Incorporating Semantic Knowledge into Latent Matching Model in Search

Shuxin Wang[1](✉), Xin Jiang[2], Hang Li[2], Jun Xu[1], and Bin Wang[3]

[1] Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China
`wangshuxin@ict.ac.cn`
[2] Huawei Noah's Ark Lab, Hong Kong, China
[3] Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China

Abstract. The relevance between a query and a document in search can be represented as matching degree between the two objects. Latent space models have been proven to be effective for the task, which are often trained with click-through data. One technical challenge with the approach is that it is hard to train a model for tail queries and tail documents for which there are not enough clicks. In this paper, we propose to address the challenge by learning a latent matching model, using not only click-through data but also semantic knowledge. The semantic knowledge can be categories of queries and documents as well as synonyms of words, manually or automatically created. Specifically, we incorporate semantic knowledge into the objective function by including regularization terms. We develop two methods to solve the learning task on the basis of coordinate descent and gradient descent respectively, which can be employed in different settings. Experimental results on two datasets from an app search engine demonstrate that our model can make effective use of semantic knowledge, and thus can significantly enhance the accuracies of latent matching models, particularly for tail queries.

Keywords: Latent Matching Model · Semantic knowledge · Learning to match · Regularized mapping to latent structures

1 Introduction

In search, given a query documents are retrieved and ranked according to their relevance, which can be represented by the matching score between the query and each of the documents, referred to as semantic matching in [9]. Traditional IR models, including Vector Space Model (VSM), BM25, and Language Models for Information Retrieval (LMIR) can be viewed as matching models for search, created without using machine learning. The models work well to some extent, but they sometimes suffer from mismatch between queries and documents.

Recently significant effort has been made on automatic construction of matching models in search, using machine learning and click-through data. The learned models can effectively deal with mismatch and outperform traditional IR models [9]. Among the proposed approaches, learning a latent space model for

S. Ma et al. (Eds.): AIRS 2016, LNCS 9994, pp. 29–41, 2016.
DOI: 10.1007/978-3-319-48051-0_3

matching in search becomes the state-of-the-art. The class of semantic matching models, called latent matching models in this paper, map the queries and documents from their original spaces to a lower dimensional latent space, in which the matching scores are calculated as inner products of the mapped vectors.

Despite the empirical success of latent matching models, the problem of query document mismatch in search is still not completely solved. Specifically, it remains hard to effectively train a matching model which works well not only for frequent queries and documents, but also for rare queries and documents, because there is not sufficient click data for rare queries and documents. This in fact belongs to the long tail phenomenon, which also exists in many different tasks in web search and data mining. One way to conquer the challenge would be to incorporate additional semantic knowledge into the latent matching models. Specifically, semantic knowledge about synonyms and categories of queries and documents can make the latent space better represent similarity between queries and documents. Suppose that "Sacramento" and "capital of California" are synonyms and it would be difficult to observe their association directly from click information(e.g., a query and the title of clicked document), because both rarely occur in the data. If we can embed the knowledge into the learned latent space, then it will help to make judgment on the matching degrees between queries and documents containing the synonyms. The technical question which we want to address in this paper is how to incorporate semantic knowledge in the learning of latent space model in a theoretically sound and empirically effective way.

In this paper, as the first step of the work, we propose a novel method for learning a linear latent matching model for search, leveraging not only clickthrough data, but also semantics knowledge such as synonym dictionary and semantic categories. The semantic knowledge can either be automatically mined or manually created. Specifically, we reformulate the learning of latent space model by adding regularization, in which way semantic knowledge can be naturally embedded into the latent space and be utilized in matching. The learning problem becomes maximization of the matching degrees of relevant query document pairs as well as the agreement with the given semantic knowledge. Regularization is also imposed on the linear mapping matrices as well as their product in order to increase the generalization ability of the learned model.

Without loss of generality, we take Regularized Mapping in Latent Space (RMLS) [13], one of the state-of-the-art methods for query document matching, as the basic latent matching model and augment it with semantic knowledge. We improve the optimization procedure of RMLS by introducing a new regularization term. We further develop a coordinate descent algorithm and a gradient descent algorithm to solve the optimization problem. The algorithms can be employed in different settings and thus the learning can be generally carried out in an efficient and scalable way. We conduct experiments on two large-scale datasets from a mobile app search engine. The experimental results demonstrate that our model can make effective use of the semantic knowledge, and significantly outperform existing matching models.

2 Related Work

Matching between queries and documents is of central importance to search [9]. Traditional information retrieval models based on term matching may suffer from term mismatch.

Topic modeling techniques aim to discover the topics as well as the topic representations of documents in the document collection, and can be used to deal with query document mismatch. Latent semantic indexing (LSI) [3] is one typical non-probabilistic topic model. Regularized Latent Semantic Indexing (RLSI) [12] formalizes topic modeling as matrix factorization with regularization of ℓ_1/ℓ_2-norm on topic vectors and document representation vectors. Probabilistic Latent Semantic Indexing (PLSI) [5] and Latent Dirichlet Allocation (LDA) [2] are two widely used probabilistic topic models. By employing one of the topic models, one can project queries and documents into the topic space and calculate their similarities in the space. However, topic modeling does not directly learn the query document matching relation, and thus its ability of dealing with query document mismatch is limited.

In a latent matching model, queries and documents are deemed as objects in two different spaces and are mapped into the same latent space for matching degree calculation (e.g., inner product). The learning of the mapping functions is performed by using training data such as click-through log in a supervised fashion, and thus is more effective to deal with mismatch. Partial Least Square (PLS) [11] is a method developed in statistics and can be utilized to model the matching relations between queries and documents. PLS is formalized as learning of two linear projection functions represented by orthonormal matrices and can be solved by Singular Value Decomposition (SVD). Canonical Correspondence Analysis (CCA) [4] is an alternative method to PLS. The difference between CCA and PLS is that CCA takes cosine as the similarity measure and PLS takes inner product as the similarity measure. Bai et al. [1] propose Supervised Semantic Indexing(SSI), which makes use of a pairwise loss function and learns a low-rank model for matching and ranking. Wu et al. [13] propose a general framework for learning to match heterogeneous objects, and a matching model called Regularized Mapping to Latent Structures (RMLS) is specified. RMLS extends PLS by replacing its orthonormal constraints with ℓ_1 and ℓ_2 regularization. RMLS is superior to PLS in terms of computation efficiency and scalability.

Recently, non-linear matching models have also been studied. For example, Huang et al. [6] propose a model referred to as Deep Structured Semantic Model (DSSM), which performs semantic matching with deep learning techniques. Specifically, the model maps the input term vectors into output vectors of lower dimensions through a multi-layer neural network, and takes cosine similarities between the output vectors as the matching scores. Lu and Li [10] propose a deep architecture for matching short texts, which can also be queries and documents. Their method learns matching relations between words in the two short texts as a hierarchy of topics and takes the topic hierarchy as a deep neural network.

3 Incorporating Semantic Knowledge into Latent Matching Model

3.1 Latent Matching Model

Let $\mathcal{X} \subset \mathbb{R}^{d_x}$ and $\mathcal{Y} \subset \mathbb{R}^{d_y}$ denote the two spaces for matching, and $x \in \mathcal{X}$ and $y \in \mathcal{Y}$ denote the objects in the spaces. In search, x and y are a query vector and a document vector, respectively. Suppose that there is a latent space $\mathcal{L} \subset \mathbb{R}^d$. We intend to find two mapping functions that can map the objects in both $\mathcal{X}$ and $\mathcal{Y}$ into $\mathcal{L}$. When the two mapping functions are linear, they can be represented as matrices: $L_x \in \mathbb{R}^{d \times d_x}$ and $L_y \in \mathbb{R}^{d \times d_y}$. The degree of matching between objects x and y is then defined as inner product of $L_x x$ and $L_y y$, which is $x^T L_x^T L_y y$.

To learn the linear mappings, we need training data that indicates the matching relations between the objects from the two spaces. In search, click-through logs are often used as training data, because they provide information about matching between queries and documents. Following the framework by Wu et al. [13], given a training dataset of positive matching pairs $\{(x_i, y_i)\}_{i=1}^n$, the learning problem is formalized as

$$\arg\max_{L_x, L_y} \frac{1}{n} \sum_{i=1}^{n} {x_i}^T L_x^T L_y y_i, \text{subject to}\, L_x \in \mathcal{H}_x, L_y \in \mathcal{H}_y. \tag{1}$$

where $\mathcal{H}_x$ and $\mathcal{H}_y$ denote the hypothesis spaces for the linear mappings L_x and L_y, respectively. This framework subsumes Partial Least Square (PLS) and Regularized Mapping to Latent Structure (RMLS) as special cases. For PLS, the hypothesis spaces are confined to matrices with orthonormal rows. RMLS replaces the orthonormal assumption with sparsity constraints on L_x and L_y. More specifically, the hypothesis spaces in RMLS become:

$$\mathcal{H}_x = \{L_x \mid \|l_{xu}\|_p \leqslant \tau_{x,p}, p = 1, 2, u = 1, \ldots, d_x\},$$
$$\mathcal{H}_y = \{L_y \mid \|l_{yv}\|_p \leqslant \tau_{y,p}, p = 1, 2, v = 1, \ldots, d_y\},$$

where l_{xu} is the u-th column vector of L_x and l_{yv} is the v-th column vector of L_y. The column vectors are actually latent representations of the elements in the original spaces, for instance, the terms in queries and documents. $\|\cdot\|_p$ denotes ℓ_p norm, and both ℓ_1 and ℓ_2 are used in RMLS. $\tau_{x,p}$ and $\tau_{y,p}$ are thresholds on the norms.

We point out that RMLS is not very robust, both theoretically and empirically. Wu et al. [13] prove that RMLS gives a degenerate solution with ℓ_1 regularization only. Specifically, the solution of L_x and L_y will be matrices of rank one and all the column vectors l_{xu} and l_{yv} will be proportional to each other. Wu et al. [13] propose addressing the problem with further ℓ_2 regularization on l_{xu} and l_{yv}. However, this does not solve the problem, which we will explain later in this Section, Our experiments also show that RMLS tends to create degenerate solutions.

We notice that RMLS does not penalize the case in which any x in one space matches any y in the other space, which may happen even when L_x and L_y are sparse. To cope with the problem, we introduce additional constraints on the matching matrix $L_x^T L_y$, whose (u, v)-th element corresponds to the matching score between the u-th basis vector from $\mathcal{X}$ and the v-th basis vector from $\mathcal{Y}$. Specifically, we add ℓ_1 and ℓ_2 norms on $L_x^T L_y$ as follows, which can limit the overall degree of matching any two objects.

The regularizations $\|L_x^T L_y\|_1 = \sum_{u,v} |l_{xu}^T l_{yv}|$, $\|L_x^T L_y\|_2^2 = \sum_{u,v} (l_{xu}^T l_{yv})^2$ can prevent the model from becoming a degenerate solution, and thus make the model more robust. With all of the constraints the hypothesis spaces of L_x and L_y become:

$$\mathcal{H}_x = \{L_x \mid \|l_{xu}\|_p \leqslant \tau_{x,p}, \|l_{xu}^T l_{yv}\|_p \leqslant \sigma_p, p = 1, 2, \forall u, v\},$$
$$\mathcal{H}_y = \{L_y \mid \|l_{yv}\|_p \leqslant \tau_{y,p}, \|l_{xu}^T l_{yv}\|_p \leqslant \sigma_p, p = 1, 2, \forall u, v\}.$$

Note that $\mathcal{H}_x$ and $\mathcal{H}_y$ are now related to each other because of the constraints on the interaction of the two mappings.

We then reformalize the learning of latent matching model, referred to as LMM for short, as the following optimization problem:

$$\arg\min_{L_x, L_y} -\frac{1}{n}\sum_{i=1}^{n} x_i^T L_x^T L_y y_i + \sum_{p=1,2} \frac{\theta_p}{2}\|L_x^T L_y\|_p^p + \sum_{p=1,2} \frac{\lambda_p}{2}\|L_x\|_p^p + \sum_{p=1,2} \frac{\rho_p}{2}\|L_y\|_p^p, \quad (2)$$

where θ_p, λ_p and ρ_p are the hyper-parameters for regularization.

In general, there is no guarantee that a global optimal solution of (2) exists, and thus we employ a greedy algorithm to conduct the optimization. Let F denote the corresponding objective function. The matching term in F can be reformulated as $\frac{1}{n}\sum_{i=1}^{n} x_i^T L_x^T L_y y_i = \sum_{u,v} c_{u,v} l_{xu}^T l_{yv}$, where $c_{u,v}$ is the (u, v)-th element of the empirical cross-covariance matrix $C = \frac{1}{n}\sum_{i=1}^{n} x_i y_i^T$.

For simplicity, in the following derivation, let us only consider the use of ℓ_2 regularization, i.e., set $\theta_1 = \lambda_1 = \rho_1 = 0$.

By setting the derivatives to zeros, the optimal values of l_{xu} and l_{yv} can be solved as:

$$l_{xu}^* = (\theta_2 \sum_v l_{yv} l_{yv}^T + \lambda_2 I)^{-1}(\sum_v c_{u,v} l_{yv}), l_{yv}^* = (\theta_2 \sum_u l_{xu} l_{xu}^T + \rho_2 I)^{-1}(\sum_u c_{u,v} l_{xu}). \quad (3)$$

The parameters of L_x and L_y are updated alternatively until convergence.

It should be noted that since the parameters are directly calculated, the convergence rate is fast for the coordinate descent algorithm. However, the calculations at each step in Eq. 3 involve inversion of two d-dimension matrices, which could become a computation bottleneck when the dimension of latent space is high. Therefore, we can obtained a gradient descent algorithm for LMM as an alternative[1], specifically for the case of high-dimensional latent space. The gradient descent algorithm has less computation at each step but generally needs more iterations to converge. Therefore, one always needs to consider selecting a more suitable optimization method in a specific situation.

---

[1] $l'_{xu} = l_{xu} + \gamma(\sum_v c_{u,v} l_{yv} - \theta_2 \sum_v l_{yv} l_{yv}^T l_{xu} - \lambda_2 l_{xu}), l'_{yv} = l_{yv} + \gamma(\sum_u c_{u,v} l_{xu} - \theta_2 \sum_u l_{xu} l_{xu}^T l_{yv} - \rho_2 l_{yv})$, where γ is the learning rate.

When Eq. 3 is applied to RMLS (by letting $\theta_2 = 0$), the updates of parameters in each iteration become $L_x^{(t+1)} = L_x^{(t)}(\lambda_2\rho_2)^{-1}CC^T$ and $L_y^{(t+1)} = L_y^{(t)}(\lambda_2\rho_2)^{-1}C^TC$. They are equivalent to conducting power iteration on each row of L_x and L_y independently. Consequently, all rows of L_x will converge to the eigenvector (with the largest eigenvalue) of the matrix $(\lambda_2\rho_2)^{-1}CC^T$, and so will be all rows of L_y. Thus, the optimal parameters L_x^* and L_y^* are both matrices of rank one. This justifies the necessity of regularization on the matching matrix $L_x^TL_y$.

3.2 Incorporating Semantic Knowledge to Latent Matching Model

A latent matching model trained with the method described in the previous section can perform well for head queries and documents, since it can capture the matching information from click-through data. However, for tail queries and documents, there is not enough click-through data, and it is almost impossible to accurately learn the matching relations between them. To alleviate this problem, we propose incorporating semantic knowledge of synonyms and semantic categories into the learning of the latent matching model.

Without loss of generality, we assume that in one space the semantic knowledge is represented as a set of pairs of similar objects (e.g., words or tags), denoted as $\{w_i^{(1)}, w_i^{(2)}, s_i\}_{i=1}^m$, where $w_i^{(1)}$ and $w_i^{(2)}$ represent the term vectors of the objects, and s_i is a scalar representing their weight. Therefore, the matching degrees of the pairs become $\sum_{i=1}^m s_i\ (w_i^{(1)})^T L^T L w_i^{(2)}$.

We extend the latent matching model (2) by incorporating the above 'regularization' term, for the two spaces $\mathcal{X}$ and $\mathcal{Y}$ respectively, into the objective function of learning:

$$\arg\min_{L_x,L_y} -\frac{1}{n}\sum_{i=1}^n x_i^T L_x^T L_y y_i + \sum_{p=1,2}\frac{\theta_p}{2}\|L_x^TL_y\|_p^p + \sum_{p=1,2}\frac{\lambda_p}{2}\|L_x\|_p^p + \sum_{p=1,2}\frac{\rho_p}{2}\|L_y\|_p^p - \frac{\alpha}{m_x}\sum_{i=1}^{m_x} s_{x,i}\ (w_{x,i}^{(1)})^T L_x^T L_x w_{x,i}^{(2)} - \frac{\beta}{m_y}\sum_{i=1}^{m_y} s_{y,i}\ (w_{y,i}^{(1)})^T L_y^T L_y w_{y,i}^{(2)}. \tag{4}$$

The hyper-parameters α and β control the importance of semantic knowledge from the two spaces.

Similarly, coordinate descent can be employed to solve the optimization (4). The optimal values of l_{xu} and l_{yv} are then given by

$$\begin{aligned} l_{xu}^* &= (\theta_2\sum_v l_{yv}l_{yv}^T + \lambda_2 I)^{-1}(\sum_v c_{u,v}l_{yv} + \alpha\sum_v r_{x,u,v}l_{xv}),\\ l_{yv}^* &= (\theta_2\sum_u l_{xu}L_{xu}^T + \rho_2 I)^{-1}(\sum_u c_{u,v}l_{xu} + \beta\sum_u r_{y,u,v}l_{yu}), \end{aligned} \tag{5}$$

where $r_{x,u,v}$ and $r_{y,u,v}$ denote the (u,v)-th elements of the empirical covariance matrices R_x and R_y respectively, where $R_x = \frac{1}{m_x}\sum_{i=1}^{m_x} s_{x,i}\ w_{x,i}^{(1)}(w_{x,i}^{(2)})^T$,

Algorithm 1. Coordinate Descent Algorithm for Latent Matching Model with Semantic Knowledge

```
1. Input: $C$, $R_x$, $R_y$, $\alpha$, $\beta$, $\theta$, $\lambda_2$, $\rho_2$, $T$.
2. Initialization: $t \leftarrow 0$, random matrices $L_x^{(0)}$ and $L_y^{(0)}$.
while not converge and $t \leqslant T$ do
    Compute $A_x = \theta_2 L_x^{(t)} (L_x^{(t)})^T + \lambda_2 I$ and its inverse $A_x^{-1}$ .
    Compute $A_y = \theta_2 L_y^{(t)} (L_y^{(t)})^T + \rho_2 I$ and its inverse $A_y^{-1}$.
    Compute $B_x = L_x^{(t)} C + \beta L_y^{(t)} R_y$.
    Compute $B_y = C (L_y^{(t)})^T + \alpha R_x (L_x^{(t)})^T$.
    for $u = 1 : d_x$ do
        Select $u$-th row of $B_y$ as $b_{yu}^T$,
        Compute $l_{xu}^{(t+1)} = A_y^{-1} b_{yu}$.
    end
    for $v = 1 : d_y$ do
        Select $v$-th row of $B_x$ as $b_{xv}^T$,
        Compute $l_{yv}^{(t+1)} = A_x^{-1} b_{xv}$.
    end
end
```

$R_y = \frac{1}{m_y} \sum_{i=1}^{m_y} s_{y,i}\, w_{y,i}^{(1)} (w_{y,i}^{(2)})^T$. Algorithm 1 shows the procedure of the coordinate descent algorithm for latent matching model with semantic knowledge. An alternative algorithm using gradient descent can also be obtained[2].

3.3 Acquisition of Semantic Knowledge

Synonyms are obviously useful semantic knowledge for our matching task. A general dictionary of synonyms such as WordNet is usually not suitable for a real-world setting, however. The reason is that synonyms usually heavily depend on domains. Here we adopt an algorithm for mining synonym pairs by exploiting click-through data. Specifically, we first try to find clusters of queries from a click-through bipartite graph (cf., [7]). Queries in one cluster are regarded as synonyms. Next, for each cluster, we extract pairs of terms sharing the same context as candidates of synonym pairs (cf., [8]). Here the context refers to the surrounding text of a term in the query. For example, given a query "download 2048 apk", the context for term "2048" is "download * apk", where '*' is the wildcard character. Then we go through all the clusters and count the numbers of occurrences (called support) for all the candidate pairs. The candidate pairs with support above a certain threshold are chosen as synonym pairs. Algorithm 2 shows the detailed procedure.

We denote the set of mined synonym pairs as $\{(w_{x,i}^{(1)}, w_{x,i}^{(2)}, s_{x,i})\}$, where $w_{x,i}^{(1)}$ and $w_{x,i}^{(2)}$ are the i-th pair of synonyms. $s_{x,i}$ is the corresponding weight for the pair, which is computed as the logistic transformation of the support. The knowledge about the synonym set for the query domain ($\mathcal{X}$) is formalized as $\sum_i s_{x,i}\, (w_{x,i}^{(1)})^T L_x^T L_x w_{x,i}^{(2)}$ in the optimization function.

---

[2] $l'_{xu} = l_{xu} + \gamma(\sum_v c_{u,v} l_{yv} + \alpha \sum_v r_{x,u,v} l_{xv}) - \gamma(\theta_2 \sum_v l_{yv} l_{yv}^T l_{xu} + \lambda_2 l_{xu}), l'_{yv} = l_{yv} + \gamma(\sum_u c_{u,v} l_{xu} + \beta \sum_u r_{y,u,v} l_{yu}) - \gamma(\theta_2 \sum_u l_{xu} l_{xu}^T l_{yv} + \rho_2 l_{yv})$, where γ is the learning rate.

Algorithm 2. Synonyms Mining Algorithm on Click Bipartite Graph

```
0. Notation: $Q$: query set, $D$: document set, $C$: click set, $q$: query, $d$: document, $t$: term.
1. Input: click bipartite graph $G = (Q, D, C)$.
2. Initialization: dictionary of candidate synonym pairs $S = [\ ]$.
for $d$ in $D$ do
    Collect $Q_d = \{q|(q, d) \in C\}$.
    Init $T = \{\ \}$.
    for $q$ in $Q_d$ do
        for $t$ in $q$ do
            Extract context $c_t$ of $t$ in $q$
            Add $(t, c_t)$ to $T$
        end
    end
    Find $P_d = \{(t_i, t_j)|c_{t_i} = c_{t_j}, (t_i, c_{t_i}) \in T, (t_j, c_{t_j}) \in T\}$
    for $(t_i, t_j)$ in $P_d$ do
        if $(t_i, t_j)$ not in $S$ then
            Add $(t_i, t_j)$ to $S$ and set $S[(t_i, t_j)] = 1$
        else
            Set $S[(t_i, t_j)] = S[(t_i, t_j)] + 1$
        end
    end
end
3. Sort $S$ by value in descending order.
4. Return top $K$ pairs of $S$ as the synonym pairs.
```

In addition to synonyms, we also utilize categories or tags in a taxonomy as semantic knowledge for the document domain. For example, in our experiment of the mobile app search, apps are given various tags by users. An app named "tiny racing car" is tagged "action, mario, racing, sports, auto, adventure, racing track". For each tag, we have a list of associated documents. We represent the title of each document as a tf-idf vector and calculate the average vector of the tf-idf vectors for each tag. We select the top k terms in the average vector and view them as the relevant terms to the tag. A set of 'tag-term' pairs is then obtained from all the tags and their relevant terms, and it is denoted as $\{(w_{y,i}, w_{y,ij}, s_{y,ij})\}$, where $w_{y,i}$ is the i-th tag, and $w_{y,ij}$ is the j-th relevant term to the i-th tag, and $s_{y,ij}$ is the corresponding average tf-idf value. We can formalize the knowledge for the document domain ($\mathcal{Y}$) as $\sum_i \sum_j s_{y,ij}\ (w^{(1)}_{y,i})^T L^T_y L_y w^{(2)}_{y,ij}$ in the objective function of learning of latent matching model.

4 Experiments

4.1 Experimental Setup

We take app search as example and use data from an app search engine. Each app is represented by its title and description and can be viewed as a document. Click-through logs at the search engine are collected and processed. We create two datasets from the click-through logs, one containing one week data and the other containing one month data. Table 1 reports some statistics of the two datasets. Each dataset consists of query-document pairs and their associated clicks, where

Table 1. Statistics of two training datasets

	#clicks	#queries	#apps
one-week click-through data	1,020,854	190,486	110,757
one-month click-through data	3,441,768	534,939	192,026

a query and a document are represented by a term-frequency vector and a tf-idf vector of the title, respectively. The queries and documents are regarded as heterogeneous data in two different spaces, because queries and documents have different characteristics.

In addition, we randomly sample two sets of 200 queries from a time period different from that of training datasets, and take them as two test datasets.

Each test set is composed of 100 head queries and 100 tail queries, according to the frequencies of them. In the following sub-sections, performance on the whole random test set as well as the head and tail subsets will be reported. For each query in the test sets, we collect top 20 apps retrieved by the app search engine and then label the query-app pairs at four levels of matching: Excellent, Good, Fair, and Bad.

As evaluation measures, Normalized Discounted Cumulative Gain (NDCG) at positions 1, 3, 5, 10 are used. We choose the conventional IR model of BM25 (with the parameters tuned for best performance in the training set), and two latent matching models of PLS (Partial Least Square) and RMLS (Regularized Mapping to Latent Structures) as the baseline methods. Our basic model is denoted as LMM (Latent Matching Model) and our augmented models are denoted as LMM-X where X stands for the type of incorporated semantic knowledge.

4.2 Experimental Results

Latent Matching Model: We conduct a series of experiments to test the performances of LMM, LMM-X and the baseline models. For RMLS, LMM, and LMM-X, the results with latent dimensionalities of 100 and 500 are reported. For PLS, only the performance with latent dimensionality of 100 is reported, due to its scalability limitation.

Table 2 report the performances of the models trained using one-week click-through data, evaluated on the test tests: random queries, head queries and tail queries respectively. From the results, we can see that: (1) all the latent matching models significantly outperform the conventional BM25 model in terms of all evaluation measures; (2) among the latent space models with the same dimension, LMM achieves the best performances in many cases. The improvements of LMM over BM25 and RMLS are statistically significant (paired t-test, p-value < 0.05); (3) the improvements of LMM over the other baseline models are larger on tail queries than on head queries, which indicates that LMM can really enhance matching performance for tail queries; (4) for LMM, the performance increases as the dimensionality of latent space increases. Note that PLS requires

Table 2. Matching performance on one week data

Model	NDCG on Random queries				NDCG on Head queries				NDCG on Tail queries			
(Dimension)	@1	@3	@5	@10	@1	@3	@5	@10	@1	@3	@5	@10
BM25	0.687	0.700	0.707	0.741	0.729	0.754	0.758	0.786	0.645	0.645	0.656	0.696
PLS(100)	0.715	0.733	0.738	0.767	0.756	0.780	0.787	0.809	0.675	0.686	0.689	0.726
RMLS(100)	0.697	0.727	0.732	0.765	0.740	0.767	0.772	0.801	0.653	0.686	0.692	0.729
LMM(100)	0.713	0.727	0.741	0.771	0.744	0.771	0.785	0.813	0.681	0.684	**0.697**	0.729
RMLS(500)	0.709	0.720	0.731	0.762	0.742	0.765	0.777	0.805	0.677	0.674	0.686	0.719
LMM(500)	**0.727**	**0.737**	**0.738**	**0.772**	**0.766**	**0.783**	**0.787**	**0.812**	**0.689**	**0.690**	0.688	**0.731**

Table 3. Matching performance on one-month data

Model	NDCG on Random queries				NDCG on Head queries				NDCG on Tail queries			
(Dimension)	@1	@3	@5	@10	@1	@3	@5	@10	@1	@3	@5	@10
BM25	0.644	0.681	0.714	0.740	0.721	0.738	0.756	0.771	0.567	0.624	0.672	0.710
PLS(100)	0.692	0.727	0.749	0.772	0.735	0.757	0.774	0.788	0.649	0.698	0.724	0.756
RMLS(100)	0.668	0.703	0.727	0.752	0.736	0.746	0.762	0.779	0.600	0.660	0.693	0.726
LMM(100)	0.692	**0.733**	**0.751**	0.775	0.744	0.765	**0.779**	0.793	0.640	0.700	0.724	0.758
RMLS(500)	0.687	0.725	0.745	0.774	**0.753**	**0.767**	0.772	0.798	0.620	0.684	0.719	0.751
LMM(500)	**0.704**	0.730	0.749	**0.780**	0.745	0.756	0.770	**0.795**	**0.662**	**0.704**	**0.729**	**0.765**

SVD and thus becomes practically intractable when the dimension is large. In that sense, RMLS and LMM exhibit their advantages over PLS on scalability.

Table 3 show the comparison results of models trained using one-month click-though data, evaluated on the tested random queries, head queries and tail queries respectively, which follows the same trends as that of one-week data, especially on tail queries.

Incorporating Semantic Knowledge: Next, we test the performances of the LMM-X models which incorporate semantic knowledge into the model. The LMM-X models have the ability to leverage multiple sources of semantic knowledge by adding regularization terms to the objective function. We consider two methods of acquiring and utilizing semantic knowledge. In the first method we mine and use synonym pairs from the click-through logs, In the second method we collect and use over 50,000 tags in the app search engine, which are described in 3.3.

We conduct experiments using LMM model and the two types of knowledge. We summarize the results in Table 4 for one-week data and Table 5 for one-month data evaluated on random queries, head queries and tail queries respectively. For each training dataset, we first separately train the LMM model augmented with the synonyms dictionary and the tag-term pairs, denoted as LMM-Synonyms and LMM-Tags, respectively. Then we train the LMM model augmented with both types of knowledge, denoted as LMM-Both. From the results we can see: (1) with knowledge embedded, the performances of the LMM model can be consistently improved; (2) the improvements of LMM-Both over LMM are statistically significant (paired t-test, p-value < 0.05) in terms of most evaluation measures; (3) more significant improvements are made on tail queries than on head

Table 4. Matching performance on one-week data

Model (Dimension)	NDCG on Random queries				NDCG on Head queries				NDCG on Tail queries			
	@1	@3	@5	@10	@1	3	5	10	@1	@3	@5	@10
LMM(100)	0.713	0.727	0.741	0.771	0.744	0.771	0.785	0.813	0.681	0.684	0.697	0.729
LMM-Synonyms(100)	0.730	0.743	0.747	0.772	0.757	0.791	0.794	0.815	0.704	0.695	0.700	0.729
LMM-Tags(100)	0.727	0.746	0.747	0.773	0.757	0.789	0.796	0.817	0.697	0.699	0.699	0.728
LMM-Both(100)	0.735	0.750	0.752	0.772	0.762	0.798	0.799	0.815	**0.709**	0.702	0.705	0.729
LMM(500)	0.727	0.737	0.738	0.772	0.766	0.783	0.787	0.812	0.689	0.690	0.688	0.731
LMM-Synonyms(500)	0.743	0.749	0.758	0.781	0.779	**0.795**	**0.802**	0.819	0.707	0.703	0.714	0.743
LMM-Tags(500)	0.743	0.747	0.759	**0.783**	0.779	0.793	0.801	**0.820**	0.707	0.702	0.716	**0.745**
LMM-Both(500)	**0.743**	**0.750**	**0.759**	0.781	**0.779**	0.793	0.801	0.819	0.707	**0.708**	**0.718**	0.743

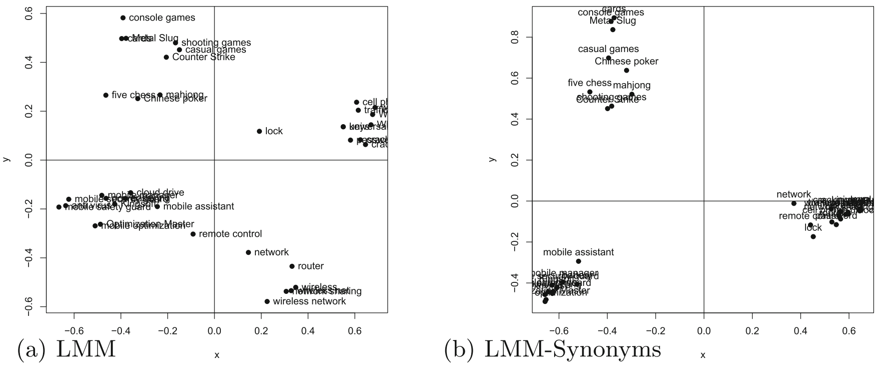
Table 5. Matching performance on one-month data

Model (Dimension)	NDCG on Random queries				NDCG on Head queries				NDCG on Tail queries			
	@1	@3	@5	@10	@1	3	5	10	@1	@3	@5	@10
LMM(100)	0.692	0.727	0.749	0.772	0.735	0.757	0.774	0.788	0.649	0.698	0.724	0.756
LMM-Synonyms(100)	0.708	0.738	0.749	0.780	0.741	0.771	0.770	0.795	0.676	0.705	0.729	0.765
LMM-Tags(100)	0.707	0.734	0.750	0.779	0.738	0.760	0.767	0.795	0.676	0.708	0.733	0.763
LMM-Both(100)	0.715	0.739	0.745	0.779	0.738	0.760	0.767	0.795	0.676	0.708	0.733	0.760
LMM(500)	0.704	0.730	0.749	0.780	0.745	0.756	0.770	**0.795**	0.662	0.704	0.729	0.765
LMM-Synonyms(500)	0.719	0.741	**0.762**	**0.783**	**0.752**	0.761	0.775	0.793	0.686	0.723	**0.748**	**0.773**
LMM-Tags(500)	0.719	0.741	0.762	0.781	0.752	0.759	**0.778**	0.794	0.686	0.723	0.746	0.769
LMM-Both(500)	**0.721**	**0.745**	0.761	0.782	0.751	**0.763**	0.777	0.793	**0.691**	**0.728**	0.745	0.771

queries; (4) the improvements of semantic knowledge augmentation are slightly less when the latent dimensionality is high (500) than when it is low (100).

We investigate the latent spaces of LMMs learned with and without incorporating synonym dictionary. The latent representations of some randomly selected words are plotted on a 2-D graph using the multidimensional scaling technique, in Fig. 1. By comparing the distributions of words in Fig. 1(a) and (b), we can clearly see that similar words are clustered closer in LMM-Synonyms than in LMM. This clearly indicates that knowledge about synonyms can be effectively incorporated into LMM-Synonyms and thus the model can further improve matching. For the latent spaces of LMMs learned with and without incorporating category tags, we observe a similar phenomenon.

We make analysis of the ranking results of LMM and LMM-X. In many cases, we find that the semantic relations embedded in LMM-X can indeed improve relevance ranking. For example, the terms "best", "most", "hardest", and "strongest" are mined as synonyms from the log, and these terms are clustered together in the latent space induced by LMM-Synonyms. In search, for the query of "best game in the history", documents about "most popular game", "hardest game" and "strongest game" are promoted to higher positions, which enhances the relevance as well as the richness of search result. However, there are also some bad cases, mainly due to noises in the synonym dictionary. For example, in one experiment our mining algorithm identifies "google" and "baidu" as synonyms. Then for the query of "google map", a document about "baidu map" is ranked higher than a document about "google earth". Therefore, improving



(a) LMM (b) LMM-Synonyms

Fig. 1. Representations of query words in latent space.

the quality of the mined semantic knowledge is one issue which we need to address in the future.

5 Conclusion

In this paper, we have studied the problem of latent semantic matching for search. We have proposed a linear latent semantic model that leverages not only click-through data, but also semantic knowledge such as synonym dictionary and category hierarchy. The semantic knowledge is incorporated into the model by imposing regularization based on synonym and/or category information. We employ two methods to acquire semantic knowledge. One is to mine synonyms from the click bipartite graph, and the other is to utilize categories of documents. We have developed a coordinate descent algorithm and a gradient descent algorithm to solve the learning problem. The algorithms can be employed depending on the settings. We have conducted experiments on two large-scale datasets from a mobile app search engine. The experimental results demonstrate that our model can make effective use of semantic knowledge, and significantly outperform existing matching models.

References

1. Bai, B., Weston, J., Grangier, D., Collobert, R., Sadamasa, K., Qi, Y., Chapelle, O., Weinberger, K.: Supervised semantic indexing. In: Proceedings of the 18th ACM Conference On Information and Knowledge Management, pp. 187–196. ACM (2009)
2. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent Dirichlet allocation. J. Mach. Learn. Res. **3**, 993–1022 (2003)
3. Deerwester, S., Dumais, S.T., Furnas, G.W., Landauer, T.K., Harshman, R.A.: Indexing by latent semantic analysis. J. Am. Soc. Inf. Sci. **41**(6), 391–407 (1990)

4. Hardoon, D., Szedmak, S., Shawe-Taylor, J.: Canonical correlation analysis: an overview with application to learning methods. Neural Comput. **16**(12), 2639–2664 (2004)
5. Hofmann, T.: Probabilistic latent semantic indexing. In: Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 50–57. ACM (1999)
6. Huang, P.-S., He, X., Gao, J., Deng, L., Acero, A., Heck, L.: Learning deep structured semantic models for web search using click through data. In: Proceedings of the 22nd ACM International Conference on Information and Knowledge Management, pp. 2333–2338. ACM (2013)
7. Jiang, D., Pei, J., Li, H.: Mining search and browse logs for web search: a survey. ACM Trans. Intell. Syst. Technol. (TIST) **4**(4), 57 (2013)
8. Lamping, J., Baker, S.: Determining query term synonyms within query context. US Patent 7,636,714, December 2009
9. Li, H., Xu, J.: Semantic matching in search. Found. Trends Inf. Retr. **8**, 89 (2014)
10. Lu, Z., Li, H.: A deep architecture for matching short texts. In: Advances in Neural Information Processing Systems, pp. 1367–1375 (2013)
11. Rosipal, R., Krämer, N.: Overview and recent advances in partial least squares. In: Saunders, C., Grobelnik, M., Gunn, S., Shawe-Taylor, J. (eds.) SLSFS 2005. LNCS, vol. 3940, pp. 34–51. Springer, Heidelberg (2006). doi:10.1007/11752790_2
12. Wang, Q., Xu, J., Li, H., Craswell, N.: Regularized latent semantic indexing: a new approach to large-scale topic modeling. ACM Trans. Inf. Syst. **31**(1), 5 (2013)
13. Wu, W., Lu, Z., Li, H.: Learning bilinear model for matching queries and documents. J. Mach. Learn. Res. **14**(1), 2519–2548 (2013)