國立交通大學

資訊科學與工程研究所

碩 士 論 文

於社群標記網路上的監督式推薦系統

Supervised FolkRank:

A recommender system in the social tagging networks

研 究 生：郭立言

指導教授：李素瑛 教授

中 華 民 國 101 年 6 月

於社群標記網路上的監督式推薦系統

研究生：郭立言　　指導教授：李素瑛

國立交通大學資訊科學與工程研究所

摘要

群眾分類法揭示著一種分散式、去中心化且協同的標記系統，不僅可降低文件認知的花成本，亦可快速適應常用字彙的變化。社群標記系統中的連結關係包含使用者產生的內容、對於文件的評比、以及對於文件的標記。一般傳統利用二分圖來描述使用者與文件之間的評比關係，加入標籤之後，可被視為在使用者與文件之間一種重要的介面，而這種介面以語義的方式描述文件。縱使社群標記網路上的推薦系統已被廣泛地探討，然而個人化行為對於推薦結果的影響仍尚待研究。

我們提出了一個推薦系統 Supervised FolkRank，以重啟式隨機漫步(Random Walk with Restart)為基礎，對於重啟的機率、停滯的機率以及使用者與關鍵字的權重進行最佳化。本推薦系統關注所有文件的相關性而非僅止於二元地將文件分為相關與否，如此一來便可掌握整個文件集合就相關性的排序情況，以求得到最佳化效果。除此之外，透過分析每個使用者的最佳化參數向量而獲得這些向量的分佈。透過這些分佈，我們發現肇因於使用者行為的不同，而使得最佳化參數向量存在相當的分歧。從實驗中，比較其他監督式與非監督式的方法，我們所提出的推薦系統表現出較佳的正確率與取回率。

檢索詞：社群網路、推薦、隨機漫步、最佳化

Supervised FolkRank: A recommender System in the social tagging networks

Student: Li-Yen Kuo Advisor: Suh-Yin Lee

Institute of Computer Science and Information Engineering

National Chiao-Tung University

ABSTRACT

Folksonomy represents a distributed, decentralized, collaborative tagging system, which lowers the cognition cost and has a quick adaption to changes in vocabulary. Social tagging systems contain huge linking relations including user-generated content, ratings and annotations. Beyond the bipartite graphs that describe the ratings of items for users, annotation by tagging could be taken as an important interface to describe content semantically. Although the recommendation in social tagging networks has been studied extensively, the influence of personal behaviors on recommending results is still unexplored.

We propose a recommendation model, Supervised FolkRank, which uses a list-wise approach to formulate the objective function so that the ranking of all items could be considered. By analyzing the relation among restart probability, self-transition probability and the ratio of the target user to the selected query, we discover the divergence of the distribution of parameter vectors with the difference of users' behaviors. To find the representatives to describe the distribution, clustering is used for analysis.

From our experiments on the LibraryThing social tagging graph, we show that our approach outperforms other recommendation systems including supervised and

unsupervised ones. Finally, by showing that the results by the same model vary with different parameter vectors, we demonstrate the influence of the parameter vectors.

Index terms: Social networks, recommendation, random walk, optimization

ACKNOWLEDGEMENT

I greatly appreciate the kind guidance of my advisor, Prof. Suh-Yin Lee. She not only helps with my research but also takes care of me. Her graceful suggestion and encouragement help me forward to complete this thesis.

Besides, I want to give my thanks to all members in the Information System Laboratory for their suggestion and instruction, especially Mr. Yi-Cheng Chen, Mr. Wei-Zen Wang, and Mr. Fan-Chung Lin.

I would express my thanks to the nicest and most beautiful lady Ms. Naomi Chung who accompanies me and always give me a happy smile.

Finally I would like to express my deepest appreciation to my parents. This thesis is dedicated to them.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

In recent years, social tagging has transformed the behavior of users from passive information receiving to active producing. The influence of authoritative pre-defined taxonomy is decreasing, and on the other hand a new type of tagging so-called folksonomy has become the most popular way to describe, categorize, and navigate content within the Web 2.0 websites.

Folksonomy is formed by three types of nodes, namely users, items and tags. Users use tags to describe items so that items could be categorized by these tagging behaviors. Figure 1-1 shows the tagging relation and categorism of each user. A user uses his familiar words as tags to describe items. Thus, each user has his own categorism.

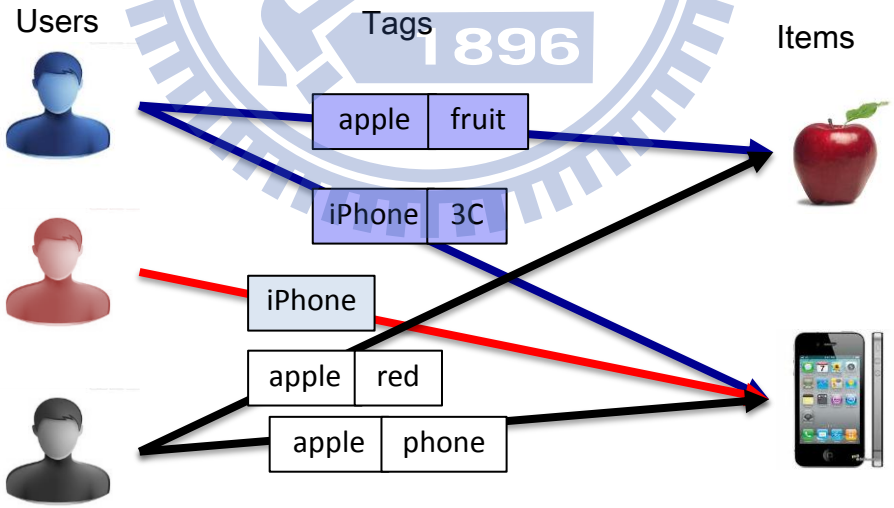

Figure 1-1 An example of folksonomy.

With the tremendously increasing of information on Internet, taxonomy, which constructs a hierarchical categorism by a single authority, would be out of date.

Unlike taxonomy, folksonomy represents a distributed, decentralized, collaborative tagging system, which lowers the cognition cost and has a quick adaption to changes in vocabulary [1, 26]. The main difference is that folksonomy respects to the largest possible extent the request of non-expert users not to be bothered with any formal modeling overhead [11].

Nevertheless, this is a benefit as well as a drawback to the tagging approach. Due to lacking of unique authoritative annotations, ambiguous and polysemous tags treated as different meanings may decline the accuracy of information retrieval, and synonyms would cause the redundancy of information. Moreover, because of unawareness of the implicit centralized controlling vocabulary, tagging of a user may contravene the mainstream categorical scheme. To reduce the influence of noise, personalized recommendation is proposed to make these 'abnormal' tags more reasonable.

A. Hotho et al. [11] propose a PageRank-like algorithm, called FolkRank, to retrieve information in folksonomies. FolkRank considers the relationship among users, tags and resources and converts the relationship formed by triadic hyper-edges into an undirected tripartite graph. Thereafter, many random walk-based recommendation systems are proposed and the parameters are tuned heuristically. The parameters could be taken as a three-tuple vector, which includes restart probability, self-transition probability and the ratio of the target user to the selected query. We are interested in the relation between the performance of recommendation and the distribution of per user optimized parameter vectors. If the distribution is centralized, we may use only one parameter vector certainly. However, if the distribution is not centralized, is that still meaningful to tune only one parameter vector heuristically?

In this thesis, we will propose a novel supervised recommendation system, called Supervised FolkRank (SFR), to enable collaborative tagging to annotate the available

content. Our model uses a list-wise learning approach to focus on the ranking of all items rather than pair-wise ones that split them into two sets. Due to lack of ground truth data of recommendation, we use tagging records to simulate the recommendation results. To make our model reliable, we define the relevant and irrelevant item. The items that are neither relevant nor irrelevant would be pruned. We analyze the distribution of the optimized parameter vector of each user. It proves our assumption that the distribution may not be centralized. To reduce the influence of the divergence of distribution, we find the representatives by clustering. Besides, we also discuss the influence of the transition matrices constructed by different protocols that reduce the influence of frequent occurring elements in the transition matrix. Though the social tagging graph is undirected, the asymmetric transition matrix would outperform the symmetric one. The weights of different directions of an edge are quite different. Hence, before normalizing each row to sum to one, the transition matrix should be built asymmetrically.

Chapter 2

Related Work

2.1 Recommendation Systems in Folksonomies

The relation in folksonomies could be basically regularized as a ternary relation among users, tags, and items [5, 11, 13], where each entry indicates a user tagging an item with a tag. Due to the impracticality of using of the ternary relation directly, all co-occurrences of users and items, items and tags, users and tags are projected from the ternary relation to undirected and weighted edges.

2.1.1 Collaborative Filtering

Lately, many researches have been revolving around recommendation systems in social networks. Collaborative filtering (CF) [10] is one of the most used and successfully applied methods for a personalized recommendation system. By and large, CF makes a recommendation from the similarity of linking behaviors. Thus, the definition of similarity becomes a critical factor. In the past the traditional CF uses a bipartite graph which means that users give preference judgments for items as ratings. There are two sorts of nodes, i.e. users and items, and weighted edges. Due to the bipartite graph, the traditional CF could be divided into two categories, i.e. user based and item based.

A user-based CF algorithm makes a prediction by first finding users who are similar to the target user, and then taking a weighted combination of the deviation from their mean ratings. The predicted rating could be written as in Eq. (1).

$$pred(u,i) = \bar{r}_u + \frac{\sum_{u' \in U}\left(r_{u',i} - \bar{r}_{u'}\right) userSim(u,u')}{\sum_{u' \in U} userSim(u,u')} \quad (1)$$

where u is the target user, i is the target item, U is the finite set of users, $r_{u',i}$ is

the rating of user u' for item i, $\bar{r}_u$ and $\bar{r}_{u'}$ are the mean ratings of users u and u'respectively, and $userSim(u,u')$ stands for the similarity between users u and u'.

In an item based CF algorithm, a prediction is made by finding items which are similar to the target item which is predicted now, and then taking a weighted combination of the deviation from their mean ratings. The predicted rating could be written as in Eq. (2).

$$pred(u,i) = \bar{r}_i + \frac{\sum_{i \in I}(r_{u,k} - \bar{r}_k)\ itemSim(i,k)}{\sum_{k \in I}\ itemSim(i,k)} \tag{2}$$

where I is finite set of items, $r_{u,k}$ is the rating of user u for item k, $\bar{r}_i$ is the mean rating of item i, and $itemSim(i,k)$ stands for the similarity between items i and k.

For calculating the similarity between users or items, several measures of similarity can be used. The most used is the Pearson correlation score defined in Eq. (3).

$$userSim(u,u') = \frac{\sum_{u' \in U}(r_{u,i} - \bar{r}_u)(r_{u',i} - \bar{r}_{u'})}{\sigma_u \sigma_{u'}} \tag{3}$$

where σ_u is the standard deviation of the ratings of user u. Moreover, I. Konstas et al. [13] combines several different similaritys based on the relation in the data. The datasets used is from Last.fm [1], which is an online radio website. They use three similaritys based on the user's playcount, users tags and users friendship, and then their weighted sum is used to obtain the compound similarity which can be written as in Eq. (4).

$$Sim(u,u') = \alpha \cdot itemSim(u,u') + \beta \cdot userSim(u,u') + \gamma \cdot tagSim(u,u') \tag{4}$$

where $\alpha + \beta + \gamma = 1$ and $itemSim(u,u')$, $userSim(u,u')$, $tagSim(u,u')$ are the similarity obtained from the user tracks, user friendships and user tags.

Even though the compound similarity is used, from the result in [13], we know that the traditional CF does not perform well without using information about tags directly.

[1] http://www.last.fm

In the CF model in [13], $tagSim(u,u')$ is the only term concerned with tags in Eq. (4). Could it elaborate the ternary relation among users, tags and items by using $tagSim(u,u')$ merely? From the definition of $tagSim(u,u')$, we know that $tagSim(u,u')$ evaluates the similarity in view of macrocosm, neglecting the variation among users. For a polysemous word such as apple, people concerned with agriculture may use this tag to refer to a kind of fruit, while others concerned with 3C products always use this tag to represent the Apple Inc. Hence, in a personalized social recommendation system, the variation among users plays an important role, and then the information about the topological structure in a social graph is concerned.

To consider the influence of tags, D. Parra-Santander et al. [18] introduce the BM25-based similarity into the classic CF model [19]. BM25 is a non-binary probabilistic model used in information retrieval to rank matching documents according to their relevance to a given search query. Given a search query, it calculates the relevance of each document in a collection. They compare the tags taken as a query with other tags used by the target user. Besides, they also compare the set of tags of items. Thus, the neighbor-weighted collaborative filtering (NwCF) which combines the BM25 model and the classic CF is proposed. The new predicted score can be written as in Eq. (5).

$$pred'(u,i) = \log\big(1 + nbr(i)\big) \cdot pred(u,i) \tag{5}$$

where $nbr(i)$ represents the similarity between the target user u and a neighbor (i.e., an item i), while $pred(u,i)$ is the classic CF. $nbr(i)$ is taken from the calculation of the Retrieval Status Value of an item i given a query q as in Eq. (6).

$$nbr(i) = \sum_{t \in q} \text{IDF} \cdot \frac{(k_1+1)\, tf_{ti}}{k_1[(1-b) + b(L_d/L_{ave})] + tf_{ti}} \cdot \frac{(k_3+1)\, tf_{tq}}{k_3 + tf_{tq}} \tag{6}$$

where L_d is the item length (i.e. the sum of the frequencies of each tag of the item i), L_{ave} is the average of the L_d of every item, tf_{ti} is the frequency of the tag t

in the set of tags of the item i and tf_{tq} stands for the frequency of the tag in the query. Besides, k_1, k_3 and b are parameters that are set to 1.2, 1.2 and 0.8, respectively, according to the results in [18].

2.1.2 Random Walk Model

PageRank [3] proposed by S. Brin and L. Page makes use of the link structure of the web to calculate a quality ranking for each web page. They use the random walk model with the probability to restart to simulate the behavior of user' surfing on the Internet. It exploits the topological structure of a social graph through the stochastic process. Moreover, the probability of restart determines the static score of each node according to the distance from the start node. Without restart, the static score distribution is not affected by the initial state and could only represent the macro characteristic of the social graph. From the viewpoint of macrocosm, a social graph represents the general behavior of people. To make the recommendation system personalized, the influence of starting nodes must be emphasized by introducing the probability of restart.

FolkRank algorithm proposed by A. Hotho et al. [11] is the first approach to apply the random walk models to folksonomies. They propose a formal model for folksonomies which is a tuple $\mathbb{F} \coloneqq (U, T, R, Y, \prec)$ where U, T and R are finite sets whose elements are called users, tags and resources respectively, Y is a ternary relation among them, i.e. $Y \subseteq U \times T \times R$, and $\prec$ is a user-specific subtag/supertag-relation, i.e. $\prec \subseteq U \times T \times T$. A folksonomy induces a topological structure which would be exploited by the ranking algorithm.

Their ranking algorithm called Adapted PageRank combines random walk with restart and lazy random walk models to provide a topic-specific ranking in a folksonomy. The vector of each step could be written as in Eq. (7).

$$\mathbf{w_{n+1}} \leftarrow \alpha \mathbf{w_n} + \beta \mathbf{A w_n} + \gamma \mathbf{p} \quad (7)$$

where $\mathbf{A}$ is the row-stochastic version of the adjacent matrix of the graph constructed, $\mathbf{p}$ is a preference vector, i.e. initial vector, $\alpha, \beta, \gamma \in [0, 1]$ are constants with $\alpha + \beta + \gamma = 1$. The constant α regulates the speed of convergence, while γ controls the influence of the preference vector.

Notice that the graph constructed is an undirected graph so that the undirectedness of the graph makes it very difficult for other nodes than those with high edge degree to become highly ranked, no matter what the initial vector is [16]. To solve this problem, a differential approach called FolkRank based on Adapted PageRank is proposed. The FolkRank algorithm computes a topic-specific ranking as follows [11, 16]:

1. Let $\mathbf{w^{(0)}}$ be the fixed point from (7) with $\mathbf{p} = \mathbf{1}$.
2. Let $\mathbf{w^{(1)}}$ be the fixed point from (7) with $\mathbf{p} = \mathbf{1}$, but $\mathbf{p}[u] = 1 + |U|$ and $\mathbf{p}[i] = 1 + |I|$.
3. $\mathbf{w} \coloneqq \mathbf{w^{(1)}} - \mathbf{w^{(0)}}$ is the final weight vector.

Eliminating the influence of the initial vector, the resulting score distribution $\mathbf{w^{(0)}}$ represents the macro behavior of all users. This resulting score distribution could be taken as the neutral preference of the graph without being affected by the initial vector. For an instance, in [11], the tags “software” or “java” are frequently used, so that the result of topic-specific ranking would be interfered by these tags. Hence, the neutral preference of the graph could be taken as the “background noise”. From their result of experiments, it shows that the FolkRank algorithm could cause better ranking by diminishing the influence of the background noise $\mathbf{w^{(0)}}$.

M. Clements et al. [5] use the lazy random walk model which integrates the user’s preference and semantically related query terms. To reduce the influence of frequently occurring elements, they use TF-IDF weighting on the input matrices. Unlike the

differential approach in FolkRank, it does not calculate $\mathbf{w}^{(\mathbf{0})}$ or $\mathbf{w}$, which costs more time and memory. By modifying the transition matrix before computing random walks, this approach is more practical in learning phase. They heuristically tune the ratio of the weight of a user to a query as an initial state vector in the lazy random walk model. Moreover, they assume that a tag assigned to an item by the user is the same as they would use as a query to retrieve the item. Thus, they only focus on the known data, neglecting what is unknown.

To evaluate the predicted content ranking, they use the Normalized Discount Cumulative Gain (NDCG) proposed by Järvelin and Kekäläinen [12]. Due to lack of relevance measures, they assume that the ratings could be taken as the relevance while predicting the content ranking. We argue that the assumption is true on the premise that the item is relevant in the topic-specific ranking. On the other hand, A. Al-Maskari et.al [1] NDCG has its limitation and could not be taken as the only one measure for evaluation.

From their result, with increasing of query length, the influence of personalization would decrease gradually. If a user puts more effort in indicating his information need by giving more query terms to the system, the influence of personalization and smoothing diminishes. These results show that the optimal model whose parameters are tuned heuristically converges to the frequency based model, i.e. non-personalized model, when the user issues longer queries.

The construction of transition matrix seems to be a standard basis in social tagging networks. So does what I. Konstas et al. [13] propose. They develop a track recommendation system integrating the traditional ternary relation and the relationship between users (i.e., the user-user relation). However, from their results, the direct relation between users might disturb the performance while considering less

social knowledge. When the main social knowledge (e.g., the user-item, user-tag, item-tag relations) is taken into account, the user-user relation might give the performance a little help [21]. Besides, comparing the results of the graph model with the ones of the standard collaborative filtering model, it shows that the former outperforms the latter. We argue that it is unfair to compare the graph model of the ternary relation with the collaborative filtering model of the binary relation.

Due to their empirical view to evaluate the proposed algorithm, the parameters are tuned heuristically without discussing the relation between parameters and results further. Besides, it is still obscure that how the variation of behavior between users affects the performance and the optimal parameters of the model. Therefore, in our model, we obtain the optimal parameters for each user by machine learning, and then from the distribution of these parameter sets, we discuss the relation among user's behavior, ranking and parameters.

2.2 Machine Learning

2.2.1 The Pairwise Approach

The learning approaches take the entire set of documents associated with a query in the training data as an input and predict their ground truth labels. The pair-wise approaches calculate the loss function to divide the training data into two sub-categories, namely relevant and irrelevant. The supervised random walks (SRW) [2] proposed by L. Backstrom et al. is an instance of pair-wise approaches.

For each edge (u, v) in the graph G, they compute the strength $a_{uv} = f_w(\psi_{uv})$, where f_w parameterized by w takes the edge feature vector ψ_{uv} as input. Thus, the strength function $f_w(\psi)$ is what they want to optimize in the training phase. Because of avoidance from the occurrence with underflow and overflow of double precision floating point numbers, the logistic edge function is suggested.

The restart probability controls the expected distance surfing from the start node before it restarts. High values of the restart probability give local random walks, while low values may diminish the influence of initial state. Notice that in SRW, the restart parameter is set heuristically rather than optimized in the training phase because from their experiments the restart probability does not affect the result much. On the contrary, in our model, the restart probability is in the parameter vector for optimization.

Similarly to formulations of Support Vector Machine, SRW introduces a loss function to "soften" the constraints. According to the loss function calculated from the stationary distribution of random walk with restart, it penalizes violated constraints. Thus, the optimal condition could be found by minimizing the loss function. There are three choices of loss function:

1. Squared loss with margin b:

$$h(x) = \max\{x + b, 0\}^2 \tag{8}$$

2. Huber loss with margin b and window $z > b$:

$$h(x) = \begin{cases} 0 & \text{if } x \leq -b, \\ (x+b)^2/2z & \text{if } -b < x \leq z - b, \\ (x+b) - z/2 & \text{if } x > z - b \end{cases} \tag{9}$$

3. Wilcoxon-Mann-Whitney (WMW) loss with width (Proposed to be used when one aims to maximize AUC [27]):

$$h(x) = \frac{1}{1 + \exp(-x/b)} \tag{10}$$

The WMW loss is suggested because the model trained with the other two loss functions does not perform better than the baseline obtained through unweighted PageRank. Compared with other learning methods such as decision tree and logistic regression, SRW algorithm obtains the highest AUC (Area Under Curve) and P@20 (Precision at top 20).

2.2.2 The Listwise Approach

Unlike pair-wise approaches, the list-wise approach assumes that the ground truth labels are given in terms of permutations, while the judgments might be in other forms [15]. Evaluation measures such as the mean average precision (MAP) and the normalized discounted cumulative gain (NDCG) can also be rewritten in the form of the permutation set. Hence, how to obtain the approximate permutation sorted by their relevance scores, which has not been known yet, becomes an important issue.

In Softrank [24], it introduces a random process in which each random variable is governed by Gaussian distribution to describe the score distribution of all items. Given the item set $x = \{x_j\}_{j=1}^{m}$ associated with a training query q, the score s_j of item x_j is treated as no longer a deterministic value but a random variable. The random variable could be written as in Eq. (11).

$$p(s_j) = N(s_j | f(x_j), \sigma_s^2) \tag{11}$$

where σ_s is the variance of the Gaussian distribution, $f(x_j)$ which is the original score output by the scoring function is the mean.

Due to the randomness in the scores, each item has the probability of being ranked at any position in the ranking. Thus, the probability of an item x_u being ranked before another item x_v can be written as in Eq. (12).

$$p_{u,v} = \int_0^{\infty} N(s | f(x_u) - f(x_v), 2\sigma_s^2) ds \tag{12}$$

Suppose that there is an item x_j already in the ranked list, while adding another one x_u, if x_u can beat x_j the rank of x_j would be increased by one. Otherwise, the rank position of x_j is unchanged. Hence, the probability of an item x_j being ranked at position r can be deduced iteratively

$$P_j^{(u)}(r) = p_j^{(u-1)}(r-1)P_{u,j} + p_j^{(u-1)}(r)(1 - P_{u,j}) \tag{13}$$

where u is the iteration index, and then the expected position of an item x_u could

be written as follows

$$\pi^q(x_u) = \sum_{r=1}^{m} P_j^{(u)}(r) \frac{1}{\log(1+r)} \tag{14}$$

So far, we can use the expected rank position of each item to approximate the evaluation measures such as NDCG or MAP which is taken as the basis of the objective function.

Besides Softrank stated above, there are some approaches that perform approximation to the rank positions using smooth functions of the ranking scores, such that the approximate evaluation measures can consequently become differentiable and easier to optimize. Qin et al. [20] approximate the rank positions by a sigmoid function. The rank position can be written as

$$\pi^q(x_j) \approx 1 + \sum_{u=1, j\neq i}^{m} \frac{\exp\left[-\alpha\left(f(x_j) - f(x_u)\right)\right]}{1 + \exp\left[-\alpha\left(f(x_j) - f(x_u)\right)\right]} \tag{15}$$

where $\alpha > 0$ is a scaling constant.

H. Valizadegan et al. [25] use a simple logistic model as a smooth function to approximate the rank positions. The rank position can be written as

$$\pi^q(x_j) \approx 1 + \sum_{u=1, j\neq i}^{m} \frac{1}{1 + \exp\left[2\left(f(x_j) - f(x_u)\right)\right]} \tag{16}$$

Notice that there is no scaling constant in it. Practically, the ranked list predicted by the logistic-based smooth function is not precise enough so that it may interfere with the optimization. In our model, we would discuss the problem and introduce a scaling constant to reinforce the precision of the predicted ranked list. Moreover, the influence of scaling constant upon precision and practicability would be also discussed in the next chapter.

Chapter 3

Supervised FolkRank

3.1 Random Walk Model

A graph is an intuitive representation of data with some topological relations. Thus, we use a random walk model over the graph to rank the relevance of items on the selected tags as a query. From [11] a folksonomy is a ternary relation $\boldsymbol{Y} = U \times I \times T = \{(u, i, t) | u \in U, i \in I, t \in T\}$, where each element indicates that a user u tagged an item i with a tag t and U, I, T are finite sets, whose elements are called users, tags, items respectively. Using the concept in [5], we can build a matrix **D**, where $\mathbf{D}(u, i, t) = 1$ if $(u, i, t) \in \boldsymbol{Y}$. Thus, this 3-dimension matrix **D** can be projected to three 2-dimension matrices:

$$\mathbf{UT}(u, t) = \sum_{i \in I} \mathbf{D}(u, i, t) \tag{17}$$

$$\mathbf{UI}(u, i) = \sum_{t \in T} \mathbf{D}(u, i, t) \tag{18}$$

$$\mathbf{IT}(i, t) = \sum_{u \in U} \mathbf{D}(u, i, t) \tag{19}$$

Semantically speaking, each position in **UI** matrix indicates how many tags that a user assigned to an item. This relation is not useful because the familiarity between a user and an item does not rely on how many tags used. An ambiguous item to a user semantically may be described by less tags, but it doesn't mean that the user dislikes the item. Hence, due to lack of clear indication of **UI**, we replace $\mathbf{UI}(u, i)$ with the rating matrix $\mathbf{R}(u, i)$ where each position indicates that the rating that a user u assigned to an item i.

To reduce the influence of frequent occurring elements in a 2-dimension matrix, we use TD-IDF weighting on each 2-dimension matrix [5, 22] with normalization. The

definitions of weightings of elements in the sub-matrices are shown below, respectively:

$$\mathbf{UT}'(u,t) = \frac{1}{N_{\mathrm{UT}}} * \mathbf{UT}(u,t) * \log\left(\frac{|U|}{\sum_{u' \in U} \mathrm{sgn}\left(\mathbf{UT}(u',t)\right)}\right) \tag{20}$$

$$\mathbf{TU}'(t,u) = \frac{1}{N_{\mathrm{UT}}} * \mathbf{UT}(u,t) * \log\left(\frac{|T|}{\sum_{t' \in T} \mathrm{sgn}\left(\mathbf{UT}(u,t')\right)}\right) \tag{21}$$

$$\mathbf{IT}'(i,t) = \frac{1}{N_{\mathrm{IT}}} * \mathbf{IT}(i,t) * \log\left(\frac{|I|}{\sum_{i' \in I} \mathrm{sgn}\left(\mathbf{IT}(i',t)\right)}\right) \tag{22}$$

$$\mathbf{TI}'(t,i) = \frac{1}{N_{\mathrm{TI}}} * \mathbf{IT}(i,t) * \log\left(\frac{|T|}{\sum_{t' \in T} \mathrm{sgn}\left(\mathbf{IT}(i,t')\right)}\right) \tag{23}$$

$$\mathbf{R}'(u,i) = \frac{1}{N_{\mathrm{R}}} * \mathbf{R}(u,i) * \log\left(\frac{|U|}{\sum_{u' \in U} \mathrm{sgn}\left(\mathbf{R}(u',i)\right)}\right) \tag{24}$$

$$\mathbf{R}_{\mathrm{T}}'(i,u) = \frac{1}{N_{\mathrm{R}_{\mathrm{T}}}} * \mathbf{R}(u,i) * \log\left(\frac{|I|}{\sum_{i' \in I} \mathrm{sgn}\left(\mathbf{R}(u,i')\right)}\right) \tag{25}$$

where N is the normalization. For example, the normalization of the $\mathbf{UT}'$ matrix is computed by:

$$N_{\mathrm{UT}} = \sum_{t' \in T} \mathbf{UT}(u,t') * \log\left(\frac{|U|}{\sum_{u' \in U} \mathrm{sgn}\left(\mathbf{UT}(u',t')\right)}\right)$$

We then modify the transition matrix proposed and combine these six normalized matrices to build the random walk stochastic transition matrix $\mathbf{A}$ depicted in Figure 3-1. In Figure 3-1 (a), the transition matrix A combined by the sub-matrices. There is no edges to link two nodes with the same type so that UU, II, TT are zero matrices. In Figure 3-1 (b), the user and the query are assigned with weights θ and $(1-\theta)$ respectively. In Figure 3-1 (c), all items ranked according to their scores in v_n would be taken as an output. Notice that $\mathbf{A}$ is not a diagonal matrix, and the tripartite graph formed by $\mathbf{A}$ is directed. We argue that in a social tagging graph, the edge direction may affect the probability of surfing to a specific adjacent node. Hence, while reducing the influence of frequent occurring elements, the edge direction would also affect the results of reduction. For an instance, if a user u only used a tag t while t is used popularly, the value of $\mathbf{UT}'(u,t)$ is less because there are many competitive

users for t. On the other hand, because the lack of competitive tags for u, the value of $\mathbf{TU}'(t,u)$ is quite large.

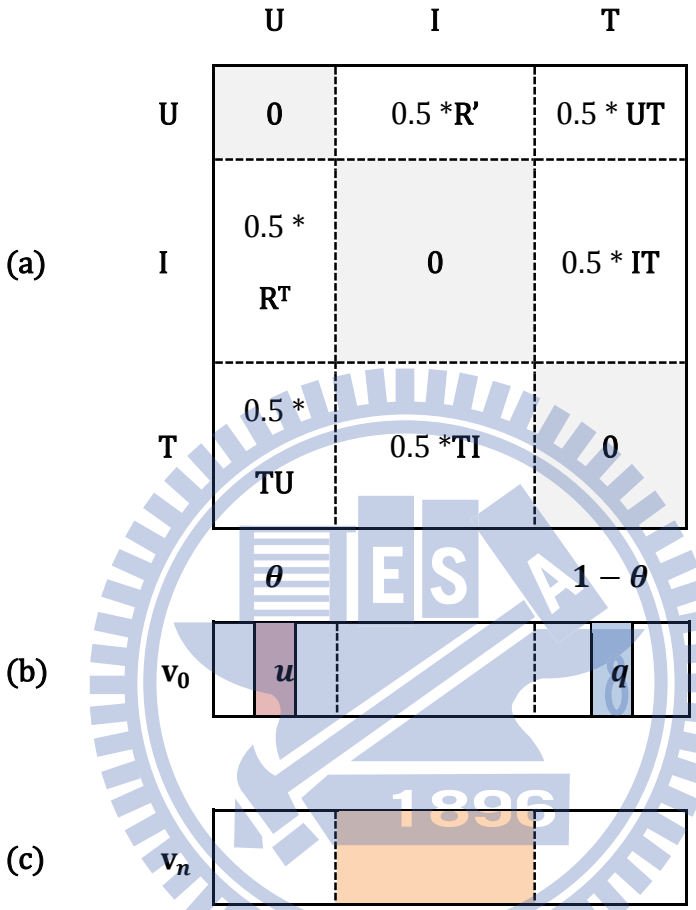

Figure 3-1 (a) The transition matrix A. (b) The initial state vector v_0. (c) The stationary state vector v_n.

Due to the normalization of the sub-matrices in Eq. (20-25), each row of each sub-matrix sums to 1. We multiply each sub-matrix in $\mathbf{A}$ by 0.5 to make each row of $\mathbf{A}$ sums also to 1. Thus, each position in $\mathbf{A}$ could be used as the transition probability.

In the initial state vector $\mathbf{v}_0$ we use θ to adjust the proportion of the target user to the query: $\mathbf{v}_0(u) = \theta$ and $\mathbf{v}_0(q) = 1 - \theta$, where u is the target user and q is the query. The more θ is, the more influence of the target user is, and vice versa.

The model we used is FolkRank in [11]. FolkRank combines two models:

PageRank [3] and Lazy Random Walk. The relation between two adjacent iterations could be stated in Eq. (26).

$$\mathbf{v}_{n+1} = \alpha\mathbf{v}_0 + \beta\mathbf{A}\mathbf{v}_n + \gamma\mathbf{v}_n \tag{26}$$

where n is the iteration index, α is the probability of restart, β is the probability of forward-transition, and γ is the probability of self-transition, $\alpha, \beta, \gamma \in [0, 1]$ are constrained by the linear relation: $\alpha + \beta + \gamma = 1$. The constant γ controls the speed of convergence, while α controls the locality. In the training phase, the speed of convergence determines the iterations of transition which affect the cost of time and memory usage

3.2 The Optimization Problem

The training data contains the ternary relation among users, tags and items. Thus we could determine whether an item is relevant or not. According to the definition of NDCG, items are ranked by its relevance. Due to the lack of the measure of relevance, we take the ratings as relevance. However, the relevance of an item to the query is not related to its rating directly. The only thing we know is that the ratings could be taken as a measure of quality while the items are all relevant. To avoid disturbing the precision, we set the rating of an irrelevant item to 0.

We modify the objective function proposed by H. Valizadegan et al. [25] to approximate NDCG in Eq. (27).

$$\bar{\mathcal{L}}(u, Q, F) = \frac{1}{|Q|}\sum_{q\in Q}\frac{1}{|I_u|}\sum_{i\in I_u}\left\langle\frac{2^{r_{u,q}(i)}-1}{\log\left(1+\pi^q(i)\right)}\right\rangle_F$$

$$= \frac{1}{|Q|}\sum_{q\in Q}\frac{1}{|I_u|}\sum_{i\in I_u}\sum_{\pi^q\in S_{I_u}}\Pr(\pi^q|F,q)\frac{2^{r_{u,q}(i)}-1}{\log\left(1+\pi^q(i)\right)} \tag{27}$$

where the notation $\langle\ \ \rangle_F$ is the expectation over all the possible rankings induced by the ranking function F, Q is a query set which we use to train for u, I_u is the item set

in which each item is relevant to u, S_{I_u} stands for the set of permutations of I_u, and π^q is an instance of permutation. Notation $\pi^q(i)$ stands for the rank position of the item i by π^q, and $r_{u,q}(i)$ stands for the conditional rating defined as follows.

$$r_{u,q}(i) = \begin{cases} \mathbf{R}(u,i), & \text{if } i \text{ is relevent to } u \text{ selecting } q \text{ as a query.} \\ 0, & \text{if } i \text{ is irrelevent to } u \text{ selecting } q \text{ as a query.} \end{cases}$$

$\bar{\mathcal{L}}(u,Q,F)$ can be simplified to $\bar{\mathcal{H}}(u,Q,F)$ in Eq. (28).

$$\bar{\mathcal{H}}(u,Q,F) = \frac{1}{|Q|}\sum_{q\in Q}\frac{1}{|I_u|}\sum_{i\in I_u}\frac{2^{r_{u,q}(i)}-1}{\log(1+\langle\pi^q(i)\rangle_F)} \tag{28}$$

Because $1/x$ is a convex function while $x>0$, therefore $\langle 1/\log(1+x)\rangle \geq 1/\langle\log(1+x)\rangle$. On the other hand, because $1/\log(1+x)$ is a concave function, therefore $\langle\log(1+x)\rangle \leq 1/\log(1+\langle x\rangle)$. From the two inequality stated above, we could get $\bar{\mathcal{L}}(u,Q,F) \geq \bar{\mathcal{H}}(u,Q,F)$.

By introducing the difference of the output scores of every two items to the logistic function, the rank position could be approximated in Eq. (29).

$$\langle\pi^q(i)\rangle = 1 + \sum_{j\in I_u, i\neq j}\langle\pi^q(i,j)\rangle \approx 1 + \sum_{j\in I_u, i\neq j}\frac{1}{1+\exp\left[\eta\left(F_{u,q}(i)-F_{u,q}(j)\right)\right]} \tag{29}$$

where $\langle\pi^q(i,j)\rangle$ is the result of the competition between item i and item j, $F_{u,q}(i)$ is the output score given by the ranking function $F_{u,q}$ which takes the target user u and the selected query q as input. Here, we use FolkRank model proposed above as the ranking function.

In the real condition, the result of a competition is that the position of a winner who has a larger score would add 0, while the position of the loser who has a smaller score would be added by one. We could translate the competition into a non-differentiable function written as:

$$\langle\pi^q(i,j)\rangle = \begin{cases} 0, & \text{if } F_{u,q}(i)-F_{u,q}(j) > 0 \\ 1, & \text{if } F_{u,q}(i)-F_{u,q}(j) < 0 \end{cases}$$

For an instance, the item which has the largest score would never get one in each

competition such that from Eq. (29), its rank position is 1. Thus, our goal is to make the gap between the approximation and the real condition as close as possible. Due to the definition of the logistic function used to simulate the rank positions, while η is set to a larger number, the results of competitions of two items are close to the real condition. The relation is depicted in Figure 3-2. The larger η is, the more precise the approximation is. With the increasing of η, it loses its differentiability gradually. However, due to the non-differentiability of $\langle \pi^q(i,j) \rangle$, if η is set too large, though the approximation is precise, the derivatives of the objective function might be too large to process for the computer because of the occurrence of overflow of double precision floating point numbers.

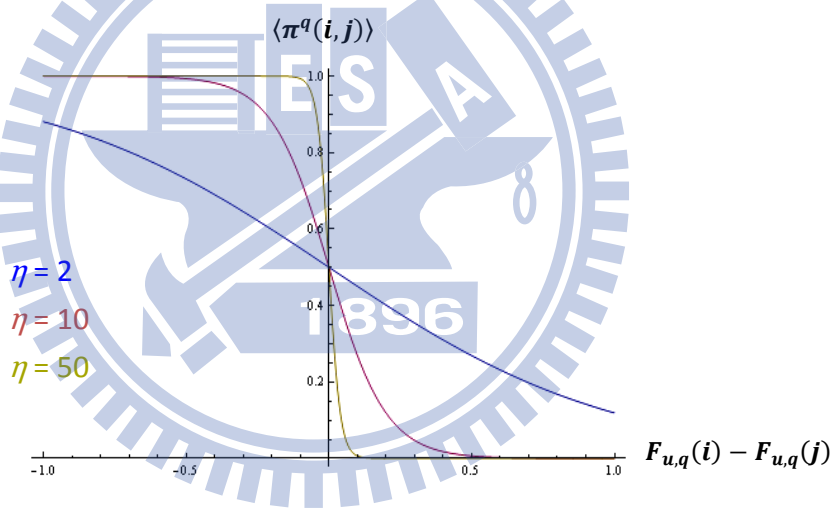

Figure 3-2 The relation between η and the precipitation of the logistic function.

Using the above approximation for $\langle \pi^q(i) \rangle$, $\bar{\mathcal{H}}(u, Q, F)$ could be written as

$$\bar{\mathcal{H}}(u, Q, F) = \frac{1}{|Q|} \sum_{q \in Q} \frac{1}{|I_u|} \sum_{i \in I_u} \frac{2^{r_{u,q}(i)} - 1}{\log(2 + A_i^q)} \tag{30}$$

where

$$A_i^q = \sum_{j \in I_u, i \neq j} \frac{1}{1 + \exp\left[\eta \left(F_{u,q}(i) - F_{u,q}(j)\right)\right]} \tag{31}$$

By the Taylor expansion around $x = 0$ while $\langle \pi^q(i) \rangle > 0$, we could get the

inequation in Eq. (32).

$$\frac{1}{\log\left(2+A_i^q\right)} \geq \frac{1}{\log(2)} - \frac{A_i^q}{2[\log(2)]^2} \tag{32}$$

, $\bar{\mathcal{T}}(u,Q,F)$ can be obtained by simplifying $\bar{\mathcal{H}}$. Hence $\bar{\mathcal{T}}$ can be written as in Eq. (33).

$$\bar{\mathcal{T}}(u,Q,F) = \frac{1}{|Q|}\sum_{q\in Q}\frac{1}{|I_u|}\sum_{i\in I_u}\left(2^{r_{u,q}(i)}-1\right)\left\{\frac{1}{\log(2)} - \frac{A_i^q}{2[\log(2)]^2}\right\} \tag{33}$$

$\bar{\mathcal{H}}$, a logarithmic function which is hard to get its differential function in the optimization phase, has been approximated as a polynomial function $\bar{\mathcal{T}}$. However, there is a conspicuous difference between $\bar{\mathcal{H}}$ and $\bar{\mathcal{T}}$ depicted in Figure 3-3. The NDCG measures may be affected by the rank position of items in the form of logarithm. On the other hand, if we approximate it by Taylor expansion, it may be affected linearly. From the definition of the NDCG function, we know that the relevance of the item in the first rank has the most influence to the NDCG measures. With the increasing of the rank position, the influence of the relevance of an item decreases gradually in the form of logarithm. Thus, if we use the objective function without approximation by the Taylor expansion, in the optimization phase, the optimized model may prefer to push a relevant item to the first rank while others may be left far behind. Nevertheless, through the approximation by the Taylor expansion, the optimization by the objective function may push each item forward equally.

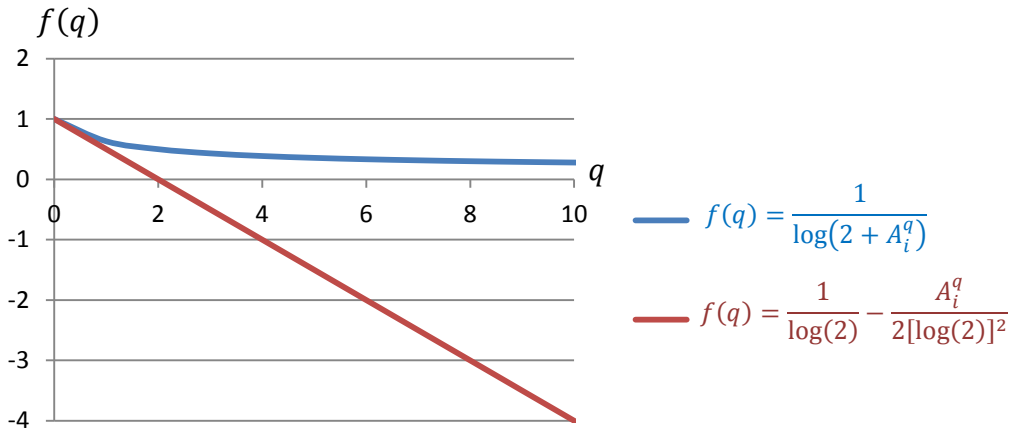

Figure 3-3 The NDCG measures may be affected by the rank position of items in the form of logarithm. On the other hand, if we approximate it by Taylor expansion, it may be affected linearly.

To simplify $\bar{\mathcal{T}}$, we neglect the constant terms, and then we could obtain the objective function in Eq. (34).

$$\mathrm{M}(u,Q) \approx \sum_{q\in Q}\sum_{i\in I_u}\left(2^{r_{u,q}(i)}-1\right)\sum_{j\in I_u, i\neq j}\frac{1}{1+\exp\left[\eta\left(F_{u,q}(i)-F_{u,q}(j)\right)\right]} \quad (34)$$

Because we want to maximize the NDCG, in the training phase, the objective function would be minimized. In our implementation, we use the BFGS algorithm [4, 7, 8, 9, 23] to find the optimization result. There are only 3 parameters that we have to train, and then we calculate the inverse Hessian matrix directly rather than approximate iteratively like the L-BFGS [14] algorithm does. Because $F_{u,q}(i)$ is polynomial, we could rewrite $\left(F_{u,q}(i)-F_{u,q}(j)\right)$ as $F_{u,q,i,j}(\alpha,\beta,\gamma)$ where α,β,γ are our parameters to be optimized. Thus the derivatives of $\mathrm{M}(u,Q)$ with respect to α,β,γ could be computed independently. For example, the derivatives of $\mathrm{M}(u,Q)$ with respect to α could be written as in Eq. (35).

$$\frac{\partial \mathrm{M}(u,Q)}{\partial\alpha} \approx \sum_{q\in Q}\sum_{i\in I_u}\left(2^{r_{u,q}(i)}-1\right)\sum_{j\in I_u, i\neq j}\frac{-\eta\exp\left[\eta\left(F_{u,q}(i)-F_{u,q}(j)\right)\right]}{\left\{1+\exp\left[\eta\left(F_{u,q}(i)-F_{u,q}(j)\right)\right]\right\}^2}\cdot\frac{\partial F_{u,q,i,j}(\alpha,\beta,\gamma)}{\partial\alpha} \quad (35)$$

Our objective function is not convex, so that the gradient descent methods may not

find the global minimum. We resolve the problem of local minimum by using several different start points and then what causes the minimum value of objective function would be selected as the optimized parameters.

Chapter 4

Methodology

4.1 Data Preparation

The dataset we use is from the LibraryThing collected by M. Clements et al. [5, 6]. LibraryThing is a social network about books. A user could give all the books ratings and tags and then personalized catalogs are created. According to the preference of a user, the system would give him a list where users may have similar interests to him or recommend books he may like.

After pruning the books and tags that appear in less than 5 user profiles [5], there are 7279 users, 10559 tags and 37232 books. The pruned dataset has 2056487 UIT relations. The derived $\mathbf{UT}$, $\mathbf{R}$, $\mathbf{IT}$ matrices have a density 5.2×10^{-3}, $2.8 \times 10^{-3}$and 2.0×10^{-3}, respectively. However, many users tag and rate items repeatedly with a little difference. We use the rating that the user gives to the item in the last record of the UIT relations and accumulate all the tags that the user gives to the item in the dataset. Thus, the real density of matrices would be a little less than the one stated above.

In Figure 4-1, we split the data into two parts, namely training set and testing set. To split the $\mathbf{D}$ matrix into two parts, we choose the UIT relations given by the first 3000 users as the training set. The others would be taken as the testing set. Thus, in training set, there are 3000 users, 8009 tags and 36596 items, while there are 4280 users, 8071 tags and 37101 items in the testing set. In training phase, we use the training set to optimize the parameter vector of our model and we validate the performance of the optimized model in the testing phase.

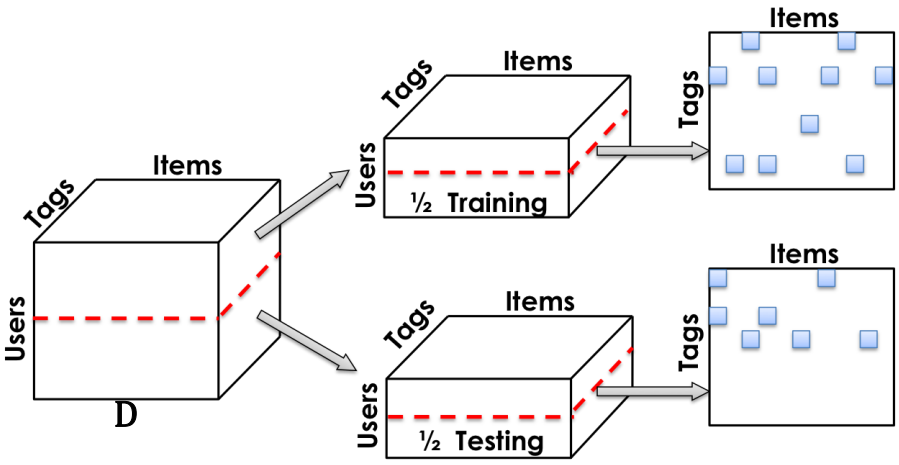

Figure 4-1 The data preparation by splitting the D matrix into two parts.

4.2 Measures for Evaluation

4.2.1 The Assumption for Relevance

The data we use for the training phase are UIT relations which only represent the condition of tagging rather than the results of recommendation. Due to the lack of the ground truth of recommendation, to determine the items relevant or not, we assume that if the user u has used the tag q to tag the item i which has been tagged by u, the tag q is irrelevant to the item i for the user u. Because of the difference of the cognition on the same word between different individuals, in the language system of the user u, q may not refer to i, even though q is relevant to i in common conditions. It may cause the difference more obvious when a polysemous word is taken as a query.

If we presumed to take other tags, which are relevant in a common case, relevant, it might harm the precision of recommendation. Moreover, to consider other tags which have not been used by the target user yet we have to cluster tags before evaluation so that the experiment would be more complicated. If we expanded the relevant tags for evaluation, we would consequently use another model to cluster tags prior to ours that

it would fall into a trap of the circular evaluation. Hence we would use the tags which have been tagged by a user rather than expand them for evaluation.

Besides, is it possible that the user u does not tag the item i with the tag q which really is relevant to i for u? Figure 4-2 shows the average distribution of tagging frequency per user. There are about 10 tags whose frequencies are more than the average frequency. The tags which are familiar to a user are not many. While tagging, a user may use a familiar tag on hand rather than an unfamiliar one unless the user tags an unfamiliar item. Nevertheless, in both the training phase and the testing phase, we only use the query whose tagging frequency is superior to the average for the target user.

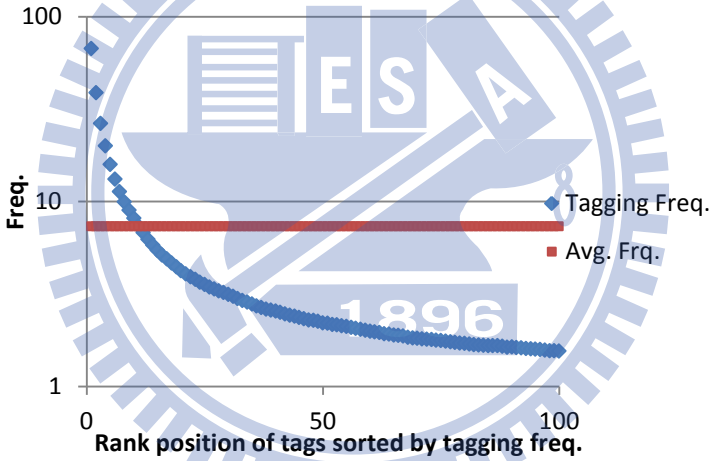

Figure 4-2 The average distribution of tagging frequency per user.

4.2.2 Normalized Discounted Cumulative Gain

To evaluate the suitability of the predicted content ranking for each item ranking task we use the Normalized Discounted Cumulative Gain (NDCG) proposed by Järvelin, K. and Kekäläinen, J. [12]. The basic concept of NDCG is that highly relevant items appearing lower in the resulting ranked list ought to be penalized. The modified relevance value is reduced logarithmically proportional to the rank position. Thus, the summation of the first p items' modified relevance values is called

Discounted Cumulative Gain(DCG), which is defined as follows

$$\mathrm{DCG}(p) = \sum_{rank=1}^{p} \frac{2^{rel(rank)} - 1}{\log_2(1 + rank)} \tag{36}$$

where $rank$ stands for the rank position, $rel(rank)$ is the relevance value of an item whose rank position is $rank$. Besides, to compare the performance of retrieval of different queries, the normalization across queries is needed. The Ideal Discounted Cumulative Gain (IDCG) is introduced to represent the ranked list from a prefect ranking algorithm by which the resulting permutation is sorted by the relevance scores of items. Practically we sort items by their relevance to obtain the IDCG and thus we could compute the NDCG which is defined as follows

$$\mathrm{NDCG}(p) = \frac{\mathrm{DCG}(p)}{\mathrm{IDCG}(p)} \tag{37}$$

Due to lack of the ground truth of the relevance scores to the query, M. Clements et al. [5, 6] create a gain vector $\mathbf{g}$ with length $|I|$ (i.e., all items) of zeros. To prevent from predicting content that has received a low rating, in this gain vector, the predicted rank positions of the held-out validation items that correspond to a positive opinion $\mathbf{r} \in \{3, 3.5, 4, 4.5, 5\}$ are assigned a value of respectively $\mathbf{g} \in \{1, 2, 3, 4, 5\}$. In other words, an item whose rating is small is taken as irrelevant. However the rating of an item does not map to its relevance score of it directly because there is no relation between quality (i.e., rating) and relevance. Given the condition stated below, an item with rating value 2.5 is relevant, and another item with rating value 5 is irrelevant. According to the evaluation in [5, 6], the relevant item would be neglected because of its small rating value, while the irrelevant one is taken to be contributive to the suitability of the predicted content ranking. Hence we rewrite the assumption that the rating of an item could map to its relevance score on the premise that the item is relevant.

We directly use the ratings of items which are relevant to the query as their relevance in the NDCG. According to our assumption of relevance stated above, relevant items are included by the ones the target user has tagged. Moreover, we assume that the rating of an irrelevant item is assigned 0. Only the items tagged by the target user u could be relevant or not to the query q for u. By considering all items in the list (i.e., $p = |I|$), Discounted Cumulative Gain (DCG) now accumulates the values of the discounted gain for each item:

$$\mathrm{DCG}(u,q) = \sum_{i \in I_u} \frac{2^{r_{u,q}(i)} - 1}{\log_2\left(1 + \pi^q(i)\right)} \tag{38}$$

where I_u stands for the set of items that the target user u has tagged, $\pi^q(i)$ is the rank position of the item i from the query q and $r_{u,q}(i)$ is the rating of the item i which is relevant to q for u.

The DCG value is normalized by dividing by the optimal DCG value, i.e., IDCG, which is computed using a static state vector in descending order. The NDCG could be written as in Eq. (39).

$$\mathrm{NDCG}(u,q) = \frac{1}{\mathrm{IDCG}} \sum_{i \in I_u} \frac{2^{r_{u,q}(i)} - 1}{\log_2\left(1 + \pi^q(i)\right)} \tag{39}$$

In our experiment we use the mean of the NDCG over all validation users. The mean of the NDCG over all validation users could be written as in Eq. (40).

$$\overline{\mathrm{NDCG}}(U) = \frac{1}{|U|} \sum_{u \in U} \overline{\mathrm{NDCG}}(u, Q_u) = \frac{1}{|U|} \sum_{u \in U} \frac{1}{|Q_u|} \sum_{q \in Q_u} \mathrm{NDCG}(u,q) \tag{40}$$

where U is the set of all validation users and Q_u is the query set for the target user u.

With the increasing of the rank position, the influence of the relevance of an item on the NDCG decreases gradually in the form of logarithm. It is possible that the ranking results in a large value of the NDCG while the precision and recall are small.

The NDCG cannot be used independently while evaluating the performance in information retrieval. Hence, we also use precision and recall.

4.2.3 Precision and Recall

In information retrieval, precision is the fraction of retrieved instances that are relevant, while recall is the fraction of relevant instances that are retrieved. Precision and recall are defined as follows

$$\text{precision} = \frac{|\{\text{relevant items}\} \cap \{\text{retrieved items}\}|}{|\{\text{retrieved items}\}|} \tag{41}$$

$$\text{recall} = \frac{|\{\text{relevant items}\} \cap \{\text{retrieved items}\}|}{|\{\text{relevant items}\}|} \tag{42}$$

The relation between precision and recall is shown in Figure 4-3.

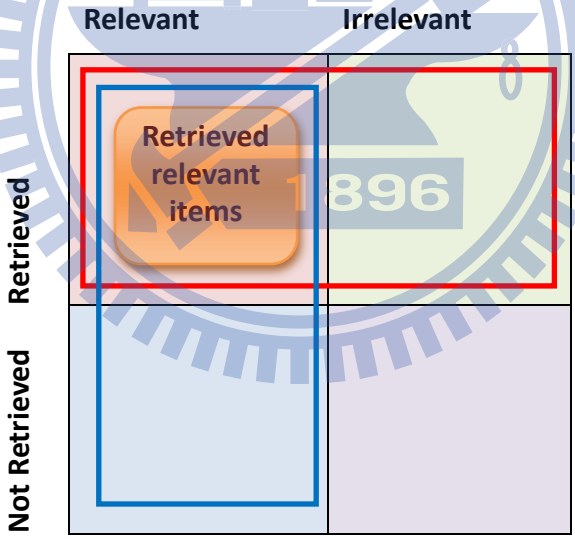

Figure 4-3 Precision and recall are the quotient of the upper left region with orange color by respectively the region with red boundary and the one with blue boundary.

A perfect precision score of 1.0 means that all the items retrieved by the search engine are relevant and a prefect recall score of 1.0 means that all the relevant items are retrieved. Notice that the two statements do not mention how many items are

retrieved. If a search engine only retrieves the item which has the highest rank score, the item would be relevant almost certainly. On the contrary, if a search engine retrieves all items whatever the query is, it always obtains the recall score of 1.0. Thus, it does not suggest that only one of the two measures is used and the other is neglected.

4.3 Evaluation

To make the recommendation system more practical, the items not only tagged by the user but also those untagged should be retrieved. Therefore we propose a pre-evaluation protocol which is modified from [13]. In every experiment we follow the pre-evaluation protocol as follows. For each individual user u in the dataset we randomly select a list of 20% of the items the user u has tagged and take them as "unseen" items which we refer to as $\boldsymbol{S_{u,t}}$. We set zeros to the elements relative to these "unseen" tags in $\mathbf{R}$ and $\mathbf{R}^{\mathbf{T}}$, and subtract one from the elements relative to these "unseen" tags in **UT**, **TU**, **IT** and **TI**. $\boldsymbol{S_{u,a}}$ is the remaining 80% of items the user u has tagged and $\boldsymbol{S_{u,n}}$ is the set of the items that the user u has not tagged yet. The protocol stated above is depicted in Figure 4-4.

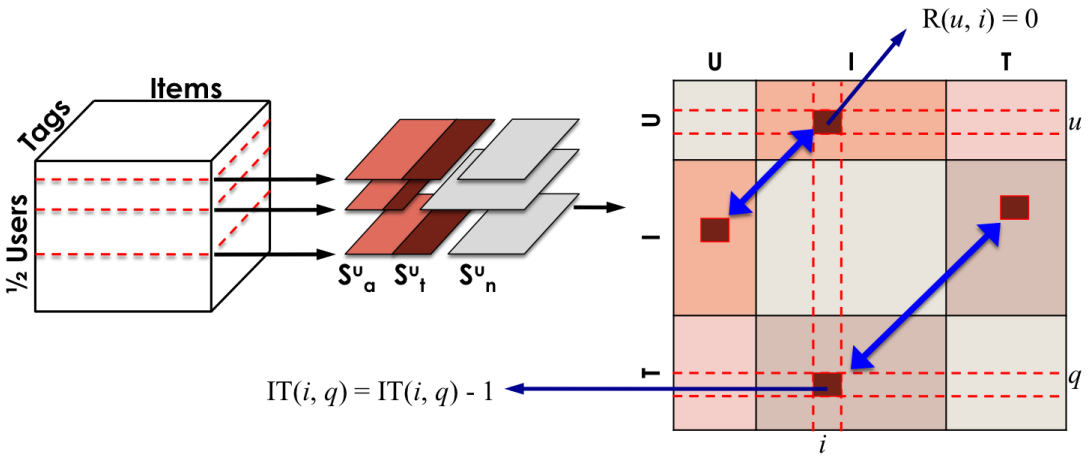

Figure 4-4 For each user, the per user pre-evaluation protocol is followed: $S_{u,t}$ is the

20% randomly selected set of items that the user u has tagged, $S_{u,a}$ is the remaining 80% of items the user u has tagged and $S_{u,n}$ corresponds to the items that the user u has not tagged yet.

Then we use TD-IDF weighting shown in Eq. (20-25) on each 2-dimension matrix with normalization to reduce the influence of frequently occurring elements. After the normalization in all 2-dimension matrices, we combine these 2-dimension matrices in the transition matrix $\mathbf{A}_{\boldsymbol{u}}$. We use our PageRank-like model in Eq. (26) to calculate the scores of all items iteratively. While the scores of items converge, the static scores are obtained. We evaluate the performance of prediction from the static scores of items. Notice that before calculating the scores of items, the rating of items which belong to $\boldsymbol{S_{u,t}}$ are set to zero, thus the construction of the transition matrix $\mathbf{A}_{\boldsymbol{u}}$ is on the premise that the user u pretends that u has not tagged the items which belong to $\boldsymbol{S_{u,t}}$ before.

Notice that our random walk model in the training phase is the same with that in the testing phase. The only difference between the training phase and the testing phase is the evaluation protocol. The evaluation protocol in the training phase computes the objective function and that in the testing phase evaluates the performance. The series of steps of the construction of transition matrix could be taken as the pre-evaluation protocol. By this pre-evaluation protocol, we could split the UIT relations into two parts. One includes the UIT relations, which does not involve u or involves the items that belong to $\boldsymbol{S_{u,a}}$, while the other is $\boldsymbol{S_{u,t}}$. The former is taken as the historical information that supports recommendation, while the later could be taken as the ground truth. To estimate the suitability of the recommendation of items which a user has not seen, we would check the relevant items which are retrieved and belong to $\boldsymbol{S_{u,t}}$. If our recommendation system can find most of the relevant items which belong

to $\boldsymbol{S_{u,t}}$, the items which the user might like but has not seen yet could be retrieved. On the other hand, due to the usage of the recommendation system, we want the relevant items seen before are still retrieved as many as possible. In both training phase and testing phase, whether the retrieved items belong to $\boldsymbol{S_{u,t}}$ or $\boldsymbol{S_{u,a}}$, we would treat them equally in the evaluation.

For each individual user u, we select the tags that u has used as the queries $\boldsymbol{Q_u}$. If a tag t used by u infrequently would be taken as a query, the precision of recommendation depends mostly on the per user pre-evaluation protocol. Because if i_t which is tagged by u with t, is selected as an "unseen" item, the value of $\mathbf{UT}(u,t)$and $\mathbf{TU}(t,u)$ would be subtracted by one. Due to the infrequent occurrence of the relation between u and t, the value of $\mathbf{UT}(u,t)$and $\mathbf{TU}(t,u)$ are small. The small value is sensitive to addition and subtraction.

Each query q in the query set $\boldsymbol{Q_u}$ is the input of our model and the permutation of items sorted by their scores in the static state vector is the output of our model. The objective function is computed from the output of our random walk model. When the user u has completed the query process, i.e. every query in the query set $\boldsymbol{Q_u}$ is used as the input of our training model once, we combine their objective function by summation.

Our objective function is based on the NDCG [12]. If we took the ratings of items as their relevance to a query, the result would be interfered with by irrelevant items which we do not recognize. Therefore, to consider not only the NDCG but also precision of an item, we define what a relevant item is. An item which has been tagged by the user with the query is relevant. Now the evaluation that combines the NDCG and the precision would be proposed as follows.

After computing the static scores, we divide all items into two parts, namely relevant items and irrelevant items. Notice that the relevant item set and irrelevant one

are independent of $\boldsymbol{S_{u,a}}$ and $\boldsymbol{S_{u,n}}$. Because we do not know whether the user u may like the items which have not seen by u before, $\boldsymbol{S_{u,n}}$ is neglected. We focus on the items that the user u has tagged, i.e. the items which belong to $\boldsymbol{S_{u,t}}$ or $\boldsymbol{S_{u,a}}$. Moreover, the items which belong to $\boldsymbol{S_{u,t}}$ or $\boldsymbol{S_{u,a}}$ could be divided into three parts, namely $\boldsymbol{S_{u,t}^{q}}$, $\boldsymbol{S_{u,a}^{q}}$ and $\boldsymbol{S_{u,n}^{q}}$, which are defined respectively as follows.

$\boldsymbol{S_{u,t}^{q}}$: the items which are relevant to the query q and $\boldsymbol{S_{u,t}^{q}} \subseteq \boldsymbol{S_{u,t}}$.

$\boldsymbol{S_{u,a}^{q}}$: the items which are relevant to the query q and $\boldsymbol{S_{u,a}^{q}} \subseteq \boldsymbol{S_{u,a}}$.

$\boldsymbol{S_{u,n}^{q}}$: the items which are irrelevant to the query q and belong to $\boldsymbol{S_{u,n}^{q}} \subseteq \boldsymbol{S_{u,t}} \cup \boldsymbol{S_{u,a}}$.

Figure 4-5 shows the relation among them.

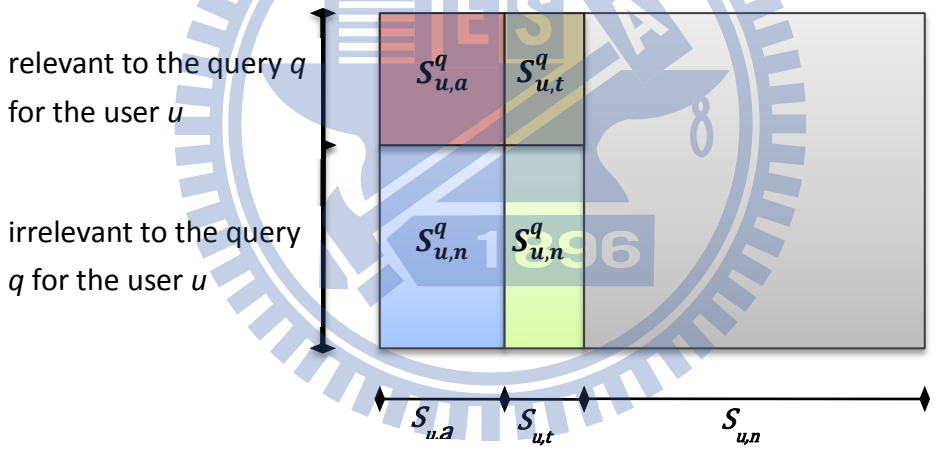

Figure 4-5: The relations between $S_{u,t}$, $S_{u,a}$, $S_{u,n}$, $S_{u,t}^{q}$, $S_{u,a}^{q}$and $S_{u,n}^{q}$.

In training phase, for each item i relevant to q, we use the Eq. (31) to predict the rank position of i by comparing the static score of i with others. Figure 4-6 shows how we compute the objective function. Notice that while computing the objective function, the ratings of the items, which belong to $\boldsymbol{S_t}$, are considered. In other words, we use the ratings in $\mathbf{R}$ which has not been processed by the per user pre-evaluation protocol yet. Besides, the items which belong to $\boldsymbol{S_n}$ are regarded irrelevant, and their ratings are set to zero. According to our assumption, items in $\boldsymbol{S_{u,n}}$ would be

neglected.

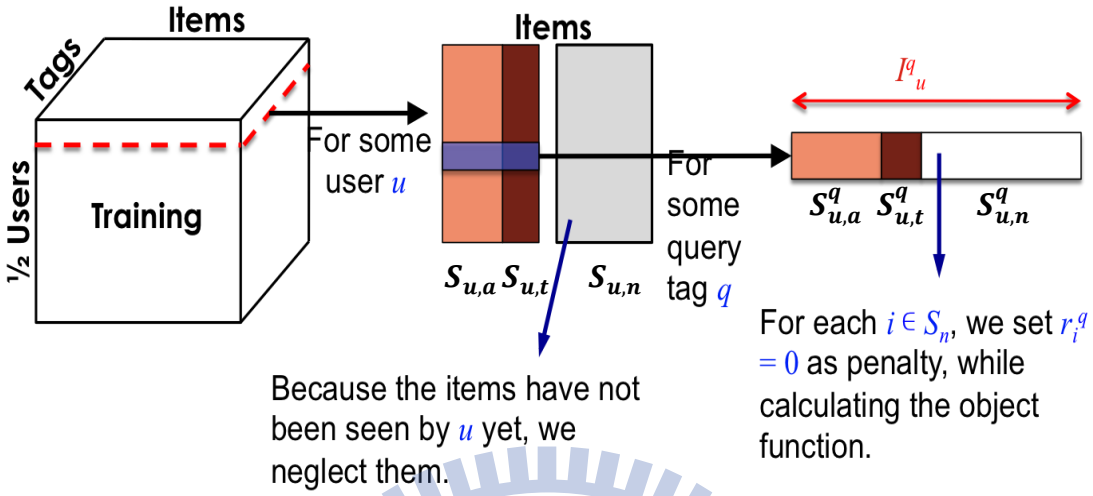

Figure 4-6: Relevant & irrelevant items in the training phase.

We optimize the parameter vector for each individual user rather than sum up the objective function for all users before optimization. In the training phase, each per user objective function is not concave; neither does the sum of all the objective functions. To analyze the personal behavior and its influence, we optimize the per user objective function for each user.

From the distribution of the per user optimized parameters, we could find some characteristics of the user behavior. For example, the parameter θ controls the proportion of the target user to the tag selected as a query. When the value of θ is large, the user is more influential than the query and vice versa. Considering a user who prefers to use ambiguous tags, the initial state may not include the query merely without considering the personal information obtained by including the target user. On the contrary, if the user prefers to use exact tags to describe items, the personal information could almost be neglected ($\theta = 0$). Besides, another condition should be considered. If the preference of the target user is ambiguous, it could not be

contributive to the precision of our model to consider the personal information ($\theta > 0$).

For each element in the per user optimized parameter vector, we compute the mean of the elements in all per user optimized parameter vectors as the global optimized parameter for the testing phase. We are afraid that the distribution of the per user optimized parameter vectors is divergent. If this distribution is divergent, the mean vector could not represent all vectors very well. Hence, we could use multiple parameter vectors rather than one in the testing phase to have a better result. To obtain the multiple optimized parameter vectors which could represent all the per user optimized parameter vectors better, we cluster the per user optimized parameter vectors by K-Means and DBSCAN, respectively. In our experiments, we would compare the performance of the two clustering methods using the same ranking task. The procedure in the training phase is shown in Alg. 1.

Training Phase ($\mathbf{D_{train}}$, $\mathbf{R}$)

foreach user $u \in U$ **do**

select the 20% of the relevant items as $\boldsymbol{S_{u,t}}$, other 80% as $\boldsymbol{S_{u,a}}$ and the others as $\boldsymbol{S_{u,n}}$

$\mathbf{D_u} = \mathbf{D_{train}} - \{\mathbf{D_{train}}(u,i,t) | \forall i \in \boldsymbol{S_{u,t}}\}$

$\mathbf{R_u} = \mathbf{R}$, where $\mathbf{R}(u,i) = 0, \forall i \in \boldsymbol{S_{u,t}}$

compute $\mathbf{UT}'$, $\mathbf{TU}'$, $\mathbf{IT}'$, $\mathbf{TI}'$, $\mathbf{R}'$and $\mathbf{R_T}'$ from (20, 21, 22, 23, 24, 25) by $\mathbf{D_u}$ and $\mathbf{R_u}$

build transition matrix $\mathbf{A_u}$ by $\mathbf{UT}'$, $\mathbf{TU}'$, $\mathbf{IT}'$, $\mathbf{TI}'$, $\mathbf{R}'$and $\mathbf{R_T}'$

select tags as queries $\boldsymbol{Q_u}$

$\mathrm{M}(u, Q_u) = 0$

foreach query $q \in \boldsymbol{Q_u}$ **do**

get the static score vector $F_{u,q}$ by $\mathbf{A_u}$, where $|F_{u,q}| = |I|$

$\boldsymbol{S_t}$: $\boldsymbol{S_t} \subseteq \boldsymbol{S_{u,t}}$ && $\boldsymbol{S_t}$ is relevant to q for u.

$\boldsymbol{S_a}$: $\boldsymbol{S_a} \subseteq \boldsymbol{S_{u,a}}$ && $\boldsymbol{S_a}$ is relevant to q for u.

$\boldsymbol{S_n}$: $\boldsymbol{S_n} \subseteq \boldsymbol{S_{u,t}} \cup \boldsymbol{S_{u,a}}$ && $\boldsymbol{S_n}$ is irrelevant to q for u.

$sum = 0$

foreach $i \in \boldsymbol{S_t} \cup \boldsymbol{S_a}$ **do**

$isum = 0$

foreach $j \in \boldsymbol{S_t} \cup \boldsymbol{S_a} \cup \boldsymbol{S_n}$ **do**

if $(i \neq j)$ **then** $isum = isum + \frac{1}{1+\exp\left[\eta\left(F_{u,q}(i) - F_{u,q}(j)\right)\right]}$

$sum = sum + \left(2^{\mathbf{R}(u,i)} - 1\right) * isum$

$\mathrm{M}(u, Q_u) = \mathrm{M}(u, Q_u) + sum$

set initial value $\mathbf{w}^{(0)}$

$t = 1$

while *not converge* **do**

execute the gradient descent method on $\mathrm{M}(u, Q_u)$ to get new $\mathbf{w}^{(t)}$ from $\mathbf{w}^{(t-1)}$

$t = t + 1$

recover the setting of the per user pre-evaluation protocol.

return $\mathbf{w} = \mathbf{w}^{(t)}$

Algorithm 1 The procedure in the training phase.

In the testing phase, we rank all the items according to their static scores in descending order. The ranked item list is the output of our model. To evaluate the performance of the output, we split the items into two parts, namely relevant and irrelevant. We only consider the relevant items and count in their ratings when computing the NDCG. The irrelevant items would be neglected. The definition of a relevant item in the training phase is the same as that in the testing phase.

To reinforce the suitability of our model, we compute the static scores according to

different optimized parameter vectors, which are obtained from the result of clustering, and thus we could get different ranked item list. We select the ranked item list which has the highest value of Recall@10 as the output of our model. The procedure in the testing phase is shown in Alg. 2.

Testing Phase ($\mathbf{D}_{\text{test}}$, R)

foreach user $u \in U$ **do**

select the 20% of the relevant items as $\boldsymbol{S_{u,t}}$, other 80% as $\boldsymbol{S_{u,a}}$ and the others as $\boldsymbol{S_{u,n}}$

$\mathbf{D_u} = \mathbf{D_{test}} - \{\mathbf{D_{test}}(u,i,t) | \forall i \in \boldsymbol{S_{u,t}}\}$

$\mathbf{R_u} = \mathbf{R}$, where $\mathbf{R}(u,i) = 0, \forall i \in \boldsymbol{S_{u,t}}$

compute $\mathbf{UT'}$, $\mathbf{TU'}$, $\mathbf{IT'}$, $\mathbf{TI'}$, $\mathbf{R'}$and $\mathbf{R_T}'$ from (20, 21, 22, 23, 24, 25) by $\mathbf{D_u}$ and $\mathbf{R_u}$

build transition matrix $\mathbf{A_u}$ by $\mathbf{UT'}$, $\mathbf{TU'}$, $\mathbf{IT'}$, $\mathbf{TI'}$, $\mathbf{R'}$and $\mathbf{R_T}'$

select tags as queries $\boldsymbol{Q_u}$

foreach query $q \in \boldsymbol{Q_u}$ **do**

get the static score vector $F_{u,q}$ by $\mathbf{A_u}$, where $|F_{u,q}| = |I|$

sort items according to the $F_{u,q}$ in descent order

$\boldsymbol{S_t}$: $\boldsymbol{S_t} \subseteq \boldsymbol{S_{u,t}}$ && $\boldsymbol{S_t}$ is relevant to q for u.

$\boldsymbol{S_a}$: $\boldsymbol{S_a} \subseteq \boldsymbol{S_{u,a}}$ && $\boldsymbol{S_a}$ is relevant to q for u.

$\boldsymbol{S_n}$: $\boldsymbol{S_n} \subseteq \boldsymbol{S_{u,t}} \cup \boldsymbol{S_{u,a}}$ && $\boldsymbol{S_n}$ is irrelevant to q for u.

$NDCG, Recall@10, Recall@20, Recall@30, Recall@40, Recall@50 = 0,$

foreach $i \in \boldsymbol{S_t} \cup \boldsymbol{S_a}$ **do**

$$NDCG = NDCG + \frac{2^{r_{u,q}(i)}-1}{\log_2(1+\pi^q(i))}$$

If $(\pi^q(i) \leq 10)$ **then** $Recall@10$++

If $(\pi^q(i) \leq 20)$ **then** $Recall@20$++

If $(\pi^q(i) \leq 30)$ **then** $Recall@30$++

If $(\pi^q(i) \leq 40)$ **then** $Recall@40{+}{+}$

If $(\pi^q(i) \leq 50)$ **then** $Recall@50{+}{+}$

recover the setting of the per user pre-evaluation protocol.

return $\mathbf{w} = \mathbf{w}^{(t)}$

Algorithm 2 The procedure in the testing phase.

In order to consider the rank positions of the relevant items in the result list of all items, the rank position of an item is considered according to the ranked list in which all items are sorted by their static scores. Figure 4-7 shows how we obtain the ranked result of items and evaluate the NDCG value according to the static state vector including both relevant and irrelevant items. While computing the NDCG, we sort all items by their rating to obtain the perfect ranked list as IDCG. However the items that the target user has not tagged before (i.e., belong to $\boldsymbol{S_{u,n}}$) are not irrelevant certainly, the DCG would be underestimated a little and so as to the IDCG. Because of the logarithmic descent of the DCG value proportional to the position of the result, the underestimation of the IDCG value would be more than that of the DCG values.

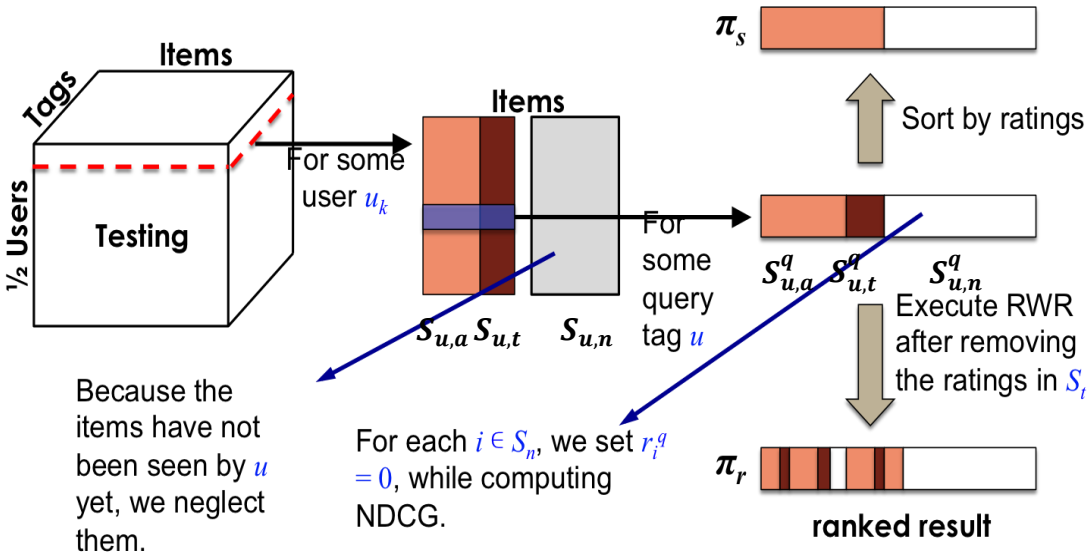

Figure 4-7 Relevant & irrelevant items in the testing phase.

Chapter 5

Experiments

In this chapter we discuss the results we got from all the experiments performed. We examine the performance of the parameter estimation and then compare Supervised FolkRank with other methods.

5.1 Distribution of optimized parameter vectors

Figure 5-1 shows the distribution of the 2-tuple (θ, α) per user optimized parameter vectors, where θ stands for the ratio of the target user to the tag selected as a query and α stands for the restart probability. Most of the optimized parameter vectors prefer small value of θ. We could explain that most of users prefer to use more explicit tags to describe items. When the query is polysemic, we may concern the behavior of the user to find what the query means for the user. Thus, if a user usually uses ambiguous tags for tagging, the random walk-based recommendation system may not get sufficient information by almost only concerning about the query (i.e., θ is small). However, a user usually has different interests, and may tag what he or she doesn't like, and therefore user behaviors are divergent.

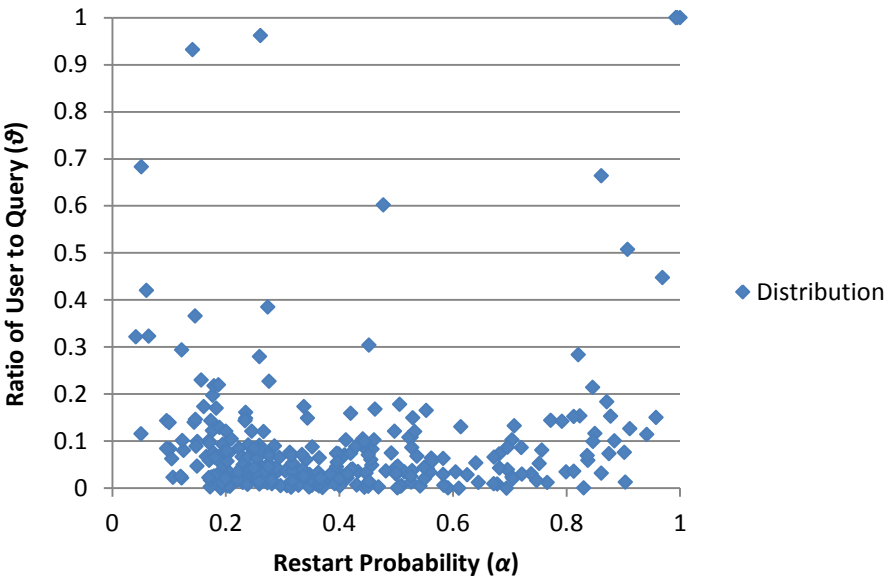

Figure 5-1 The distribution of the 2-tuple per user optimized parameter vector.

Notice that the distribution of these per user optimized parameter vectors do not concentrate tightly. It is impossible to find a parameter vector to fit all users. Therefore, to obtain more suitable recommendation, we would not only use a single optimized parameter vector but also multiple parameter vectors. We divide the optimized parameter vectors by clustering and then select the representative vector for each cluster. Each cluster could represent a sort of user behavior. In our experiments, there are two clustering methods to cluster these per user optimized parameter vectors. For each cluster, we compute the mean vector where each element is the mean of the elements in all optimized parameter vectors in the cluster. Figure 5-2 and 5-3 shows the clustering result by K-Means and DBSCAN, respectively.

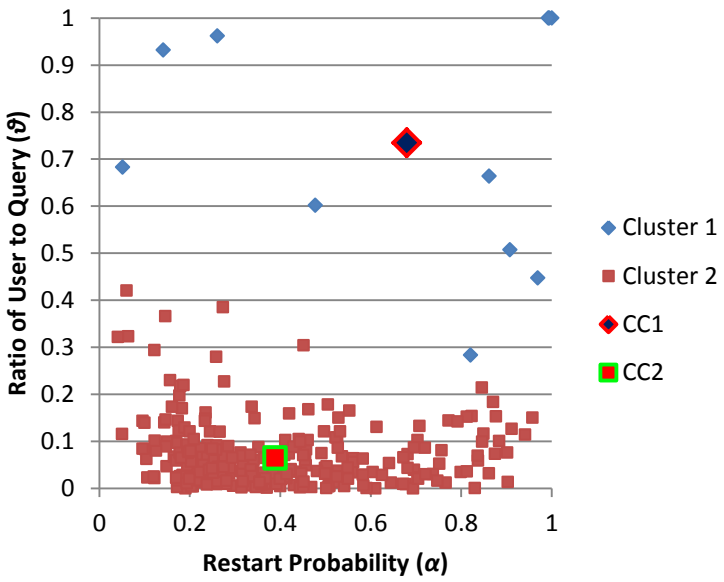

Figure 5-2 The clustering result by K-Means. The per user optimized parameter vector is 2-tuple (θ, α).

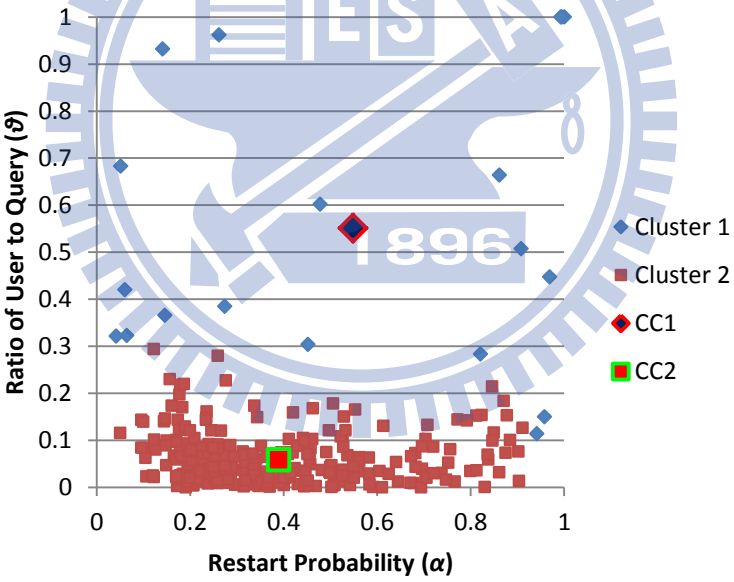

Figure 5-3 The clustering result by DBSCAN. The per user optimized parameter vector is 2-tuple (θ, α).

For K-Means, the distribution of the 2-dimension relation graph can be clustered into two sets, and then we could give the initial set of K-Means where the values of θ are separated. On the other hand, for DBSCAN, either the initial means of clusters or the number of clusters we don't need to give. DBSCAN requires two parameters: the

search radius (eps) and the minimum number of points required to form a cluster (minPts). Both of the parameters are easier to be tuned. From our experiments, the result by these two clustering methods are similar. One cluster locates at the area where the ratio of user (i.e., θ) is quite small (i.e., 0 ~ 0.2), and the other could be taken as the outliers. Because the outliers are too many to be neglected, the mean vector of all outliers is still used as a parameter vector for recommendation. Thus, our random walk model using this parameter vector would fit the users who are taken as outliers because of their linking behaviors.

Figure 5-4 shows the distribution of the 3-tuple (θ, α, γ) per user optimized parameter vectors, where the three parameters stand for the ratio of user to query, the restart probability and the self-transition probability, respectively. Like the distribution of the 2-tuple vectors, most of the optimized parameter vectors prefer small θ. We still explain that most of users prefer to use more explicit tags to describe items. Moreover, from the projection of α-γ plane, there are three clusters: one locates along the line α+γ=0.8, another locates near the y-axis (i.e., $\alpha \approx 0$) and the other locates on the x-axis (i.e., $\gamma \approx 0$). The first cluster infers that, the walk goes forward with the probability of 0.2 whatever the ratio between α and γ is. Most of vectors are located in this cluster. The second cluster shows that the Supervised FolkRank (SFR) could be reduced to the self-transition model that M. Clements et al. propose [5]. As shown in Figure 5-5, the distribution of PageRank-based model always assign the highest value to the nodes closest to the starting position, while in the lazy random walk model the distant nodes are more relevant [6].

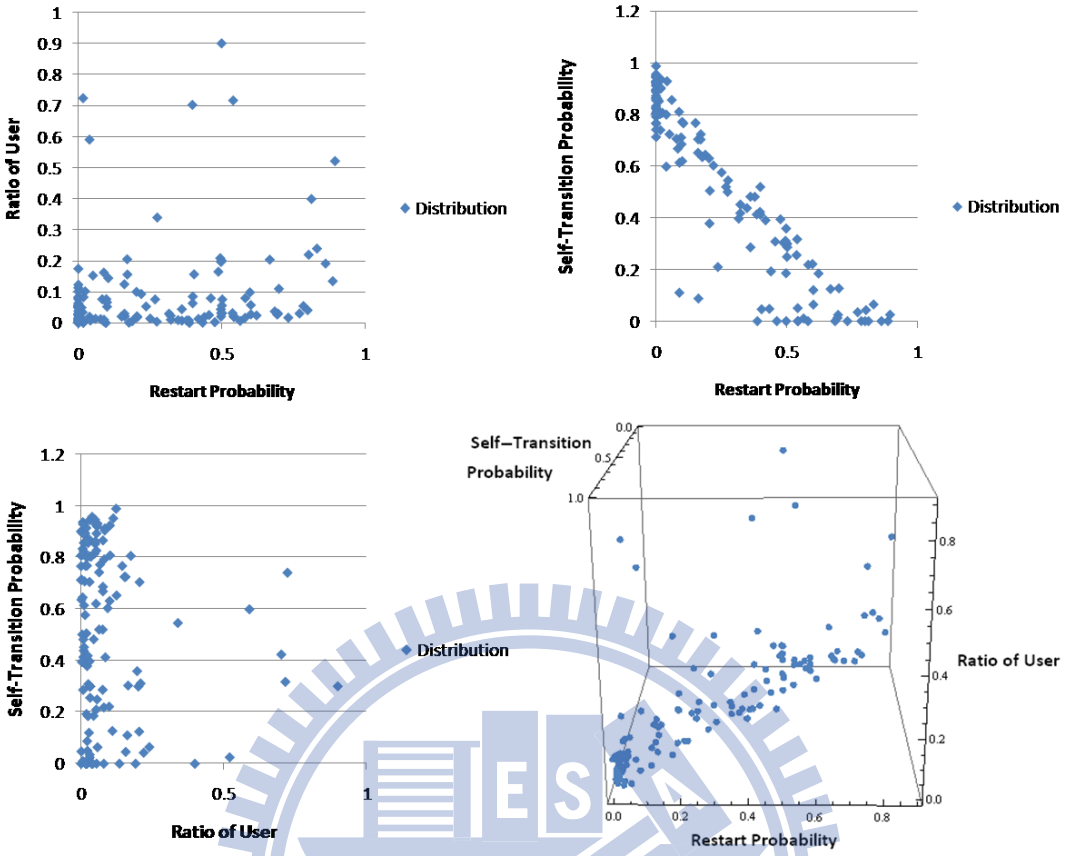

Figure 5-4 The distribution of the 3-tuple $(\theta,\ \alpha,\ \gamma)$ per user optimized parameter vector. (a) The projection of $\theta-\alpha$ plane. (b) The projection of $\gamma-\alpha$ plane. (c) The projection of $\gamma-\theta$ plane. (d) The distribution of $\theta,\ \alpha$ and γ.

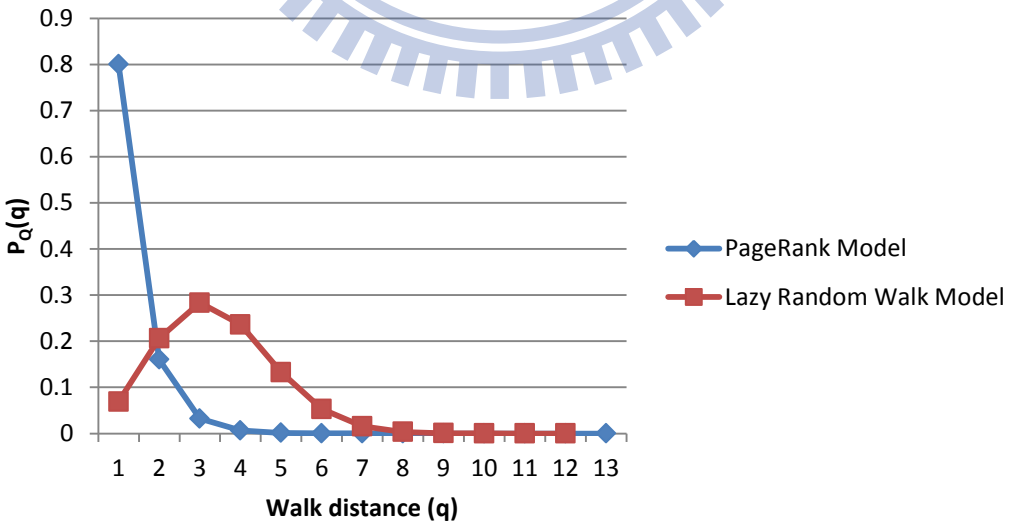

Figure 5-5 The Probability mass function (PMF) of the walk distance after a fixed number of steps through the social graph, for restart probability and self-transition probability are both 0.8.

Unlike the distribution of the 2-dimension relation graph, the distribution of 3-dimension graph cannot be figured out easily. Moreover, the initial set of K-Means would affect the clustering. Figure 5-6 and Figure 5-7 show that the result of clustering by K-Means and DBSCAN. Among a multi-dimension relation graph, it needs other algorithm to find an initial set of K-Means. Without further information about the data distribution, the number of clusters is unknown. Moreover, the K-Means divides nodes into clusters according to distance and we know that the distance-based clustering methods do not adapt to certain data sets, whose clusters are not circle-like, very well. From Figure 5-4, the distribution could not be divided into circle-like clusters. For DBSCAN, three clusters are obtained without assigning the number of clusters. The clustering result is similar to our discussion stated above. If we use the centers of clusters obtained by DBSCAN as the initial set of K-Means, the result by K-Means is similar. Again, the mean vector of all outliers is used. The search radius (eps) and the minimum number of points required to form a cluster (minPts) could be predicted according to the range of each dimension.

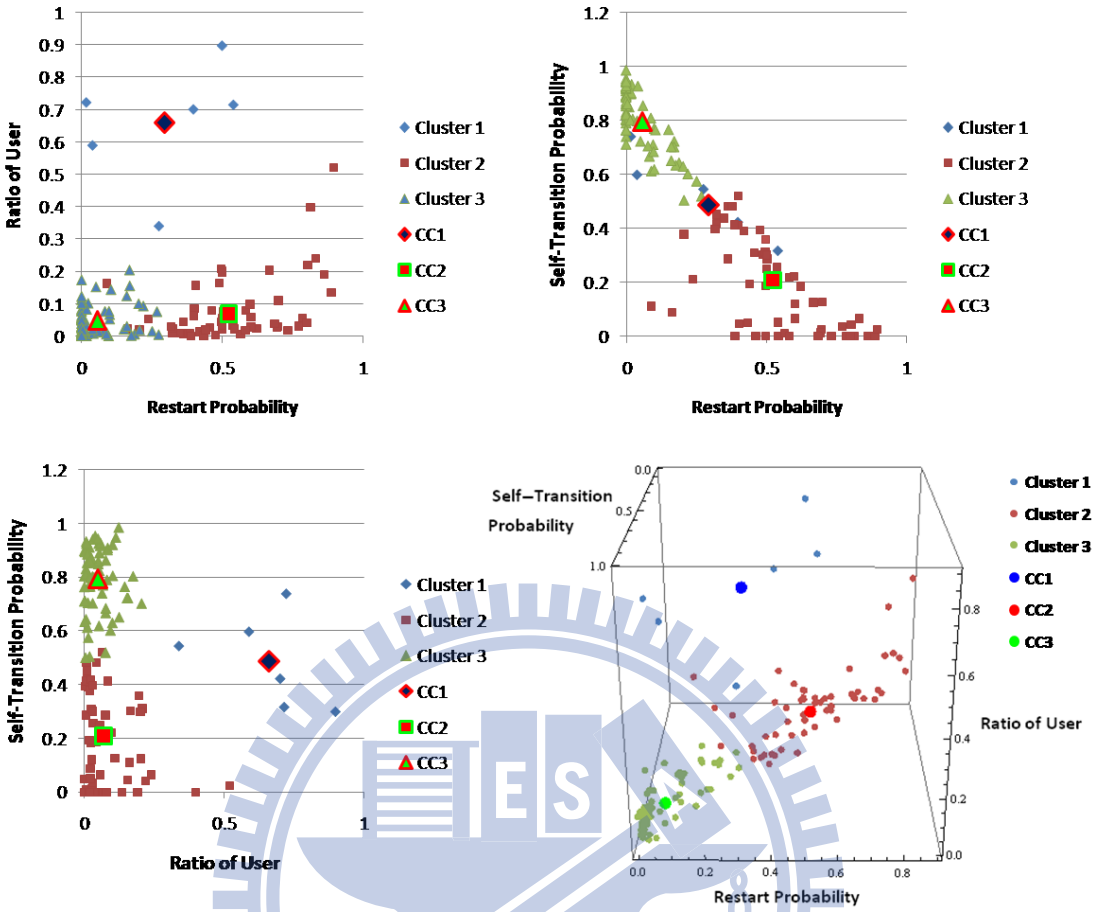

Figure 5-6 The clustering result by K-Means. The per user optimized parameter vectors are 3-tuple $(\theta,\ \alpha,\ \gamma)$. (a) The projection of $\theta - \alpha$ plane. (b) The projection of $\gamma - \alpha$ plane. (c) The projection of $\gamma - \theta$ plane. (d) The distribution of θ, α and γ.

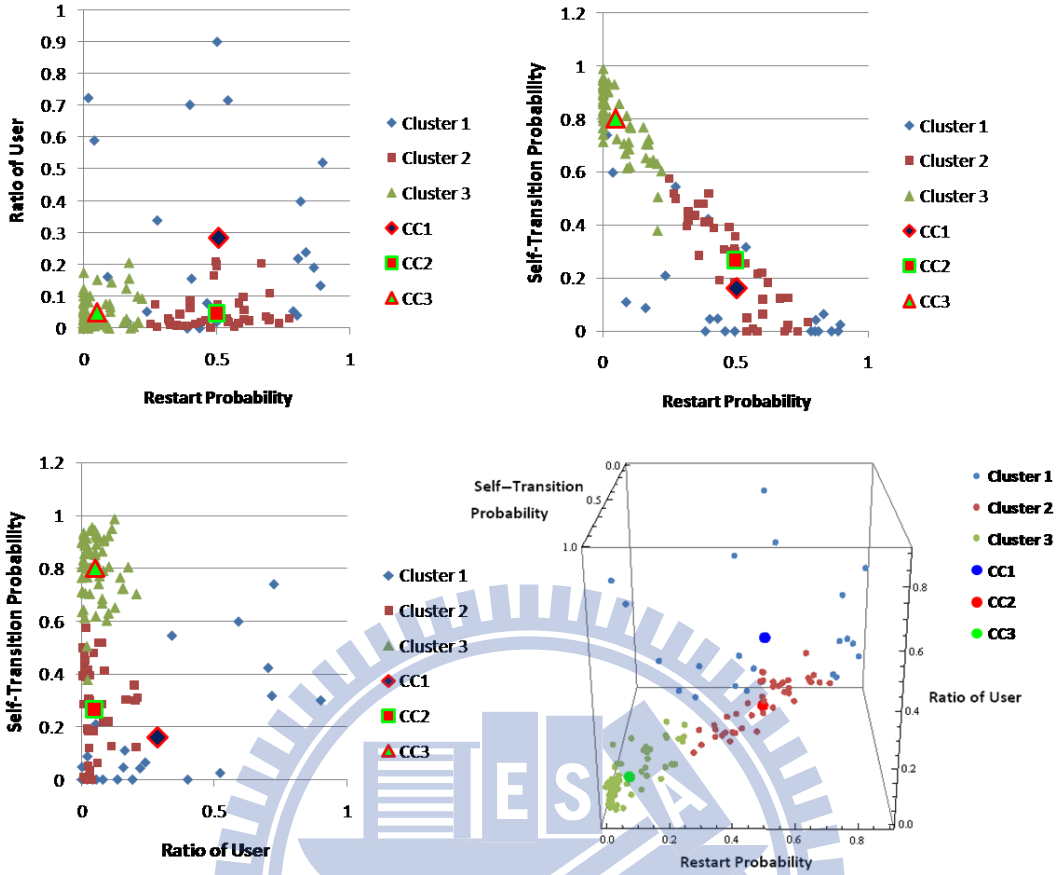

Figure 5-7 The clustering result by DBSCAN. The per user optimized parameter vectors are 3-tuple $(\theta,\ \alpha,\ \gamma)$. (a) The projection of $\theta-\alpha$ plane. (b) The projection of $\gamma-\alpha$ plane. (c) The projection of $\gamma-\theta$ plane. (d) The distribution of θ, α and γ.

5.2 Comparison with other methods

We compare the predictive performance of Supervised FolkRank with other methods including supervised and unsupervised methods. Table 5-1 shows the result for the LibraryThing dataset in terms of NDCG. SFR_2 represents Supervised FolkRank in 2-dimension. SFR_2_Merge represents SFR_2 with multiple optimal parameters. SFR_3 represents Supervised FolkRank in 3-dimension. SFR_3_Merge represents SFR_3 with multiple optimal parameters. SRW represents Supervised Random Walk. LRW represents Lazy Random Walk. uLRW represents Lazy Random Walk using the asymmetric transition matrix. RWR represents Random Walk with

Restart. BM25_CF represents Collaborative Filtering with BM25 model.

The first seven methods are supervised. The seventh method, called Supervised Random Walk (SRW), proposed by L. Backstrom et al. [2] bias the random walk by assigning the transition probability of each edge. In SRW, there are several attributes to describe edges. The weights of attributes can be annotated as the edge feature vector, where the length equals to the number of attributes. By tuning the weights in the edge feature vector, the random walk would visit the relevant items more likely. In our case, edges have only one value i.e., the length of the edge feature vector is 1. Therefore, we modify the problem formulation of SRW. The parameters we tune are the same as our model rather than the weights of elements in the edge feature vector. Nevertheless, we still use their method for optimization.

	NDCG
SFR_2	0.703407
SFR_2 _DBSCAN	0.711197
SFR_2 _KMeans	0.7095989
SFR_3	0.7044617
SFR_3_DBSCAN	0.712282
SFR_3_KMeans	0.7081372
SRW	0.6983309
RWR	0.6896602
URW	0.6552698
LRW	0.5976361
uLRW	0.3833435
BM25_CF	0.6619233

Table 5-1 Results of NDCG for the LibraryThing dataset.

The last five methods are unsupervised. Lazy Random Walk proposed by M. Clements et al. [5] walks in an undirected graph. Due to wondering the difference of predictive performance between directed graph and undirected one, we use this method in the directed graph, annotated as LRW. uLRW is the origin method in [5]. RWR (heuristic) is the random walk with restart model where the parameters is tuned heuristically while all the parameters in RWR (unweighted) is set to 0.5. The last method annotated as BM25 – CF is the collaborative filtering with the BM25 model.

First, comparing the result of NDCG, our models (SFR_2, SFR_2_KMeans, SFR_3, SFR_3_DBSCAN) achieve a significant improvement over other methods while SFR_3_DBSCAN performs the best. In terms of NDCG, SFR_3_DBSCAN gets 8.7% relative improvement to the unweighted random walk model and 3.2% relative improvement to the random walk model where the parameter vector is tuned heuristically. Except for our models, only two methods, namely SWR and BM25 - CF, could get the NDCG values over 0.7, while the worst method is LRW using an asymmetric transition matrix. The unsupervised random walk model, whose parameter vectors are unweighted or tuned heuristically, gets the NDCG value over 0.65. We explain that the random walk-based models perform well, and can be reinforced through optimizing the parameter vectors by machine learning. Besides, BM25 - CF performs better than the unsupervised random walk models.

However, due to the logarithmic decent of the NDCG value proportional to the rank position, the result of recommendation may get a high NDCG value by giving a relevant item at the top rank while other relevant items get low rank positions. Hence, we would compare precision and recall respectively.

Table 5-2 and Figure 5-8 show the result of precision, where P@n stands for the precision at top n. For P@10, BM25_CF performs the best and the second is

FR_3_DBSCAN. However for P@20, the best and the second are FR_3_DBSCAN and FR_3_DBSCAN. In terms of NDCG and P@10, though the relevant items retrieved by BM25_CF are more, the rank positions of the relevant items retrieved by FR_3_DBSCAN precede the rank positions of the relevant items by BM25_CF. Notice that in terms of P@20 and P@30, the result of BM25_CF outperforms SFR_2 and SFR_3. Hence, by using only one parameter vector, the adaption may not cover the linking behaviors of all users. Comparing with all PageRank-based models, SFR_3_DBSCAN gets 21.1% improvement relative to the modified Supervised Random Walk model and 42% relative improvement to the random walk model where the parameter vector is tuned heuristically.

precision / model	P@10	P@20	P@30	P@40	P@50
SFR_2	0.4575253	0.318664	0.241182	0.192541	0.15967
SFR_2_DBSCAN	0.4764794	0.3264877	0.24471	0.1943747	0.160565
SFR_2_KMeans	0.4725113	0.3251586	0.244013	0.19375	0.160328
SFR_3	0.4597796	0.3200149	0.241895	0.192908	0.159917
SFR_3_DBSCAN	0.4883557	0.3320404	0.24529	0.194049	0.160116
SFR_3_KMeans	0.4739067	0.3253684	0.24247	0.19257	0.159522
SRW	0.4031109	0.286734	0.224481	0.184143	0.155311
RWR	0.3439359	0.243693	0.19375	0.162473	0.13926
URW	0.3238302	0.236082	0.190182	0.159995	0.137823
LRW	0.355818	0.26024	0.205359	0.169973	0.144820
uLRW	0.1484502	0.140693	0.129866	0.117421	0.106468
BM25_CF	0.488969	0.325332	0.242129	0.192276	0.159488

Table 5-2 Results of the precision for the LibraryThing dataset at the rank position 10,

20, 30, 40 and 50, respectively.

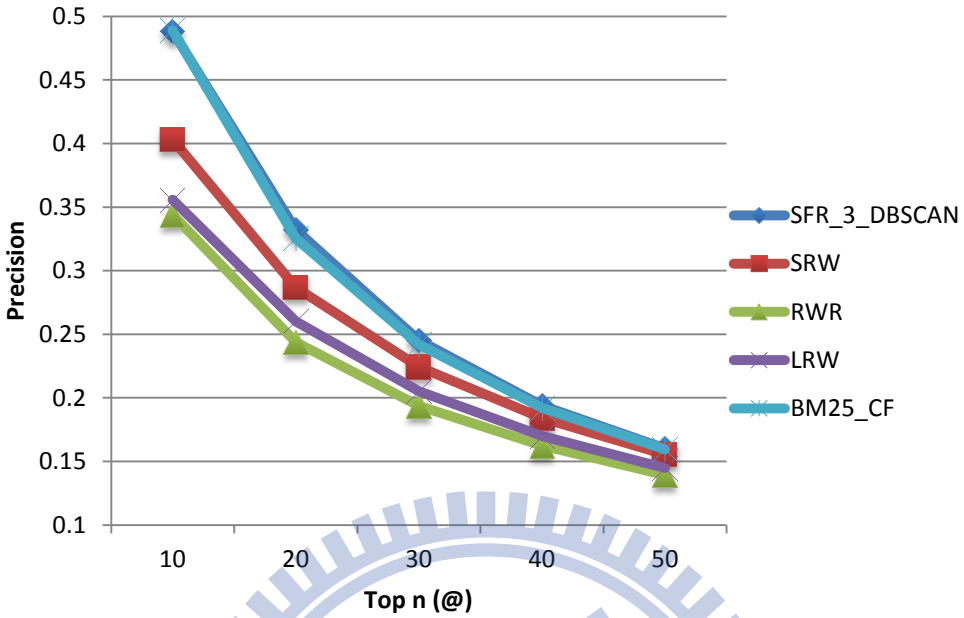

Figure 5-8 Performance of Supervised FolkRank and other methods in terms of precision. Notice that SFR and BM25-based Collaborative Filtering perform similarly.

Table 5-3 and Figure 5-9 show the result of recall, where R@n stands for the recall at top n. In terms of recall at top 10 and 20, SFR_2_DBSCAN and SFR_3_DBSCAN are the best two methods, while at top 40 and 50, SFR_2_DBSCAN and SFR_2_KMeans performs the best. In terms of R@10, SFR_3_DBSCAN gets 20% relative improvement to the modified Supervised Random Walk model and 44.9% relative improvement to the random walk model where the parameter vector is tuned heuristically. Though BM25_CF performs well in terms of precision, SFR_2_DBSCAN and SFR_3_DBSCAN both outperform BM25_CF in terms of recall.

model \ recall	R@10	R@20	R@30	R@40	R@50
SFR_2	0.6202254	0.7760892	0.8463632	0.8846496	0.9081167

SFR_2_DBSCAN	0.64548	0.792365	0.855814	0.890612	0.911547
SFR_2_KMeans	0.6441519	0.7936372	0.8578678	0.8918045	0.9115475
SFR_3	0.6234566	0.7791072	0.8482266	0.8858644	0.9090599
SFR_3_DBSCAN	0.660385	0.803793	0.857673	0.889815	0.909688
SFR_3_KMeans	0.6419471	0.7904646	0.8498648	0.8845853	0.9070997
SRW	0.550053	0.7065371	0.7985334	0.8562116	0.8907223
RWR	0.4558299	0.6017594	0.7011443	0.7721735	0.8175204
URW	0.4348568	0.587865	0.6923667	0.7642607	0.8119878
LRW	0.4851459	0.6560767	0.749214	0.8089915	0.8482832
uLRW	0.1913874	0.3544839	0.4873156	0.5817193	0.6520374
BM25_CF	0.630947	0.7623796	0.8306035	0.871583	0.8982363

Table 5-3 Results of the recall for the LibraryThing dataset at the rank position 10, 20, 30, 40 and 50, respectively.

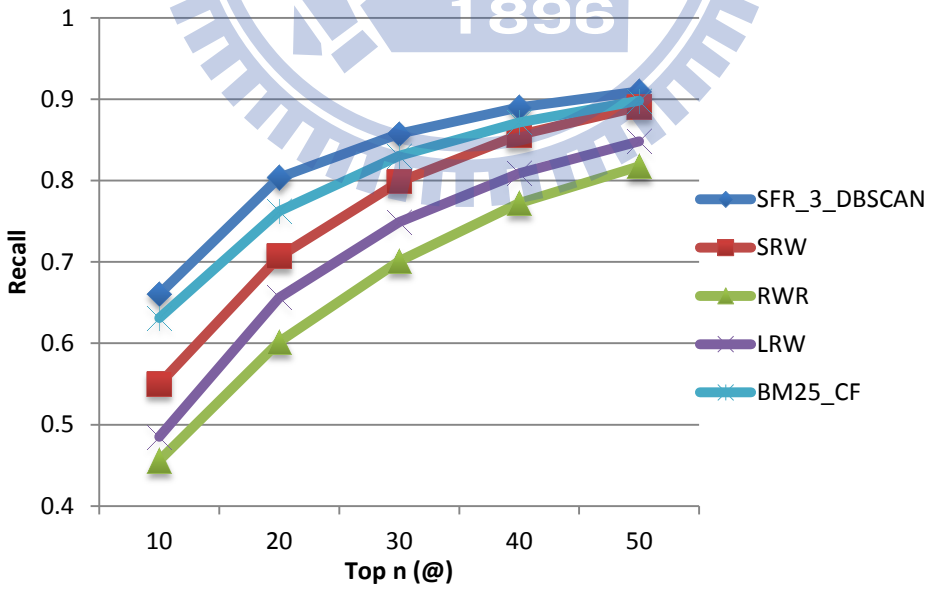

Figure 5-9 Performance of Supervised FolkRank and other methods in terms of recall. SFR outperforms other methods.

5.3 The influence of transition matrix

The relation in folksonomies could be basically regularized as a ternary relation between users, tags, and items. All co-occurrences of users and items, items and tags, users and tags are projected from the ternary relation to undirected and weighted edges in a social graph.

To reduce the influence of frequent occurring elements in a 2-dimension matrix, IDF-TF weighting is used on each matrix. [5] normalize the matrices, namely **UT**, **IT** and **R**, and then combine these sub-matrices and their transposes in the transition matrix **A**. In our model, we compute **TU**, **TI** and $\mathbf{R_T}$ respectively rather than the transpose matrices. Notice that by our definition, $\mathbf{R_T}$ is not the transpose of **R**.

We compare the results of SFR_3_DBSCAN that uses asymmetric transition matrix and symmetric one, denoted as ASYM and SYM as shown in Table 5-4 and Figure 5-10. ASYM gets 31.4% improvement in terms of NDCG. In terms of precision, ASYM gets 52.76% improvement, while SYM also gets 35.67% improvement in terms of recall. Because of our definition of relevance, the tags relevant to an item for the target user are fewer so that the difference of the results between SYM and ASYM diminishes with the growing of the reference items in Figure 5-10.

	ASYM	SYM
NDCG	0.72029	0.54814
P@10	0.49273	0.26204
P@20	0.33497	0.21104
P@30	0.24714	0.17799
P@40	0.19511	0.15288

P@50	0.16084	0.13268
R@10	0.66977	0.34355
R@20	0.81274	0.53298
R@30	0.86606	0.65824
R@40	0.8961	0.73974
R@50	0.91486	0.79141

Table 5-4 Results of ASYM and SYM in terms of NDCG, precision and recall. The model that uses the transition matrix that we modify outperforms the other.

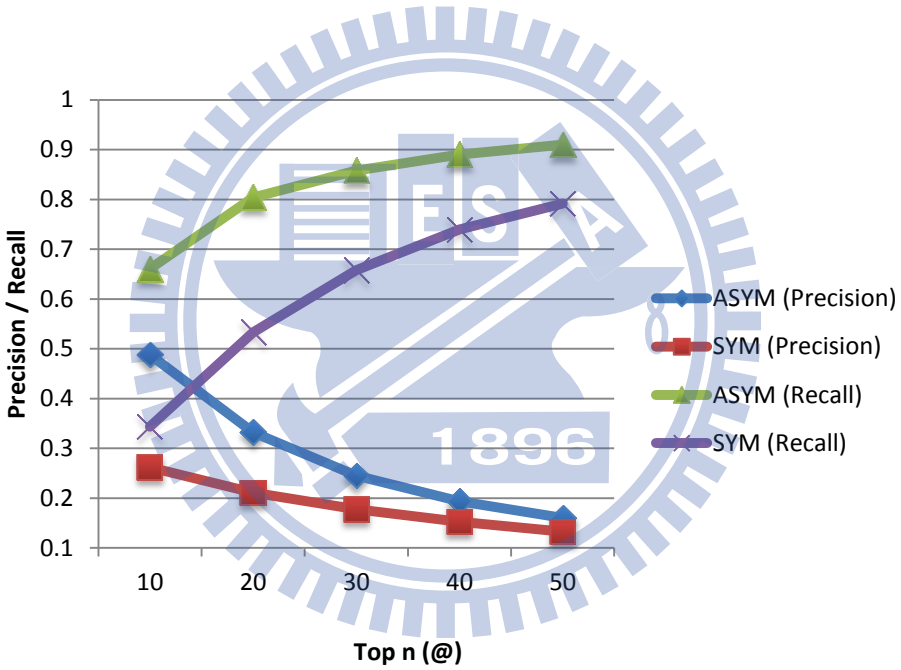

Figure 5-10 Results of ASYM and SYM in terms of precision and recall. ASYM outperforms SYM.

By the definition of normalization, the value of an element in the matrix is different from that in its transpose. The influence of a frequent occurring element could be reduced logarithmically proportional to the sparsity of the elements located in the same column. Thus, the normalization that we modify can represent the local linking relation around the elements more precisely.

Chapter 6

Conclusions and Future Work

We have proposed a novel learning model, the Supervised FolkRank (SFR), for link recommendation in social tagging networks. By approximating the NDCG to be the objective function, we consider the rank position of each item rather than split items into two sets. Moreover, to make our model reliable, we define the relevance of an item so that items that a target user has never tagged before would be pruned. Both the search space of relevant and irrelevant items would be reduced to the items that the target user has tagged before.

The transition matrix in our model is similar to most of random walk-based methods in the social tagging networks [5, 6, 11, 13]. However, our transition matrix is not symmetric. We argue that the asymmetric transition matrix would adapt to the real condition though edges in the social graph are undirected. The Supervised FolkRank provides two types. We use the random walk with restart model as our basic type, and we introduce the probability of self-transition to our model to combine the PageRank-like model and the Lazy Random Walk model. While computing the objective function in the training phase, by our definition of relevance, we compute the NDCG-based objective function where the rating of a relevant item is taken as the relevance score. Thus, the irrelevant items would not be counted in to affect the predictive result.

By optimizing the parameters of our model, we analyze the linking behavior of a user that would affect the result much. Due to the divergence of users' linking behaviors, we argue that the prediction by PageRank-based model with only one parameter vector may not adapt to the real datasets. Thus, by clustering, we could find the representatives for each cluster by computing the mean of each cluster. Each

cluster represents a sort of user behavior. While recommending, we use the parameter vector, which belongs to the cluster that is similar to the target user. Thus, the prediction could be enhanced.

Experiments on LibraryThing demonstrate good performance of the Supervised FolkRank. Comparing with supervised (e.g., modified Supervised Random Walks) and unsupervised methods, Supervised FolkRank (SFR) outperforms other methods. The list-wise learning method we utilize could obtain more precise distribution than the pair-wise one such as Supervised Random Walks. Besides, due to the learning techniques, SFR could make reliable prediction without the requirement of network features discovery and extraction.

Supervised FolkRank is a robust model that can be applied to the problems which require ranking nodes in a social tagging graph, such as keyword search and recommendation.

Bibliography

[1] Al-Maskari, A., Sanderson, M. and Clough P. The Relationship between IR effectiveness measures and user satisfaction. In *SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, p.p. 23-27, 2007.

[2] Backstrom, L. and Leskovec, J. Supervised Random Walks: Predicting and recommending links in social networks. In *WSDM '11: Proceedings of the 4th International Conference on Web Search and Web Data Mining*, p.p. 635-644, 2011.

[3] Brin, S. and Page, L. The anatomy of a large-scale hypertextual web search engine. In *Computer Networks and ISDN Systems*, 30(1-7):107-117, April 1998.

[4] Broyden, C. G. The convergence of a class of double-rank minimization algorithms. In *Journal of the institute of Mathematics and Its Applications 6*, p.p. 76-90.

[5] Clements, M., Vries, A. P., and Reinders, M. I. J. The influence of personalization on tag query length in social media search. In *Information Processing and Management*, p.p. 403-412, 2010.

[6] Clements, M., Vries, A. P., and Reinders, M. I. J. The Task-Dependent Effect of Tags and Ratings on Social Media Access. In *ACM Transactions on Information Systems*, Vol. 28, No. 4, Article 21, Nov. 2010.

[7] Fletcher, R. A New Approach to variable metric algorithms. In *Computer Journal 13(3)*, p.p. 317-322.

[8] Fletcher, Roger. *Practical methods of optimization* (2nd ed.), New York: John Wiley & Sons.

[9] Goldfarb, D. A family of variable metric updates derived by variational means. In *Mathematics of Computation 24 (109)*, p.p. 23-26.

[10] Herlocker, J. L., Konstan, J. A., Borchers, A., and Riedl, J. An algorithmic framework for performing collaborative filtering. In *SIGIR '99: Proceedings of the 22nd annual international ACM SIGIR conference o Research and development in information retrieval*, p.p. 230-247, 1999.

[11] Hotho, A., Jäschke, R., Schmitz, C. and Stumme, G. Information retrieval in folksonomies: search and ranking. In *ESWC '06: Proceedings of the 3rd*

European conference on The Semantic Web: research and applications, p.p. 411-426, 2006.

[12] Järvelin, K. and Kekäläinen, J. Cumulative gain-based evaluation of IR techniques. In *ACM Transactions on Information Systems* 20, p.p. 422-446, 2002.

[13] Konstas, I., Stathopoulos, V. and Jose, J. M. On social networks and collaborative recommendation. In *SIGIR '09: Proceedings of the 32th annual international ACM SIGIR conference on Research and development in information retrieval*, p.p. 19-23, 2009.

[14] Liu, D. and Nocedal, J. On the limited memory bfgs method for large scale optimization. In *Mathematical Programming*, pp. 45:503-528, 1989.

[15] Liu, T.Y. Learning to Rank for Information Retrieval. *Springer-Verlag Berlin Heidelberg*.

[16] Marinho, L. B., Nanopoulos, A., Schmidt-Thieme, L., Jäschke, R., Hotho, A., Stumme, G. and Symeonidis, P. Social tagging recommendation systems. In *Recommender System Handbook*, p.p. 615-632, 2011

[17] .Milicevic, A. K., Nanopoulos, A. and Ivanovic, M. Social tagging in recommender systems: A survey of the state-of-art and possible extensions. In *Artificial Intelligence Review, Volume 33 Issue 3, March 2010*, p.p. 187-209, 2010.

[18] Parra, D. and Brusilovsky, P. Collaborative filtering for social tagging systems: An experiment with CiteULike. In *Rec*Sys '09: *Proceedings of the 2009 ACM conference on Recommender Systems*, p.p. 237-240, 2009.

[19] Parra-Santander, D. and Brusilovsky, R. Improving collaborative filtering in social tagging systems for the recommendation of scientific articles. In *IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, vol. 1, pp.136-142, 2010.

[20] Qin T., Liu, T.Y., Li, H. A general approximation framework for direct optimization of information retrieval measures. In *Information Retrieval* 13(4), p.p. 375-397, 2009.

[21] Rae, A., Sigurbjörnsson, B. and Zwol, R. Improving Tag Recommendation Using Social Networks. In *RIAO' 10: 9th Recherche d'Information Assistée par Ordinateur Conference Adaptivity, Personalization and Fusion of Heterogeneous Information*, 2010.

[22] Salton, G. and Buckley, C. Term-weighting approaches in automatic text

retrieval. In *Information Processing and Managemen*t, 24(5), p.p. 513-523, 2010.

[23] Shanno, David F. Conditioning of quasi-Newton methods for function minimization. In *Math. Comput. 24(111)*, p.p. 647-656.

[24] Taylor, M., Guiver, J., et al. Softrank: optimising non-smooth rank metrics. In *WSDM '08: Proceedings of the 1st International Conference on Web Search and Web Data Mining*, p.p. 77-86, 2008.

[25] Valizadegan, H., Jin, R., Zhang, R. and Mao J. Learning to rank by optimizing NDCG measure. In *Neural Information Processing Systems*, 2010.

[26] Wu, X., Zhang, L. and Yu, Y. Exploring social annotations for the semantic web. In *WWW '06: Proceedings of the 15th international conference on World Wide Web*, p.p. 417-426, 2006.

[27] Yan, L., Dodier, R., Mozer, M. and Wolniexicz, R. Optimizing classifier performance via an approximation to the Wilcoxon-Mann-Whitney statistic. In *ICML '03: The 20th International Conference on Machine Learning*, p.p. 848-855, 2003.