e., $\tilde{n}^y$) and each document is clicked for 1.46 queries (i.e., $\tilde{n}^x$) in the one week data set. In the half year data set, each query has on average 2.92 clicked documents and each document on average is clicked for 4.04 queries.

We extract features from two sources. First, we took the words in queries and the words in URLs and titles of documents as features. After stemming and removing stop words, queries and documents are represented as tf-idf vectors [22] in a word space. There are respectively $101,904$ and $271,561$ unique words in one week data and half year data. Then we followed [3] and took the numbers of clicks of documents as features of queries, and the numbers of clicks of queries as features of documents. We concatenated the features from word and those from click to create a long vector. The resulted query space and document space are of high dimensions but very sparse. Table 1 gives the statistics on query and document features. Note that the notations in the table are the same as in Section 3.

	d_x	d_y	c_x	c_y	N_x	N_y
one week	$2.1 \cdot 10^5$	$2.0 \cdot 10^5$	4.0	5.9	1.7	3.4
half year	$4.9 \cdot 10^6$	$6.6 \cdot 10^6$	5.5	8.6	7.2	5.9

Table 1: Statistics on query and document features

For baseline methods, we chose BM25 [20] which is a classic relevance model in information retrieval. We also compared RMLS with PLS, LSI and random walk on click-through bipartite [8] ("RW" for short). We implemented two versions of LSI in this paper: 1) LSI [10] on a document-term matrix was implemented, denoted as LSI_{dt}, and 2) LSI on a query-document matrix with each element representing the click number, denoted as LSI_{qd}. In LSI_{qd} and random walk, the information from word space is not taken into account, to make a fair comparison, we also linearly combine them with BM25.

We obtained relevance data consisting of judged query-document pairs from the search engine in a different time period with the click-through data. There are five level 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.

In BM25, the default setting is used. In PLS, LSI_{dt}, and LSI_{qd}, the parameter is d, the dimensionality of latent space. We set d in the range of $\{100, 200, \ldots, 1000\}$ for all methods. In random walk, we followed the conclusion in [8] and fixed the self-transition probability as 0.9. We chose the number of transition steps from $\{1, \ldots, 10\}$. We also treated the combination weights in the linear combination models as parameters and tuned them within $\{0.1, 0.2, \ldots, 0.9\}$. In RMLS, we checked d in the same range as PLS, LSI_{dt}, and LSI_{qd}. We fixed θ_x and θ_y as 1, and tuned β and γ from $\{0.001, 0.005, 0.1, 0.5, 1, 5, 10\}$.

	MAP	NDCG@1	NDCG@3	NDCG@5
RMLS	**0.554**	**0.686**	**0.732**	0.729
PLS	**0.552**	0.676	**0.728**	**0.736**
RW	0.484	0.655	0.704	0.704
RW+BM25	0.497	0.671	0.718	0.716
LSI_{qd}	0.449	0.588	0.665	0.676
LSI_{qd}+BM25	0.481	0.649	0.705	0.706
LSI_{dt}	0.460	0.616	0.675	0.680
BM25	0.462	0.637	0.690	0.690

Table 2: Relevance ranking result on one week data

To evaluate the performances of different methods, we employ Mean Average Precision (MAP) [2] and Normalized Discounted Cumulative Gain (NDCG) [14] at positions of 1, 3, and 5 as evaluation measures.

5.2 Results on One Week Data

We conducted experiments on a workstation with 24 AMD Opteron 6172 processors and 96 GB RAM. After parameter tuning on the validation set, we chose 1000 as the value of d for PLS, LSI_{dt}, and LSI_{qd}, and 5 as the number of transition steps for random walk. We set combination models as $0.8\,\text{RW} + 0.2\,\text{BM25}$ and $0.8\,\text{LSI}_{\text{qd}} + 0.2\,\text{BM25}$. For RMLS, we chose 1000 as the value of d and 0.1 as the value of β and γ. 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 ($p < 0.01$ from sign test).

We also tested the efficiency of RMLS and compared it with PLS. In PLS, the linear mappings are learned through SVD. For a fair comparison, we tried to use the most efficient implementation of SVD. More specifically, we implemented the power method in [25] with C++, and further optimized the data structure for our tasks. This SVD implementation can handle large data set on which state-of-the-art SVD tools like SVDLIBC [4] fails. Since the efficiency of algorithms is influenced by implementation strategies, e.g. 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 took 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 complexity 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) still makes 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 1. Clearly the running time decreases with with the number of threads. The improvement becomes slow after 10 threads. With 20 threads, RMLS only takes 277 seconds to achieve a comparable performance with PLS.

5.3 Results on Half Year Data

We further tested the performance of RMLS on a half year data set with millions of queries and documents. On such a large scale, SVD-based methods and random walk become almost infeasible (e.g., taking months to run). We therefore only compared RMLS with BM25 and PLS with word features. With only word features ($d_x = d_y = 271,561$), PLS is slow but still feasible. We tuned parameters in the same way as in one week data.

[4] `http://tedlab.mit.edu/~dr/SVDLIBC/`

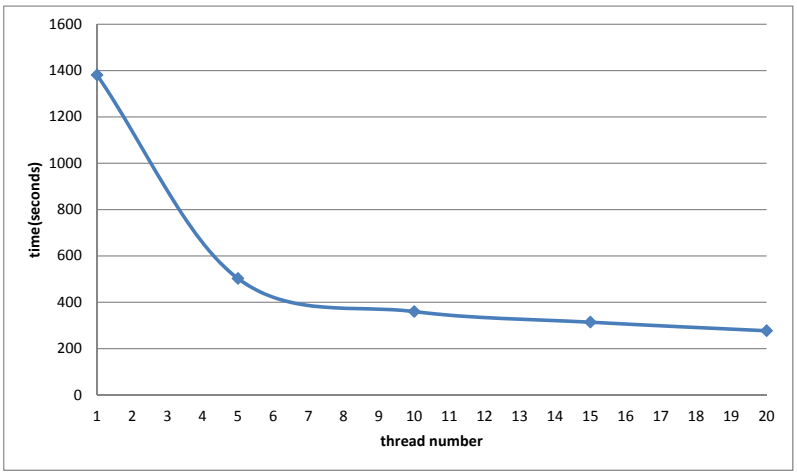

Figure 1: Time cost trend of RMLS under multiple processors.

	MAP	NDCG@1	NDCG@3	NDCG@5
RMLS	**0.578**	**0.717**	**0.743**	**0.753**
PLS (word)	0.474	0.638	0.666	0.677
BM25	0.441	0.643	0.663	0.670

Table 3: Relevance ranking result on half year data

From Table 3, RMLS can leverage high dimensional features to outperform all the baselines ($p < 0.01$ from sign test). We hypothesize that the performance of PLS with full features can achieve results comparable with RMLS, but PLS its high computation complexity prevented us from testing it. With RMLS, it took $20,523$ seconds to achieve the result using 20 threads. For PLS, although it only uses word features, it took $1,121,440$ seconds to finish learning. In other words, parallelized RMLS can be used to tackle web search problem of real-world scale.

6 Related Work

Matching heterogeneous data from two spaces is well studied in statistics. Projection to Latent Structures, a.k.a. Partial Least Squares (PLS) [cf., 21, 23], and Canonical Correlation Analysis (CCA) [cf., 12] are classic tools in statistics for studying the relation between two or more sets of data. The underlying idea in PLS and CCA is to model collinearity between different data sets. In PLS, the collinearity is modeled by covariance (i.e., dot product), while in CCA, it is modeled by correlation (i.e., *cosine*). Both PLS and CCA can be viewed as models for matching heterogeneous data via common latent structures. However they require solving an eigenvalue problem, and are prohibitively expensive for web scale applications. Our work provides a rather flexible framework, allowing rather scalable implementations.

Matching pairs of objects with a similarity function defined as dot product is not new. When the pair of objects are from the same space, the similarity function becomes positive semi-definite, and the matching problem is essentially finding a good kernel [cf., 9, 15, 7]. Recently, the learning of a similarity function for object pairs from two different spaces has also emerged as a hot research topic [11, 1]. Our model belongs to the latter category, but is tailored for web search and tries to solve problems central to that, e.g., scalability.

Our model, when applied to web search, is also obviously related to the effort on learning to rank [16, 17]. However, we focus on learning to match queries and documents, while learning to rank are 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.

In web search, Bai et al. [4, 5] recently propose learning a low rank model for ranking documents, which is also in a sense 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 learn a ranking model using hinge loss, while we employ a point-wise input and learn a matching model using alignment.

The matching problem is also widely studied in collaborative filtering (CF) whose goal can be viewed as matching users and items [13, 24, 1]. The characteristics of the problems CF attempts to solve, e.g. the sampling assumption and the nature of "ratings", are different from the matching problems in web search. We leave it as an open question on whether our work can be applied to CF problems.

7 Conclusion and Future Work

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. Moreover, we devised an algorithm called Regularized Mapping to Latent Structures as a special case, which can achieve comparable performance as PLS but much more scalable. For future work, we consider studying other objective functions such as square loss and hinge loss, as well as other forms of regularization on large scale problems.

References

[1] 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.

[2] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1999.

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

[4] 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.

[5] B. Bai, J. Weston, D. Grangier, R. Collobert, K. Sadamasa, Y. Qi, C. Cortes, and M. Mohri. Polynomial semantic indexing. *NIPS'09*, 22:64–72, 2009.

[6] W. Chen, T.Y. Liu, and Z. Ma. Two-layer generalization analysis for ranking using rademacher average. *NIPS*, 23:370–378, 2010.

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

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

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

[10] S. Deerwester, S. Dumais, G. Furnas, T. Landauer, and R. Harshman. Indexing by latent semantic analysis. *JASIS*, 41:391–407, 1990.

[11] 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.

[12] 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.

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

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

[15] 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.

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

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

[18] C.H. Papadimitriou, H. Tamaki, P. Raghavan, and S. Vempala. Latent semantic indexing: A probabilistic analysis. In *Proceedings of the seventeenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, pages 159–168. ACM, 1998.

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

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

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

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

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

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

[25] 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.

[26] W. Wu, H. Li, and J. Xu. Learning query and document similarities from click-through bipartite graph with metadata. *Microsoft Research Technical Report, MSR-TR-2011-126*, 2011.

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

[28] 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.