Recommendation over a Heterogeneous Social Network

Jing Zhang[#1], Jie Tang[#2], Bangyong Liang[*3], Zi Yang[#4], Sijie Wang[#5], Jingjing Zuo[#6], and Juanzi Li[#7]

[#]*Department of Computer Science and Technology, Tsinghua University*
[1,2,6,7]*{zhangjing, tangjie, zjj, ljz}@keg.cs.tsinghua.edu.cn*
[4]*yangtseyangtse@163.com,* [5]*thuarmymen@gmail.com*
[*]*NEC Laboratories, China*
[3]*liangbangyong@research.nec.com.cn*

Abstract

With the Web content having been changed from homogeneity to heterogeneity, the recommendation becomes a more challenging issue. In this paper, we have investigated the recommendation problem on a general heterogeneous Web social network. We categorize the recommendation needs on it into two main scenarios: recommendation when a person is doing a search and recommendation when the person is browsing the information. We formalize the recommendation as a ranking problem over the heterogeneous network. Moreover, we propose using a random walk model to simultaneously ranking different types of objects and propose a pair-wise learning algorithm to learn the weight of each type of relationship in the model. Experimental results on two real-world data sets show that improvements can be obtained by comparing with the baseline methods.

1. Introduction

Recommendation is an effective way to reduce the cost for finding information and also a powerful way to attract customers. It has been widely used in many e-commence applications, e.g., Amazon.com, CDNOW.com, eBay.com, Reel.com, and so on.

Recently, many methods have been proposed for recommendation, for example, content-based filtering [2], collaborative filtering [8], clustering model [4], classification model [3], graph model [1], and association rule approach [10]. The proposed approaches have been applied to the traditional Web applications, which usually need recommend only one type of information (e.g., Amazon recommends books, news.baidu.com recommends news, and movielens.com recommends movies). Nowadays, social networks consisting of different types of information become popular (e.g., a common social network indicated by Fig. 1 is composed of users, categories, resources, tags, and complex relationships between them). The flourish of the heterogeneous social networks provides a new environment for validating the recommendation methods, at the same time brings new challenges, e.g., how to recommend the heterogeneous information simultaneously?

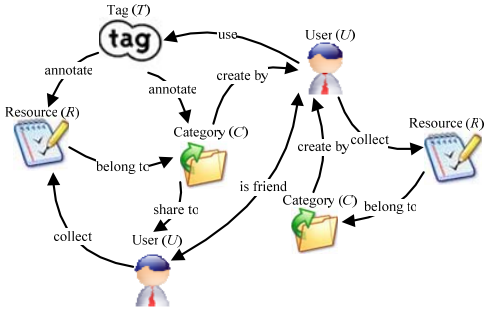

Figure 1. **A common social network**

In this paper, we intend to conduct a complementary study for the recommendation problem in the heterogeneous Web social networks. We will mainly focus on non-personalized recommendation, as it can be extended to the personalized setting by combing the user behavior logs and user preference profiles. Hence, the major problems addressed here are: 1) How to formalize the recommendation problem in a heterogeneous Web social network? 2) How to determine the strategies for recommending different types of objects simultaneously in different scenarios, as the requirements for different scenarios might be different also?

Specifically, for a heterogeneous Web social network, we categorize the recommendation needs into two main scenarios and design corresponding strategies: (a) recommendation of different types of objects when a person searches for one type of object and (b) recommendation of different types of objects when the person is browsing one specific object. We formalize the recommendation as that of ranking over a heterogeneous graph and propose using a random

walk model for ranking. For specifying the weights of different relationships in the random walk model, we propose a pair-wise learning algorithm. We conducted experiments on two real-world systems: an online information sharing system (Powazi, www.powazi.com) and an academic social network system (Arnetminer, www.arnetminer.org). Both experimental results show that the proposed method can outperform the baseline methods of using language model and PageRank without learning the weights of relationships.

Advantages of the proposed approach for recommendation include: 1) it is not necessary to manually specify the weights for different types of relationships; 2) the approach can recommend different types of objects simultaneously; 3) the approach has been empirically verified on two real-world systems.

The rest of the paper is organized as follows. In Section 2, we formalize the recommendation problem in a heterogeneous network. In Section 3, we explain our approach for recommendation. In Section 4, we present our experiments and in Section 5, we review the related works. Finally, we conclude the paper.

2. Problem setting

The structure of a Web social network varies largely depending on the application itself. This work intends to address the recommendation problem in a general setting, which thus can be easily extended to deal with different specific scenarios.

The notations used in the paper can be summarized as follows. Assume there are n objects with types $\{C, U, R, T\}$, where C indicates category, U indicates user, R indicates resource, and T indicates tag. In particular, there are n_R resources r_k (k=1,...,n_R) collected by n_U users u_j (j=1,..., n_U). Moreover, these resources belong to n_C categories c_i (i=1, ... , n_C) and are annotated by n_T tags t_l (l=1,…, n_T). (Here, a resource can be referred to as a Web page, a musical item, or a movie. The notion of category and tag is very popular in Web 2.0 applications.)

The relationships in "category-user-resource-tag" form a directed heterogeneous graph (as shown in Fig. 2). Specifically, we define the graph as $G=(V_C \cup V_U \cup V_R \cup V_T, E_{UC} \cup E_{CU} \cup E_{UR} \cup E_{RU} \cup E_{UT} \cup E_{TU} \cup E_{UU} \cup E_{CT} \cup E_{TC} \cup E_{CR} \cup E_{RC} \cup E_{RT} \cup E_{TR})$, where V_C is a set of categories, V_U is a set of users, V_R is a set of resources, and V_T is a set of tags. The relationship between a category and a user is recorded in E_{CU}, likewise for the others. We consider recommendation in a direct graph, thus each undirected edge in the graph is represented as two directed edges, i.e. $\{c_j, u_i\}=(c_j, u_i) \cup (u_i, c_j)$.

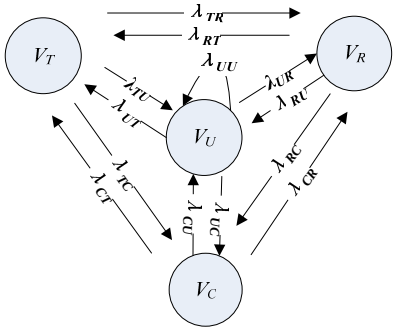

Figure 2. **A formalized heterogeneous graph**

Further, we define the transition probabilities λ between different types of nodes (cf. Fig. 2). The transition probability λ has an intuitive explanation. It can be viewed as the probability of a user jumps/clicks the targeted type of object when she/he is viewing the source type of object or it can be viewed as the weight of the relationship between two types of objects. According to the transition probability theory, we need:

$$\lambda_{UT}+\lambda_{UC}+\lambda_{UR}+\lambda_{UU}=1,\quad \lambda_{TR}+\lambda_{TU}+\lambda_{TC}=1,$$
$$\lambda_{CT}+\lambda_{CU}+\lambda_{CR}=1,\ \lambda_{RT}+\lambda_{RU}+\lambda_{RC}=1,\lambda_{XY}>0 \qquad (1)$$

In real world social networks, the relationships types would be rather complex. The difficulty lies in how to define λ for different types of relationships. We will describe the problem in Section 3.2.

Based on the scenario that a person is using the Web, the recommendation needs can be classified into two main categories:

1) Browsing: do recommendations when a person is browsing one object. The problem is that we need to 'understand' what the person is interested in when she/he is browsing the object. A simple method might be finding the similar objects based on the content. However, in the Web social network, we need also consider the relationships. For example, the relationships between users and the current browsed resource.
2) Search: do recommendation of different types of objects when a person searches for one type of object by a query. Of course, we can simply find similar objects of other types based on the content. The challenge is still how to make use of the relationships between different types of objects.

3. Our approach

Our approach is aimed at dealing with the recommendation problems of the two categories. The approach mainly consists of three steps:

1) Global importance estimation. We use a random walk model to estimate the global important score for each object. The process is

similar as that in PageRank [14]. The difference is that we are addressing a heterogeneous graph (with different types of objects) while PageRank is concerned with a homogeneous graph. We need to consider the weight of different types of relationships.

2) Relevance estimation. We calculate the relevance score of each object with the current scenario that the person is using the Web. When a person is conducting a search, the relevance score is calculated based on the query. When a user is browsing an object, the relevance score is calculated based on the key terms extracted from the profile of the object (e.g., the profile of a resource can be defined as the concatenation of its title, description, and content).
3) Recommendation. We propose several strategies for the recommendations under different scenarios, which can be viewed as the combination of the above two steps. Section 3.2 will give the details of the strategies.

For the first step, we formalize the problem in a random walk model and propose a pair-wise learning algorithm for automatically adjusting the weights of different types of relationships. For the second step, we employ the language model to estimate the relevance. For the third step, we further categorize the two recommendation scenarios into several sub problems and propose different strategies for each of them. We especially focus on the first and the third steps.

3.1. Global importance estimation

This section first formalizes the problem using a random walk model and then presents a pair-wise learning algorithm to estimate parameters in the model.

3.1.1. Random walk over a heterogeneous graph. Based on the random walk theory, the transition graph (cf. Fig. 2) formalizes a random surfer's behaviour as: when a random surfer is at a user node u_i, then he/she will have λ_{UT} probability to browse (also called "jump to") tags used or created by the current user, or have λ_{UC} probability to browse related categories, or have λ_{UR} probability to browse related resources, or have λ_{UU} probability to browse the friends of the current user.

We consider random walk over the heterogeneous graph by transforming the entire network into a transition matrix, denoted as **M**. Each element m_{ij} is denoted as a probability walking from node i to node j. For example, let node i be $u_i \in V_U$ and node j be $r_j \in V_R$:

$$m_{u_i r_j} = \frac{1}{Out\_Degree(u_i \to V_R)} \quad (2)$$

where Out_Degree($u_i \to V_R$) is the number of directed edges from u_i to nodes in V_R (in essence, indicates the number of resources that user u_i is collected). Similarly, we can easily define all the transition probabilities between different types of nodes in the network.

In addition, we need consider λ. The intuition is that when a random surfer is at node i with type X, she/he has a probability of λ_{XY} to jump to node set with type Y and then has a probability of $\lambda_{XY}*m_{ij}$ to jump to a particular node j with type Y. Thus the transition probability from u_i to r_j can be defined as:

$$P(r_j \mid u_i) = \lambda_{UR} m_{u_i r_j} \quad (3)$$

DEFINITION 1. The importance score vector is a stationary distribution of the matrix **M**:

$$\mathbf{s} = \mathbf{A}\mathbf{s}, \mathbf{A} = \mathbf{M}^{\mathbf{T}} \quad (4)$$

Furthermore, similar to PageRank, we introduce a random jump parameter α, which allows the random surfer to randomly jump to all the other nodes:

$$\mathbf{M}' = (1-\alpha)\mathbf{M} + \alpha\mathbf{E}, \mathbf{E} = (1/n,...,1.n)^{\mathbf{T}}(1,...,1) \quad (5)$$

where n is the total number of nodes in the network, i.e. $n=|V_U|+|V_C|+|V_R|+|V_T|$. Given this, we can easily use the iterative method to find the scores for each node type:

$$\mathbf{s}_U = \alpha\mathbf{E} + (1-\alpha) \times (\lambda_{CU}\mathbf{M}_{CU}^{\mathbf{T}}\mathbf{s}_C + \lambda_{RU}\mathbf{M}_{RU}^{\mathbf{T}}\mathbf{s}_R + \lambda_{TU}\mathbf{M}_{TU}^{\mathbf{T}}\mathbf{s}_T + \lambda_{UU}\mathbf{M}_{UU}^{\mathbf{T}}\mathbf{s}_U) \quad (6)$$

$$\mathbf{s}_C = \alpha\times\mathbf{E} + (1-\alpha)\times(\lambda_{UC}\mathbf{M}_{UC}^{\mathbf{T}}\mathbf{s}_U + \lambda_{RC}\mathbf{M}_{RC}^{\mathbf{T}}\mathbf{s}_R + \lambda_{TC}\mathbf{M}_{TC}^{\mathbf{T}}\mathbf{s}_T) \quad (7)$$

$$\mathbf{s}_T = \alpha\times\mathbf{E} + (1-\alpha)\times(\lambda_{CT}\mathbf{M}_{CT}^{\mathbf{T}}\mathbf{s}_C + \lambda_{UT}\mathbf{M}_{UT}^{\mathbf{T}}\mathbf{s}_U + \lambda_{RT}\mathbf{M}_{RT}^{\mathbf{T}}\mathbf{s}_R) \quad (8)$$

$$\mathbf{s}_R = \alpha\times\mathbf{E} + (1-\alpha)\times(\lambda_{CR}\mathbf{M}_{CR}^{\mathbf{T}}\mathbf{s}_C + \lambda_{UR}\mathbf{M}_{UR}^{\mathbf{T}}\mathbf{s}_U + \lambda_{TR}\mathbf{M}_{TR}^{\mathbf{T}}\mathbf{s}_T) \quad (9)$$

where, $\mathbf{s}_U$ is the vector of scores for all users, likewise for $\mathbf{s}_C$, $\mathbf{s}_R$, and $\mathbf{s}_T$; $\mathbf{M}_{CU}$, $\mathbf{M}_{RU}$, $\mathbf{M}_{TU}$, and $\mathbf{M}_{UU}$ respectively represent the transition probabilities from categories to users, from resources to users, from tags to users, and from users to users.

We can simplify (6)-(9) formulas in a general form:

$$\mathbf{s}_Y = \alpha\times\mathbf{E} + (1-\alpha)\times\sum_{\lambda_{XY}\in\Lambda}\lambda_{XY}\mathbf{M}_{XY}^{\mathbf{T}}\mathbf{s}_X \quad (10)$$

where X and Y are any node types, $\mathbf{s}_X$ and $\mathbf{s}_Y$ are the rank vectors for type X and Y, λ_{XY} is the transition probability from type X to type Y, Λ is the set of λ_{XY}, and $\mathbf{M}_{XY}$ is the transition matrix corresponding with the relationship type XY.

Now, for performing the random walk on the heterogeneous graph, we need quantify the value of each λ_{XY}. Previously, the values are often assigned by manual or empirically, e.g., [16]. Several approaches have been proposed for learning the parameters automatically, e.g., a list-wise method [13]. In this paper, we propose a pair-wise learning algorithm.

3.1.2. A pair-wise learning algorithm for parameters learning. The heterogeneous graph may have many parameters λ_{XY}. We can see from Fig. 2, there is a set of parameter Λ= {λ_{CT}, λ_{CU}, λ_{CR}, λ_{UC}, λ_{UT},

$\lambda_{UR}, \lambda_{UU}, \lambda_{RC}, \lambda_{RU}, \lambda_{RT}, \lambda_{TC}, \lambda_{TU}, \lambda_{TR}\}$. Actually, in a real world system, the objects and the relationships would be rather complex. It would be highly infeasible to assign them manually. The other simple way is to averagely specify the values for all the parameters, thus, the random walk on the heterogeneous graph is decayed as a random walk on a homogeneous graph by viewing all the objects with the same type, which is the same as the PageRank on the traditional Web. However, we argue that the weight of each type of relationship should not be treated equally. For example, when a random surfer stays at a user node, she/he may jump to the user's collected resources with higher probability than the tags that the user used. The setting of the parameters might be critical to the final recommendation results. Actually, this is also a general problem for ranking in the Web 2.0 applications.

The main idea of the proposed algorithm is: given a training data set, we aim at finding a ranking function with parameters that can best fit the training data.

The training data is denoted as a set $\mathbf{A} = \{(i, j)\}$, where, for each component, i and j are the selected pair of objects of the same type with the importance score of i larger than j. Our objective then is to make the importance score pairs estimated by our random walk algorithm identical with those in the training data. Thus, the objective function is defined as:

$$\begin{aligned} &\max\ L = \sum_{(i,j)\in A} \hat{y}_{ij} y_{ij} \\ &s.t.\ \ \lambda_{UT} + \lambda_{UC} + \lambda_{UR} + \lambda_{UU} = 1, \\ &\qquad \lambda_{TR} + \lambda_{TU} + \lambda_{TC} = 1, \quad \lambda_{CT} + \lambda_{CU} + \lambda_{CR} = 1, \\ &\qquad \lambda_{RT} + \lambda_{RU} + \lambda_{RC} = 1, \ \ \lambda_{XY} > 0 \end{aligned} \tag{11}$$

where y_{ij} is an indicator function ($\hat{y}_{ij}$ is the true value of training data, y_{ij} is the estimated value by the random walk algorithm):

$$y_{ij} = \begin{cases} 1, & s_i - s_j \geq 0 \\ -1, & s_i - s_j < 0 \end{cases} \tag{12}$$

For each node i, its importance score s_i is estimated by equation (10). s_i-s_j is the difference between two importance score, which can by caculated by:

$$s_i - s_j = (1-\alpha) \sum_{\lambda_{XY}\in\Lambda} \lambda_{XY} \Big(\sum_{type(ki)=XY} m_{ki} s_k - \sum_{type(kj)=XY} m_{kj} s_k \Big) \tag{13}$$

We put the equations (12) and (13) into equation (11) and set the derivative function equal to zero:

$$\frac{\partial L}{\lambda_{XY}} : \Delta\lambda_{XY} = \sum_{\substack{(i,j)\in A \\ \cap \hat{y}_{ij} \neq y_{ij}}} \hat{y}_{ij} (1-\alpha) \Big(\sum_{type(ki)=XY} m_{ki} s_k - \sum_{type(kj)=XY} m_{kj} s_k \Big) \tag{14}$$

The derivative function can be viewed as the learning ratio to parameter λ_{XY}. We use another parameter μ to control the learning step length and thus the final updating function for λ_{XY} can be written as:

$$\lambda_{XY}' = \lambda_{XY} + \mu \cdot \Delta\lambda_{XY} \tag{15}$$

The learning algorithm is an iterative process, of which each iteration is comprised of two steps. In the first step, we fix λ_{XY} and update all importance scores s_i. In the second step, we fix s_i and update all λ_{XY}. The iterative process continues until some stop conditions are satisfied. The algorithm is summarized in Fig. 3.

In Fig. 3, τ is a threshold to control the stop condition (we empirically set τ= 0.01 in experiments).

Input: A heterogeneous graph G with parameters Λ and a training data set A
Output: The optimal values of parameters Λ
Algorithm: A pair-wise learning algorithm for adjusting optimal parameters in a heterogeneous graph
Step 1: //Initialization.
1. Initialize all λ_{XY} with average values satisfying equation (1);
2. Initialize all the transition probabilities between nodes using equation (2);
Step 2: // Iterative updating.
3. $L_{old} \leftarrow -\infty$;// initialize old objective function value;
4. do
   // update all λ_{XY};
5. for (each pair $(i,j) \in$ A with $y_{ij} \neq \hat{y}_{ij}$ in the ranking results)
6. update λ_{XY} by equation (15);
7. normalize λ_{XY} to satisfy equation (1);
8. end for
9. update all s_i by using equation (10);
10. calculate L_{new} by using equation (11);
11. until($| L_{new} - L_{old} | < \tau$)

Figure 3. **The pair-wise learning algorithm**

3.2. Recommendation

After calculating the global importance score using random walk model, we calculate the relevance score of an object to the query (in search scenario) or to the extracted key terms (in browsing scenario) and finally combine them by different recommendation strategies.

Given a query, the relevance score of an object is calculated by using the language model:

$$P(q \mid o) = \prod_{t_i \in q} \left\{ \omega \cdot \frac{tf(t_i, o)}{|o|} + (1-\omega) \cdot \frac{tf(t_i, O)}{|O|} \right\}, \omega = \frac{|o|}{|o| + \upsilon} \tag{16}$$

where q is a query and o is the profile of an object (e.g., the profile of a resource can be defined as the concatenation of its title, description, and content). $|o|$ is the length of the profile; $tf(t_i, o)$ is the term frequency of term t_i in o; $|O|$ is the number of objects in collection O; $tf(t_i, O)$ is the term frequency of term t_i in O; ω is a parameter ranging in [0, 1]; υ is a smoothing parameter and is commonly set as the average length of object profile in O. Language model uses a generating probability to describe the relevance of an object profile to a query.

Now, we explain our recommendation strategies based on the above two steps.

3.2.1. Browsing: do recommendation when a person browses an object. We again classify the scenario into

several sub-scenarios and propose the strategy for each of them (for each scenario, we use the recommendation of categories and users as examples):

a) Recommendation of categories and users when browsing a category. For recommending categories, firstly, we use the title or tags assigned to the browsing category as the query q. If a category does not contain title and tags, we use a keyword extraction tool to extract key terms from the profile of the category (e.g., the concatenation of titles of the resources contained in the category) as the query q. Secondly, we estimate the relevance score of other categories by calculating $P(q|c)$, where c denotes a category. Thirdly, we find all the users who share the current category, and rank all their categories by the importance score s_c. Finally, we combine the second step and the third step by linear interpolating of $P(q|c)$ and s_c, and recommend the top ranked categories. For recommending users, the strategy is similar. The difference lies in the third step. We find all users who share the current category, and then rank them based on s_u.

b) Recommendation of categories and users when browsing a user. For recommending categories, firstly, we find top two ranked categories (by s_c) of the browsing user. Then we use the concatenation of the title or tags of the two categories as the query. We also extract key terms if title and tags are not available. Secondly, we estimate $P(q|c)$ for all the categories. Thirdly, we find all collaborators (users who share the same categories) of the browsed user, and then rank all their categories by s_c. Finally, we combine $P(q|c)$ and s_c in the same way as that in a). For recommending users, the difference lies in the third step. We rank all the collaborators by s_u, not their categories.

c) Recommendation of categories and users when browsing a resource. For recommending categories, firstly, we use the title, tag, or extracted key terms of the browsing resource as the query. Secondly, we estimate $P(q|c)$ for all the categories. Thirdly, we find the owner of the browsing resource, and rank his/her categories by s_c. Finally, we combine $P(q|c)$ and s_c. For recommending users, the difference lies in the third step. We rank all the collaborators of the owner by s_u.

3.2.2. Search: do recommendation when a person searches objects with a query. We also classify the scenario into two sub-scenarios:

a) Recommendation of categories and users when searching categories/resources. In the searching scenario, we use the searching query q to estimate the relevance score. For recommending different types of objects, we first estimate the relevance score of each user by calculating $P(q|u)$. Secondly, we combine $P(q|u)$ and s_u and recommend the top scored users. For recommending the objects with the same type, i.e., categories, we do not calculate the relevance score to avoid recommending similar objects as those in the search results. Our strategy then is to get the recommended users, rank their categories by s_c, and finally recommend the top ranked categories.

b) Recommendation of categories and users when searching users. For recommending categories, the strategy is the same as a). For recommending users, we get the above recommended categories first, then rank them by s_u, and finally recommend the top ranked users.

4. Experiments

We evaluated the proposed method using the data from two real-world systems, Powazi (www.powazi.com) and Arnetminer (www.arnetminer.org).

4.1. Experiments on Powazi

4.1.1. DataSet. Powazi system (www.powazi.com) is a platform on which users can create/collect resources. A user can create multiple projects (each project can be viewed as a category) and share his projects with other users. Each project may contain multiple resources and tags can be assigned to projects and resources. Besides a user may search and browse the resources, projects, users, and tags in the system. Our goal is to recommend to the person (when she/he conducts searching or browsing) with different types of objects (including, users (U), resources (R), projects (P) or tags (T)) that might interest him/her. The system has been in operation on an intranet since July, 2007. So far, we have 132 users, 340 projects, 1403 resources, and 336 tags. In total, there are 726 project-user relationships, 881 project-resource relationships, 259 project-tag relationships, 1403 user-resource relationships, and 336 user-tag relationships. Finally in the experiment, we did not collect resource-tag and user-user relationship, thus our random walk algorithm only considers ten parameters, i.e.,

$\Lambda = \{\lambda_{PT}, \lambda_{PU}, \lambda_{PR}, \lambda_{UP}, \lambda_{UT}, \lambda_{UR}, \lambda_{RP}, \lambda_{RU}, \lambda_{TP}, \lambda_{TU}\}$.

4.1.2. Experiments of recommendation. We implemented the recommendation strategies for 5 scenarios. The scenarios and strategies are as follows (cf. section 3.2 for details):

Recommend projects and users when browsing a project. We define two baselines. The first baseline is to recommend with only relevant projects by calculating $P(q|p)$ and relevant users by calculating $P(q|u)$, (construction of a query q can be referred to section 3.2). We call this baseline as language model

(shortly LM), as we use language model to calculate $P(q|p)$ and $P(q|u)$. The second baseline is to recommend with only important projects by calculating s_p and important users by calculating s_u, we call this baseline as random walk (shortly RW). Our strategy is to recommend with both relevant and important projects by combining $P(q|p)$ (with the weight 0.5) and s_p (with the weight 0.5) using linear interpolation, likewise for users (shortly LM+RW).

Recommend projects and users when browsing a user. We implemented LM, RW, and LM+RW.

Recommend projects and users when browsing a resource. We implemented LM, RW, and LM+RW.

Recommend projects and users when searching users. For recommending projects, we have implemented the LM method as baseline and our strategy LM+RW. For recommending users, we first get the recommended projects, and then recommend their owners with the highest importance score. We call the strategy as LM+RW.

Recommend projects and users when searching projects. We implemented LM and LM+RW.

To evaluate the performance of our recommendation strategies, we have manually annotated a ground truth data. For each search scenario, we first selected 12 most frequent queries from the log of Powazi. Next, for each query, we collected candidates by pooling the recommending results of the implemented strategies, LM, RW, and LM+RW. Then 7 annotators (including graduates, college faculties, and technical staffs) were asked to annotate whether or not they are satisfied with each candidate. Finally, we obtained the ground truth by majority voting on the 7 answers. For each browsing scenario, we selected 12 most frequent browsed objects and annotated the ground truth in the same way.

We use MRR, P@3, and MAP as the evaluation measures (See [5] for details). Table 1 shows the evaluation results. In Table 1, "-" means that there are less than 3 recommendations for calculating P@3. We did not implement LM and RW for some search scenarios (cf. Section 3.2.2 for details). From the results, we can see LM+RW outperforms LM and RW in most of the scenarios.

Table 2 shows top three recommended projects in two specific scenarios. We asked several users for feedbacks about the results. The feedbacks show that the users are satisfied with most recommended results.

We compared our strategy LM+RW with the traditional method by combing LM with PageRank, which is called LM+PageRank [14] (equal to LM+RW but with an identical weight for all types of relationships). In this experiment, we want to show the advantage of the proposed pair-wise learning algorithm. Fig. 4 gives the comparison results of P@3 for the 10 recommendations in Table 1. We can see that our strategy performs better than LM+PageRank in many recommendations. On several tasks, e.g., SUP and BRP, the improvements are significant (from 10% to 20%).

Table 1. **Performances of recommendations in Powazi (%)**

Scenario	Recommend	Strategy	MRR	P@3	MAP
Search Scenarios					
Search Projects	Projects	LM+RW	70.00	61.90	53.94
	Users	LM	68.18	83.33	59.21
		LM+RW	100.00	66.67	60.69
Search Users	Projects	LM	66.67	41.67	51.34
		LM+RW	50.00	52.18	68.19
	Users	LM+RW	68.18	83.33	59.21
Browsing Scenarios					
Browse a Project	Projects	LM	40.00	-	22.00
		RW	60.00	58.33	38.09
		LM+RW	80.00	66.67	52.77
	Users	LM	100.00	66.67	47.43
		RW	100.00	77.78	54.76
		LM+RW	100.00	100.00	95.12
Browse a User	Projects	LM	60.00	-	23.33
		RW	90.00	73.33	60.86
		LM+RW	90.00	73.33	67.31
	Users	LM	100.00	-	30.00
		RW	90.00	79.12	78.57
		LM+RW	76.67	88.89	62.88
Browse a Resource	Projects	LM	60.00	66.67	13.36
		RW	58.89	44.44	52.83
		LM+RW	100.00	55.56	68.98
	Users	LM	50.00	50.00	18.34
		RW	60.00	50.00	33.72
		LM+RW	90.00	75.00	70.80

Table 2. **Example recommendations in Powazi**

Recommend projects when searching users using "Java"	Recommend projects when browsing a project with title "Information Extraction"
Eclipse TPTP	Profiling
Thinking in java 4th	semantic calendar
关于 Javascript (about Javascript)	Expertise information search

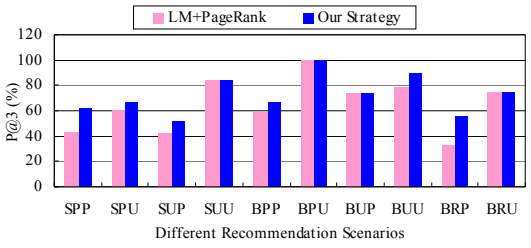

Figure 4. **The effect of assigning heterogeneous weights**

Finally, we conducted an additional experiment to evaluate the performance of the pair-wise learning algorithm itself. Specifically, for all objects in the Powazi system (including projects, users, resources, and tags), we randomly selected 400 pairs, and asked 7 annotators to annotate which one in each pair is more

'important' based on his/her preference. We pooled the results from all annotators and used 'majority voting' to obtain the ground truth. We use four-fifth of the ground truth data as training data for learning the parameters and test the obtained parameters on a held-out data set. The final accuracy is 81.8%.

We also compared the results with those obtained by a manually tuned method. For the manually tuned method, we range each λ from 0.1 to 0.9 with interval 0.1 and use the parameters setting, which results in the highest accuracy on the training data as final results. Thus, the manually tuned results can be viewed as upper bounds for our method. We also test it on the same held-out data set. The accuracy is 82%. We can see that the results obtained by our method are close to the upper bounds. This confirms the effectiveness of our method. Table 3 shows the parameters learned by our method (Shortly, Learn) and the manually tuned method (Shortly, Tune).

Table 3. **The parameters by our pair-wise learning algorithm and the manually tuned method in Powazi**

Method	λ_{PT}	λ_{PU}	λ_{PR}	λ_{UP}	λ_{UT}
Learn	0.50	0.12	0.38	0.15	0.10
Tune	0.50	0.10	0.40	0.20	0.10
Method	λ_{UR}	λ_{RP}	λ_{RU}	λ_{TP}	λ_{TU}
Learn	0.75	0.69	0.31	0.61	0.39
Tune	0.70	0.80	0.20	0.60	0.40

4.2. Experiments on Arnetminer

We also evaluated our method on ArnetMiner (www.arnetminer.org) [15], which is an academic social networking system, containing 448,365 researchers, 880,522 papers, and 4,203 conferences.

In this experiment, the task is to recommend researchers, papers, and conferences simultaneously when searching for one types of object, that is, recommend papers, conferences simultaneously when searching researchers, likewise for searching papers and conferences. We selected seven most frequent queries from the log of ArnetMiner for evaluation purpose, and annotated the ground truth in the same way as that in Powazi. We conducted evaluations on a subset of the data set in ArnetMiner. The data set contains 853 persons, 10,778 papers, and 222 conferences. We use the citation, authorship, and paper-publish-at as relationships to create a graph. In total, we create 15,169 citation relations, 2,122 bi-directional authorship relationships, and 717 bi-directional paper-publish-at relationships.

Given a search query q, we recommended each object by combining relevance score with importance score (LM+RW). We also compared with LM, which only considers the relevance score. The results are given in Table 4. We can see from the table that LM+RW outperforms LM for most of the recommendations.

Table 4.**The performances of recommendations in Arnetminer (%)**

Recommend	**Strategy**	**MRR**	**P@3**	**MAP**
Papers	LM	47.22	29.63	36.06
	LM+RW	49.44	29.63	37.31
Researchers	LM	68.52	59.26	55.84
	LM+RW	77.78	70.37	67.16
Conferences	LM	63.89	48.15	46.41
	LM+RW	66.67	51.85	48.13

We give an example when searching using query "support vector machine" in Table 5. We can see the results are reasonable, which confirms the effectiveness of our method for simultaneously recommending different types of objects.

Table 5.**Example recommendations for query "support vector machine"**

Persons	**Conferences**
Vladimir Vapnik	NIPS
Olvi L. Mangasarian	Machine Learning
Glenn Fung	ICML
Papers	
Support Vector Regression Machines	
Active Support Vector Machine Classification	
Supervised clustering with support vector machines	

5. Related work

5.1. Recommendation

Content-based filtering [2] recommends items for users based on correlations between the content of the items and the user's preferences. This method creates a profile for each user or item to characterize their nature.

Collaborative filtering [4][6] is a popular approach for recommendation, which recommends items for users based on the similarity between users or items. Model-based approaches use machine learning methods to train a model off-line that will be used to predict the ratings for unknown items. Related works include classification model [3], cluster model[4], graph model[1], and latent semantic model [9].

Association rules between users and items have been mined to help recommendation, e.g., [10][12].

Many e-commerce Web sites have utilized the recommendation function to better sell their products, e.g., Amazon.com, CDNOW.com, Reel.com, and eBay.com. Other Web sites like movielens.com, YouTube.com, and douban.com have also employed recommendation to attract more users/clicks.

Most of aforementioned approaches and Websites deal with recommendation of homogeneous objects or separately deal with different types of objects and only

a few of them consider simultaneously recommending of heterogeneous objects.

5.2. Random walk

With the large number of Web social networks becoming available, random walk theory has gained more and more popularity. Many research efforts have been made on analysing link structures to better understand the Web-based networks. PageRank is a state-of-the-art algorithm proposed by Brin and Page for estimating the importance of a Web page based on the other pages pointing to it [14].

Recently, many efforts for enhancing and extending the algorithm to a special environment have been made. For instance, Xi et al. [16] propose a unified link analysis framework called link fusion to consider both the inter- and intra-type link structure among multi-type inter-related data objects. Nie et al. [13] propose an object-level link analysis model, called PopRank, to rank the objects within a specific domain. Liu et al. [11] propose building a weighted, directed co-authorship network in digital libraries, and use an AuthorRank algorithm to rank authors. See also [7][17].

Most of the previous works focus on homogeneous graph (that is, the type of objects in the network is unique, e.g., only Web pages). Some efforts have also been placed for addressing the heterogeneous graph, e.g., [13] and [16]. The major difference of our work from the existing works lies in that we propose a learning-based random walk model over a heterogeneous network for the recommendation context.

6. Conclusion

In this paper, we have investigated the problem of recommendation over heterogeneous social networks. We formalize the recommendation as a ranking problem and propose a random walk model to estimate the importance of each object in the heterogeneous network. A pair-wise learning algorithm has been proposed for learning the weight for each type of relationship. We categorize the recommendations into different scenarios and propose corresponding strategies based on random walking results. Experimental results on two real-world systems show that improvements can be obtained by comparing with several baseline methods.

Acknowledgment

The work is supported by the National Natural Science Foundation of China (90604025, 60703059), Chinese National Key Foundation Research and Development Plan (2007CB310803), and Chinese Young Faculty Research Funding (20070003093).

References

[1] C. Aggarwal, J. Wolf, K. Wu, and P. Yu. Horting Hatches an Egg: A New Graph-theoretic Approach to Collaborative Filtering. In Proc. of KDD'99. pp. 201-212

[2] M. Balabanovic, Y. Shoham. Content-Based Collaborative Recommendation. Commun. ACM, Vol. 40(3), 1997.

[3] D. Billsus, M. J. Pazzani. Learning Collaborative Information Filters. In Proc. of ICML'98. pp. 46-53

[4] J. Breese, D. Heckerman, and C. Kadie. Empirical Analysis of Predictive Algorithms for Collaborative Filtering. In Proc. of UAI'98. pp. 43-52

[5] N. Craswell, A. de Vries, and I. Soboroff. Overview of the Trec-2005 Enterprise Track. In TREC 2005 Conference Notebook. 2005, pp. 199-205

[6] M. Deshpande and G. Karypis. Item-based Top-n Recommendation Algorithms. ACM Trans. Inf. Syst., Vol. 22(1):143-177, 2004.

[7] E. Garfield. Citation Indexing-Its Theory and Application in Science, Technology, and Humanities. John Wiley & Sons Inc, 1979.

[8] D. Goldberg, D. Nichols, D. M. Oki, AND D. Terry. Using Collaborative Filtering to Weave an Information Tapestry. Commun. ACM, Vol. 35(12):61-70, 1992.

[9] T. Hofmann. Latent Semantic Models for Collaborative Filtering. ACM Trans. Info. Syst., Vol. 22(1):89-115, 2004.

[10] W. Lin, S. Alvarez, and C. Ruiz. Collaborative Recommendation Via Adaptive Association Rule Mining. In Proc. of the Workshop on WEBKDD'00.

[11] X. Liu, J. Bollen, M. L. Nelson, and H. V. d. Sompel. Co-authorship Networks in the Digital Library Research Community. Information Processing and Management: an International Journal, Vol. 41(6):681-682, 2005.

[12] B. J. Mirza, B. J. Keller, and N. Ramakrishnan. Studying Recommendation Algorithms by Graph Analysis. J. Intel. Inf. Syst., Vol. 20(2):131-160, 2003.

[13] Z. Nie, Y. Zhang, J. Wen, and W. Ma. Object-level Ranking: Bringing Order to Web Objects. In Proc. of WWW'05. pp. 567-574

[14] L. Page, S. Brin, R. Motwani and T. Winograd. The PageRank Citation Ranking: Bringing Order to the Web. Stanford Digital Library working paper SIDL-WP-1999-0120, Nov.1999.

[15] J. Tang, D. Zhang, and L. Yao. Social Network Extraction of Academic Researchers. In Proc. of ICDM'07. pp. 292-301

[16] X W. Xi, B. Zhang, Y. Lu, Z. Chen, S. Yan, H. Zeng, W.Y. Ma, and E. A. Fox. Link Fusion: A Unified Link Analysis Framework for Multi-type Interrelated Data Objects. In Proc. of WWW'04. pp. 319-327

[17] J. Zhang, M. S. Ackerman, and L. Adamic. Expertise Networks in Online Communities: Structure and Algorithms. In Proc. of WWW'07. pp. 221-230