Learning Recommendations in Social Media Systems by Weighting Multiple Relations

Boris Chidlovskii

Xerox Research Centre Europe
6, chemin de Maupertuis, F–38240 Meylan, France

Abstract. We address the problem of item recommendation in social media sharing systems. We adopt a multi-relational framework capable to integrate different entity types available in the social media system and relations between the entities. We then model different recommendation tasks as weighted random walks in the relational graph. The main contribution of the paper is a novel method for learning the optimal contribution of each relation to a given recommendation task, by minimizing a loss function on the training dataset. We report results of the relation weight learning for two common tasks on the Flickr dataset, tag recommendation for images and contact recommendation for users.

1 Introduction

Social media sharing sites like Flickr and YouTube contain billions of image and videos uploaded and annotated by millions of users. Tagging the media objects is proven to be a powerful mechanism capable to improve media sharing and search facilities [19]. Tags play the role of metadata, however they come in a free form reflecting the individual user's choice. Despite this freedom of tag choice, some common usage topics can emerge when people agree on the semantic description of a group of objects.

The wealth of annotated and tagged objects on the social media sharing sites can form a solid base for a reliable tag recommendation [10]. There are two most frequent modes of using the recommendation systems on the media sharing sites. In the bootstrap mode, the recommender system suggests the most relevant tags for newly uploaded media objects by observing the characteristics of the objects as well as their social context. In the query mode, an object is annotated with one or more tags and the system attempts to suggest the user how to extend the tag set. In both modes, the system assists the user in the annotation and help expand the coverage of tags on objects.

Modern social media sharing sites attract users' attention by offering a multitude of valuable services. They organize users activities around entities of different types, such as images, tags, users and groups and relations between them. The tag recommendation for images or videos is one of possible scenarios in such a relational world, other scenarios may concern user contact recommendation, group recommendation, etc. This paper addresses the recommendation tasks in the multi-relational setting and out target is to determine *the optimal combination of the available relations for a given recommendation task*.

In the multi-relational setting, entities of different types get connected and form two types of relations. The first type is the relations between entities of the same type,

D. Gunopulos et al. (Eds.): ECML PKDD 2011, Part I, LNAI 6911, pp. 328–342, 2011.

such as a user-to-user relation of friendship or tag-to-tag relation of co-occurrence. The second types of relations connect entities of two different types, such as user-to-image relation of ownership. All available relations can be represented through the notion of a relational graph.

In such a representation, the recommendation task corresponds to a prediction query on an unfolding of the relational graph. In the following, we will use the Flickr as the example of social media sharing system. We will address two most popular recommendation tasks on the Flickr dataset, tag recommendation for images and contact recommendation for users.

We show that a recommendation task can be modelled as a random walk on the relational graph. To provide a better response to a recommendation query, we compose a *weighted relational graph* where the contribution of each relation is expressed by a non-negative weight. We then show how to efficiently learn the optimal weights of relational random walks from the training dataset.

We adopt the Markov chain model for walking on the relational graph. Any relation can be projected on the recommendation task and thus contribute to the random walk with its proper non-negative weight. The relational random walk is therefore expressed as an additive function on relation weights.

We then develop a technique for learning the weights of relations contributing to the weighted random walk. Stated as the supervised learning problem, it attempts to minimize the loss function over a training set of examples, where function is a metric between estimated and true probability distributions on the recommendation objects. We use the gradient descent methods to solve the optimization problem and to effectively estimate the optimal relation weights. The main advantage is that the gradient and Hessian matrix of loss function can be directly integrated in the random walk algorithm.

The remainder of the paper is organized as follows. A short review of the prior art on the tag recommendation systems and random walks in Section 2. In Section 3 we present the relational graph for entity types and their relations. Section 3 recalls the Markov chain model and presents an weighted extension of random walks on relational graphs. The core contribution on the weight learning of the random walks from available observations is presented in Section 4. Section 5 reports the evaluation results of two recommendation tasks on the Flickr dataset. Section 6 concludes the paper.

2 Prior Art

Existing methods for tag recommendation often target the social network as a source of collective knowledge. Recommender systems based on collective social knowledge have been proven to provide relevant suggestions [9,15,18]. Some of these systems aggregate the annotations used in a large collection of media objects independently of the users that annotate them [18]. In other systems, recommendations can be personalized by using the annotations for the images of a single user [9]. Both approaches come with their advantages and drawbacks. When the recommendations are based on collective knowledge the system can make good recommendations on a broad range of topics, but is likely to miss some recommendations that are particularly relevant in a personal context.

Random walks on weighted graphs is a well established model [8,13]. The random walks have been used for ranking Flickr tags in [14], where the walks are executed on one relation only, such as a image-to-tag graph for the tag recommendation. Similar techniques have used used in [2] for analysing the user-video graph on Youtube site and for providing personalized suggestions.

Many *recommendation algorithms* are inspired by the analysis of the MovieLens collection and Netflix competition; they focus only on using ratings information, while disregarding information about the context of the recommendation process. With the growth of social sites, several methods have been proposed for predicting tags and adding contextual information from social networks to improve the performance of recommendation systems.

One common approach is to address the problem as the *multi-label classification* in a (multi-)relational graph. Techniques of the label propagation on the relational graph is proposed in [17]. It develops an iterative algorithm for the inference and learning for the multi-label and multi-relational classification. Inference is performed iteratively by propagating scores according to the multi-relational structure of the data. The method extends the techniques of collective classification [11] in order to handle multiple relations and to perform multi-label classification in multi-graphs.

The concept of *relational graph* for integrating contextual information is used in [3]. It makes it straightforward to include different types of contextual information. The recommendation algorithm in [3] models the browsing process of a user on a movie database website by taking non-weighted random walks over the relational graph.

The approach closest to our is [7]; it tries to automatically learn ranking function for searching in typed (entity-relation) graphs. User input is in the form of a partial preference order between pairs of nodes, associated with a query. The node pairs are instances in *learning the ranking function*. For each pair the method assigns a label representing the relative relevance. It then trains a classification model with the labelled data and makes use existing classification methodologies like SVM, Boosting, etc. [5].

The pairwise approaches to learning the rank are known for several important limitations [6]. First, the objective of learning is formalized as minimizing errors in classification of node pairs, rather than minimizing errors in node ranking. Second, the training process is computationally costly, as the number of document pairs is very large. Third, the number of generated node pairs varies largely from query to query; this results in training a model biased toward queries with more node pairs.

All three issues are particularly critical in the social networking environment. One alternative to the pairwise approach is based on the *listwise approach*, where node lists and not node pairs are used as instances in learning [6].

What we propose in this paper is a probabilistic method to calculate the *listwise loss function* for the recommendation tasks. We transform both the scores of nodes in the relational graph assigned by a ranking function and the the node annotations by humans into probability distributions. We can then utilize any metric between probability distributions as the loss function. In other words, our approach represents a listwise alternative to the pairwise one in [7] and shows its application the recommendation tasks to social media sites.

Finally, the weight learning for random walks in social networks is recently addressed in [1]. In the link prediction setting, authors try to infer which interactions among existing network members are likely to occur in the near future. They develop an algorithm based on *supervised random walks* that combines the information from the network structure with node and edge level attributes. To guide a random walk on the graph, they formulate a supervised learning task where the goal is to learn a function that assigns strengths to edges in the network such that a random walker is more likely to visit the nodes to which new links will be created in the future.

3 Relational Graph

The relational graph aims at representing all available entity types and relations between them in one uniform way. The graph is given by $G = (E, R)$, where an entity type $e_k \in E$ is represented as a node and relation $r_{kl} \in R$ between entities of types e_k and e_l is represented as a edge between the two nodes.

The relational graph for (a part of) Flickr web site[1] is sketched in Figure 1. Nodes represent five entities types, E={`image`, `user`, `tag`, `group`, `comment`}. Edges represent relations between entities of the same or different types. One example is relation *tagged_with* between `image`and `tag`entities indicating which images are tagged with which tags. Another example is relation *contact* between `user`entities which encodes the list of user contacts. The Flickr example reports one relation between entities of e_k and e_l. In the general case, the model can accommodate any number of relations between any two entity types. Another advantage of the relational graph is its capacity to add any new type of contextual information.

The relational setting reflects the variety of user activities ans services on Flickr and similar sites. Users can upload their images and share them with other users, participate in different interest groups, browse images of other users, comment and tag them, navigate through the image collection by tags, groups, etc.

Each individual relation $r_{kl} \in R$ is expected to be internally homogeneous. In other words, bigger values in the relation tend to have a higher importance. Instead, different relations may have different importance for a given recommendation task. For example, relation *annotated_with*(`image`,`tag`) is expected to be more important For the tag recommendation task than the relation *member*(`user`, `group`). On the other side, for the user contact recommendation, the importance of these two relations may be opposite. In the following, we model the importance of a relation toward a given recommendation task with a non-negative weight.

Every relation $r_{kl} \in R$ is unfolded (instantiated) in the form of matrix $A_{kl} = \{a_{kl}^{ij}\}, k = 1, \ldots, |e_i|, l = 1, \ldots, |e_j|$, where a_{ij}^{kl} indicates the relation between entity $i \in e_k$ and entity $j \in e_l$. Values are binary (for example, in the *tagged_with* relation, $a_{ij} = 1$ if image i is tagged with tag j, 0 otherwise). In the general case, a_{ij} are non-negative real values. Any matrix A_{kl} is non-negatively defined. For the needs of random walks, we assume A_{kl} is a probability transition matrix, which can be obtained by the row normalization.

[1] `http://www.flickr.com`

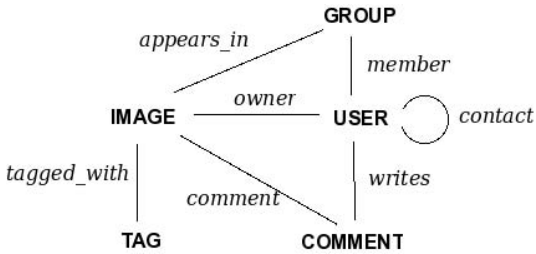

Fig. 1. Relational graph for (a part of) Flickr web site

Random Walks

We combine the relations that induce a probability distribution over entities by learning a Markov chain model, such that its stationary distribution is a good model for a specific prediction task. Constructing Markov chains whose stationary distributions are informative has been used in multiple applications, including the Google PageRank algorithm [16] and HITS-like algorithms [4].

A Markov chain over a set of states S is specified by an initial distribution P_0 over S, and a set of state transition probabilities $P(S_t|S_{t-1})$. A Markov chain defines a distribution over sequences of states, via a generative process in which the initial state S_0 is first sampled according to distribution P_0, and then states S_t (for $t = 1, 2, \ldots$) are sampled according to the transition probabilities. The stationary distribution of the Markov chain is given by $\pi(s) = \lim_\infty P(S_t = s)$, if the limit exists.

To ensure that the Markov chain has a unique stationary distribution, the process can be reset with a probability $\alpha > 0$ according to the initial state distribution P_0. In practice this prevents the chain from getting stuck in nodes having no transitions and small loops. Having the Markov chain $S_0, S_1, \ldots$ with the initial state S_0 distributed according to P_0, state transitions given by P and resetting probability α, it is straightforward to express the stationary distribution π as follows:

$$\pi = \alpha \sum_{t=0}^{\infty} (1-\alpha)^t P_0 P^t. \tag{1}$$

Equation (1) can be used to efficiently compute π. Because terms corresponding to large t have very little weight $(1-\alpha)^t$, when computing π, this sequence may be truncated after the first few (on the order $1/\alpha$) terms without incurring significant error.

Weighted Relational Random Walks

In this section we extend the Markov chain model to the unfolded relational graph. Assume the relational graph includes b entity types, $e_1, \ldots, e_b$. The total number of entities is denoted $N = \sum_{k=1}^{b} |e_k|$. The *unfolded relational graph* is composed of b^2 blocks, one block for each (e_k, e_l) pair, $k, l = 1, \ldots, b$. Available relations fill up some blocks, other blocks can be left empty or filled up with composed relations using

the relation transitivity rule $A_{kl} = A_{km}A_{ml}$, where A_{km} and A_{ml} are basic or other composed relations. Note that there might exist several ways to compose a relation; a particular choice often depends on the recommendation task.

In the Flickr relational graph (Figure 1), there are seven basic relations, which fill up the corresponding blocks and can be used to compose other relations. The *tag co-occurrence* relationship is an example of composed relation. If matrix A_{IT} describes relation *tagged_with* (`image`,`tag`), the tag co-occurrence matrix can be obtained by $A_{TT} = A'_{IT}A_{IT}$. Higher values in A_{TT} indicate that more images are tagged with a given tag pair.

When a random walk moves through relation $r_{kl} \in R$ between entities of types e_k and e_l, contribution of r_{ij} to the walk is expressed by a non-negative weight w_{kl}. The random walk is therefore performed over matrix A which is a weighted sum over relations, $A = \sum_{kl} w_{kl}A_{kl}$. Matrix A is ensured to be the probability transition matrix if $\sum_j w_{kl} = 1$, $l = 1, \ldots, b$. Also, $\pi(s)_j$ denotes a projection of the stationary distribution π on the entity type j.

To initiate the random walk, the initial distribution P_0 is composed of b vectors $\delta_j, j = 1, \ldots, b$, with all elements relevant to the query. For the tag recommendation task, we compose three vectors δ_I, δ_U, and δ_T, for images, users and tags. When recommending tags for image i, i-th element in the image vector is 1 with all other elements $\delta_{Ij}, j \not= i$ set to 0. Similarly, in the user vector δ_U only the owner u of image i is set to 1. The default choice for the tag vector δ_t is **1** vector. We however prefer to add a bias on the *user tag vocabulary* and preferences. We set t-th of vector δ_T the log value of frequencies of using t by user u in the collection, $\delta_{Tt} = log(A_{UT}(u,t) + 1)$. Then the initial distribution P_0 is defined as normalization of the composed vector $(\delta_1, \delta_2, \ldots, \delta_b)$.

If weights w_{kl} are known or recommended by an expert, equation (1) can be used for estimating the stationary distribution π and its projection π_j. If the weights are unknown a priori, we propose a method which determines such values for weights w_{kl} which minimize a loss function on the training set.

4 Weight Learning

To learn relation weights in the random walk, we approximate the stationary distribution π with the truncated version and look for an such instantiation of the Markov model where weights minimize a prediction error on a training set T. We express the optimization problem on weights w_{kl} as minimization of loss function on the training set.

The weighted random walk defined by a Markov chain query produces a probability distribution. Nodes having more links (with higher weights) with query nodes will accumulate more probability than nodes having less links and of lower weights.

Probability Estimation Loss

We define a scoring function H that assigned a [0,1] value to an entity of type e_j. The task is to learn the function H from a set of known relations between entities. The function H estimates the probability p for a given object i. Let y denote the true

probability of i and let p its estimation by H. The price we pay when predicting p in place of y is defined as a loss function $l(y, p)$. We use the *square loss*[2] between y and p in the following form:

$$l_{sq}(y, p) = y(1-p)^2 + (1-y)p^2. \tag{2}$$

Note that its first and second partial derivatives in p are $\frac{\partial}{\partial p} l_{sq}(y, p) = 2(p - y)$ and $\frac{\partial^2}{\partial^2 p} l_{sq}(y, p) = 2$, respectively.

Multi-label Square Loss

Without loss of generality, in the following sections we assume to cope with the tag recommendation. Assume the tag set includes L tags. For a given image, let Y_B denote a binary vector $Y_B = (y_1, \ldots, y_L)$ where y_i is 1 if the image is tagged with tag i, 0 otherwise, $i = 1, \ldots, L$. The probability distribution over the tag set is $Y = (y_1, \ldots, y_n)$ where y_i is 0 or $1/|Y_B|$, $i = 1, \ldots, L$.

Let P denote an estimated tag probability distribution, $P = (p_1, \ldots, p_L)$, where $\sum_{i=1}^{L} p_i = 1$. To measure the loss of using estimated distribution P in the place of true distribution Y, we use a loss function which is symmetric in y and p and equals 0 only if $y = p$. The best candidate is the multi-label square function defined as follows

$$L_{sq}(Y, P) = (l_{sq}(y_i, p_i)), i = 1, \ldots, L. \tag{3}$$

For the square loss function L_{sq}, we take its gradient as follows

$$\nabla L_{sq}(Y, P) = \left(\frac{\partial}{\partial p_i} l_{sq}(y_i, p_i) \right)_{i=1,\ldots,L} = 2(y_i - p_i),$$

and similarly we have

$$\frac{\partial}{\partial P} \nabla L_{sq}(Y, P) = \left(\frac{\partial^2}{\partial^2 p_i} l_{sq}(y_i, p_i) \right)_{i=1,\ldots,L} = 2\mathbf{1}.$$

If we dispose a training set T of images with the tag probability distribution Y, we try to define such a scoring function H which minimizes the empirical loss over T. It is defined as follows

$$Loss(H) = \frac{1}{|T|} \sum_{j \in T} L_{sq}(Y_j, P_j), \tag{4}$$

where Y_j is the true probability vector for image j and P_j is the prediction probability distribution.

The weighted sum of composed of b distinct entity types $A = \sum_{kl}^{b} w_{kl} A_{kl}$. Bigger values of w_{kl} indicate higher importance of relation between entities e_k and e_l to the task. We assume that matrix A_{kl} for relation r_{kl} is normalized with each raw forming a state transition distribution. The mixture matrix A satisfies the same condition if the

[2] Other loss functions will be equally tested in the evaluation section.

constraint $\sum_j w_{kl} = 1, w_{kl} \geq 0$ holds. The matrix A is however is not required to be symmetric, so $w_{kl} \neq w_{lk}$ in the general case. Thus we obtain the following optimization problem:

$$\begin{gathered} min_{w_{kl}} Loss(H) \\ s.t. \\ 0 \leq w_{kl} \leq 1 \\ \sum_l w_{kl} = 1, k = 1, \ldots, b. \end{gathered} \tag{5}$$

The constrained optimization problem (5) is transformed into unconstrained one by introducing variables v_{kl}, $k, l = 1, \ldots, b$ and representing $w_{kl} = e^{v_{kl}} / \sum_m e^{v_{lm}}$. The problem constrained on w_{kl} becomes unconstrained on v_{kl}.[3]

We use the L-BFGS method to solve numerically the problem. L-BFGS algorithm is a member of the broad family of quasi-Newton optimization methods. They approximate the well-known Newton's method, a class of hill-climbing optimization techniques that seeks a stationary point of a (twice continuously differentiable) function. The Broyden-Fletcher-Goldfarb-Shanno (BFGS) method is one of the most popular members of quasi-Newton methods.

The L-BFGS uses a limited memory variation of the BFGS to approximate the inverse Hessian matrix. Unlike the original BFGS method which stores a dense $n \times n$ approximation, L-BFGS stores only a few vectors that represent the approximation implicitly. We use the open source implementation of L-BFGS routine available in Python via SciPy library. It preforms an iterative scheme along all w_{kl} dimensions, including the gradient $\nabla Loss(H)$ and the inverse of the Hessian matrix H for $Loss(H)$ at W, where $W = \{w_{kl}\}, k, l = 1, \ldots, b$. This produces an iterative sequence of approximated solutions $W_0, W_1, \ldots$ which converges to a (generally local) optimum point.

In order to deploy the quasi-Newton methods for solving the optimization problem (5), we first obtain the derivatives of the loss function with respect to variables w_{kl}:

$$\frac{\partial Loss(H)}{\partial w_{kl}} = \frac{1}{|T|} \sum_{j \in T} \nabla L_{sq}(Y_j, P_j) \frac{\partial P_j}{\partial w_{kl}}, \tag{6}$$

where $P_j = \alpha \sum_{t=1}^{k} (1-\alpha)^t P_0^j A^t$ and P_0^t is the initial probability distribution for image j.

The power series A^t, $t = 1, 2, \ldots$ are the only terms in P_j depending on w_{kl}. To compute the derivative of a composite function, we use the chain rule for matrices. We obtain the recursive for the first derivatives at step t

$$\frac{\partial A^t}{\partial w_{kl}} = \frac{\partial (A^{t-1} A)}{\partial w_{kl}} = \frac{\partial A^{t-1}}{\partial w_{kl}} A + A^{t-1} A_{kl}. \tag{7}$$

Algorithm 1 presents a meta-code for the loss function $Loss(H)$ and its gradient $\nabla Loss(H)$ needed for solving the optimization problem (5) with quasi-Newton method.

[3] A regularization term on v_{kl} can be added to the objective function.

Algorithm 1. Loss function and its gradient

Require: Training dataset T, the restarting probability α
Require: Relation matrices A_{kl}, weights $(w_{kl}), k, l = 1, \ldots, b$
Ensure: Loss function value $Loss(H)$ and the gradient $\nabla Loss(H)$
1: $A = \sum_{kl} w_{kl} A_{kl}$
2: **for** $j = 1$ to $|T|$ **do**
3: Set the initial distribution P_0 for object j
4: **for** $t = 1$ until convergence **do**
5: $P_j^t = \alpha P_0 + (1 - \alpha) P_j^{t-1} A$
6: **for all** w_{kl} **do**
7: Update $\frac{\partial P_j^t}{\partial w_{kl}}$ using (7)
8: **end for**
9: **end for**
10: Set $L(Y_j, P_j)$ using (4)
11: $Loss(H) = Loss(H) + L(Y_j, P_j)$
12: **for all** w_{kl} **do**
13: Set $\frac{\partial L(Y_j, P_j)}{\partial w_{kl}}$ using (6)
14: $\frac{\partial Loss(H)}{\partial w_{kl}} = \frac{\partial Loss(H)}{\partial w_{kl}} + \frac{\partial L(Y_j, P_j)}{\partial w_{kl}}$
15: **end for**
16: **end for**
17: Return $Loss(H)$ and the gradient $\nabla Loss(H)$ at (w_{kl}), $k, l = 1, \ldots, b$.

Hessian Matrix

The Hessian matrix H for the loss function may help the quasi-Newton method to faster converge to an optimum point. The matrix requires the mixed derivatives for weights w_{kl} and $w_{k'l'}$:

$$\frac{\partial Loss(H)}{\partial w_{kl} \partial w_{k'l'}} = \frac{1}{|T|} \sum_{j \in T} \frac{\partial}{\partial P_j} \nabla L_{sq}(Y_j, P_j) \frac{\partial^2 P_j}{\partial w_{kl} \partial w_{k'l'}}, \tag{8}$$

The second derivatives in the Hessian matrix H can be developed by using the chain rule for matrices, similarly to (7). We obtain a recursive formula where the values at iteration t depend on the function values and its gradient on the previous step $t - 1$ of the random walk.

$$\begin{aligned} &\frac{\partial^2 A^t}{\partial w_{kl} \partial w_{k'l'}} = \\ &\frac{\partial}{\partial w_{k'l'}} \left(\frac{\partial A^{t-1}}{\partial w_{kl}} A + A^{t-1} A_{kl} \right) = \\ &\frac{\partial^2 A^{t-1}}{\partial w_{kl} \partial w_{k'l'}} A + \frac{\partial A^{t-1}}{\partial w_{kl}} A_{k'l'} + \frac{\partial A^{t-1}}{\partial w_{k'l'}} A_{kl}. \end{aligned} \tag{9}$$

Algorithm 1 can be extended with the evaluation of the Hessian matrix in the straightforward way. The extension requires to expand lines 6 to 8 of Algorithm 1 with the evaluation of second derivatives $\frac{\partial^2 P_j}{\partial w_{kl} \partial w_{k'l'}}$ for a given object j using the rule (9). Then lines 12 to 15 should be expanded to get the Hessian matrix for the entire set T using formula (8).

The power iteration for the gradient and Hessian can be prohibitive for large full matrices. Luckily, matrices A_{kl} are all sparse and the matrix product is proportional to the number of non-zero elements in the matrix. This ensures a relative speed when performing truncated random walks on the relational graph.

5 Evaluation

In this section we describe the real dataset used in the experiments, the evaluation setting and the results of experiments for two different recommendation tasks.

Flickr dataset. In all evaluations, we use a set of Flickr data which has been downloaded from Flickr site with the help of social-network connectors (Flickr API) [12]. The API give an access to entities and relational data, including users, groups of interests, images with associated comments and tags.

We test the method of relation weight learning for the random walks described in Section 4 on three entity types, E={`image`, `tag`, `user`}. Three core relations associated with the three types are image-to-tag relation R_{IT}= *tagged_with* (`image`,`tag`), user-to-image relation R_{UI}= *owner* (`user`,`image`) and user-to-user relation R_{UU}= *contact* (`user`, `user`).

We use a fragment of the Flickr dataset with 100,000 images; these images are owned by 1,951 users and annotated with 127,182 different tags (113,426 tags after normalization). Matrices of the three core relations are sparse, their elements follow the power low distribution which is very common in the social networks. In the image-to-tag matrix, an image has between 1 and 132 tags, with the average of 5.65 tags per image. The user-to-image matrix contains 1 to 384 images per user, the average number is 27.32 images. The number of contacts in the user-to-user matrix is between 0 and 43, the average is 1.24 contacts.[4]

We run a series of experiments on the dataset where two tasks are *tag recommendation* for images and *contact recommendation* for users. In either task, we use the core relations to compose other relations. The way the composed relations are generated depends on the recommendation task:

Tag recommendation: the image-to-image matrix is composed as $A_{II} = A_{IT}A'_{IT}$. Other composed relations are tag-to-tag $A_{TT} = A_{IT}A'_{IT}$ and user-to-tag $A_{UT} = A_{UI}A_{IT}$, and their inversion.

User contact recommendation: The image-to-image matrix is composed as $A_{II} = A_{UI}A'_{UI}$ and user-to-tag matrix is given by $A_{UT} = A_{UI}A_{IT}$.

An all cases, the matrix A is block-wise ; the optimization problem (5) is solved for b^2 weights w_{kl}.

The image tag recommendation runs either in the bootstrap or query mode. In *bootstrap mode*, the task is to predict tags for a newly uploaded image. In *query mode*, an image may have some tags and the task is to extend them. In both modes, we measure the performance of predicting the top 5 and $|size|$ tags where the number $|size|$ of tags vary from image to image but is known in advance (and equals to the test tag set). The user contact recommendation task has been tested in the query mode only.

[4] The multi-relational Flick dataset is available from authors upon request.

To evaluate the correctness of different methods we use conventional precision, recall and F1 evaluation metrics adopted for the multi-label classification case. Let Y_j and P_j denote the true and recommended tag vectors for image j in the test set, respectively. Then, precision and recall metrics for the test set are defined as $Pr = \sum_j \frac{|Y_j \cup P_j|}{|Y_j|}$ and $Re = \sum_j \frac{|Y_j \cup P_j|}{|P_j|}$, respectively. $F1$ score is then defined as $2\frac{Pr \cdot Re}{Pr+Re}$.

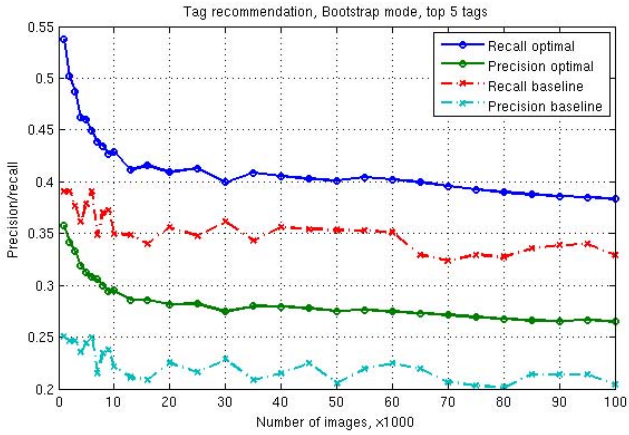

Fig. 2. Image tag recommendation in the bootstrap mode for top 5 tags

Due to non-availability of [7], we compare the performance of relation weight learning to the *baseline* methods, which correspond to an unweighted combination of relations. We test two instantiations of the unweighted combinations. In one instance, we simulate the mono-relation setting, where the core relation is only used, weights of all other relations are set to 0. For the image tag recommendation, the core relation is given by the image-to-tag matrix A_{IT}. For the user contact recommendation, the core relation is the user-to-user matrix A_{UU}. In another instance, all available relations (both core and composed ones) are combined with the equal weights $w_{kl} = 1/b$. In the following, we report the best of the two baseline methods and observe the gain of the weight learning method over the unweighted ones.

In all experiments, the average values are reported over 5 independent runs, the resetting probability α is 0.05, the loss function (2) is used. We have also tried other loss functions satisfying the symmetry condition. We have tested three following versions: the absolute loss $l_{abs}(y,p) = |y-p|$, the exponential loss $l_{exp}(y,p) = exp^{|y-p|} - 1$ and the huber loss $huber(y,p) = \begin{cases} \frac{|y-p|^2}{2}, & \text{if} |y-p| \leq 0.5 \\ |y-p| - \frac{1}{2}, & \text{otherwise,} \end{cases}$. All three are differentiable but their derivatives are not continuous. This makes them a poor alternative to the square loss which is twice continuously differentiable. Tests with these three losses unveil their under-performance, where only the huber loss can hardly beat the baseline method.

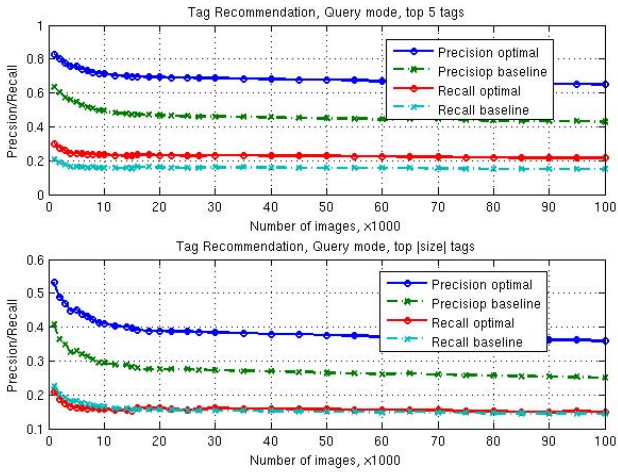

Fig. 3. Recall and precision values in the query mode: a) Top 5 tags; b) Top $|size|$ tags

Image Tag Recommendation

Bootstrap mode. Figure 2 reports the recall and precision values for the image tag recommendation in the bootstrap mode, where the number of images vary from 1,000 to 100,000. The test is performed in 5 folds, with the top 5 predicted tags being compared to the true tag set. In each run, 80% of images are randomly selected for training and remaining 20% are used for testing. Both recall and precision decrease with the growth of the data set, due to the growing number of tags to predict. The learning the relation weights permits to gain 4-17% in recall and 5-11% in precision over the best unweighted schema. We note that the gain resists well to the growth of the dataset.

Query mode. Figure 3.a reports the recall and precision values for the query tag recommendation, where the size of the image set vary from 1K to 100K. In this evaluation, 50% of tags are randomly selected to form the query for a given image, remaining 50% tags are used for testing, where the 5 top predicted tags are compared to the true tag set. The gain in precision and recall over the best baseline method is 17% and 4% respectively.

Figure 3.b reports precision/recall values for the same setting, where the number of predicted tags is not 5 but equals to the test tag set which is 50% of tags for any image.

User Contact Recommendation

The second task for the method of relation weight learning is the user contact recommendation. Similarly to the previous runs, 50% of a user's contacts are randomly selected to form a query, remaining 50% contacts are used for testing. Figure 4 reports precision and recall values for the top 5 recommended contacts; the number of users vary from 100 to 1900. One can observe that the weight learning permit to gain up to 10% in precision while the gain in recall remains limited to 2%.

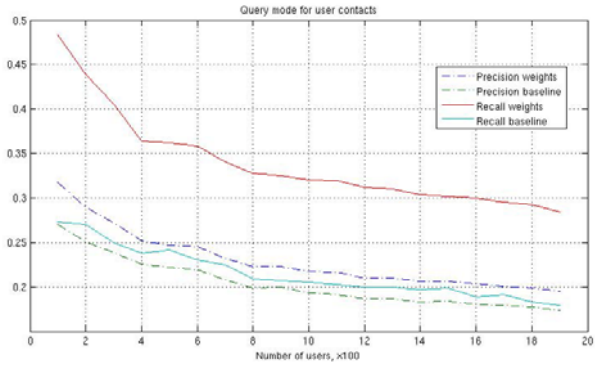

Fig. 4. User contact recommendation in the query mode: precision and recall for the top 5 contacts

Resetting Coefficient

Figure 5 shows the impact of resetting coefficient α on the performance of the weight learning. It reports the relative precision values for the tag recommendation in the bootstrap mode. Three cases for 1K, 50K and 100K images are presented. As the figure shows, there exists a convenient range of values α between 0.05 and 0.2 where the maximum precision is achieved in all cases.

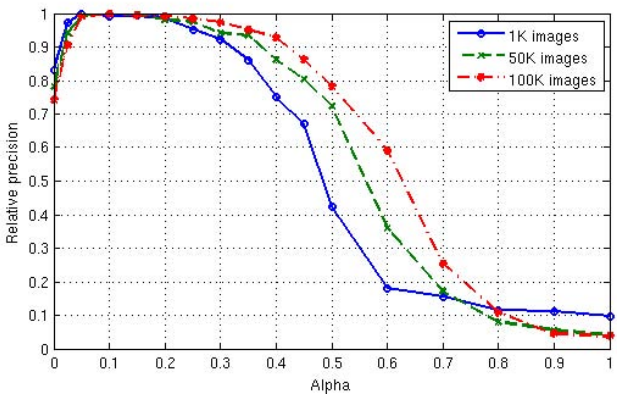

Fig. 5. Tag recommendation for different values of the resetting coefficient

Number of Iterations

Another evaluation item concerns the convergence of the random walks in Equation (1). Figure 6 shows the impact of truncating after varying number of iterations on the performance of weight learning. It reports the precision and recall values for the tag recommendation in the bootstrap mode. Two cases of 1,000 and 50,000 images are presented, when the random walk is truncated after 1,2,...,15 iterations. As the figure suggests both

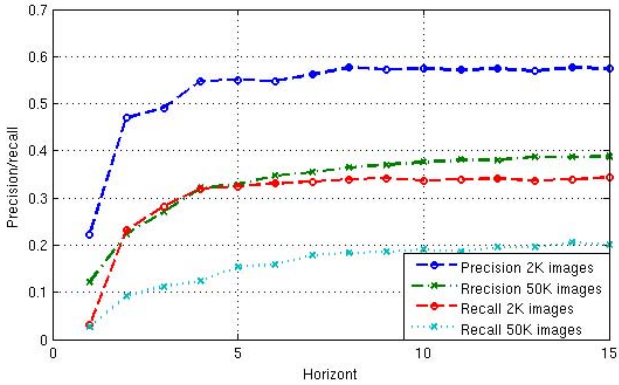

Fig. 6. Query-based prediction of user contacts

precision and recall achieve their top values after 5 to 7 iterations in the first case and after 10 to 12 iterations in the second case.

Finally, we tracked the evaluation of Hessian matrix (9) in addition to the gradient in Algorithm 1. For small datasets (less 5,000 images), using the Hessian helps to faster converge to the local optimum, with the saving of 25% for evaluation on 1000 images. The situation changes drastically for large datasets where evaluation of Hessian matrices become an important handicap. For this reason, Hessians were excluded from the valuation for 10,000 images and more.

6 Conclusion

We presented the relational graph representation for multiple entity types and relations available in a social media system. We implemented the Markov chain model and presented its weighted extension to the relational graph. We has shown how to learn the relation weights by minimizing the loss between predicted and observed probability distributions. We reported the evaluation results of the image tag and user contact recommendation tasks on the Flickr dataset. The results of experiments confirm that the relation weights learned from the training set provide a constant gain over the unweighted methods.

References

1. Backstrom, L., Lescovec, J.: Supervised random walks: Predicting and recommending links in social networks. In: Proc. ACM WSDM (2011)
2. Baluja, S., Seth, R., Sivakumar, Y.J., Yagnik, J., Kumar, S., Ravichandran, D., Aly, M.: Video suggestion and discovery for youtube: Taking random walks through the view graph. In: Proc. WWW 2008, pp. 895–904 (2008)
3. Bogers, T.: Movie recommendation using random walks over the contextual graph. In: Proc. 2nd Workshop on Context-Aware Recommender Systems, CARS (2010)

4. Borodin, A., Roberts, G.O., Rosenthal, J.S., Tsaparas, P.: Link analysis ranking: algorithms, theory, and experiments. ACM Trans. Internet Technol. 5(1), 231–297 (2005)
5. Burges, C.J.C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N., Hullender, G.N.: Learning to rank using gradient descent. In: Proc. ICML, pp. 89–96 (2005)
6. Cao, Z., Qin, T., Liu, T.-Y., Tsai, M.-F., Li, H.: Learning to rank: from pairwise approach to listwise approach. In: Proc. ICML, pp. 129–136 (2007)
7. Chakrabarti, S., Agarwal, A.: Learning parameters in entity relationship graphs from ranking preferences. In: Proc. PKDD, pp. 91–102 (2006)
8. Coppersmith, D., Doyle, P., Raghavan, P., Snir, M.: Random walks on weighted graphs and applications to on-line algorithms. J. ACM 40(3), 421–453 (1993)
9. Garg, N., Weber, I.: Personalized, interactive tag recommendation for flickr. In: Proc. ACM RecSys 2008, pp. 67–74 (2008)
10. Gupta, M., Li, R., Yin, Z., Han, J.: Survey on social tagging techniques. ACM SIGKDD Explorations Newsletter 12, 58–72 (2010)
11. Jensen, D., Neville, J., Gallagher, B.: Why collective inference improves relational classification. In: Proc. ACM KDD 2004, pp. 593–598 (2004)
12. Ko, M.N., Cheek, G.P., Shehab, M., Sandhu, R.: Social-networks connect services. IEEE Computer 43(8), 37–43 (2010)
13. Toutanova, K., Manning, C.D., Ng, A.Y.: Learning random walk models for inducing word dependency distributions. In: Proc. ICML (2004)
14. Liu, D., Hua, X.-S., Yang, L., Wang, M., Zhang, H.-J.: Tag ranking. In: Proc. WWW 2009, pp. 351–360 (2009)
15. Overell, S., Sigurbjörnsson, B., van Zwol, R.: Classifying tags using open content resources. In: Proc. ACM WSDM 2009, pp. 64–73 (2009)
16. Page, L., Brin, S., Motwani, R., Winograd, T.: The PageRank citation ranking: Bringing order to the Web (1998)
17. Peters, S., Denoyer, L., Gallinari, P.: Iterative annotation of multi-relational social networks. In: Proc. ASONAM 2010, pp. 96–103 (2010)
18. Sigurbjörnsson, B., van Zwol, R.: Flickr tag recommendation based on collective knowledge. In: Proc. WWW 2008, pp. 327–336 (2008)
19. Tian, Y., Srivastava, J., Huang, T., Contractor, N.: Social multimedia computing. IEEE Computer 43, 27–36 (2010)