Content-based and Graph-based Tag Suggestion

Xiance Si, Zhiyuan Liu, Peng Li, Qixia Jiang, Maosong Sun

State Key Lab on Intelligent Technology and Systems
National Lab for Information Science and Technology
Dept. of Computer Science&Technology, Tsinghua University, Beijing 100084, China
{adam.si, lzy.thu, pengli09, qixia.jiang}@gmail.com, sms@tsinghua.edu.cn
http://nlp.csai.tsinghua.edu.cn

Abstract. Social tagging is a popular and convenient way to organize information. Automatic tag suggestion can ease the user's tagging activity. In this paper, we exam both content-based and graph-based methods for tag suggestion using the BibSonomy dataset, and describe our methods for ECML/PKDD Discovery Challenge 2009 submissions . In content-based tag suggestion, we propose a fast yet accurate method named Feature-Driven Tagging. In graph-based tag suggestion, we apply DiffusionRank to solve the problem, and get a better result than current state-of-the-art methods in cross-validation.

1 Introduction

Social tagging, aka, folksonomy, is a popular way to organize resources like documents, bookmarks and photos. Resource, tag and user are three essential parts in a social tagging system, a user uses tags to describe resources. Tag suggestion system eases the process of social tagging. It can suggest tags to new resources based on previous tagged resources.

To promote related research, ECML/PKDD organizes a open contest of tag suggestion systems, named Discovery Challenge 2009 (DC09 in short). In this contest a snapshot of users, documents and tags in the online bookmarking system BibSonomy is provided. Each team trains their suggestion system on the snapshot, and test the performance on the same test dataset. There are 3 tasks in the contest. Task 1 focuses on suggesting tags by the content of the resources, i.e, content-based tag suggestion. Task 2 focuses on suggesting tags by the tripartite links between resources, tags and users, i.e., graph-based tag suggestion. Task 3 puts the suggestion system into real-life situation by integrating it with BibSonomy website, and see which system predicts the user's intention best.

In this paper, we describe our methods for the three tasks. For Task 1 and 3, we propose a fast tag suggestion method called Feature-Driven Tagging (FDT). FDT indexes tags by features, where feature can be word, resource ID, user ID or others. For each feature, FDT keeps a list of weighted tags, the higher the weight, the more likely the tag is suggested by the feature. For a new resource, each feature in it suggests a list of weighted tags, the suggestions are combined according to the importance of features to get the final suggestion. Compared to other methods, FDT provides suggestions faster, and the speed is only related with the number of features in the resource(number of words in the content).

For Task 2, we apply two existing methods, most popular tags and FolkRank, for graph-based suggestion. Furthermore, we propose to use a new graph-based ranking model, DiffusionRank, for tag suggestion. The method of "most popular tags" is the simplest collaborative-filtering based methods. It recommends the most popular tags of the resources used by other users. FolkRank is based on PageRank [1] on user-resource-tag tripartite graph, which was first proposed as a tag suggestion method in [2]. DiffusionRank was originally proposed for combating web spam [3], which has also been successfully used in social network analysis [4] and search query suggestion [5]. DiffusionRank is motivated by the heat diffusion process, which can be used for ranking because the activities flow on the graph can be imagined as heat flow, the edge from a vertex to another can be treated as the pipe of an air-conditioner for heat flow. Compared to PageRank, DiffusionRank provides more flexible mechanism to make the ranking scores related to initial values of the vertices, which is important for graph-based tag suggestion.

The paper is organized as follows. Section 2 formulates the problem of tag suggestion. Section 3 introduces our method for content-based tag suggestion. Section 4 introduces our method for graph-based tag suggestion. Section 5 describes the dataset, experiment settings and the result. Section 6 introduces related work on tag suggestion. Section 7 concludes the paper.

2 Problem Formulation

We adopt the model of social tagging proposed by Jaschke et al [2]. A social tagging data set is defined as a tuple $\mathbb{F} := (U, T, R, Y)$, where U is the set of users, T is the set of tags and R is the set of resources. Y is a ternary relation between U, T and R, $Y \subseteq U \times T \times R$. $(u, r, t) \in Y$ is called a tag assignment, which means user u assigned the tag t to resource r. A resource $r \in R$ can be described with a piece of text, like titles of a paper or user-edited description of a website. We denote the words in the text as $\{w_i\}$.

Resources, users and tags form a graph $G = (V, E)$, where $V = U \bigcup R \bigcup T$, and $E = \{\{u,t\}, \{u,r\}, \{r,t\} | (u,t,r) \in Y\}$. The goal of tag suggestion is to predict the set of tags $\{t\}$ for a given pair of user and resource (u, r).

In related literature, social tags are also called folksonomy, the pair of a resource and a user is also called a post.

3 Content-based Tag Suggestion

In this section, we propose a content-based tag suggestion method named Feature-Driven Tagging(FDT). Briefly speaking, FDT is a voting model, where each feature in the resource votes for their favorite tags, and the final scores of tags are averaged by the importance of the features. Figure 1 illustrates the tagging procedure of FDT, it consists of 3 steps: feature extraction, feature weighting and tag voting. For a resource with content, FDT first extracts features from the content. Features include but are not limited to words, resource ID and user ID. Then, FDT weights each feature by their importance in the resource, we explain different ways to compute the importance of features later in this section. In the voting step, each feature contributes a weighted list of tags, the higher the weight, the more likely we should suggest the tag. Weight of a tag from different features are combined by the importance of each feature, thus creates the final weighted list of tags. In the tagging process, all parameters are indexed by feature, we do not need to iterate over all tags (as in text categorization approaches) or resources (as in neighborhood-based approaches), so it is called Feature-Driven Tagging.

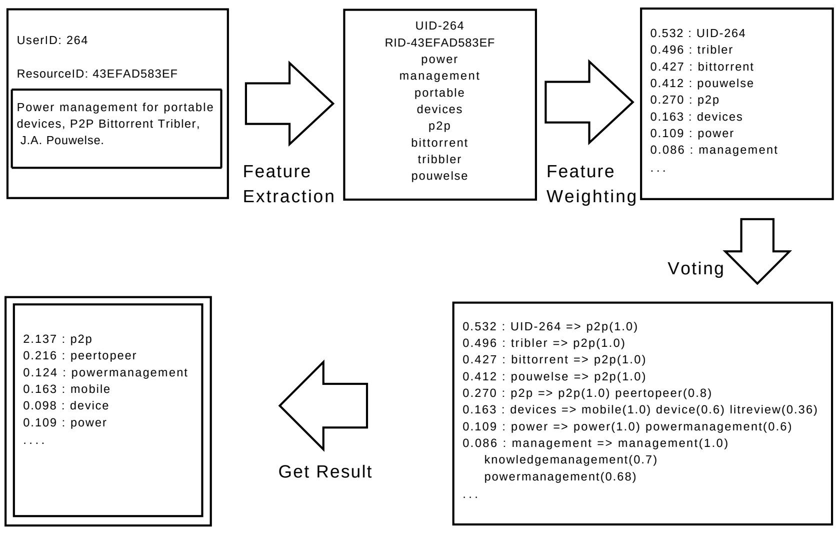
UserID: 264
ResourceID: 43EFAD583EF
Power management for portable devices, P2P Bittorrent Tribler, J.A. Pouwelse.
Feature Extraction
UID-264
RID-43EFAD583EF
power
management
portable
devices
p2p
bittorrent
tribbler
pouwelse
Feature Weighting
0.532 : UID-264
0.496 : tribler
0.427 : bittorrent
0.412 : pouwelse
0.270 : p2p
0.163 : devices
0.109 : power
0.086 : management
...
Voting
0.532 : UID-264 => p2p(1.0)
0.496 : tribler => p2p(1.0)
0.427 : bittorrent => p2p(1.0)
0.412 : pouwelse => p2p(1.0)
0.270 : p2p => p2p(1.0) peertopeer(0.8)
0.163 : devices => mobile(1.0) device(0.6) litreview(0.36)
0.109 : power => power(1.0) powermanagement(0.6)
0.086 : management => management(1.0) knowledgemanagement(0.7) powermanagement(0.68)
...
Get Result
2.137 : p2p
0.216 : peertopeer
0.124 : powermanagement
0.163 : mobile
0.098 : device
0.109 : power
....

Fig. 1. The procedure of Feature-Driven Tagging.

3.1 Feature Extraction

We extract features from different sources. Word features are extracted from textual content of resources, we use them to capture the relationship between words and tags. For bibtex, the textual content is *title* + *bibtexAbstract* + *journal* + *booktitle* + *annote* + *note* + *description*; For bookmark, it is *description* + *extended*. We also include *simhash*1 and the user ID of a resource as a feature. The same publication or website share the same *simhash*1, we use it to capture the tags assigned by other users. We use user ID as a feature so as to model a user's preferences of tagging.

3.2 Compute the Importance of Features

We use two methods to assess the importance of features in a resource. The first and most intuitive one is $TF \times IDF$. $TF \times IDF$ is widely used in information retrieval, text categorization and keyword extraction [6]. We use *log*-version of $TF \times IDF$, which computes as

$$TF \times IDF(f) = log(\frac{n(f)}{N} + 1) * log(\frac{|R|}{df(f)} + 1) \tag{1}$$

where $n(f)$ is the number of occurrences of f in this resource, and N is the total number of features occurred in this resource. $|R|$ is the total number of resources, $df(f)$ is the number of resources f has occurred in. The $+1$ is to avoid zero or negative weights.

The other method we used is $TF \times ITF$, ITF stands for Inverse Tag Frequency, it computes as follows,

$$TF \times ITF(f) = log(\frac{n(f)}{N} + 1) * log(\frac{|T|}{ntag(f)} + 1) \tag{2}$$

where $|T|$ is the total number of tags, and $ntag(f)$ is the number of tags f has co-occurred with. ITF implies that the more tags a feature co-occurs with, the less specific and important the feature is.

3.3 Feature-Tag Correlation

In FDT, each feature is associated with a weighted list of tags. We denote this as a matrix Θ, where $\theta_{i,j}$ is the weight of tag t_j to feature f_i, the size of Θ is $|F| \times |T|$, F is the set of all features. Although Θ is large, it is extremely sparse, so each feature only associates with a small number of tags.

We use three different methods to compute Θ offline, they are co-occurrence count(CC), Mutual Information (MI) and χ^2 statistics (χ^2). Co-occurrence count is computed by

$$CC(f,t) = n(f,t)/n(t) \tag{3}$$

where $n(f,t)$ is the number of co-occurrences of feature f and tag t, and $n(t)$ is the total number of occurrences of tag t. CC is a naive way to find the most important tags for a feature.

In MI, we model each feature or tag as a binary-valued probabilistic variable, the value of which means occur in a document(1) or not(0). Then, we can compute the Mutual Information between a feature and a tag by

$$MI(f,t) = \sum_{f' \in f, \bar{f}} \sum_{t' \in t, \bar{t}} p(f',t') log(\frac{p(f',t')}{p(f')p(t')}) \tag{4}$$

where $f' = f$ means feature f occurs in the resource, and $f' = \bar{f}$ means it doesn't occur, the same is for t'. MI computes the shared information between f and t, the higher it is, the more correlated f and t are.

χ^2 has been used for feature selection in text categorization [7], it also find the correlation between a feature and a category, here we use the tag as category. χ^2 is computed as follows,

$$\chi^2(f,t) = \frac{N(AD - BC)^2}{(A+C)(B+D)(A+B)(C+D)} \tag{5}$$

where $A = n(f,t)$, $B = n(f,\bar{t})$, $C = n(\bar{f},t$, $D = n(\bar{f},\bar{t})$.

After we get Θ by one of the above methods, we make Θ sparse by picking the largest K values in Θ and set other values to 0. We test $K = 30000, 50000$ and 100000, as K increases, the F1 measure increases. When $K > 50000$, the F1-measure doesn't change a lot, so we use $K = 50000$ in all experiments. For each row in Θ, we first find the largest value $\theta_{i,max}$, then set all values in this row to $\theta_{i,j} = \theta_{i,j}/\theta_{i,max}$. We compare the performance of these 3 methods in the experiment section.

FDT has low computation complexity when tagging. For a resource with n features, the complexity of tagging is $O(nm)$, where m is the average tags for each feature in Θ. m is usually a small number, in our model it is 4.63 for bibtex and 5.81 for bookmark. Note that the complexity of FDT is not related to the total number of training documents, tags or users. Nearest neighbor methods have to search in the entire training data set, so the complexity is at least $O(|R|)$. Multi-label classifier methods have to train a classifier for each one of tags, so the complexity is at least $O(|T|)$. Furthermore, the model of FDT is related with K, which is around 10^5, it is small enough to load in the main memory.

4 Graph-based Tag Suggestion

4.1 Method Preliminaries

The basic idea of graph-based tag suggestion is to construct a graph with users, resources and tags as vertices and build edges according to user tagging behaviors. After building the graph, we can adopt some graph-based ranking algorithms to rank tags for a specific user and resource. Then the top-ranked tags are recommended to users.

To describe the graph-based methods more clearly, we first give some mathematical notations. For the folksonomy $\mathbb{F} := (U, T, R, Y)$, we firstly convert it into an undirected tripartite graph $G_{\mathbb{F}} = (V, E)$. In $G_{\mathbb{F}}$, the vertices consists of users, resources and tags, i.e., $V = U \bigcup R \bigcup T$. For each tagging behavior of user u assigning tag t to resource r, we will add edges between u , r and t, i.e., $E = \{\{u,r\},\{u,t\},\{r,t\}|(u,t,r) \in Y\}$.

In $G_{\mathbb{F}}$, we have the set of vertices $V = \{v_1, v_2, \cdots, v_N\}$ and the set of edges $E = \{(v_i, v_j) \mid \text{There is an edge between } v_i \text{ and } v_j\}$. For a given vertex v_i, let $N(v_i)$ be the set of vertices that are neighbors of v_i. We have $w(v_i, v_j)$ as the weight of the edge (v_i, v_j). For an undirected graph, $w(v_i, v_j) = w(v_j, v_i)$. Let $w(v_i)$ be the degree of v_i, and we have

$$w(v_i) = \sum_{v_j \in N(v_i)} w(v_j, v_i) = \sum_{v_j \in N(v_i)} w(v_i, v_j) \tag{6}$$

Based on the graph, we can employ various graph-based ranking methods to recommend tags. In this paper, we first introduce two existing methods, including "most popular tags" and "FolkRank". Furthermore, we propose to use a new ranking model, DiffusionRank, for graph-based tag suggestion.

4.2 Most Popular Tags

We first introduce a simple but effective method for tag suggestion. Some notations are given as below, which is identical with [2]. For a user $u \in U$, we denote all his/her tag assignments as $Y_u := Y \bigcap (\{u\} \times T \times R)$. Accordingly, we have Y_r and Y_t. Based on the same principle, we can define $Y_{u,t} := Y \bigcap (\{u\} \times \{t\} \times R)$ for $u \in U$ and $t \in T$. We also have $Y_{t,r}$ accordingly. Furthermore, we denote all tags that user $u \in U$ have assigned as $T_u := \{t \in T | \exists r \in R : (u,t,r) \in Y\}$.

There are variants of "most popular tags" as shown in [8], which are usually restricted in different statistical range. For example, most popular tags of *folksonomy* recommends the most popular tags of the whole set of folksonomy. Therefore, it recommends the same set of tags for any user and resource, which suffers from cold-start problems and has no consideration on personalization.

A reasonable variant of "most popular tags" is recommending the tags that globally are most specific to the resource. The method is named as most popular tags by *resource*:

$$T(u,r) = \underset{t \in T}{\overset{n}{\operatorname{argmax}}}(|Y_{t,r}|) \tag{7}$$

Since users might have specific preferences for some tags, which should have been used by him/her, thus we can use the most popular tags by *user*. As shown in [8], the performance is poor if we use most popular tags by *user* in isolation. If we mix the most popular tags of *user* and *resource*, the performance will be much better than each of them. The simplest way to mix the effect of users and resources on tags is to add the counts and then sort:

$$T(u,r) = \underset{t \in T}{\overset{n}{\operatorname{argmax}}}(|Y_{t,r}| \times |Y_{u,t}|) \tag{8}$$

4.3 FolkRank

FolkRank is originally proposed in [2] which is based on user-resource-tag tripartite graph. In FolkRank, two random surfer model is employed on the tripartite graph. The ranking values of vertices are computed using the following formula:

$$PR(v_i) = \lambda \sum_{v_j \in N(v_i)} \frac{w(v_j, v_i)}{w(v_j)} PR(v_j) + (1-\lambda) p(v_i) \tag{9}$$

where $PR(v_i)$ is the PageRank value and p_{v_i} is the preference to v_i. Suppose we have an adjacent matrix $\mathbf{A}$ to represent the graph $G_{\mathbb{F}}$:

$$A(i,j) = \begin{cases} 0 & \text{if } (v_i, v_j) \notin E \\ \frac{w(v_i,v_j)}{w(v_j)} & \text{if } (v_i, v_j) \in E \end{cases}$$

With the matrix, we can rewrite the Equation 9 as:

$$\mathbf{s} = \lambda \mathbf{A}\mathbf{s} + (1-\lambda)\mathbf{p} \tag{10}$$

where $\mathbf{s}$ is the vector of PageRank scores of vertices, and $\mathbf{p}$ is the vector of preferences of vertices.

A straightforward idea of graph-based tag suggestion is to set preference to the user and resource to be suggested for, and then compute ranking values using PageRank in Eq. (10). However, as pointed out in [8], this will make it is difficult for other vertices than those with high edge degrees to become highly ranked, no matter what the preference values are.

Based on above analysis, we described FolkRank as follows. To generate tags for user u and resource r, we have to:

1. Let $\mathbf{s}^{(0)}$ be the stable results of Eq. (10) with $\mathbf{p} = \mathbf{1}$, i.e., the vector composed by 1's.
2. Let $\mathbf{s}^{(1)}$ be the stable results of Eq. (10) with $\mathbf{p} = \mathbf{0}$, but p(u) = 1 and p(r) = 1.
3. Compute $\mathbf{s} := \mathbf{s}^{(1)} - \mathbf{s}^{(0)}$.

Therefore, we can rank tags according to their final values in $\mathbf{s}$, where the top-ranked tags are suggested to user u for resource r.

4.4 DiffusionRank

DiffusionRank was originally proposed for combating web spam [3], which has also been successfully used in social network analysis [4] and search query suggestion [5]. DiffusionRank is motivated by the heat diffusion process, which can be used for ranking because the activities flow on the graph can be imagined as heat flow, the edge from a vertex to another can be treated as the pipe of an air-conditioner for heat flow.

For a graph $G = \{V, E\}$, denote $f_i(t)$ is the heat on vertex v_i at time t, we construct DiffusionRank as follows. Suppose at time t, each vertex v_i receives an amount of heat, $M(v_i, v_j, t, \Delta t)$, from its neighbor v_j during a period Δt. The received heat is proportional to the time period Δt and the heat difference between v_i and v_j, namely $f_j(t) - f_i(t)$. Based on this, we denote $M(v_i, v_j, t, \Delta t)$ as

$$M(v_i, v_j, t, \Delta t) = \gamma (f_j(t) - f_i(t)) \Delta t$$

where γ is heat diffusion factor, i.e. the thermal conductivity. Therefore, the heat difference at node v_i between time $t + \Delta t$ and time t is equal to the sum of the heat that it receives from all its neighbors. This is formulated as:

$$f_i(t+\Delta t) - f_i(t) = \sum_{v_j \in N(v_i)} \gamma (f_j(t) - f_i(t)) \Delta t \tag{11}$$

The process can also be expressed in a matrix form:

$$\frac{\mathbf{f}(t+\Delta t) - \mathbf{f}(t)}{\Delta t} = \gamma \mathbf{H} \mathbf{f}(t) \tag{12}$$

where $\mathbf{f}$ is a vector of heat at vertices at time t, and $\mathbf{H}$ is

$$H(i,j) = \begin{cases} -1 & \text{if } i = j \\ 0 & \text{if } (v_i, v_j) \notin E \\ \frac{w(v_i,v_j)}{w(v_j)} & \text{if } (v_i, v_j) \in E \end{cases} \tag{13}$$

If the limit $\varDelta t \to 0$, the process will become into

$$\frac{d}{dt}\mathbf{f}(t) = \gamma \mathbf{H}\mathbf{f}(t) \tag{14}$$

Solving this differential equation, we have $\mathbf{f}(t) = e^{\gamma t \mathbf{H}}\mathbf{f}(0)$. Here we could extend the $e^{\gamma t \mathbf{H}}$ as

$$e^{\gamma t \mathbf{H}} = \mathbf{I} + \gamma t \mathbf{H} + \frac{\gamma^2 t^2}{2!}\mathbf{H}^2 + \frac{\gamma^3 t^3}{3!}\mathbf{H}^3 + \cdots \tag{15}$$

The matrix $e^{\gamma t \mathbf{H}}$ is named as the diffusion kernel in the sense that the heat diffusion process continues infinitely from the initial heat diffusion.

γ is an important factor in the diffusion process. If γ is large, the heat will diffuse quickly. If γ is small, the heat will diffuse slowly. When $\gamma \to +\infty$, heat will diffuse immediately, and DiffusionRank becomes into PageRank.

As in PageRank, there are random relations among vertices. To capture these relations, we use a uniform random relation among different vertices as in PageRank. Let $1 - \lambda$ denote the probability that random surfer happens and λ is the probability of following the edges. Based on the above discussion, we can modify DiffusionRank into

$$\mathbf{f}(t) = e^{\gamma t \mathbf{R}}\mathbf{f}(0),\ \mathbf{R} = \lambda \mathbf{H} + (1-\lambda)\frac{1}{N}\mathbf{1} \tag{16}$$

In application, a computation of $e^{\gamma t \mathbf{R}}$ is time consuming. We usually to approximate it to a discrete form

$$\mathbf{f}(t) = (\mathbf{I} + \frac{\gamma}{M}\mathbf{R})^{Mt}\mathbf{f}(0) \tag{17}$$

Without loss of generality, we use one unit time for heat diffusion between vertices and their neighbors, we have

$$\mathbf{f}(1) = (\mathbf{I} + \frac{\gamma}{M}\mathbf{R})^{M}\mathbf{f}(0) \tag{18}$$

We could iteratively calculate $(\mathbf{I}+\frac{\gamma}{M}\mathbf{R})^{M}\mathbf{f}(0)$ by applying the operator $(\mathbf{I}+\frac{\gamma}{M}\mathbf{R})$ to $\mathbf{f}(0)$. Therefore, for each iteration, we could diffuse the heat values at each vertices using the following formulation:

$$\mathbf{s} = (1 - \frac{\gamma}{M})\mathbf{s} + \frac{\gamma}{M}(\lambda \mathbf{A}\mathbf{s} + (1-\lambda)\frac{1}{N}\mathbf{1}) \tag{19}$$

where M is the number of iterations. As analyzed in [3], for a given threshold ϵ, we can compute to get M such that $\|((\mathbf{I} + \frac{\gamma}{M}\mathbf{R})^M - e^{\gamma \mathbf{R}})\mathbf{f}(0)\| < \epsilon$ for any $\mathbf{f}(0)$ whose sum is one. Similar to [3], in this paper we set $M = 100$ for DiffusionRank.

Different from FolkRank, in DiffusionRank we set the initial values $\mathbf{f}(0)$ for vertices to indicate the preferences. To suggest tags to user u for resource r, we set $\mathbf{f}(0) = \mathbf{0}$, but for $\mathbf{f}_u(0) = 1$ and $\mathbf{f}_r(0) = 1$. After running DiffusionRank on the tripartite graph, we rank tags according to their ranking scores and the top-ranked tags are suggested to user u for resource r.

5 Experiments

5.1 Data Set

We use the given BibSonomy data set to validate our methods, it is a snapshot of the BibSonomy system until Jan 1, 2009. The data set contains two parts, *bibtex* and *bookmark*. In *bibtex*, the resources are citation of research papers or books, with title, author and other information. In *bookmark*, the resources are website URLs with a user-provided short description. Additionally, the contest organizer provide two *postcore-2* data sets. In the *postcore-2* data sets, the organizer removed all users, tags, and resources which appear in only one post. The process was iterated until convergence and got a core in which each user, tag, and resource occurs in at least two posts. Batagelj et al [9] provided a detailed explanation of postcore building . The basic statistics of these data sets are lists in Table 1

Table 1. Basic statistics of the full bibtex and bookmark data sets. Mean Len. is the mean number of words in the corresponding text content.

Name	#posts	#tags	#users	#words	Mean Length	Mean #tags/user
bibtex	158,912	50,855	1,790	278,106	47.67	60.75
bookmark	263,004	56,424	2,679	293,026	11.83	57.78
bibtex(pcore2)	22,852	5,816	788	48,401	59.21	31.75
bookmark(pcore2)	41,268	10,702	861	47,689	12.23	60.26

To validate and tune our methods, we split each of the four dataset into 5 equal-sized subset randomly, and perform 5-fold cross validation on them.

5.2 Evaluation Metrics

We use precision, recall and F1 measure as the evaluation metrics. Precision is the number of correct suggested tags multiplied by the total number of tags suggested. Recall is the number of correct suggested tags multiplied by the total number of tags of original post. F1 measure is a geometry mean of precision and recall, $F1 = 2Precsion \times Recall/(Precision + Recall)$. For each post, we only consider the first 5 tags suggested.

5.3 Content-based Tag Suggestion

To test the performance of our content-based method, we run 5-fold cross validation using the given training data. Additionally, for each fold, we remove all posts in the postcore set from the test data, since posts in postcore will not appear in the final test data. We remove stopwords, punctuation marks and all words shorter than 2 letters from the data set, and convert all text to lowercase. We remove words, resource IDs and user IDs appear in less than 5 post. We treat bibtex and bookmark separately.

We use search-based kNN as our baseline method, this is proposed by Mishne [10] for suggesting tags to blog posts. In our experiment, we index the training data by Lucene[1] indexing package. For a test post, we use $TF \times IDF$ to select 10 top words. Then, we use these words to construct a weighted query, and search the training data with it. We take all tags from Lucene returned top-k documents, weight each tag using the corresponding document's relevance score, and sum the weights of duplicated tags. We take the first 5 tags as the suggested tags. In search-based kNN, k is a parameter to tune. After using $k = 1, 2, 3, 4, 5$, we use $k = 1$ as the final k, since it has the best F1 measure.

We list the mean precision, recall and F1 value for bibtex and bookmark data in Table 2 and 3 respectively. We experimented with the different combination of methods for weighting features and estimating Θ matrix.

In the bibtex dataset, FDT(TFITF+MI) has the similar performance as the search-based kNN methods. In the bookmark dataset, FDT(TFITF+MI) has the best performance, which is 3 percentage better than search-based kNN.

Table 2. P, R and F1 of search-based kNN and different learning methods for FDT on the bibtex dataset. All averaged over 5 folds.

Method	Precision	Recall	F1
search-based kNN	**0.2792**	0.2324	**0.2537**
FDT(TFIDF+CC)	0.2517	0.2152	0.2320
FDT(TFIDF+MI)	0.1822	0.1652	0.1733
FDT(TFIDF+χ^2)	0.2261	0.2235	0.2248
FDT(TFITF+CC)	0.2513	0.2173	0.2330
FDT(TFITF+MI)	0.2432	**0.2526**	0.2478
FDT(TFITF+χ^2)	0.2216	0.2246	0.2231

In the training data, the number of post from each user roughly follows the power law distribution, where most users have less than 100 posts, and the top 4 users have 50% of all posts. If we treat all posts as equal, then the model may bias to the preference of several super users. To know the performance of the methods on super users and common users, we run other two experiments. In the first experiment, we train the model using posts from all users, then check its performance on each of the top n users and all the rest users separately. In

[1] http://lucene.apache.org

Table 3. P, R and F1 of search-based kNN and different learning methods for FDT on the bookmark dataset. All averaged over 5 folds.

Method	Precision	Recall	F1
kNN	0.2935	0.2409	0.2646
FDT(TFIDF+CC)	**0.4036**	0.2411	**0.3018**
FDT(TFIDF+MI)	0.3580	0.1892	0.2475
FDT(TFIDF+χ^2)	0.2675	0.2486	0.2577
FDT(TFITF+CC)	0.3633	0.2404	0.2893
FDT(TFITF+MI)	0.3143	**0.2610**	0.2852
FDT(TFITF+χ^2)	0.2284	0.1831	0.2033

the second experiment, we train and test models using only posts from each of the top n users and all the rest users separately. For bibtex dataset, we choose $n = 4$, for bookmark dataset, we choose $n = 5$. The results for bibtex and bookmark are listed in Table 4 and Table 5 respectively. In these experiments, we use FDT(TF*ITF+MI) for bibtex data and FDT(TF*IDF+CC) for bookmark data. In the result table, the column *Trained(ALL)* means all methods are trained on full training data. The column *Trained(USER)* means each method is trained using only posts from corresponding group of users. In the method name, *kNN(2463)* means the method used is search-based kNN, and test data set are all post of user 2463, rest means all other users. The same naming rule applies to *FDT(xxxx)*.

For each group of test data, we have 4 different models, they are kNN trained by all users, kNN trained by this group, FDT trained by all users and FDT trained by this user. As the result shows, for groups of super users, kNN-based models have best performance. For common users (the *rest* group), FDT-based models performs better. This result follows our intuition. In this data set, super users have different tag preference than common users. kNN suggest tags using most similar resources, it is less affected by the overall distribution of resources, so it fits to the . FDT relies on the global statistics of feature-tag relationship, it is less effective to fit a special user's preference. In practical situation, we can get the best performance by choosing different model for different group of users.

One interesting observation is about the user #2732. When trained with all posts, FDT performs much better(0.6308 vs 0.2300) on #2732 than trained with #2732's own posts. We examined the posts of #2732, found that many posts contains only three tags: *genetic*, *programming* and *algorithm*, and the number of posts by #2732 is large. When we use all posts to train FDT(TFITF+MI), these three tags have a large Mutual Information value with many features, especially the user id feature "UID-2732", so FDT can predict tags for posts of #2732 with high accuracy. When trained only with #2732's posts, the Mutual Information between features and these three tags is much smaller, since these three tags appears everywhere and can be seen as stopwords in tags. Small Mutual Information of these three tags means FDT will make wrong prediction about most posts of #2732, which leads to a decreasing in F1-measure.

Table 4. P, R and F1 of search-based kNN and FDT on different set of users on the bibtex dataset. Trained(ALL) means that we train the model using posts from all users, and test the performance on given set of users. Train(USER) means that both training and testing use posts from the given set of users. The % column indicates the size of corresponding user group, as the percentage in all posts. All averaged over 5 folds.

Trained(ALL)	P	R	F1	Trained(USER)	P	R	F1	%
kNN(2463)	0.1323	0.1352	0.1337	kNN(2463)	0.1376	0.1377	0.1376	24.27
kNN(2651)	0.1561	0.1573	0.1567	kNN(2651)	0.1953	0.1910	**0.1932**	12.40
kNN(3180)	0.4278	0.2771	0.3364	kNN(3180)	0.4440	0.2807	0.3440	9.20
kNN(2732)	0.6267	0.3915	0.4819	kNN(2732)	0.6517	0.4422	0.5269	3.78
kNN(rest)	0.3207	0.2530	0.2829	kNN(rest)	0.3202	0.2579	0.2857	50.35
FDT(2463)	0.1066	0.1869	0.1358	FDT(2463)	0.1100	0.2055	**0.1429**	24.27
FDT(2651)	0.1022	0.1285	0.1138	FDT(2651)	0.1126	0.1818	0.1391	12.40
FDT(3180)	0.3656	0.3334	**0.3488**	FDT(3180)	0.3688	0.3274	0.3469	9.20
FDT(2732)	0.8763	0.4927	**0.6308**	FDT(2732)	0.3142	0.1814	0.2300	3.78
FDT(rest)	0.3101	0.2516	0.2778	FDT(rest)	0.3260	0.2559	**0.2867**	50.35

Table 5. P, R and F1 of search-based kNN and FDT on different set of users on the bookmark dataset. Trained(ALL) means that we train the model using posts from all users, and test the performance on given set of users. Train(USER) means that both training and testing use posts from the given set of users. The % column indicates the size of corresponding user group, as the percentage in all posts. All averaged over 5 folds.

Trained(ALL)	P	R	F1	Trained(USER)	P	R	F1	%
kNN(1747)	0.5523	0.4877	0.5180	kNN(1747)	0.6513	0.5566	**0.6003**	19.90
kNN(2977)	0.4554	0.4072	0.4299	kNN(2977)	0.5567	0.5154	**0.5353**	9.48
kNN(483)	0.1002	0.1365	0.1156	kNN(483)	0.2375	0.2227	**0.2299**	3.56
kNN(275)	0.2102	0.1947	0.2022	kNN(275)	0.3413	0.3059	**0.3226**	3.40
kNN(421)	0.2749	0.0867	0.1318	kNN(421)	0.2787	0.1080	0.1557	2.26
kNN(rest)	0.1921	0.1627	0.1762	kNN(rest)	0.2007	0.1643	0.1807	61.41
FDT(1747)	0.5306	0.4169	0.4670	FDT(1747)	0.3592	0.2325	0.2823	19.90
FDT(2977)	0.4437	0.3622	0.3988	FDT(2977)	0.4162	0.3367	0.3722	9.48
FDT(483)	0.1684	0.2653	0.2060	FDT(483)	0.1642	0.2637	0.2024	3.56
FDT(275)	0.1887	0.1610	0.1738	FDT(275)	0.2531	0.1760	0.2076	3.40
FDT(421)	0.4044	0.1258	0.1920	FDT(421)	0.4328	0.1339	**0.2045**	2.26
FDT(rest)	0.2462	0.2133	0.2286	FDT(rest)	0.2502	0.2204	**0.2344**	61.41

For final test, we use FDT(ITF+MI) for bibtex and FDT(IDF+CC) for bookmark. The test data of DC09 has a different distribution with the training data. Most top ranked users don't appear in the test data. So we removed the top ranked users from the training data, use the *rest* group of users to train the model for final suggestion. The p/r/f1 on final test data are 0.1388/0.1049/0.1189 respectively. Compared to the cross validation results, the performance dropped a lot on final test data. One reason is that FDT does not suggest tags that are not in the training data. There are 93756 tags in the training data and 34051 tags in the test data, the overlapped tags are only 15194. To achieve better performance, suggesting new tags should be considered in the future.

5.4 Graph-based Tag Suggestion

In experiments, we compare the results of three graph-based methods, most popular tags, FolkRank and DiffusionRank.

Here we first demonstrate the results using 5-fold cross validation on training dataset. In Table 6, we show the best performance of various methods on bibtex dataset. In this table, we also demonstrate the performance of the content-based method kNN , which achieves the best result when $k = 2$. For the method of most popular tags, we use "mpt+resource" to indicate most popular tags by *resource*, and "mpt+mix" to indicate most popular tags by mixing *resource* and *user*. For FolkRank, the best result is achieved when damping factor $\lambda = 0.01$ with 100 iterations. DiffusionRank obtains the best result when damping factor $\lambda = 0.85$, maximum number of iterations $max_i t = 10$ and diffusion factor $\gamma = 0.1$. From the table, we can see that most popular tags by mix achieves the best F_1-measure, which has the largest precision. While for DiffusionRank, it achieves the best recall.

Table 6. Best performance of various methods on bibtex training dataset. All values are averaged over 5 folds.

Method	Precision	Recall	F_1-measure
kNN	0.3664	0.4307	0.3959
mpt+resource	0.3949	0.3765	0.3855
mpt+mix	**0.4211**	0.4014	**0.4110**
FolkRank	0.3222	0.4459	0.3741
DiffusionRank	0.3347	**0.4630**	0.3885

In Table 7, we show the best performance of various methods on bookmark dataset. kNN achieves the best performance when $k = 2$. For FolkRank, the best result is achieved when damping factor $\lambda = 0.0001$ with 10 iterations. DiffusionRank obtains the best result when damping factor $\lambda = 0.85$, maximum number of iterations $max_i t = 10$ and diffusion factor $\gamma = 0.01$. Furthermore, we also restrict the scores of suggested tags should be no less than 1/5 of score of first-ranked tags. From the table, we can see that DiffusionRank achieves the best F$_1$-measure, which has the largest precision.

Table 7. Best performance of various methods on bookmark training dataset. All values are averaged over 5 folds.

Method	Precision	Recall	F$_1$-measure
kNN	0.2855	0.2892	0.2873
mpt+resource	0.3345	0.2798	0.3047
mpt+mix	0.3606	0.3017	0.3285
FolkRank	0.3288	**0.3309**	0.3298
DiffusionRank	**0.3772**	0.3266	**0.3501**

From the above two tables, we find that on the bibtex dataset the method of most popular tags by mix is the best, and on bookmark dataset DiffusionRank achieves the best result. Therefore, for task 2 of rsdc'09, we use the two methods to train ranking models separately on bibtex and bookmark. Using the original result and evaluation program provided by the challenge organizer, we obtain the evaluation results on test dataset, as shown in Table 8. From the table, we find that the absolute values are much smaller than what are shown in Table 6 and 7.

Table 8. Evaluation result on test dataset of rsdc'09.

Tag Number	Precision	Recall	F$_1$-measure
1	0.1483	0.4229	0.2196
2	0.2301	0.3477	0.2769
3	0.2960	0.3113	0.3034
4	0.3418	0.2840	0.3102
5	0.3760	0.2601	0.3075

Besides the above analysis, we want to investigate the performance of FolkRank and DiffusionRank as their parameters change.

In Table 9 and 10, we demonstrate the performance of FolkRank on bibtex training dataset and bookmark training dataset as its parameters, the damping factor λ and maximum number of iterations (denoted as "max-it" in tables) change. From the both tables, we find the performance of FolkRank improves as damping factor shrinks, which indicates the effect of preference values are growing larger. That is to say the generalization of FolkRank by passing values iteratively on graphs may harm the performance. Moreover, it seems that the maximum number of iterations of FolkRank does not effect the results significantly.

In Table 11 and 12, we demonstrate the performance of DiffusionRank on bibtex training dataset and bookmark training dataset as its parameters, the

Table 9. Performance of FolkRank on bibtex training dataset. All values are averaged over 5 folds.

λ	max-it	Precision	Recall	F_1-measure
0.85	10	0.2943	0.4072	0.3417
0.5	10	0.3053	0.4225	0.3545
0.1	10	0.3198	0.4425	0.3713
0.01	10	**0.3222**	**0.4459**	**0.3741**
0.01	100	**0.3222**	**0.4459**	**0.3741**
0.001	10	0.3219	0.4455	0.3738
0.0001	10	0.3219	0.4455	0.3738

Table 10. Performance of FolkRank on bookmark training dataset. All values are averaged over 5 folds.

λ	max-it	Precision	Recall	F_1-measure
0.85	10	0.2989	0.3008	0.2998
0.5	10	0.3038	0.3058	0.3048
0.1	10	0.3198	0.3218	0.3208
0.01	10	0.3275	0.3297	0.3286
0.01	100	0.3275	0.3297	0.3286
0.001	10	**0.3288**	**0.3309**	**0.3298**
0.0001	10	**0.3288**	**0.3309**	**0.3298**

diffusion factor γ and maximum number of iterations (denoted as "max-it" in tables) change. Here the damping factor λ is set to 0.85. We also find that the performance of DiffusionRank improves as diffusion factor shrinks, which indicates the effect of initial values is growing larger. Similar to FolkRank, the generalization of DiffusionRank by passing values iteratively on graphs may also harm the performance. It is also the same as FolkRank that the maximum number of iterations of DiffusionRank does not effect the results significantly.

From the experiments on both bibtex and bookmark training datasets, we can see that DiffusionRank always outperforms FolkRank with some specific parameters, which is more significant on bookmark dataset. Although in this dataset, FolkRank does not outperform the method of most popular tags, in [8] we know that in some datasets, FolkRank outperforms most simple methods including the method of most popular tags. Therefore, more experiments still need to be done to investigate the efficiency of DiffusionRank compared to FolkRank and other graph-based methods for tag suggestion.

Furthermore, the number of suggested tags should be specified in advance in FolkRank and DiffusionRank. However in some conditions, we do not have to recommend as many tags as specified. For DiffusionRank, we set the maximum number of suggested tags is 5. If we further require the suggested tags should have the ranking values no less than 1/5 of the ranking value of the first-ranked tag, the performance of precision, recall and F_1-measure will be improved to 0.3772, 0.3266 and 0.3501 on bookmark training dataset. Therefore, we use the altered DiffusionRank for the bookmark test set of task 2 in rsdc'09.

Table 11. Performance of DiffusionRank on bibtex training dataset. In all experiments, damping factor λ is set to $\lambda = 0.85$. All values are averaged over 5 folds.

γ	max-it	Precision	Recall	F_1-measure
2.0	10	0.3279	0.4537	0.3807
1.0	10	0.3331	0.4609	0.3867
0.1	10	**0.3347**	**0.4630**	**0.3885**
0.1	100	**0.3347**	**0.4630**	**0.3885**
0.01	10	**0.3347**	**0.4630**	**0.3885**

Table 12. Performance of DiffusionRank on bookmark training dataset. In all experiments, damping factor λ is set to $\lambda = 0.85$. All values are averaged over 5 folds.

γ	max-it	Precision	Recall	F_1-measure
2.0	10	0.3336	0.3357	0.3346
1.0	10	0.3370	0.3392	0.3381
0.1	10	0.3403	0.3425	0.3414
0.1	100	0.3403	0.3425	0.3414
0.01	10	**0.3406**	**0.3428**	**0.3417**

6 Related Work

Ohkura et al [11] proposed a Support Vector Machine-based tag suggestion system. They train a binary classifier for each tag to decide if this tag should be suggested. Katakis et al [12] use a hierarchical multi-label text classifier to find the proper tags for a document. They cluster all tags using modified k-means, use one classifier to decide which clusters a document belongs to, then use another cluster-specific classifier to decide which tags in the cluster belongs to the document. Mishne [10] use a search-based nearest neighbor method to suggest tags, where the tags of a new document is collected from the most relevant documents in the training set. Lipczak et al [13] extract keywords from the title of a document, then filter them with a user's used tags to get the final suggestion. These methods all use the content of a document, we call them content-based methods. Tatu et al [14] combine tags from similar documents and extracted keywords to provide tag suggestions. They have the best performance in the first ECML/PKDD Discovery Challenge task.

Another class of tag suggestion system is based on the links between users, tags and resources, which does not take the content of resources into consideration. Since the method of "most popular tags" also does not consider the content of resources, in this paper we regard it as a member of graph-based tag suggestion approach. Xu et al [15] use collaborative filtering to suggest tags for URL bookmarks. Jaschke et al [2] proposed FolkRank, a PageRank-like iterative algorithm to find the most related tags for a resource in its neighbor users and tags. PageRank is originally used for ranking web pages only according to the topology of web graph. However, in PageRank we can set preference values to a subset of pages to make the PageRank values biased to these pages and their neighbors. In fact, FolkRank is used to compute the relatedness between tags and the specific user and resource by setting the given user and resource to high preference values in PageRank.

Recently, a new graph-based ranking method, DiffusionRank [3], is proposed for anti-spam of web pages. DiffusionRank is motivated by the heat diffusion process, which can be used for ranking because the activities flow on the graph can be imagined as heat flow, the edge from a vertex to another can be treated as the pipe of an air-conditioner for heat flow. Based on the property of heat always flow from high to low, the ranking values of DiffusionRank are related to initial values of vertices. Therefore, DiffusionRank provides a more flexible method to rank tags by setting high initial values to the given user and resource. In this paper, we for the first time propose to use DiffusionRank for graph-based tag suggestions.

7 Conclusion

In this paper, we study the problem of tag suggestion and describe our methods for content-based and graph-based suggestion. For content-based tag suggestion, we propose a new method named Feature-Driven Tagging for fast content-based tag suggestion. Cross validation on the training data shows that FDT outperforms wildly-used search-based kNN, especially when suggesting tags for long-tail users. For graph-based tag suggestion, we study most popular tags, FolkRank, and propose a DiffusionRank-based method. Experiments show that on bibtex dataset the method of most popular tags by mixing of user and resource performs best, and on bookmark dataset, DiffusionRank outperforms other methods.

Work remains to be done. First, currently we use empirical methods to estimate the parameters for FDT, like CC, MI and ITF. We will consider learn a Θ matrix directly by optimizing a tag-related loss function. Second, evaluation using final test data of DC09 shows that the F1 value drops a lot than cross validation on the training data, especially for content-based methods. This suggests we should pay attention to out-of-vocabulary tags. Third, more information should be considered, such like time-stamp, to suggest better tags in real-world situation.

Acknowledgments

This work is supported by the National Science Foundation of China under Grant No. 60621062, 60873174 and the National 863 High-Tech Project under Grant No. 2007AA01Z148.

References

1. Page, L., Brin, S., Motwani, R., Winograd, T.: The pagerank citation ranking: Bringing order to the web. Technical report, Stanford Digital Library Technologies Project (1998)

2. Jaschke, R., Marinho, L., Hotho, A., Schmidt-Thieme, L., Stumme, G.: Tag recommendations in folksonomies. In: Proceedings of the 11th European conference on Principles and Practice of Knowledge Discovery in Databases, Springer-Verlag Berlin, Heidelberg (2007) 506–514
3. Yang, H., King, I., Lyu, M.R.: Diffusionrank: a possible penicillin for web spamming. In: Proceedings of SIGIR. (2007) 431–438
4. Ma, H., Yang, H., Lyu, M.R., King, I.: Mining social networks using heat diffusion processes for marketing candidates selection. In: Proceeding of CIKM. (2008) 233–242
5. Ma, H., Yang, H., King, I., Lyu, M.R.: Learning latent semantic relations from clickthrough data for query suggestion. In: Proceeding of CIKM. (2008) 709–718
6. Manning, C., Raghavan, P., Schtze, H.: Introduction to information retrieval. Cambridge University Press New York, NY, USA (2008)
7. Yang, Y., Pedersen, J.: A comparative study on feature selection in text categorization. In: MACHINE LEARNING-INTERNATIONAL WORKSHOP THEN CONFERENCE-, MORGAN KAUFMANN PUBLISHERS, INC. (1997) 412–420
8. Jaschke, R., Marinho, L., Hotho, A., Schmidt-Thieme, L., Stumme, G.: Tag recommendations in social bookmarking systems. AI Communications **21**(4) (2008) 231–247
9. Batagelj, V., Zaversnik, M.: Generalized cores (2002)
10. Mishne, G.: Autotag: a collaborative approach to automated tag assignment for weblog posts. In: Proceedings of the 15th international conference on World Wide Web, ACM New York, NY, USA (2006) 953–954
11. Ohkura, T., Kiyota, Y., Nakagawa, H.: Browsing system for weblog articles based on automated folksonomy. In: Proceedings of the WWW 2006 Workshop on the Weblogging Ecosystem: Aggregation, Analysis and Dynamics, at WWW. Volume 2006. (2006)
12. Katakis, I., Tsoumakas, G., Vlahavas, I.: Multilabel text classification for automated tag suggestion. ECML PKDD Discovery Challenge 2008 75
13. Lipczak, M.: Tag Recommendation for Folksonomies Oriented towards Individual Users. ECML PKDD Discovery Challenge 2008 84
14. Tatu, M., Srikanth, M., D'Silva, T.: Rsdc'08: Tag recommendations using bookmark content. ECML PKDD Discovery Challenge 2008
15. Xu, Z., Fu, Y., Mao, J., Su, D.: Towards the semantic web: Collaborative tag suggestions. In: Collaborative Web Tagging Workshop at WWW2006. (2006)