Location-Sensitive Resources Recommendation in Social Tagging Systems

Chang Wan Ben Kao David W. Cheung
Department of Computer Science, The University of Hong Kong, Pokfulam, Hong Kong
{cwan, kao, dcheung}@cs.hku.hk

ABSTRACT

In social tagging systems, resources such as images and videos are annotated with descriptive words called *tags*. It has been shown that tag-based resource searching and retrieval is much more effective than content-based retrieval. With the advances in mobile technology, many resources are also geo-tagged with location information. We observe that a traditional tag (word) can carry different semantics at different locations. We study how location information can be used to help distinguish the different semantics of a resource's tags and thus to improve retrieval accuracy. Given a search query, we propose a *location-partitioning method* that partitions all locations into regions such that the user query carries distinguishing semantics in each region. Based on the identified regions, we utilize location information in estimating the ranking scores of resources for the given query. These ranking scores are learned using the Bayesian Personalized Ranking (BPR) framework. Two algorithms, namely, LTD and LPITF, which apply Tucker Decomposition and Pairwise Interaction Tensor Factorization, respectively for modeling the ranking score tensor are proposed. Through experiments on real datasets, we show that LTD and LPITF outperform other tag-based resource retrieval methods.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Retrieval models

General Terms

Algorithms, Experimentation, Performance

Keywords

ranking, resources recommendation, location-sensitive

1. INTRODUCTION

In a social tagging system (e.g., Flickr and YouTube), resources (e.g., photos and videos) are often annotated with descriptive words called *tags*. Previous studies have shown that searching resources based on their tags leads to very effective and accurate resource retrieval [2, 6]. However, in some cases, tags are ambiguous. For example, the word *mouse* could refer to an animal, a cartoon character (Mickey), or a computer peripheral device. Tag ambiguity lowers the effectiveness of tag-based search. Over the years, researchers have spent much effort in proposing techniques to deal with the tag-ambiguity problem [3, 5].

With the advances in mobile technology and the global positioning system (GPS), resources, especially images and videos, are geo-tagged with location information. Our observation is that resources' location information often provides useful hints in identifying the semantics of their tags. As an example, consider a set of photos that are tagged with the tag "sand". Depending on the locations at which these photos were taken, the semantics of the tag could be inferred. For example, the tag "sand" of a photo taken in the Sahara very likely refers to the concept "desert"; for photos taken in Hawaii and Singapore, the same tag probably refers to "beach" and the casino hotel "Marina Bay Sands", respectively. The semantics of a user query can likewise be inferred. For example, a user located in the Sahara who issues the query "sand" is probably looking for photos of the desert.

In the above example, we suggested that the semantics of a tag or a query could be deduced by the *region* in which the resource was obtained or the query issuer was located. Intuitively, a region is a set of locations within which a tag or a query carries a unique distinguishing meaning. An interesting question is how these regions (such as *The Sahara*, *Hawaii*, and *Singapore*) be automatically identified. We remark that different tags induce different partitionings of the world into regions: while there are multiple regions for the tag "sand", there are two regions for the tag "football" (*American football* in North America and *soccer* anywhere else in the world), and there is only one region (the whole world) for a tag that refers to some globally known concept, such as "iPhone". Another interesting question is if regions with respect to tags and queries could be identified, how could this information be leveraged in order to achieve good resource retrieval results.

The objective of this paper is to address the above two questions. First, given a tag t, we identify the regions induced by t by using machine learning techniques to train a function $f_t(l)$, which maps a location l into a region. Second, given a query q issued by a user located at location l, we estimate the ranking scores of resources with respect to q. We extend the BPR framework [9] to include region information in the score estimation. As we will see later in Section 5, our solutions lead to very good resource retrieval results. Here, we summarize our contributions:

- We propose an algorithm for computing the region mapping function $f_t(l)$.
- We propose algorithms for estimating the ranking scores of resources with respect to a query.

CIKM'12, October 29–November 2, 2012, Maui, HI, USA.

- We perform an extensive experiment using real datasets to evaluate the performance of our algorithms in terms of resource retrieval effectiveness.

2. RELATED WORK

LSI is a popular semantic analysis technique that has proven to be very successful in IR. It attempts to overcome the word ambiguity problem by extracting concepts from words in an unstructured text collection, which is realized by performing Singular Value Decomposition (SVD) on a term-document matrix and by measuring the tag-pair similarities so derived. With the extracted concepts, one can rank resources w.r.t. a given query by measuring the similarities between resources and the query on the concept level. Bi et al. [3] observes that user information can improve resource retrieval results. They extend LSI to the CubeLSI method by applying three-dimensional Tucker Decomposition (TD) on a third-order tensor whose dimensions are resources, users, and tags.

Besides resource retrieval, tag recommendation is also a popular topic in the study of social tagging systems. Factorization models are a popular method. Rendle et al. [10] model the likelihood score that a user u would tag a resource r with a tag t as a 3D tensor, which is approximated using TD. By using AUC (area under the ROC-curve) as the ranking criteria, [10] optimizes the model parameters of TD to obtain optimal values of the entries in the tensor. [9] extends the work of [10] by applying the Bayesian Personalized Ranking (BPR) framework in model parameter optimization. In [11], the Pairwise Interaction Tensor Factorization (PITF) model is proposed, which is shown to be more efficient than the TD model used in [10]. These techniques for tag recommendation can also be employed for solving the resources retrieval problem.

Some studies [8, 4] utilize location information in resource retrieval. Lu, Lu, and Cong [8] present a hybrid index tree called IUR-tree, which combines location proximity with textual similarity. They further design a branch-and-bound search algorithm to retrieve resources that are the closest and the most relevant to a query. Our method is significantly different from theirs. In particular, we propose a query-driven location partitioning algorithm to transform the location dimension to a region dimension, which significantly improves searching performance.

3. PROBLEM STATEMENT

Our approach to integrating location information in solving the tag-based resource retrieval problem consists of two components: (1) A location-partitioning scheme that derives a region-mapping function $f_t(l)$ for each tag t, and (2) a score function that ranks resources in terms of their relevancy to a given query q. In this section we formally define these two problems.

We consider four attributes of a social tagging system: a set of users (taggers) U, a set of tags T, a set of resources R, and a set of locations L[1]. We assume that each location $l \in L$ is represented by a longitude-latitude pair (x_l, y_l). Resources in the systems are tagged by users. We assume that the tagging information is represented by a set $S \subseteq U \times T \times R \times L$ of tagging records. Each tagging record is a quadruple $(u, t, r, l) \in S$, which indicates that a user u assigns a tag t to a resource r. l is the geo-tag location of resource r. Although l is dependent on r, to simplify our discussion, we explicitly include l in a tagging record. A user query $q = (u_q, t_q, l_q)$ is a triplet which indicates that a user u_q, who is located at location l_q, issues a query with the tag t_q. For simplicity, we assume that each query consists of only one tag. Our technique can be easily extended to handle multi-tag queries, for example, by considering all the tags in a multi-tag query as a single *complex tag*.

Given a tag t, the *problem of location partitioning* is to partition L into a set of k_t regions $\mathcal{H}_t = \{H_1, \ldots, H_{k_t}\}$ such that tag t refers to the same concept within each region H_i and dissimilar concepts in different regions. This partitioning can be represented by a *region mapping function* (RMF) $f_t : L \rightarrow [1 \ldots k_t]$, which maps a location $l \in L$ to the region $H_{f_t(l)} \in \mathcal{H}_t$. We say that tag t induces the region set $\mathcal{H}_t$.

Given a query q, we first obtain the RMF f_{t_q}. Let $\mathcal{H}_q$ be the set of regions induced by t_q. Note that while the query tag t_q (such as "sand") can assume different meanings at different locations of the world, it refers to a unique concept within each region $H_i \in \mathcal{H}_q$. For notational convenience, we use H_q to denote $H_{f_{t_q}(l_q)}$, which is the region that encompasses the location (l_q) of the query.

The *resource ranking problem* is to derive a score function $\hat{y}(u, t, H, r)$: $U \times T \times \mathcal{H}_q \times R \rightarrow \mathbb{R}$, which gives a relevancy score of resource r with respect to a user query issued by user u, who is located in the region H, with the query tag t. As we will see later, we solve this problem by representing the score function $\hat{y}$ by a 4D tensor $\hat{Y}$, called the *score tensor*, such that $\hat{y}(u, t, H, r)$ is represented by the entry $\hat{y}_{u,t,H,r}$ of the score tensor. Finally, the top-N resources with the highest relevancy scores are returned as the answer set of query q.

$$\mathcal{TN}(q, N) := \arg \max_{r \in R}^{N} \hat{y}_{u_q, t_q, H_q, r}. \quad (1)$$

4. ALGORITHMS

In this section we describe the algorithms for solving the location partitioning problem and the resource ranking problem. For the former, we show how to train an RMF f_{t_q} for the query tag t_q. For the latter, we show how to extend the BPR framework to include region information in estimating the 4D score tensor $\hat{Y}$.

4.1 Location Partitioning

Our objective is to derive f_{t_q} based on the tagging record set S. Our approach is based on two intuitive assumptions: (1) Users in close proximity are likely to have the same understanding of a tag. (2) If t_q's associations (e.g., co-occurrences) with other tags in resources at a certain location l_1 is similar to those at another location l_2, then l_1 and l_2 should belong to the same region. Procedurally, we first retrieve all the tagging records that involve t_q into a set S_q, i.e., $S_q = \{(u, t_q, r, l) \in S\}$. Next, we create a set of training instances $\mathbf{X}$ in the following way. Given any tag t and a resource r that is tagged by t_q, we define the distance of t from t_q w.r.t. r by

$$d_q(t, r) = \frac{c(t, r) - c(t_q, r)}{\max_i \{|c(t, r) - c(t_i, r)|\}}, \quad (2)$$

where $c(t, r)$ is the number of times resource r is tagged by t. Essentially, $d_q(t, r)$ measures the difference in the tagging frequencies of tags t_q and t for the resource r, scaled to the range [-1,1]. Next, for each tuple $(u, t_q, r, l) \in S_q$, we generate an instance $\mathbf{x} = (d_q(t_1, r), \ldots, d_q(t_{|T|}, r), l)$ and put it in $\mathbf{X}$.

Based on these instances (observations), we create partitions of $\mathbf{X}$ to maximize a posteriori (MAP) estimate of parameters as follows. Suppose the instances in $\mathbf{X}$ are observations of the mixture of k multivariate normal distributions of $|T| + 2$ dimensions, let $\mathbf{z} = (z_1, z_2, \ldots, z_{|\mathbf{X}|})$ be the latent variables that determine which distributions the observations come from. We have:

$$\mathbf{X}|(z_j = i) \sim \mathcal{N}_d(\boldsymbol{\mu}_i, \Sigma_i), \quad 1 \leq i \leq k. \quad (3)$$

[1] L includes all possible locations in the world, not only the locations that are associated with the resources in the social tagging system.

Let $\mathcal{U} = [\boldsymbol{\mu}_i]$, $\boldsymbol{\Sigma} = [\Sigma_i]$, and $P(z_j = i) = \varphi_i$ such that $\sum_{i=1}^{k} \varphi_i = 1$. Let $\boldsymbol{\varphi} = [\varphi_i] \in \mathbb{R}_+^{k \times 1}$. The model parameters to be estimated is denoted by $\Theta = (\boldsymbol{\varphi}, \mathcal{U}, \boldsymbol{\Sigma})$. The likelihood function is:

$$L(\Theta; \mathbf{X}, \mathbf{z}) = P(\mathbf{X}, \mathbf{z}|\Theta) = \prod_{j=1}^{|\mathbf{X}|} \sum_{i=1}^{k} \mathbb{I}(z_j = i) \varphi_i f_P(\mathbf{x}_j; \boldsymbol{\mu}_i, \Sigma_i), \quad (4)$$

where $\mathbb{I}$ is the indicator function and f_P is the probability density function of a multivariate normal. We apply expectation-maximization (EM) algorithm to find MAP estimates of Θ with the following equations to update $\boldsymbol{\varphi}, \mathcal{U}, \boldsymbol{\Sigma}$:

$$\boldsymbol{\varphi}^{(t+1)} = \arg\max_{\boldsymbol{\varphi}} Q(\Theta|\Theta^{(t)}) = \arg\max_{\boldsymbol{\varphi}} (\sum_{j=1}^{|\mathbf{X}|} \sum_{i=1}^{k} P_{ji}^{(t)} \log \varphi_i^t), \quad (5)$$

$$\boldsymbol{\mu}_i^{(t+1)} = \frac{\sum_{j=1}^{|\mathbf{X}|} P_{ji}^{(t)} \mathbf{x}_j}{\sum_{j=1}^{|\mathbf{X}|} P_{ji}^{(t)}}, \quad (6)$$

$$\Sigma_i^{(t+1)} = \frac{\sum_{j=1}^{|\mathbf{X}|} P_{ji}^{(t)} (\mathbf{x}_j - \boldsymbol{\mu}_i^{(t+1)})(\mathbf{x}_j - \boldsymbol{\mu}_i^{(t+1)})\top}{\sum_{j=1}^{|\mathbf{X}|} P_{ji}^{(t)}}, \quad (7)$$

where P_{ji} is the conditional probability of $\mathbf{x}_j$ belonging to the i-th distribution, given by:

$$P_{ji}^{(t)} = P(z_j = i|\mathbf{x}_j; \Theta^{(t)}) = \frac{\varphi_i^{(t)} f_P(\mathbf{x}_j; \boldsymbol{\mu}_i^{(t)}, \Sigma_i^{(t)})}{\sum_{i=1}^{k} \varphi_i^{(t)} f_P(\mathbf{x}_j; \boldsymbol{\mu}_i^{(t)}, \Sigma_i^{(t)})}. \quad (8)$$

Let $L_q = \{l|(u, t_q, r, l) \in S_q\}$. After the EM algorithm is done, for each location $\tilde{l} \in L_q$, the RMF value $f_{t_q}(\tilde{l})$ is determined by

$$f_{t_q}(\tilde{l}) = \arg\max_i \varphi_i f_P(\mathbf{x}; \boldsymbol{\mu}_i, \Sigma_i). \quad (9)$$

For other locations $l \in L - L_q$, we first determine the location $\tilde{l}_* \in L_q$ that gives the smallest Euclidean distance $D(l, \tilde{l}_*)$ from l (i.e., $\tilde{l}_* = \arg\min_{\tilde{l} \in L_q} D(l, \tilde{l})$). The region of $\tilde{l}_*$ is taken as the region of l, i.e., $f_{t_q}(l) = f_{t_q}(\tilde{l}_*)$.

So far, we have assumed that the number of regions, k, is known. We can determine the value of k by gradually increasing its value from 1 until the likelihood value (Equation 4) stops increasing. Algorithm 1 summarizes the location partitioning algorithm LP.

Algorithm 1 LP

INPUT t_q, S_q, L, L_q.
OUTPUT The RMF f_{t_q}.

```
Generate the set of observations X;
k = 0;
repeat
    k = k + 1;
    Initialize Θ = (φ, U, Σ);
    repeat
        Update P_ji, φ, U, Σ according to Equations 5 - 8;
    until convergence
until L(Θ; X, z) does not increase
∀l̃ ∈ L_q, f_tq(l̃) = arg max_i φ_i f_P(x; μ_i, Σ_i);
∀l ∈ L − L_q, l̃_* = arg min_{l̃∈L_q} D(l, l̃); f_tq(l) = f_tq(l̃_*);
RETURN f_tq
```

4.2 Estimating the Score Tensor

Our next task is to estimate the score tensor $\hat{Y}$. We generate the values of the elements in $\hat{Y}$ by assuming a certain model. Let Θ represents the model's parameters. We determine Θ by first constructing a set of observations $\mathbf{Y}$ and then use these observations to find the optimal Θ following the Bayesian Personalized Ranking (BPR) framework. In the following, we first give the details of how the training set $\mathbf{Y}$ is constructed and how BPR is applied. Then, we show how $\hat{Y}$ is determined using the Tucker Decomposition (TD) and PITF as two possible models of $\hat{Y}$.

First, for each record $(u, t, l, r) \in S$, we replace the location l of the record by the region $H \in \mathcal{H}_q$ to which l belongs. Let $S_H = \{(u, t, H, r)|((u, t, l, r) \in S) \wedge (H = H_{f_{t_q}(l)})\}$. Suppose a user u tags a resource r_1, which is located in region H, with the tag t, then $(u, t, H, r_1) \in S_H$. Since u makes such a tagging, resource r is relevant to the tag t from the perspective of u. On the other hand, if the same user does not tag another resource r_2 with t, then r_2 is not relevant to t according to u. In other words, $\hat{y}_{u,t,H,r_1}$ should be large and $\hat{y}_{u,t,H,r_2}$ should be small. Define $\Delta\hat{y}(u, t, H, r_1, r_2) = \hat{y}_{u,t,H,r_1} - \hat{y}_{u,t,H,r_2}$. We construct the set $\mathbf{Y}$ as

$$\mathbf{Y} = \{(u, t, H, r_1, r_2)|((u, t, H, r_1) \in S_H) \wedge ((u, t, H, r_2) \notin S_H)\}. \quad (10)$$

Given a model and its parameters Θ, we follow [9] in estimating Θ. Specifically, based on Bayes' theorem,

$$p(\hat{Y}|\mathbf{Y}) \propto p(\mathbf{Y}|\hat{Y}) p(\hat{Y}). \quad (11)$$

For each observation $w = (u, t, H, r_1, r_2) \in \mathbf{Y}$, the probability $p(w|\hat{Y})$ is estimated by,

$$p((u, t, H, r_1, r_2)|\hat{Y}) = \sigma(\Delta\hat{y}(u, t, H, r_1, r_2)), \quad (12)$$

where σ is the logistic function $\sigma(x) = \frac{1}{1+e^{(-x)}}$. We assume that model parameters are drawn from a normal distribution $\boldsymbol{\Theta} \sim \mathcal{N}(0, \sigma_{\boldsymbol{\Theta}}^2 \mathbf{I})$. Let $\lambda_{\boldsymbol{\Theta}}$ be the regularization constant of $\sigma_{\boldsymbol{\Theta}}$ and α be the learning rate of BPR. Algorithm 2 shows the model optimization procedure BPR-OPT for training model parameters $\boldsymbol{\Theta}$.

Algorithm 2 Model Optimization Using Bayesian Theorem and MAP (BPR-OPT)

INPUT $\mathbf{Y}$, $\lambda_{\boldsymbol{\Theta}}$, α
OUTPUT the predictive model parameters $\boldsymbol{\Theta}$.

```
Initialize Θ;
repeat
    Draw (u, t, H, r_1, r_2) from Y;
    Θ ← Θ + α((1 − σ(Δŷ(u, t, H, r_1, r_2))) ∂/∂Θ Δŷ(u, t, H, r_1, r_2) − λ_Θ Θ);
until convergence
RETURN Θ
```

Factorization models are very popular models in recommendation systems and resource searching systems. We use TD and PITF as two models for predicting the score tensor $\hat{Y}$. We show how these models' parameters can be learned using BPR-OPT.

[Tucker Decomposition (TD)] TD factorizes a high-order cube $\hat{Y}$ into a core tensor $\hat{C} \in \mathbb{R}^{p_u \times p_t \times p_h \times p_r}$ and four factor matrices $\hat{U} \in \mathbb{R}^{|U| \times p_u}$, $\hat{T} \in \mathbb{R}^{|T| \times p_t}$, $\hat{H} \in \mathbb{R}^{|\mathcal{H}_q| \times p_h}$ and $\hat{R} \in \mathbb{R}^{|R| \times p_r}$, where p_u, p_t, p_h, and p_r are parameters which control the sizes of $\hat{C}$'s dimensions. Each entry in $\hat{Y}$ can be formulated as:

$$\hat{y}_{u,t,H,r}^{\text{TD}} = \sum_{\tilde{u}} \sum_{\tilde{t}} \sum_{\tilde{H}} \sum_{\tilde{r}} \hat{c}_{\tilde{u},\tilde{t},\tilde{H},\tilde{r}} \hat{u}_{u,\tilde{u}} \hat{t}_{t,\tilde{t}} \hat{H}_{H,\tilde{H}} \hat{r}_{r,\tilde{r}}. \quad (13)$$

The model parameters Θ include all the entries of $\hat{C}$, $\hat{U}$, $\hat{T}$, $\hat{H}$, and $\hat{R}$. For learning the model parameters using BPR-OPT, the gradients $\frac{\partial}{\partial\theta}\hat{y}_{u,t,H,r}$ used in Algorithm 2 are:

$$\begin{aligned}
\frac{\partial\hat{y}^{\text{TD}}_{u,t,H,r}}{\partial\hat{c}_{\tilde{u},\tilde{t},\tilde{H},\tilde{r}}} &= \hat{u}_{u,\tilde{u}}\hat{t}_{t,\tilde{t}}\hat{H}_{H,\tilde{H}}\hat{r}_{r,\tilde{r}}\\
\frac{\partial\hat{y}^{\text{TD}}_{u,t,H,r}}{\partial\hat{u}_{u,\tilde{u}}} &= \sum_{\tilde{t}}\sum_{\tilde{H}}\sum_{\tilde{r}}\hat{c}_{\tilde{u},\tilde{t},\tilde{H},\tilde{r}}\hat{t}_{t,\tilde{t}}\hat{H}_{H,\tilde{H}}\hat{r}_{r,\tilde{r}}\\
\frac{\partial\hat{y}^{\text{TD}}_{u,t,H,r}}{\partial\hat{t}_{t,\tilde{t}}} &= \sum_{\tilde{u}}\sum_{\tilde{H}}\sum_{\tilde{r}}\hat{c}_{\tilde{u},\tilde{t},\tilde{H},\tilde{r}}\hat{u}_{u,\tilde{u}}\hat{H}_{H,\tilde{H}}\hat{r}_{r,\tilde{r}}\\
\frac{\partial\hat{y}^{\text{TD}}_{u,t,H,r}}{\partial\hat{H}_{H,\tilde{H}}} &= \sum_{\tilde{u}}\sum_{\tilde{t}}\sum_{\tilde{r}}\hat{c}_{\tilde{u},\tilde{t},\tilde{H},\tilde{r}}\hat{u}_{u,\tilde{u}}\hat{t}_{t,\tilde{t}}\hat{r}_{r,\tilde{r}}\\
\frac{\partial\hat{y}^{\text{TD}}_{u,t,H,r}}{\partial\hat{r}_{r,\tilde{r}}} &= \sum_{\tilde{u}}\sum_{\tilde{t}}\sum_{\tilde{H}}\hat{c}_{\tilde{u},\tilde{t},\tilde{H},\tilde{r}}\hat{u}_{u,\tilde{u}}\hat{t}_{t,\tilde{t}}\hat{H}_{H,\tilde{H}}.
\end{aligned} \tag{14}$$

We call our solution of estimating $\hat{Y}$ using TD, **LTD**. The complexity of LTD is $O(p_u \cdot p_t \cdot p_h \cdot p_r)$.

[Pairwise Interaction Tensor Factorization] (PITF) The time complexity of TD is high. To improve efficiency, PITF is proposed in [11], which only considers two-way interactions between r and other dimensions. We extend PITF to a 4D version to include the region dimension H. We factorize $\hat{Y}$ into four factor matrices $\hat{U} \in \mathbb{R}^{|U|\times p}$, $\hat{T} \in \mathbb{R}^{|T|\times p}$, $\hat{H} \in \mathbb{R}^{|\mathcal{H}_q|\times p}$ and $\hat{R} \in \mathbb{R}^{|R|\times p}$, where p is a parameter that controls the size of the matrices. Each entry in $\hat{Y}$ can be formulated as:

$$\hat{y}^{\text{PITF}}_{u,t,H,r} = \sum_p \hat{u}_{u,p}\cdot\hat{r}^U_{r,p} + \sum_p \hat{t}_{t,p}\cdot\hat{r}^T_{r,p} + \sum_p \hat{H}_{H,p}\cdot\hat{r}^H_{r,p} \tag{15}$$

BPR-OPT can then be applied in a way similar to that of TD, with the following gradients:

$$\begin{aligned}
&\frac{\partial\hat{y}^{\text{PITF}}_{u,t,H,r}}{\partial\hat{u}_{u,p}} = \hat{r}^U_{r,p}, \frac{\partial\hat{y}^{\text{PITF}}_{u,t,H,r}}{\partial\hat{t}_{t,p}} = \hat{r}^T_{r,p}, \frac{\partial\hat{y}^{\text{PITF}}_{u,t,H,r}}{\partial\hat{H}_{H,p}} = \hat{r}^H_{r,p},\\
&\frac{\partial\hat{y}^{\text{PITF}}_{u,t,H,r}}{\partial\hat{r}^U_{r,p}} = \hat{u}_{u,p}, \frac{\partial\hat{y}^{\text{PITF}}_{u,t,H,r}}{\partial\hat{r}^T_{r,p}} = \hat{t}_{t,p}, \frac{\partial\hat{y}^{\text{PITF}}_{u,t,H,r}}{\partial\hat{r}^H_{r,p}} = \hat{H}_{H,p}.
\end{aligned} \tag{16}$$

We call the resulting solution **LPITF**, whose complexity is $O(p)$.

5. EXPERIMENTS

In this section we compare the effectiveness of our algorithms in ranking resources for answering search queries.

5.1 Datasets

We conducted experiments on data collected from two social tagging systems: Flickr and Picasa. They are photo sharing systems that allow users to annotate photos with tags. Since the raw data is very noisy and sparse, we performed some cleaning and preprocessing. First, we removed system-generated tags such as "uploaded:by=instagram". Also, to eliminate outliers, any user, tag, or resource that has appeared in less than 10 records of S are removed. Table 1 shows some statistics of the two datasets after cleaning.

5.2 Metrics

Let Q be a set of queries. For each query $q \in Q$, a ranking algorithm returns a ranked list of N resources. We evaluate the effectiveness of a ranking algorithm by three metrics, namely, Mean Reciprocal Rank (MRR), Precision (P@N), and Normalized Discounted Cumulative Gain (NGCG@N).

Dataset	$\lvert U\rvert$	$\lvert T\rvert$	$\lvert R\rvert$	$\lvert S\rvert$
Flickr	8,199	28,049	26,522	323,527
Picasa	509	1,545	1,041	18,135

Table 1: Data Statistics

The *reciprocal rank* of a ranked list is the multiplicative inverse of the rank of the first relevant resource in the list. The MRR score of an algorithm A is the average reciprocal rank obtained by the ranked lists given by A w.r.t. the query set Q. Formally,

$$MRR = \frac{1}{|Q|}\sum_{i=1}^{|Q|}\frac{1}{rank_i}, \tag{17}$$

where $rank_i$ is the rank of the first relevant resource in the ranked list for the i-th query. The Precision of a ranked list is the fraction of the resources in the list that are relevant to the corresponding query. The Precision score of an algorithm is the average precision of its ranked lists, which is given by:

$$P@N = \frac{1}{|Q|}\sum_{q\in Q}\frac{\sum_{i=1}^{N} rel(q,i)}{N}, \tag{18}$$

where $rel(q,i)$ is an indicator function whose value is 1 if the i^{th}-ranked resource in the list for query q is relevant and 0 otherwise. Finally, the NDCG score [7] is computed as follows:

$$NDCG@N = \frac{1}{|Q|}\sum_{q\in Q}\left(Z_q\sum_{i=1}^{N}\frac{(2^{rel(q,i)}-1)}{\log(i+1)}\right),$$

where Z_q is a normalization factor (see [7] for details).

We invited 10 judges to participate in the experiments. We randomly selected 100 tagging records in S to generate the query set Q. For each selected record (u,t,r,l), we created a query $q = (u,t,l)$. We then applied the various ranking algorithms to return their ranked lists for each query. Each judge was given 10 queries and was asked to judge whether each returned resource was relevant to the corresponding query. The judge was not told which algorithm's ranked list did a resource appear. The results were then used to compute the scores for the various metrics.

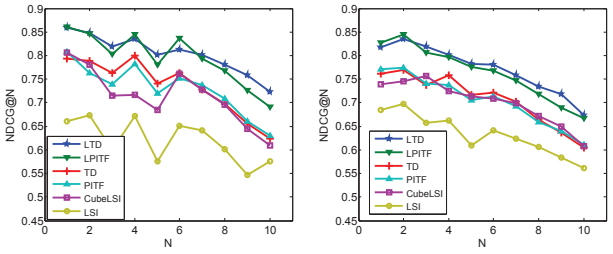
5.3 Algorithms and Settings

Besides LTD and LPITF, we evaluate four other algorithms for ranking resources. They are LSI [3], CubeLSI [3], TD [9] and PITF [11]. Due to space limitations, readers are referred to the references for details. In the experiments, the BPR-OPT parameters are $\alpha = 0.05$ and $\lambda_\theta = 5\cdot 20^{-5}$. For LP's initialization step (Algorithm 1, Line 5), we first perform k-means to cluster the instances in $\mathbf{X}$ into k clusters[2]. This result is used to initialize $\mathcal{U}$ and $\mathbf{\Sigma}$. Also, we set $\varphi_i = \frac{|cluster(i)|}{|\mathbf{X}|}$, where $cluster(i)$ is the i-th cluster of the k-means result. The model parameters of TD, PITF, LTD, and LPITF are all initialized with $\mathcal{N}(0, 0.01)$. Also, we set $p = p_u = p_t = p_h = p_r = 64$.

5.4 Results

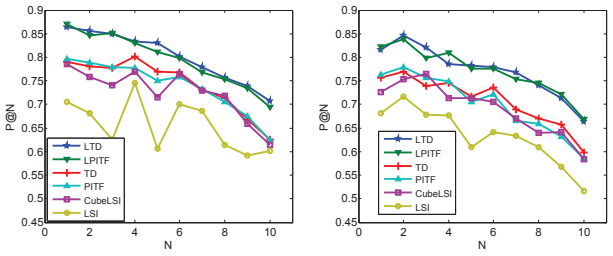
Figures 1-3 show the scores NDCG@N, P@N, and MRR, respectively, of the algorithms when they are applied to the two datasets. From the figures, we see that LSI, which considers only the dimensions tag and resource performs worst. The three algorithms CubeLSI, TD, and PITF, which consider the user dimension as

[2]The l dimension of $\mathbf{x}$ is represented by a longitude dimension and a latitude dimension, both of them are normalized to the [-1,1] range.



(a) Flickr (b) Picasa

Figure 1: NDCG@N



(a) Flickr (b) Picasa

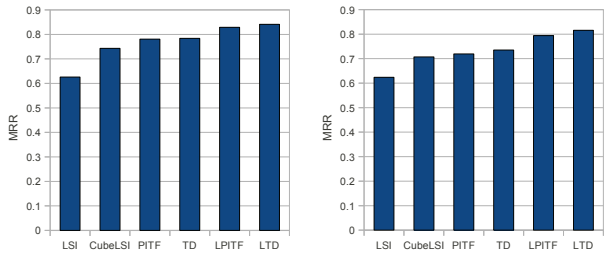
Figure 2: $P@N$

well perform better than LSI. These three algorithms give comparable performances among themselves. By considering the location dimension and extending the 3D tensor to 4D, LTD and LPITF clearly outperform the other four algorithms. The performance of LTD and LPITF is very similar, despite the fact that LPITF uses a simpler factorization model. Given that LPITF is much more efficient than LTD, LPITF is the algorithm of choice for solving the resource ranking problem.

5.5 Parameter Sensitivity

The parameters p, p_u, p_t, p_h, and p_r control the sizes of the core tensor and the factorization matrices of TD and PITF. These values affect the accuracy of our approximation of the score tensor $\hat{Y}$. We have conducted an experiment varying the values of these parameters. We observe that in general, setting them to 64 is enough for $\hat{Y}$ to converge. Therefore, in our performance experiment, we set those parameters to 64.

6. CONCLUSION



(a) Flickr (b) Picasa

Figure 3: MRR

In this paper we studied the impact of location information on resource recommendation. We addressed two problems, namely, the location partitioning problem and the resource ranking problem. We proposed the LP algorithm which computes a region mapping function to solve the location partitioning problem. We also proposed the LTD algorithm and the LPITF algorithm for estimating the score tensor for ranking resources. Through an experimental study on real datasets, we showed that our location-sensitive algorithms outperform other competitors.

Acknowledgements

This project is supported by HKU Research Grant 201011159070.

7. REFERENCES

[1] *SIGMOD 2011, Athens, Greece, June 12-16, 2011*. ACM, 2011.

[2] S. Bao, G.-R. Xue, X. Wu, Y. Yu, B. Fei, and Z. Su. Optimizing web search using social annotations. In *WWW*, pages 501–510. ACM, 2007.

[3] B. Bi, S. D. Lee, B. Kao, and R. Cheng. CubeLSI: An effective and efficient method for searching resources in social tagging systems. In *ICDE*, pages 27–38. IEEE Computer Society, 2011.

[4] X. Cao, G. Cong, C. S. Jensen, and B. C. Ooi. Collective spatial keyword querying. In *SIGMOD Conference* [1], pages 373–384.

[5] S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman. Indexing by latent semantic analysis. *JASIS*, 41(6):391–407, 1990.

[6] P. Heymann, G. Koutrika, and H. Garcia-Molina. Can social bookmarking improve web search? In *WSDM*, pages 195–206. ACM, 2008.

[7] K. Järvelin and J. Kekäläinen. Cumulated gain-based evaluation of IR techniques. *ACM Trans. Inf. Syst.*, 20(4):422–446, 2002.

[8] J. Lu, Y. Lu, and G. Cong. Reverse spatial and textual k nearest neighbor search. In *SIGMOD Conference* [1], pages 349–360.

[9] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme. BPR: Bayesian Personalized Ranking from Implicit Feedback. In *UAI*, pages 452–461. AUAI Press, 2009.

[10] S. Rendle, L. B. Marinho, A. Nanopoulos, and L. Schmidt-Thieme. Learning optimal ranking with tensor factorization for tag recommendation. In *KDD*, pages 727–736. ACM, 2009.

[11] S. Rendle and L. Schmidt-Thieme. Pairwise interaction tensor factorization for personalized tag recommendation. In *WSDM*, pages 81–90. ACM, 2010.