Improved Recommendation based on Collaborative Tagging Behaviors

Shiwan Zhao[1], **Nan Du**[2], **Andreas Nauerz**[3], **Xiatian Zhang**[1], **Quan Yuan**[1], **Rongyao Fu**[1]
[1]IBM China Research Laboratory, Beijing, 100094, China
{zhaosw,xiatianz,quanyuan,furongy}@cn.ibm.com
[2]Beijing University of Posts and Telecommunications,Beijing, 100876, China
dunan@bupt.edu.cn
[3]IBM Research and Development,Schoenacher STR. 220, Boeblingen,71032, Germany
andreas.nauerz@de.ibm.com

ABSTRACT
Taking into consideration the natural tendency that people usually follow direct or indirect cues of activities of others, most collaborative filtering-based recommender systems in web personalization often predict the utilization of pages for a particular user according to the pages previously rated by other similar users. Consequently, effective searching for the most related neighbors is critical for the success of recommendation.

Recent years have seen the flourishing development of the World Wide Web as the main social media that enables individuals to easily share opinions, experiences and expertise across the world. Collaborative tagging systems with social bookmarking as their key component of modern Web 2.0 applications allow users to freely bookmark and assign semantic descriptions to various shared resources on the web. While the favorite list of web pages indicates the interests or taste of each user, the assigned tags can further provide useful hints about how a user may think of the pages.

Therefore, in this paper, we propose a new collaborative filtering approach *TBCF* (Tag-based Collaborative Filtering) based on the semantic distance among tags assigned by different users to improve the effectiveness of neighbor searching. That is, two users could be considered similar not only if they rated the web pages similarly, but also if they have similar cognitions over these pages. We tested *TBCF* with two different real-life datasets. The experimental results show that our approach has significant improvement against the traditional cosine-based recommendation method.

Author Keywords
Web2.0, Tag, Recommendation, Collaborative Filtering

ACM Classification Keywords
H.5.3 [Information Interfaces and Presentation]: Group and Organization Interfaces-Collaborative computing

INTRODUCTION

Collaborative filtering, as one of the most successful technologies for recommender systems, has been developed and improved over the past decades. In the common formulation, the recommendation problem is reduced to the problem of estimating the utilization for the items that have not been seen by a user[7, 12, 15]. Generally speaking, collaborative filtering approaches predict the rating of items for a particular user (active user) based on the ratings from other users with similar interests. For example, in a movie recommender system, one user's rating to a movie is a numeric score ranging from zero to five, which indicates how much the user likes or dislikes the movie. People holding similar rating patterns can form a neighborhood, and the ratings from these like-minded neighbors be used to achieve predictions for the active user.

Although the previous rating scheme can represent the extent to which a particular user may prefer a given item, it can not reveal the user's own opinions and understanding over such an item. In other words, this rating scheme only demonstrates the strength of preference along a single dimension by scalar quantities. Yet, various users may give the same rating to an item from different points of view. For instance, both Bob and Tom may rate the movie Transformers by five stars, which indicates they all like this movie very much. Nevertheless, as a 3D fan, Bob appreciates this movie for its high quality 3D animations, while John may think that it is a wonderful action movie. Therefore, if users show approximate strength of interests from similar points of view, they can be regarded as neighbors with much more closer relationship.

In most practical applications, it is often annoying for users to provide ratings explicitly, so many researchers focus on the retrieval of user profiles from the interactions with the application system alternatively. For example, web server logs could indicate the usage patterns of a user on certain web sites. Such implicit acquisition of user preferences is often referred as log-based collaborative filtering[18] and usually gives binary-valued ratings. If some web page is visited by a user, the rating is one implicitly which means relevance; otherwise it is zero which indicates irrelevance. The development of modern Web 2.0 applications, such as flickr, del.icio.us and YouTube, indicates a fundamental shift in the ease of publishing contents. People now can easily share their photos, bookmarked web pages, or movies with each other at the push of a single button. As a result,

tagging has become the key part of the social bookmarking systems and a powerful tool for semantically describing the shared resources. Additionally, tagging can be regarded as another important way of implicit rating with semantics. In a social bookmarking system, users usually save their favorite web resources, such as web pages, photos, and images, which could indicate the positive attitude (like) to these bookmarked resources. Beyond that, people can also freely add tags based on their personal tendency, their preferences and beliefs.

On the other hand, one common problem of collaborative recommendation systems is that the number of ratings already obtained by each user is very small, which is often referred as the sparsity problem[1]. Since the success of any collaborative system depends on the availability of a critical mass of users, effective prediction from a small number of examples is important. Compared with the traditional binary-valued ratings, tags can reflect both the user preferences and their opinions. That is, two users could be considered similar not only if they rated the web pages similarly, but also if they have similar cognitions over these pages. This kind of additional semantic information can be used to address the sparsity problem further. Therefore, by taking the semantic distance among tags assigned by different users into consideration, we propose a new collaborative filtering approach *TBCF* (Tag-based Collaborative Filtering) to improve the effectiveness of neighbor searching.

The remainder of the paper is organized as follows: In sections 2 we review some related work. Section 3 describes our tag-based recommendation approaches. The experimental results and analysis are given in section 4; and we conclude our work in section 5.

RELATED WORK

Research within the field of rating-based collaborative recommendations can be classified into two general groups: memory-based and model-based. In terms of the memory-based approach, every rating example could be accessed just in time as long as it is needed to find similar neighbors and make predictions. It seems that all these examples are "memorized" by the recommendation system. During the prediction stage, similar neighbors are ranked according to the memorized ratings. Then, based on the ratings of the most similar users, a recommendation for the active user is generated. Existing algorithms of memory-based algorithms include the weighted predictive methods, user clustering and the item correlation approach[4, 5, 11].

With respect to the model-based algorithms, they usually learn experiences or models from the collection of ratings. Their predications are based on these models being constructed. Breese[4] proposes two probabilistic models which are based on the simple Bayesian classification and Bayesian network respectively. Billsus and Pazzani[3] provide a collaborative filtering approach in the machine learning framework where various machine learning techniques can be used, such as the reinforcement learning and artificial neural networks. Ungar and Foster[17] have presented a statistical model by using K-means clustering and Gibbs sampling to estimate the model parameters. Other interesting methods include the linear regression model[13], the maximum entropy[6], a Markov decision process[14], and the probabilistic latent semantic approach[8]. Compared with memory-based algorithms, the model-based methods can solve the data sparsity problem to some extent, but they usually require a significant number of parameters and hidden variables to be tuned, which often prevents them to be used in practice.

As mentioned before, although the rating-based approaches depend on the ratings of items. In many real-life applications, people hesitate to give ratings explicitly. As a result, log-based approaches are more favorable in practice, where implicit interest functions usually generate binary-valued preferences. More formally, let U and I be the set of all users and items respectively. The specific value of the unknown rating $r(u_i, c)$ of item c for user $u_i (u_i \in U, c \in I)$ is often estimated from the ratings $r(u_j, c)$ given to item c by other similar users $u_j \in U, j \neq i$. In log-based collaborative recommendation,$r(u_j, c) = 1$ if u_j has once browsed or accessed item c; otherwise $r(u_j, c) = 0$. Suppose U_N denotes the set of N users who have accessed item c and are the most similar to u_i. One of the most popular approaches to define $r(u_i, c$ is to use the weighted sum as follows

$$r(u_i, c) = k \sum_{u_j \in U_N} sim(u_i, u_j) \times r(u_j, c)$$

The weight $sim(u_i, u_j)$ is the similarity measure between users u_i and u_j. The more similar users u_i and u_j are, the more weight rating $r(u_j, c)$ will carry in the estimation of $r(u_i, c)$. k is the normalizing factor such that the absolute values of the weights sum to unity. Since that $sim(u_i, u_j)$ is a heuristic function, different recommendation systems in separate application domains may use different similarity measures of their own. Let I_{uv} be the set of all items accessed by both user u and v together.One commonly adopted similarity measure is to treat user u and v as two vectors in m-dimension space where $m = |I|$ accordingly. The similarity value between u and v is thus calculated as the cosine of the angle between the corresponding two vectors.

$$sim(u, v) = \frac{\overrightarrow{u} \cdot \overrightarrow{v}}{\| \overrightarrow{u} \| \cdot \| \overrightarrow{v} \|} = \frac{\sum_{c \in I_{uv}} r(u, c) \times r(v, c)}{\sqrt{\sum_{c \in I_u} r(u, c)^2} \times \sqrt{\sum_{c \in I_v} r(v, c)^2}}$$

Sood and Owsley[16] directly apply information retrieval techniques to contents part of the browsed web pages in order to compute $sim(u, v)$. However, it is obvious that not all shared resources can be compared directly by their contents.Hence, our proposed method uses tags as the feature vector of a user. Thus *TBCF* in general can recommend uniquely identifiable resources, such as pages, images, videos and even people.

TAG-BASED RECOMMENDATION

Our recommendation approach mainly consists of two parts. First we will adopt two approaches to calculate the semantic

similarity among tags. Then based on this new similarity metric we present our collaborative recommendation engine.

WordNet-based Tag Similarity

WordNet is a public lexical database that provides a large repository of English lexical items. Each word in WordNet is stored in a structure called *synset* which is the basic unit of the whole dictionary. Every *synset* includes the word, its meaning and the corresponding synonyms. The meaning of a word is often referred as *gloss* which actually defines the specific concept. Different meanings of a word correspond to different *synsets*. Terms with the synonymous meanings lie in the same *synset*.For example, the word *love* and *passion* constitute a *synset* with the *gloss*:*any object of warm affection or devotion*. All the *synsets* are organized by some basic semantic relations, such as "the part of" and "is a kind of". As a result, the whole dictionary can be treated as a large graph with the node being the single *synset* and the edge representing the semantic relation.

As discussed above, the tagging technique allows people to freely annotate their shared resources from photo tagging (*flickr*), to web page tagging (*del.icio.us*) to social tagging(*Facebook*). The tags used tremendously improve organizing, browsing, navigating and searching for resources. Tagging provides users with means to categorize content autonomously, independent from any central administrative instance. However, every coin always has two sides. Since tagging systems do not enforce fixed or controlled vocabularies for tag selection, the free choice of tags also result in many problems. First, multiple tags can have the same meanings, which is referred as synonymy. Two tags may be morphological variation (apple vs. apples) or semantically similar (love vs. passion). Second, single tags can have multiple meanings, which is often referred as polysemy. For instance, a web page tagged with "apple" may be a post about fruits or can be interpreted as introducing iPod.

To address these problems, we first adopt Porter's stemming algorithm[9] to remove the common morphological and inflexional endings of tags, so the morphological variation can be solved. Then we use Satanjeev Banerjee's algorithm[2] to get rid of the semantic ambiguity of a particular tag in certain contexts. The basic idea of this approach is to count the number of words that are shared between two given glosses. The more common words they share, the more close they would become. For example, suppose we have two tags "apple" and "orange". According to *WordNet*, "apple" has two meanings:

1. fruit with red or yellow or green skin and sweet to tart crisp whitish flesh
2. native Eurasian tree widely cultivated in many varieties for its firm rounded edible fruits

The tag "orange" has five different meanings:

1. round yellow to orange fruit of any of several citrus trees
2. orange color or pigment; any of a range of colors between red and yellow
3. any citrus tree bearing oranges
4. any pigment producing the orange color
5. a river in South Africa that flows generally westward to the Atlantic Ocean

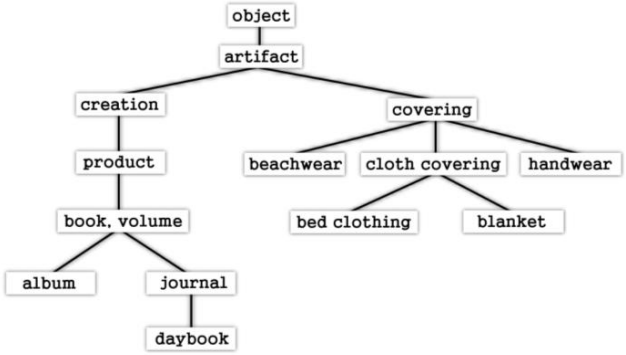

Figure 1. WordNet Concept Tree

Applying Leks's algorithm to these tags, we can find that the first meaning of "apple" and the first meaning of "orange" shares the word "fruit". As a consequence, these two glosses are selected to define "apple" and "orange" when we put them together. Since that the organization of *WordNet* can be regarded as an undirected graph shown in, single tag with multiple meanings or glosses can appear in multiple vertices. However, once we have identified the specific gloss of a tag, the corresponding position or vertex of the tag in the graph is thus fixed. Therefore, the most straightforward method to calculate the semantic similarity between two given tags is to find the shortest path connecting them in the graph. The shorter the path would be, the more similar they would become. In figure 1, we see that "book" and "volume" are within the same *synset* indicating that they have the same meaning exactly. Thus the distance is zero. While, the distance between "journal" and "book" is 1, "bed clothing" and "blanket" is 2, and "album" and "blanket" is 7. For the tags that are not included in *WordNet*, we use *Levenshtein* algorithm[10] to calculate the edit-distance between them. *Levenshtein* distance can be computed by finding the cheapest way to transform one string into another. Transformations are the one-step operations of insertion, deletion and substitution. In the simplest version substitutions cost about two units except when the source and target are identical, in which case the cost is zero. Insertions and deletions costs half that of substitutions. Therefore, the total similarity measure is defined as

$$sim(x, y) = \frac{1}{dis(x, y) + 1}$$

if x and y are contained in *WordNet*. $dis(x, y)$ is the shortest path length between x and y. Otherwise,

$$sim(x, y) = 1 - \frac{Lev(x, y)}{maxlength(x, y)}$$

Table 1. Tag-based User-Item Matrix

	Item 1	Item 2	Item 3	Item 4
Alice	Art, photo	Home, Products	Writing , Design	Learning, Education
Daniel	Photo, Album, Image	$\emptyset$	Typewriter	Tutorial, Training
Sherry	$\emptyset$	Cleaning	$\emptyset$	Language, Study
Maggie	Photography	$\emptyset$	Ovens	$\emptyset$

where either x or y is not contained, and $Lev(x, y)$ is the value of Levenshtein distance.

Algorithm 1 *GetSemanticDistance*(t_i,t_j)

1: apply Porter Stemming Algorithm on t_i and t_j
2: **if** t_i and t_j ∈*WordNet* **then**
3: select the appropriate meanings of t_i and t_j by Micheal's algorithm
4: calculate the shortest path length $dis(t_i, t_j)$
5: **return** $\frac{1}{dis(x,y)+1}$
6: **else**
7: calculate Levenshtein distance $Lev(t_i, t_j)$
8: **return** $1 - \frac{Lev(t_i,t_j)}{maxlength(t_i,t_j)}$
9: **end if**

Tag-based Recommendation Engine

The basic idea of our recommendation approach is to find the top-n nearest neighbors by using the semantic similarity among tags. Formally, let U be the set of all users, I be the set of all items that can be recommended, and T be the set of all tags rated on all the items by users in U. The new user-item rating matrix with tags is shown in Table 1

We can observe that each element in the matrix is now the set of rated tags rather than the binary-value 1 or 0 compared with traditional log-based method. Since that the feature vector of each user is no longer in the m-dimension space $m = |I|$ of real numbers, we cannot directly calculate the cosine of the angle between these vectors. Alternately, we turn to compute the user similarity based on the similarity value of tag sets. In the first step, given two users $u, v \in U$, we obtain the set of common items I_{uv} by intersecting I_u with I_v. For each element $c \in I_{uv}$ the tag sets of user u and v on item c are defined as $T(u, c)$ and $T(v, c)$ respectively, where $T(u, c), T(v, c) \subseteq T$.

In the second step, the similarity calculation of two tag sets is formulated as computing a maximum total matching weight of a bipartite graph, which is derived from the classic *optimal assignment problem* that tries find the optimal assignment of workers to jobs that maximizes the sum of ratings,given all non-negative ratings $cost[i, j]$ of each worker i to each job j. Similarly, G can be partitioned into two disjoint node sets $T(u, c)$ and $T(v, c)$ such that every edge connects a tag in $T(u, c)$ with a tag in $T(v, c)$ carrying the similarity value $sim(x, y)$ $x \in T(u, c), y \in T(v, c)$ as the weight. The Hungarian method is thus adopted to obtain the set of matching pairs defined as M. Algorithm 2 presents the process. We then define the similarity value of user u

Algorithm 2 *GetSimilarity*(T_i,T_j)

1: **for** each tag $t_i \in T_i$ **do**
2: **for** each tag $t_j \in T_i$ **do**
3: $sim(t_i, t_j) = GetSemanticDistance(t_i, t_j)$
4: **end for**
5: **end for**
6: $M \leftarrow$ Hungarian Algorithm applied on bipartite graph $G(T_i,T_j)$
7: $sum \leftarrow 0$
8: **for** each pair $(t_i, t_j \in M)$ **do**
9: $sum \leftarrow sum + sim(t_i, t_j)$
10: **end for**
11: **return** sum

and v as follows:

$$sim(u, v) = \sum_{c_i \in I_{uv}} \sum_{(x,y) \in M_{c_i}} sim(x, y)$$

where $x \in T(u, c_i), y \in T(v, c_i)$.

Based on this similarity measure, we are going to find the top-n nearest neighbors U_n of the particular user u. For each element $c_k \in \overline{I} = I - I_u$, the predicted rating of c_k is defined as

$$R(u, c_k) = \sum_{v_i \in U_n} w(v_i) \times sim(u, v_i)$$

where $w(v_i) = 1$ if v_i rates c_k; otherwise, $w(v_i) = 0$. In the end, we only return the top M predicted items of to user u. The whole procedure is described in the following algorithm.

Algorithm 3 $TBCF$(User-Item matrix $U \times C$, tag set T, active user $u \in U$)

1: **for** each user $v_i \in U$, $v_i \neq u$ **do**
2: $I_{uv_i} \leftarrow I_u \cap I_{v_i}$
3: $sim(u, v_i) \leftarrow 0$
4: **for** each $c_i \in I_{uv_i}$ **do**
5: $sim(u, v_i) \leftarrow sim(u, v_i)$+$GetSimilarity(T(u, c_i), T(v_i, c_i))$
6: **end for**
7: **end for**
8: $U_n \leftarrow$ top-n nearest neighbors of u
9: **for** each $c_i \in \overline{I_u}$ **do**
10: $R(u, c_k) = \sum_{v_i \in U_n} w(v_i) \times sim(u, v_i)$
11: **end for**
12: sort $\overline{I_u}$ by the descending order of $R(u, c_k)$, $c_k \in \overline{I_u}$
13: **return** top M recommended items from $\overline{I_u}$

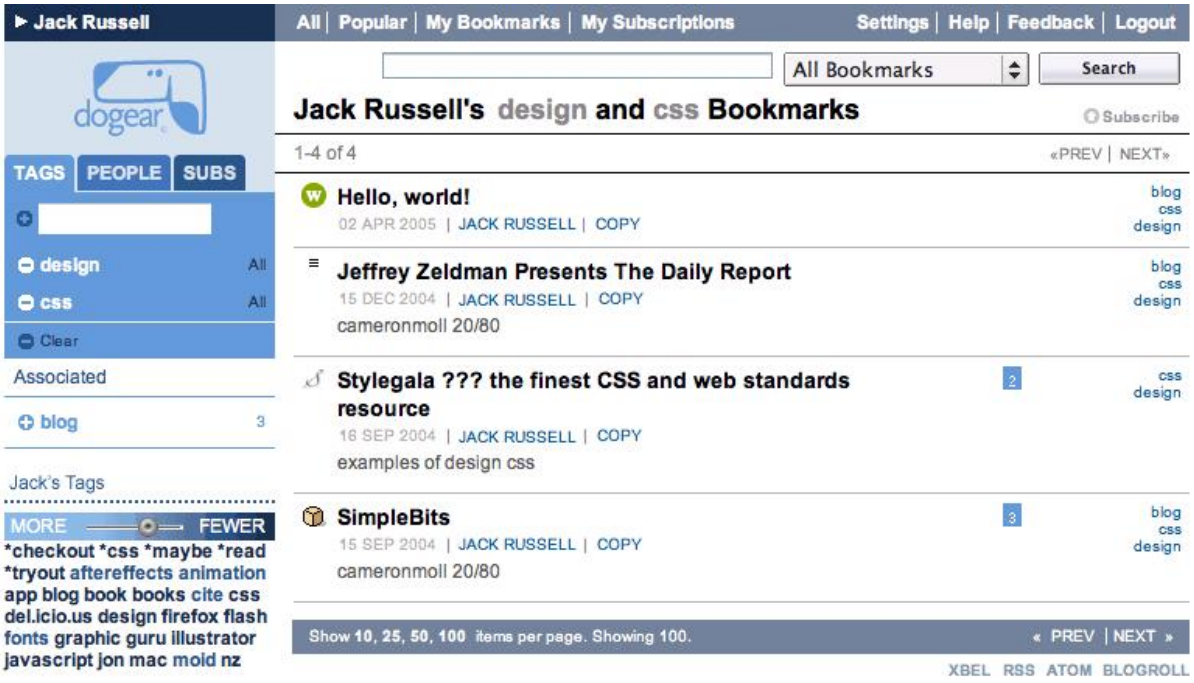

Figure 2. Dogear system in IBM

EXPERIMENTAL EVALUATION

In this section, we will describe the datasets, evaluation criteria and different protocols used in the experiments. Based on the experimental results, we present a detailed analysis and discuss the predictive accuracy.

Datasets Preparation

We demonstrate the working of our approach on the datasets extracted from the web logs of Dogear shown in Figure 2 which is an enterprise collaborative bookmarking system within IBM comparable with delicious. Dogear exploits the enterprise by allowing people to bookmark pages within their Intranet.

Moreover, it uses enterprise directories to authenticate and record the user's identity, which allows people to find experts on specific topics within the company. For example, an employee looking for someone knowledgeable in Java can look at the dogear "java" tag to see who has been bookmarking pages around that topic. Dogear will also show tags associated with "java," which may help to refine the search. Once users have found a potential expert, they can see that person's bookmarks, internal blog, and contact information. This form of expertise location helps spur collaboration and sharing of resources within the company.

We use a triple relation $< userID, pageID, tag >$ to record which user puts which tag on which page. The majority of the data in dogear comes from two sources mainly: the first one includes all the pages and tags created within the intranet, and the second one contains the data imported from the users' own bookmarked pages in their browser. Based on the above triple relation, we have extracted two datasets with different characteristics shown in Table2.

Table 2. Tag-based User-Item Matrix

	Dataset 1	Dataset 2
Total Users	8000	2756
Total Pages	5315	12000
Total tags	7670	9006
Mean pages per user	7	7
Mean pages per page	10	1.6
Mean pages per page	6.8	4.8

Evaluation Criteria

The effectiveness of a collaborative filtering algorithm depends on the ways in which recommendations will be presented to the user. Traditionally, there exist two basic styles of recommendations. In terms of the first one, individual items are presented one-at-a-time to the users along with a rating indicating potential interest in the topic. Thus each item has an estimated rating value, and which allows to compare this value with that of the actual rating of the active user to make an evaluation.

With respect to the second one, it presents the user with a ranked list of items by a descending order of their estimated ratings. Users can go through the list to find their most interested items. Since in our experiments the item ratings are literal tags but not real numbers, we decided to adopt the second method to generate a list of web pages being recommended, and evaluate their acceptance by the active user.

Table 3. Statistical all but one results

Dataset 1		
Algorithm	Average Precision	Average Ranking
$TBCF$	0.27	2.8
$cosine$	0.13	1.5
Dataset 2		
$TBCF$	0.037	1.4
$cosine$	0.015	1.2

Table 4. Random all but one results

Random generated subset	Average Precision $TBCF$	Average Precision cosine
500	0.208	0.121
1000	0.211	0.112
2000	0.182	0.118
3000	0.209	0.185
4000	0.202	0.173
5000	0.188	0.156
6000	0.209	0.180

Protocols and Results

We perform three classes of experiments to compare the performance of *TBCF* with that of the classic rating-based algorithm which depends on a cosine-based similarity measure.

In the first protocol, every single page that is tagged before by the active user is withheld alternately. We then try to determine a top-n list of recommendations and predict whether this withheld page could appear or not. We cycle through all the users and calculate the average value of the portion that the predicted number of pages occupies against all the tagged pages each active user statistically. This protocol is termed as "statistical all but one" and we formulate this protocol as follows. For each active user $u_i \in U$, I_{u_i} represents the set of pages that u_i has tagged. Let R_{u_i} be the set of pages that appear in the corresponding recommendation list. The average precision is thus defined as

$$SABONE = \frac{\sum_{v_i \in U_n} \frac{|R_{ui}|}{|I_{u_i}|}}{|U|}$$

The "statistical all but one" experiments measure the algorithms' performance when given as much data as possible from each active user. The higher *SABONE* would become, the more accurate the algorithm can behave. In our experiments, for each active user, we search for his top-5 most similar neighbors, and generate a top-10 recommendation list each time. The results of our *TBCF* approach and the cosine-based method are given in table 3.

In table 3, the average ranking value reflects the average position that the withheld pages hold in the corresponding top-10 recommendation list. We see that the number of users in dataset 1 is greater than the number of the pages. By contrast, in dataset 2, we have exactly the opposite situation. Therefore, although the number of the mean pages tagged by each user is the same in both datasets, the users in dataset 1 have a higher probability that two more users can tag on the same page than dataset 2. As a result, the average precision of the two algorithms in dataset 1 is greater than that in dataset 2, and *TBCF* can correctly recommend more pages than cosine in both the two situations.

In the second protocol, for a randomly selected active user, we search for his similar neighbors in a randomly generated subset of the user space instead of the whole user space. This protocol is termed as "random all but one", which focus on the performance of the neighborhood selection. In our experiments, we have randomly generated eight subsets with the total user being as 500, 1000, 2000, 3000, 4000, 5000, and 6000 from dataset 1 accordingly. Table 4 shows the experimental results.

The crucial step in collaborative filtering recommendation systems is the selection of the neighborhood. The neighbors of the active user is entirely determine his predictions. It is therefore critical to select neighbors who are most similar to the active user. In pure collaborative filtering systems, the similarity among users is only determined by the ratings given to co-rated items; items that have not been rated by both users are ignored.

However, in *TBCF*, the similarity is not only based on the rating patterns of the users, but also based on their cognitions on the same items. We claim that this feature of TBCF makes it possible to select a better, more representative and accurate neighborhood. For example, consider two users u and v with four common pages about Java technology. User u often browses these pages searching for the GUI programming techniques, but the other one often focuses on the aspects of *jsp* and *B/S* architecture. Pure collaborative filtering would consider them similar. By contrast, *TBCF* would not think so and may further find user w holding three common pages with v and having similar tags as *jsp*, tomcat and struts.

Consequently, although the number of common pages between w and v is less than that between u and v, user w could be more closer to user v than user u and therefore they would be considered neighbors. We believe that this superior selection of neighbors is one of the reasons that *TBCF* outperforms pure cosine-based collaborative filtering approach shown in figure 3.

In the third protocol, we randomly select 2 and 5 tagged pages from each user as the observed pages, and attempt to predict the remaining ones. This protocol is called k-given where $k = 2$ and 5 respectively. We use *recall* and *precision* from the field of information retrieval to evaluate the returned items. For a given number of returned items, *recall* is the percentage of relevant items that were returned and *precision* is the percentage of returned items that are relevant. More formally, for each active user $u_i \in U$, I_{u_i} represents the set of pages that u_i has tagged. Suppose we use set K to store the observed pages, and set R to represent the recommended top-n (in our experiment $n = 10$) list. The

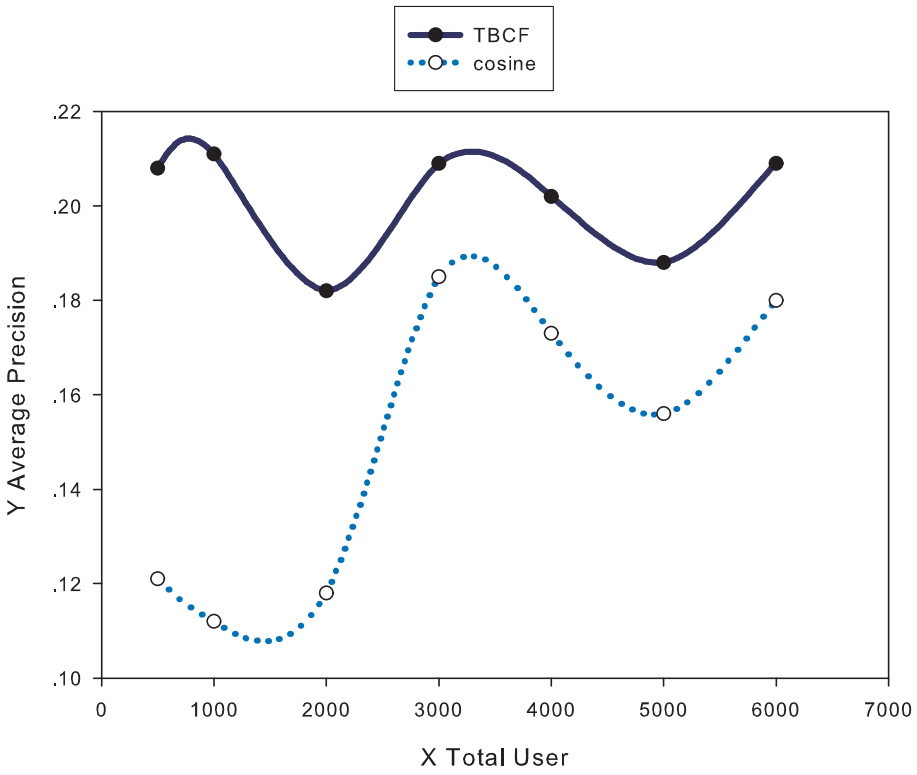

Figure 3. Random All but One

Table 5. Random all but one results

Dataset 1				
Algorithms	Given	Recall	Precision	ValidUser
TBCF	2	0.28	0.16	6247
cosine	2	0.23	0.13	6247
TBCF	5	0.27	0.15	4832
cosine	5	0.23	0.12	4832
Dataset 2				
TBCF	2	0.042	0.023	1284
cosine	2	0.037	0.019	1284
TBCF	5	0.041	0.039	700
cosine	5	0.034	0.033	700

recall is defined as

$$recall = \frac{|(I_{u_i} - K) \cap R|}{|I_{u_i}| - |K|}$$

and $precision$ is given as

$$precision = \frac{|(I_{u_i} - K) \cap R|}{|R|}$$

Given $k = 2$ and 5, we go through the whole user space and calculate the average value of *recall* and *precision* respectively, shown in table 5. The various Given experiments look at users with less data available, and examine the performance of the algorithms when there is relatively little known about an active user. In running the tests, if a user did not have adequate tagged pages for the "Given" value, he will be eliminated from the evaluation. Thus the ValidUser evaluated under each protocol is different from each other. Table 5 demonstrates that even in the extremely sparse situation, *TBCF* also outperforms the cosine method due to its better neighborhood searching mechanism.

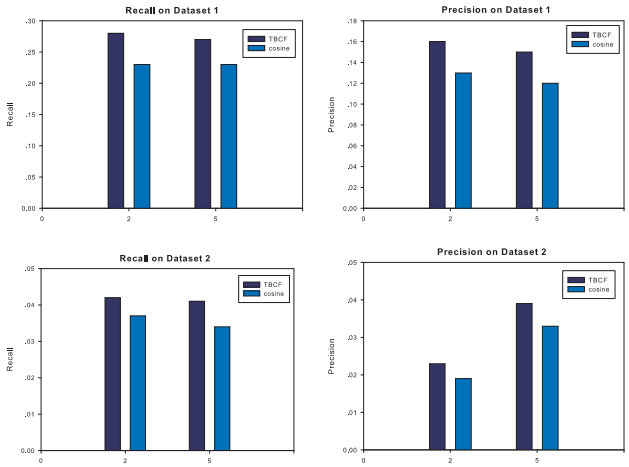

Figure 4. Network Modularity Q and f

CONCLUSION AND FUTURE WORK

Incorporating the semantic information of the tags associated with the shared resources into collaborative filtering can

significantly improve predictions of a recommender system. In this paper, we have provided an effective way to achieve this goal. We have shown that how tag-based collaborative filtering outperforms the pure cosine-based collaborative filtering method.

TBCF elegantly uses the semantic similarity among the tags to significantly improve the neighborhood searching process which is a critical step for the collaborative recommendation system and directly determines the accuracy of the final predictions. We have tested *TBCF* with two different datasets from the real-life applications. It can also overcome the sparsity disadvantage to some extent.

Although *TBCF* performs consistently better than pure collaborative filtering, the difference in performance is not very large. For the future work, we could further improve the calculation of the semantic distance among new tags that are not stored in *WordNet* yet, and we could use the community wisdom in social network analysis to improve the neighbor searching as well.

ACKNOWLEDGMENTS

We thank Rich Thompson for his generous help and the valuable comments greatly.

REFERENCES

1. M. Balabanović and Y. Shoham. Fab: content-based, collaborative recommendation. *Commun. ACM*, 40(3):66–72, 1997.

2. S. Banerjee and T. Pedersen. An adapted lesk algorithm for word sense disambiguation using wordnet. In *CICLing '02: Proceedings of the Third International Conference on Computational Linguistics and Intelligent Text Processing*, pages 136–145, London, UK, 2002. Springer-Verlag.

3. D. Billsus and M. J. Pazzani. Learning collaborative information filters. In *Proc. 15th International Conf. on Machine Learning*, pages 46–54. Morgan Kaufmann, San Francisco, CA, 1998.

4. J. S. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. pages 43–52, 1998.

5. J. Delgado. Memory-based weightedmajority prediction for recommender systems, 1999.

6. D. M. P. Dmitry Y. Pavlov. A maximum entropy approach to collaborative filtering in dynamic, sparse, high-dimensional domains. In *Conference on Neural Information Processing Systems*, 2002.

7. W. Hill, L. Stead, M. Rosenstein, and G. Furnas. Recommending and evaluating choices in a virtual community of use. In *CHI '95: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 194–201, New York, NY, USA, 1995. ACM Press/Addison-Wesley Publishing Co.

8. T. Hofmann. Collaborative filtering via gaussian probabilistic latent semantic analysis. In *SIGIR '03: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 259–266, New York, NY, USA, 2003. ACM Press.

9. W. Kraaij and R. Pohlmann. Porter's stemming algorithm for dutch, 1994.

10. V. I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. Technical Report 8, 1966.

11. A. Nakamura and N. Abe. Collaborative filtering using weighted majority prediction algorithms. In *ICML '98: Proceedings of the Fifteenth International Conference on Machine Learning*, pages 395–403, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc.

12. P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. Grouplens: an open architecture for collaborative filtering of netnews. In *CSCW '94: Proceedings of the 1994 ACM conference on Computer supported cooperative work*, pages 175–186, New York, NY, USA, 1994. ACM Press.

13. B. M. Sarwar, G. Karypis, J. A. Konstan, and J. Reidl. Item-based collaborative filtering recommendation algorithms. In *World Wide Web*, pages 285–295, 2001.

14. G. Shani, D. Heckerman, and R. I. Brafman. An mdp-based recommender system. *J. Mach. Learn. Res.*, 6:1265–1295, 2005.

15. U. Shardanand and P. Maes. Social information filtering: Algorithms for automating word of mouth. In *Proceedings of ACM CHI'95 Conference on Human Factors in Computing Systems*, volume 1, pages 210–217, 1995.

16. S. Sood, S. Owsley, K. Hammond, and L. Birnbaum. Tagassist: Automatic tag suggestion for blog posts. March 2007.

17. L. Ungar and D. Foster. Clustering methods for collaborative filtering. In *Proceedings of the Workshop on Recommendation Systems*. AAAI Press, Menlo Park California, 1998.

18. J. Wang, A. P. de Vries, and M. J. Reinders. A user-item relevance model for log-based collaborative filtering. In *Proc. of European Conference on Information Retrieval (ECIR 2006), London, UK*, 2006.