Electronic Commerce Research and Applications 9 (2010) 73–83

Contents lists available at ScienceDirect

Electronic Commerce Research and Applications

journal homepage: www.elsevier.com/locate/ecra

Collaborative filtering based on collaborative tagging for enhancing the quality of recommendation

Heung-Nam Kim *, Ae-Ttie Ji, Inay Ha, Geun-Sik Jo

Department of Computer and Information Engineering, INHA University, 253 Yonghyun-dong, Nam-gu, Incheon 402-751, Republic of Korea

ARTICLE INFO

Article history:
Received 15 June 2008
Received in revised form 5 August 2009
Accepted 17 August 2009
Available online 26 August 2009

Keywords:
Collaborative tagging
Collaborative filtering
Recommender system

ABSTRACT

We propose a collaborative filtering method to provide an enhanced recommendation quality derived from user-created tags. Collaborative tagging is employed as an approach in order to grasp and filter users' preferences for items. In addition, we explore several advantages of collaborative tagging for data sparseness and a cold-start user. These applications are notable challenges in collaborative filtering. We present empirical experiments using a real dataset from *del.icio.us*. Experimental results show that the proposed algorithm offers significant advantages both in terms of improving the recommendation quality for sparse data and in dealing with cold-start users as compared to existing work.

© 2009 Elsevier B.V. All rights reserved.

1. Introduction

The prevalence of digital devices and the development of Web 2.0 technologies and services enable end-users to be producers as well as consumers of media content. Even in a single day, an enormous amount of content including digital video, blogging, photography, and wikis, is generated on the Web. It is getting more difficult to make a recommendation to a user about what he/she will prefer among those items automatically because of, not only their huge amount, but also the difficulty of automatically grasping their meanings. Recommender systems, which have emerged in response to the above challenges, provide users with recommendations of items that are likely to fit their needs (Sarwar et al. 2000).

One of the most successful technologies among recommender systems is Collaborative Filtering (CF). Numerous commercial on-line companies (e.g., Amazon.com, Half.com, and cdnow.com) apply this technology to provide recommendations to their customers. CF has an advantage over content-based filtering which is the ability to filter any type of items, e.g. text, music, videos, and photos (Herlocker et al. 2000). Because the filtering process is only based on historical information about whether or not a given target user (also called the active user) has preferred an item before, analysis of the actual content, itself, is not necessarily required. However, despite its success and popularity, CF encounters two serious limitations with quality evaluation: the *sparsity problem* and the cold *start problem*. A number of studies have attempted to address problems related to collaborative filtering (Sarwar et al. 2001, Schein et al. 2002, Deshpande and Karypis 2004).

The *sparsity problem* occurs when available data is insufficient for identifying similar users or items (neighbors) due to an immense amount of users and items (Sarwar et al. 2001). In practice, even though users are very active, each individual has only expressed a rating (or purchase) on a very small portion of the items (Linden et al. 2003). Likewise, very popular items may have been rated (or purchased) by only a few of the total number of users. Accordingly, it is often the case that there is no intersection at all between two users or two items and hence the similarity is not computable at all. Even when the computation of similarity is possible, it may not be very reliable, because of insufficient information processed (Papagelis et al. 2005).

The second problem is the so called *cold start problem*. This problem can be divided into *cold-start items* and *cold-start users* (Schein et al. 2002). A *cold-start user*, the focus of the present research, describes a new user that joins a CF-based recommender system and has presented few opinions. With this situation, the system is generally unable to make high quality recommendations (Massa and Avesani 2004).

To solve these limitations, in our research, we propose a new and unique approach to provide an enhanced recommendation quality derived from user-created tags. Collaborative tagging, which allows many users to annotate content with descriptive keywords (i.e., tags) (Golder and Huberman 2006), is employed as an approach in order to grasp and filter users' preferences for items. Tagging is not new, but has recently become useful and popular as one effective way of classifying items for future search, sharing information, and filtering. In terms of user-created tags, they imply

* Corresponding author. Tel.: +82 32 875 5863; fax: +82 32 872 7446.
E-mail addresses: nami4596@gmail.com, nami@eslab.inha.ac.kr (H.-N. Kim).

1567-4223/$ - see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.elerap.2009.08.004

users' preferences and opinions about items as well as metadata about them. The proposed approach first determines similarities between users with user-created tags and subsequently identifies the latent tags for each user, collectively called the *Candidate Tag Set*. Furthermore, we present, in this paper, a method of applying the candidate tags to the item recommendations.

This paper presents three specific contributions toward recommender systems. First, by introducing a new and unique recommendation algorithm via collaborative tags of users, we overcome some of the limitations in CF systems. Our algorithm is designed to run in collaborative tagging systems. Therefore, we define 3 two-dimensional matrices which are transformed from three-dimensional space for collaborative tagging system. And we present a detailed method of combining CF and collaborative tagging at different stages in the recommendation process. Second, we present a new method of building a pre-computed model from user-created tags. We, then, present a method of applying the model to item recommendations via a probabilistic approach. Third, we provide detailed experimental evaluations with a real dataset and show that how well collaborative tags effectively work in terms of improving the recommendation quality for very sparse datasets and cold-start users.

The rest of this paper is organized: Section 2 describes an overview of recent studies related to collaborative filtering and collaborative tagging. In Section 3, we describe our collaborative tagging approach and provide a detailed description of how the system uses collaborative tags for recommendation. In Section 4, we present the effectiveness of our approach through experimental evaluations comparing our approach with existing work using *del.icio.us* (http://del.icio.us/) data. Finally, we conclude with a discussion and future directions for our work.

1.1. Problem statement

We now elaborate on the need to address some problems of collaborative filtering due to insufficiency of information about users. The problems of CF, caused by the filtering process mainly depending on co-occurrence of item selections of different users, include sparsity of the user–item matrix and filtering for cold-start users.

Suppose there are three users, Alice, Bob, and Cathy, who want to find some information, useful for them, by browsing the Web. Each user makes a selection of Web contents by bookmarking. We have the assumption that a user bookmarking a Web page means that the user implicitly has an interest in the Web page.

As illustrated in Fig. 1, Alice bookmarked two Web sites, "iGoogle (www.google.com/ig)" and "Netvibes (www.netvibes.com)", Bob bookmarked "iGoogle," "Netvibes," and "MS Virtual Earth (www.maps.live.com)," and Cathy bookmarked "MS Virtual Earth" and "Windows Live (www.live.com)". In the case of using only the bookmarking information, some items may not be recommended to a user because of the data sparseness even though the user is most likely to prefer the items. If Alice is a target user who needs to get a recommendation, the system finds her neighbors who have similar preferences to hers and then recommends some items with the highest possibility among the Web contents that the other neighbors have bookmarked. "MS Virtual Earth" which Bob bookmarked can be provided to Alice because Alice and Bob have the same selections, "iGoogle" and "Netvibes." Thus, Bob can be a neighbor of Alice. However, Cathy cannot be a neighbor of Alice because they do not have any of the same selection of items and so no similarity relationship between them can be derived as you can see in the user–item matrix in Fig. 1. Since Alice bookmarked Web sites supporting personal and search services, it is likely that she would be interested in "Windows Live" which Cathy has bookmarked. Nevertheless, "Windows Live" cannot be recommended to Alice because Cathy is not her neighbor. Moreover, you can see that Cathy also bookmarked "MS Virtual Earth," however her opinion does not affect recommending "MS Virtual Earth" to Alice. That is, users who do not have any of the same item selections with a target user cannot affect the recommendation process as mentioned above. That also can be a problem when the target user is a cold-start user (e.g., Eric) who has made only a few item selections. In the user–item matrix of Fig. 1, only Cathy can be a neighbor of Eric. Therefore, only "MS Virtual Earth" can be considered to be recommended to Eric.

In order to treat the problem of data insufficiency about users, we use annotated tags of the item for enriching users' preference information.

2. Related work

In this section, we briefly explain concepts that we used in our research such as recommender systems, especially related to a user-based CF, an item-based CF, and collaborative tagging.

2.1. Collaborative filtering

CF is based on the fact that "word of mouth" opinions of other people have considerable influence on the decision making of

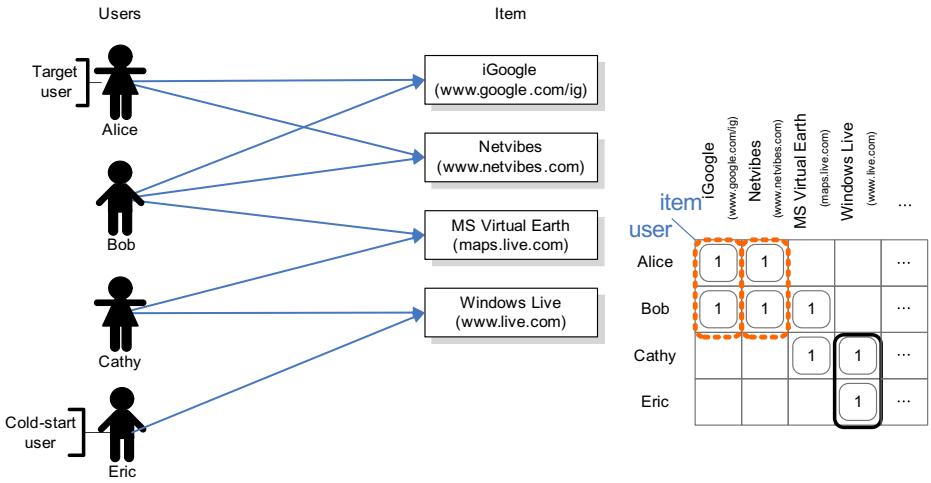

Fig. 1. An example of a user–item matrix in classic CF.

buyers (Shardanand and Maes 1995). If advisors have similar preferences with the buyer, he/she is much more likely to be affected by their opinions. Although the field of CF research has a large number of information filtering problems, generally a traditional CF domain starts with a list of l users $U = \{u_1, u_2, \ldots, u_l\}$, a list of n items $I = \{i_1, i_2, \ldots, i_n\}$, and a mapping between user–item pairs and a set of numerical ratings which can be represented as a $l \times n$ user–item matrix R (Sarwar et al. 2000). In this case, the preference indicator has the form $\langle u, i, r \rangle$ where u is a user index, i is an item index, and r is a rating value. That is, user u explicitly rated an item i with a numerical value r. Preference indicators used for CF can also be represented by click behaviors, purchase history, or Web logs, implicitly (Breese et al. 1998). In those cases, the preference indicator is represented as a form $\langle u, i \rangle$ such that user u viewed or purchased an item i (e.g., page or product). Therefore, the value of the entry for the matrix R is usually binary.

In CF-based recommendation schemes, two approaches have mainly been developed: memory-based CF (also known as user-based CF) and model-based CF (Breese et al. 1998). Following the proposal of GroupLens (Resnick et al. 1994), the first system to generate automated recommendations, user-based CF approaches have seen the widest use in recommendation systems. User-based CF uses a similarity measurement between neighbors and the target users to learn and predict preference towards new items or unrated products by a target user. However, despite the popularity of user-based CF algorithms, they have some serious problems relating to the complexity of computing each recommendation as the number of users and items grow. Besides, sparsity problems due to the insufficiency of users' historical information should be considered seriously (Sarwar et al. 2001, Papagelis et al. 2005). In order to improve scalability and real-time performance in large applications, a variety of model-based recommendation techniques have been developed. Model-based approaches, such as our algorithm, provide item recommendations by first developing a pre-computed model (Hofmann 2004). In comparison to user-based approaches, model-based CF is typically faster in terms of recommendation time, though the method may have an expensive learning or model building process. A new class of model-based CF, called an item-based CF, has been proposed (Sarwar et al. 2001, Miller et al. 2004, Deshpande and Karypis 2004) and applied to commercial recommender systems such as Amazon.com (Linden et al. 2003). Instead of computing similarities between users, an item-based CF reviews a set of items the target user has rated (or purchased) and selects the most similar items based on the similarities between items. Fig. 2 shows examples of calculating the similarity of two users, u and w, compared with calculating the similarity of two items, i and j, in the process of predicting the value of item i for user u in the user–item matrix.

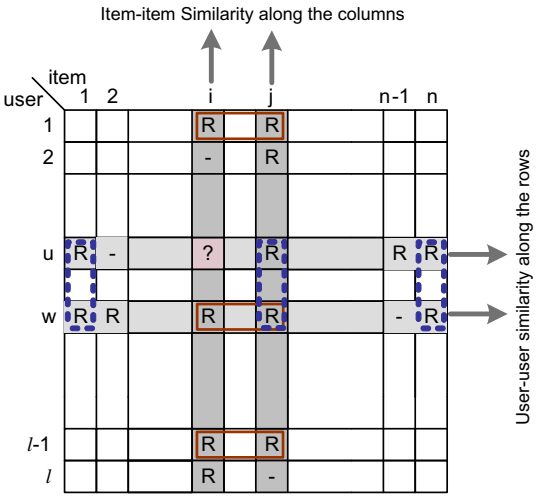

Fig. 2. The similarity computation in a user-based CF and an item-based CF.

Usually, CF systems take two steps: first, the neighbor group, the users who have a similar preference with the target user (for user-based CF) or the set of items that is similar to the item selected by the target user (for an item-based CF), should be determined by using a variety of similarity computing methods. Based on the group of neighbors, the prediction values of particular items, estimating how the target user is likely to prefer the items, are obtained and then the top-N items with a higher predicted value that will be of interest to the target user are identified.

2.1.1. Neighborhood formation

As stated above, neighbors simply mean a group of like-minded users with a target user or a set of similar items with the items that have been already been identified as being preferred by the target user. Finding nearest neighbors, a variety of similarity methods have been researched, such as cosine similarity, which is widely used, the Pearson correlation (Resnick et al. 1994), weight amplification and inverse user frequency, and default rating (Breese et al. 1998), including probability-based approaches. According to the results of the selected similarity measure, particular users or items with highest similarity are identified as neighbors. The number of neighbors may be varied depending on the characteristics of the domains and the application. However, it also has significant impact on the quality of results from the CF (Sarwar et al. 2000). The recommender systems have to determine the size of the neighborhood in order to compute the prediction results effectively. That is, if the size of the neighborhood is too small, it becomes difficult to obtain accurate results, whereas the large size of neighborhood brings about the complexity of computation even though the results might be more accurate (Sarwar et al. 2000).

2.1.2. Predictions and recommendations

Once k neighbors are found, various methods can be used to combine the ratings of neighbors to compute a prediction value on unrated items for the target user. The preference rating of each neighbor is usually weighted by the similarity value which is computed when the neighbors are determined. The more a neighbor is similar to a target user or item, the more influence he or she has for calculating a prediction value. After predicting how a target user will like particular items which have not been rated yet by the target user, the top-N item set, a set of ordered items with a higher predicted value, is identified and recommended. The target user can present feedback of whether the target user actually likes the recommend top-N items or how much he/she prefers those items as scaled ratings. The feedback can be used for updating a similarity model in order to help the model reflect changes of preferences as well as a measurement of whether the recommender system performs well or not.

2.2. Collaborative tagging and folksonomy

Collaborative tagging is the practice of allowing any user to freely annotate the content with any kind of tags (Golder and Huberman 2006). In some circumstances, such as the Web, where there is no "librarian" to classify items or there are too many items to classify by a single authority, collaborative tagging is one of the most useful ways of categorizing or indexing content (Golder and Huberman 2006). Moreover, tags are directly published and discussed on the Web and may be applied to any kinds of items, even people. Collaborative tagging can play a key role in sharing content in social networks.

Collaborative tagging is described as "folksonomy," in contrast with typical "taxonomy," even though there is some debate about the appropriateness of this term (Marlow et al. 2006). In contrast with taxonomy, tagging performs in a horizontal and inclusive way for classification and therefore can have an advantage over hierarchical taxonomy in some cases. In a taxonomy, a category having a more general concept includes more specific ones. Even though a hierarchical category assures a user that all of the items exist in one corresponding stable place, the user cannot be sure that all relevant items are returned by a query. To avoid fruitless searching, the user needs to check multiple locations. Unlike a hierarchical search, in a collaborative tagging system such items can be annotated with a variety of terms simultaneously: general tags and specific ones. In addition, tags can filter out all relevant items and return only those items tagged with those tags. As users can provide tags without any intricate implementation, a tagging system can be an effective and easy way to help identify correct items and make search results more relevant. Golder and Huberman (2006) have discussed such possible advantages over taxonomy as well as other significant issues of tagging systems.

Marlow et al. (2006) define several dimensions of tagging system design according to their possible implications. We review, briefly, two of them that are related to our work. Firstly, from the user's authority of tagging behavior, a tagging system can be classified into self-tagging, permission-based, and free-to-all. Self-tagging, where users only tag the content they created for future personal retrieval, is provided by *Technorati* (http://www.technorati.com/) and *YouTube* (http://www.youtube.com/). As in *Flickr* (http://www.flickr.com/), permission-based tagging is provided as a way of specifying different levels of permission that users can annotate the content with tags. These two forms of tagging are also mentioned as narrow folksonomies (Wal 2005). Strictly speaking, they partially or do not support *collaborative* tagging. *Del.icio.us* and *Yahoo! MyWeb* (http://myweb.yahoo.com/) provide free-to-all tagging allowing any user to tag any items. Free-to-all tagging, the focus of the present research, is also known as a broad folksonomy (Wal 2005).

Secondly, according to the aggregation of tags, a tagging system is divided into a bag-model and a set-model. A set-model does not allow any repetition of tags. So the system shows users only the "set" of tags attached for the item (e.g., *Flickr*, *YouTube*, and *Technorati*). In contrast with a set-model, a bag-model system, the focus of the present research, allows duplicated tags for the same item from different users (e.g., *del.icio.us*, *Yahoo! MyWeb*). Based on the statistical information of tag frequencies, the system is able to present the item with the collective opinions of the taggers.

3. Collaborative filtering based on collaborative tagging

Fig. 3 illustrates our method with two phases: a building model phase and a probabilistic recommendation phase. In the model building phase, we first generate the latent tags that will be of interest to a target user, collectively called a *Candidate Tag Set* (*CTS*), by using a collaborative filtering scheme. Based on the *CTS*, a *naïve Bayes* approach is applied to decide which items to recommend stochastically.

As mentioned previously, the tagging system in our research is free-to-all and the bag-model system. Therefore, the preference indicators used for our algorithm are a co-occurrence triplet such as $\langle u, i, t \rangle$ where u is a user index, i is an item index, and t is a tag index. The triplet implies the fact that user u annotated or bookmarked item i with tag t. Conceptually, we can represent the triplet in the form of a three-dimensional data cube, as in Fig. 4.

In our research, the data cube is transformed into 3 two-dimensional matrices. Before describing the algorithms, some definitions of the matrices are introduced.

Definition 1 (*User–item binary matrix, R*). If there is a list of l users $U = \{u_1, u_2, \ldots, u_l\}$, a list of n items $I = \{i_1, i_2, \ldots, i_n\}$, and a mapping between user–item pairs and the opinions, user–item data can be represented as a $l \times n$ binary matrix, R, referred to as a user–item matrix. The matrix rows represent users, the columns represent items, and $R_{u,i}$ represents the historical preference of a user u on an item i. Each $R_{u,i}$ is set to 1 if a user u has selected (or tagged) an item i or 0 otherwise.

Definition 2 (*User–tag frequency matrix, A*). For a set of m tags $T = \{t_1, t_2, \ldots, t_m\}$, tag usages of l users can be represented as a $l \times m$ *user–tag matrix*, A. The matrix rows represent users, the columns represent tags, and $A_{u,t}$ represents the number of items that a user u has tagged with a tag t.

Definition 3 (*Tag–item frequency matrix, Q*). This is a $m \times n$ matrix of tags against items that have as elements the frequencies of tags to items. The matrix rows represent tags, the columns represent items; and $Q_{t,i}$ implies the number of users who have tagged an item i with a tag t.

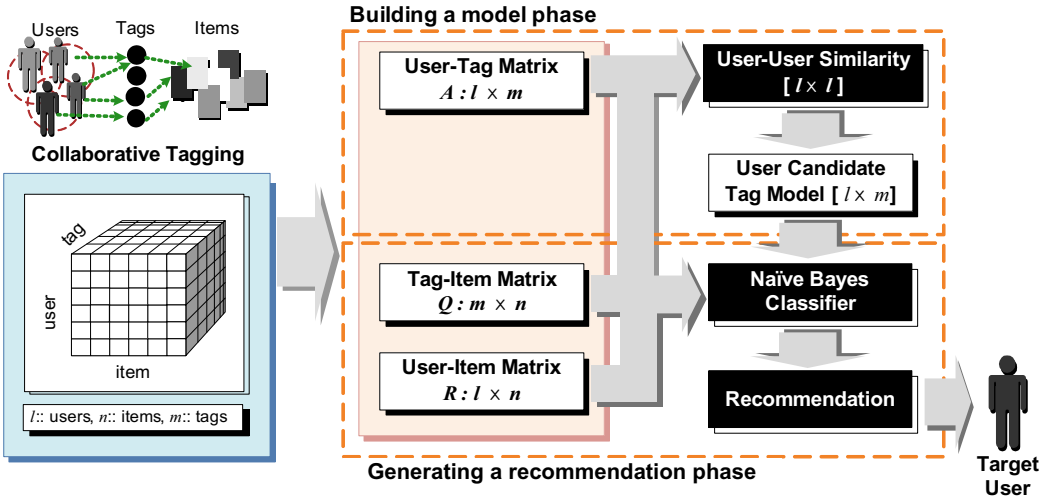

Fig. 3. An overview of the proposed approach for item recommendations.

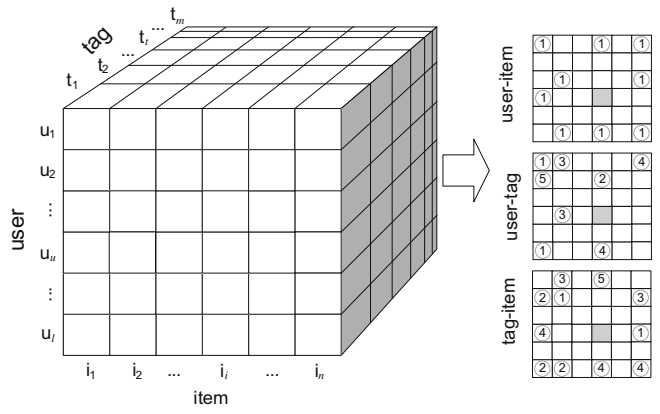

Fig. 4. A three-dimensional data cube representation for collaborative tagging system.

3.1. Building a user candidate tag model

The basic idea of generating candidate tags starts from assuming that a target user is likely to prefer tags that have been used by similar users or by the target user before. The *Candidate Tag Set* (*CTS*) for a certain user includes tags which implied his/her latent preference (see Fig. 5).

Concentrate on the same situation as the example described in Section 1.1. However, it is different from the previous condition where the system constructs a Candidate Tag Model that the target user tagged when he or she bookmarked. Suppose that Alice used "*personal*" and "*portal*" tags many times among the tags while she made her bookmarks as shown in the user–tag matrix of Fig. 6. The system finds other tags similar to her tags by using the collaborative filtering process in order to construct the *CTS*. If her *CTS* consists of {*personal, portal, search*}, the "Windows Live" page can be recommended to Alice because of the tag "*personal*" and "*search*" which the "Windows Live" page already has. As you can see from the example, tags, which were tagged when the item was selected, enrich information about users and items such as building the *CTS* model reflecting user's preferences even though the users do not select the items directly and so the data sparsity problem can be partially improved by using tags.

In this context, the information insufficiency problem due to cold-start users who have few item selections also can be dealt with by using tags. Let us assume that a new user Eric makes his first bookmark, "Windows Live," with a tag such as "*search*". If Eric's *CTS* constructed by the system consists of {*personal*, *Web2.0*, *portal*}, "iGoogle" and "Netvibes" pages can be considered for recommending to Eric because of the tag "*personal*," "*Web2.0*," and "*portal*" which those pages already have. That is, the recommender system combined with annotated tags can recommend Alice and Bob's bookmarks to Eric even though they are not his neighbors. Practically, a user uses several tags for one contents selection. If Eric makes his first bookmark with several tags, the system is more likely to make relatively proper recommendations. In our collected del.*icio.us* dataset for experiments, the number of tags used by each user averages 2.98 a bookmark. We observe that the tags can be more helpful to grasp a user's preference than only a few bookmarks by alleviating the sparsity of the data matrix as shown in a user–tag matrix comparing with user–item matrix of Fig. 6. That is, tags make the information used for analyzing users' preference extendable to more than twice than the original; therefore, the system is more likely to recommend relatively suitable items for cold-start users (see Fig. 7).

3.1.1. Neighborhood formation using tagging

The most important task in CF-based recommendations is the similarity measurement because different measurements lead to different neighbor users, in turn, leading to different recommendations. Since the *user–item matrix R* is usually very sparse, which is one of the limitations of CF, it is often the case that two users do not share a sufficient number of items selected in common for computing similarity. For this reason, in our research, we select the best neighbors, often called *k nearest neighbors*, with tag frequencies of the corresponding user in the *user–tag matrix, A*.

In order to find the *k* nearest neighbor (*KNN*), cosine similarity, which quantifies the similarity of two vectors according to their angle, is employed to measure the similarity values between a target user and every other user. *KNN* includes users who have a higher similarity score than the other users and means a set of users who prefer more similar tags with a target user. In cosine similarity between users, two users are treated as two vectors in the *m*-dimensional space of tags. In addition, we also consider the number of users for tags, namely *the inverse user frequency*. Consider two tags, t_1 and t_2, both having been tagged by user u and v; however, just 10 users used tag t_1, whereas 100 users used tag t_2. In this situation, tag t_1, tagged by fewer users, is relatively more reliable for the similarity of user u and v than tag t_2 tagged by many users. Therefore, the *inverse user frequency* as described in Breese et al. (1998) for our algorithm is slightly modified such as Definition 4. Likewise with the inverse document frequency, the main idea is that tags used by many users present less contribution with regard to capturing similarity, than tags used by a smaller number of users.

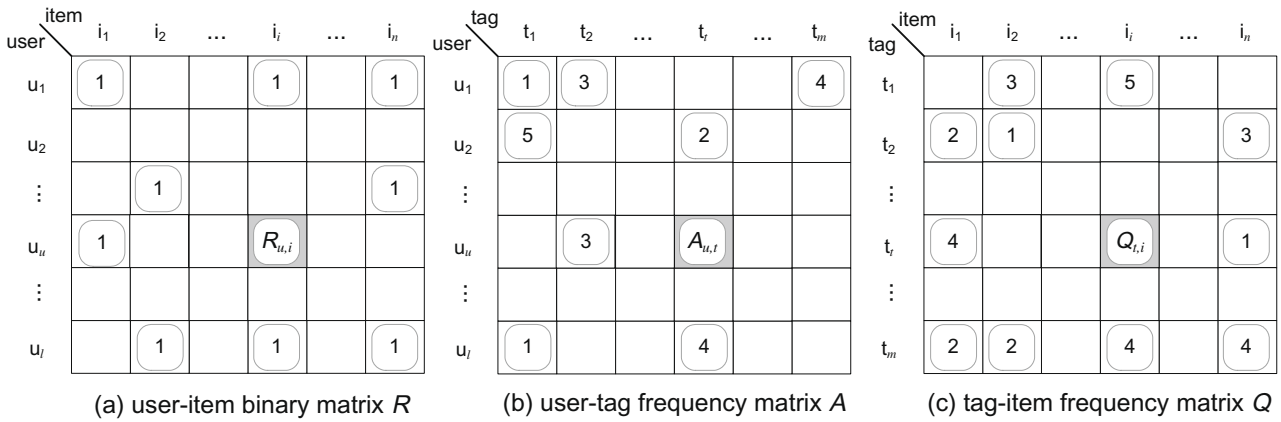

Fig. 5. Three matrices for a tag-based collaborative filtering system.

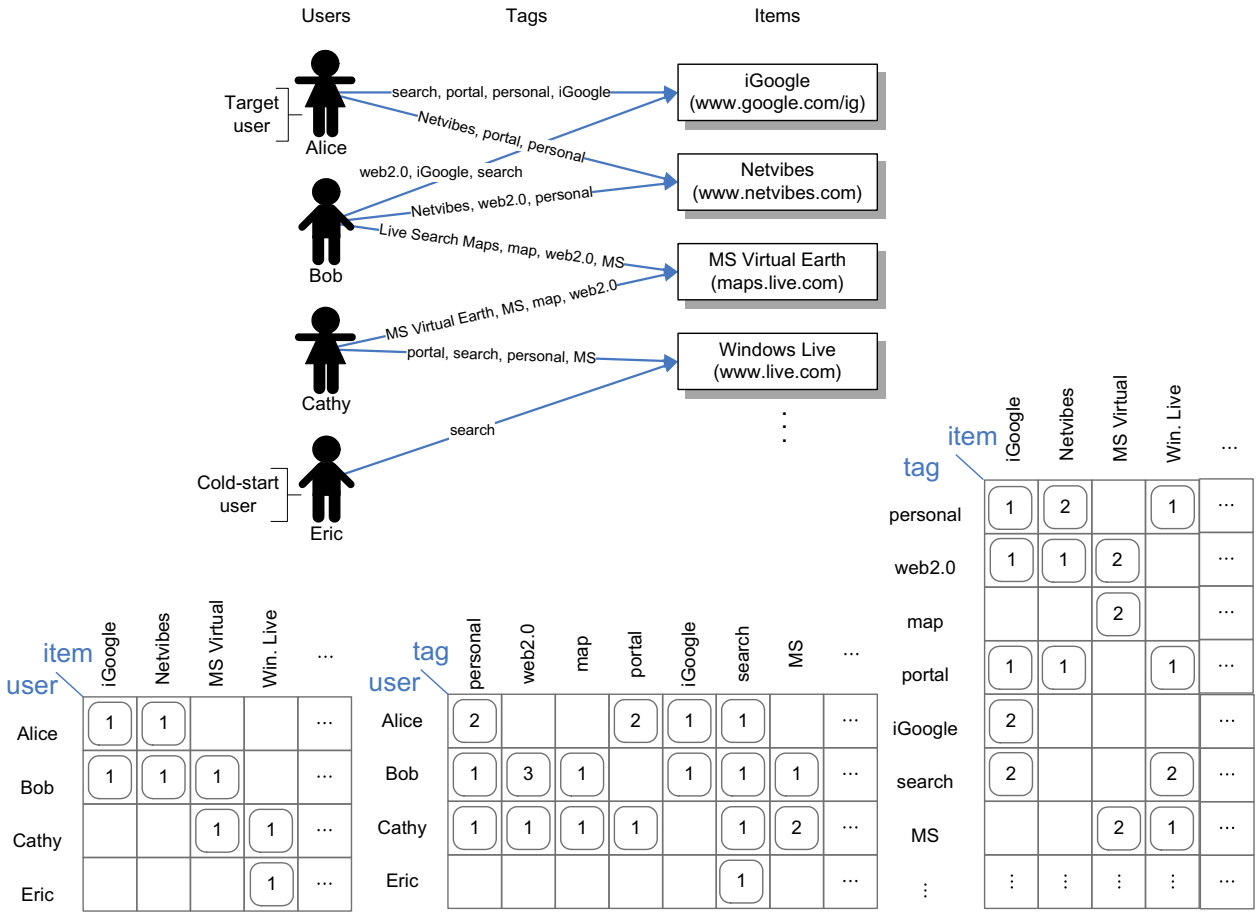

Fig. 6. An example of a user–item matrix, a user–tag matrix, and a tag–item matrix.

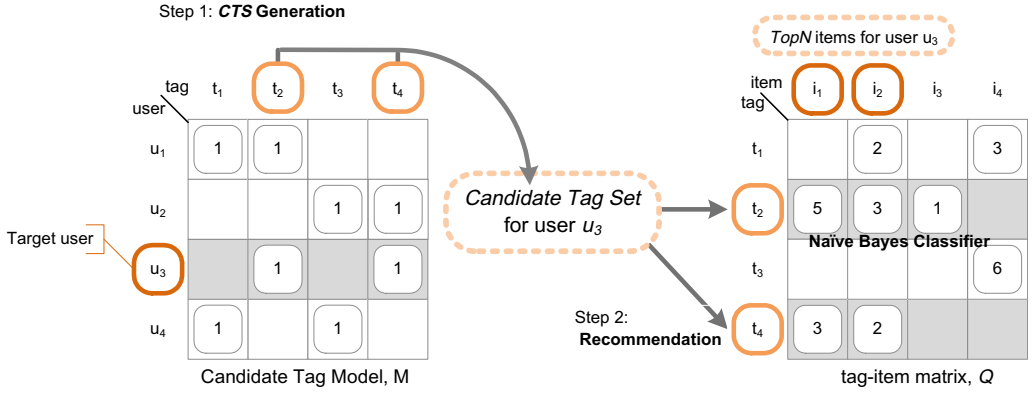

Fig. 7. The process of item recommendation via a naïve Bayes approach.

Definition 4 (*Inverse user frequency*). Let l be the total number of users in the system and n_t the number of users tagging with a tag t. Then, the *inverse user frequency* for a tag t, iuf_t, is computed: $iuf_t = log(l/n_t)$. If all users have tagged using tag t, then the value of iuf_t is zero, $iuf_t = 0$.

When *the inverse user frequency* is applied to the cosine similarity technique, the similarity between two users, u and v, is measured by the following equation:

$$sim(u,v) = \cos(\vec{u},\vec{v}) = \frac{\sum_{t\in T}(A_{u,t}\cdot iuf_t)(A_{v,t}\cdot iuf_t)}{\sqrt{\sum_{t\in T}(A_{u,t}\cdot iuf_t)^2}\sqrt{\sum_{t\in T}(A_{v,t}\cdot iuf_t)^2}} \quad (1)$$

where T is a set of tags and $A_{u,t}$ and $A_{v,t}$ denote the tag frequencies of users u and v each in the *user–tag matrix, A*. In addition, iuf_t refers to the inverse user frequency of tag t. The similarity score between two users is in the range of [0, 1]. The higher score a user has, the more similar he/she is to a target user. Finally, for l users, we compute a $l \times l$ user–user similarity matrix D.

3.1.2. Tag preference generation

For constructing a candidate tag model, first a user's preference should be predicted for a tag. To this end, we use the "All But 1" protocol from Breese et al. (1998). In this protocol, we withhold a single randomly selected tag for a target user in the entire tag

set. We, then, try to predict its value. In order to compute the prediction value of the target user u for tag t, the sum of the tag frequency given by neighbor users on the tag t are used. Each frequency is weighted by the corresponding similarity between the target user and each neighbor. The weighted sum leads to a higher prediction value for a more similar user to a target user. Formally, the measurement of how much the target user u prefers tag t is given by:

$$S_{u,t} = \sum_{o \in KNN(u)} (A_{o,t}) \cdot sim(u, o) \tag{2}$$

where $KNN(u)$ is a set of k nearest neighbors of user u, and $A_{o,t}$ is the tag frequency of neighbor user o on tag t. $sim(u, o)$ represents the similarity between users, u and o, which is calculated as mentioned in Eq. (1). The tag preference of users can be represented as a matrix, S, where rows represent users and columns represent tags. An entire $l \times m$ user–tag preference matrix, S, can be filled in using Eq. (2). The algorithm for computing user–tag preferences is shown in Algorithm 1.

Algorithm 1. Tag preference prediction algorithm

```
Input: total user list U; size of KNN k; user–tag
  matrix A; user–user similarity matrix D;
Output: user–tag preference matrix S
computingTagPreference(U, k, A, D)
01: for each u ∈ U
02:   for i ← 1 to l // l is row count of matrix A
03:     add D_{u,i} to itemset KNN
04:   for each o ∈ KNN // get KNN of each user
05:     if o ≠ among the k largest values i · n KNN
        then
06:       remove o from KNN
07:   for each t ∈ T // compute the preference
      matrix S
08:     for each o ∈ KNN
09:       S_{u,t} ← S_{u,t} + (A_{o,t} × D_{u,o})
10: return S
```

3.1.3. Building a candidate tag model

Algorithm 2. Candidate tag model construction algorithm

```
Input: total user list U; model size w; user–tag
  preference matrix S;
Output: user–tag candidate tag matrix M
buildingModel(U, S, w)
01: for each u ∈ U
02:   for i ← 1 to m // m is column count of matrix S
03:     M_{u,i} ← S_{u,i}
04:   for i ← 1 to m
05:     if M_{u,i} = 0 ∨ M_{u,i} ≠ among the w
        highest values in M_{u,*}
06:       then M_{u,i} ← 0
07:       else M_{u,i} ← 1
08: return M
```

Once we have calculated the predictions for users on tags, a user candidate tag model is constructed as shown in Algorithm 2. The input is the $l \times m$ user–tag preference matrix, S, described in Section 3.1.2 and a parameter w that determines the number of candidate tags. The term w is the model size. The output is a candidate tag model represented by a $l \times m$ user–tag matrix, M. The matrix S is binarized: $M_{u,t}$, which implies the uth user for the yth tag, is set to 1 if the corresponding prediction value $S_{u,t}$ is greater than the w highest prediction value in the uth row of S and 0, otherwise. Non-zero entries per each row, collectively called the *Candidate Tag Set*, are used to recommend items for the target user.

Definition 5 (*Candidate Tag Set, CTS*). Given the set of all tags T, the Candidate Tag Set for user u, denoted as the $CTS_w(u)$, is defined as a set of preference tags of user u such that $CTS(u) \subseteq T$. In particular, $CTS(u)$ that have a model size of w, written as $CTS_w(u)$, indicates an ordered set of tags such that $|CTS_w(u)| \leqslant w$ and $M_{u,t} = 1$, $t \in CTS_w(u)$.

3.2. Item recommendation via a probabilistic approach

The final step in our algorithm is to generate a recommendation such as a list of N items that the target user will like the most. In our research, an item recommendation for a CF is treated as a classification problem. Based on a candidate tag model for each user, top-N items are recommended stochastically with the multinomial event model of a *naïve Bayes assumption* (McCallum and Nigam 1998). To apply the *naïve Bayes* classifier to the recommendation process of the target user u, an item is referred to the class item and all of the tags in the candidate tag model for user u, $CTS_w(u) = \{t_1, t_2, \ldots, t_w\}$, are used as features. Suppose that there are n class items, $I = \{i_1, i_2, \ldots, i_n\}$. Given a tag instance t_j in $CTS_w(u)$ as a set of feature variables, the classifier predicts that $CTS_w(u)$ belongs to the class item having the highest posterior probability conditioned on $CTS_w(u)$. We make a similar naïve Bayes assumption that the tags are independent given the items. The *posterior* probability, as a preference probability $P_{u,y}$ of user u with $CTS_w(u)$ for an item i_y is obtained:

$$P_{u,y} = P(i_y | CTS_w(u)) = P(I = i_y) \prod_{j=1}^{w} P(t_j | I = i_y)^{(A_{u,j}+1)} \tag{3}$$

where $A_{u,j}$ denotes the tag frequency of the uth user for the jth tag in the user–tag matrix, A. In addition, a *priori* probability, $P(I = i_y)$, and the probability of tag t_j in item i_y, $P(t_j \mid I = i_y)$ are computed:

$$P(I = i_y) = \frac{\sum_{u=1}^{k} R_{u,y}}{\sum_{y=1}^{n} \sum_{u=1}^{k} R_{u,y}}, \quad P(t_j | I = i_y) = \frac{1 + Q_{j,y}}{m + \sum_{t=1}^{m} Q_{t,y}} \tag{4}$$

where $R_{u,y}$ and $Q_{j,y}$ denote the binary value of the uth user for the yth item in the user–item matrix, R, and the tag frequency of the jth tag for the yth item in the tag–item matrix, Q, respectively. Avoiding the situation where $Q_{j,y}$ turn out to be zero, we use the Laplacean prior in Eq. (4) (McCallum and Nigam 1998).

Once the preference probabilities of the target user for items, which he/she has not yet selected, are computed, the items are sorted in order of descending probability values $P_{u,y}$. Finally, a set of N ordered items that have obtained the higher values are identified for user u. Then, those items are recommended to user u (the top-N recommendation) (Deshpande and Karypis 2004). The entire recommendation process proposed is described in Algorithm 3.

Definition 7 (*Top-N recommendation*). Let I be a set of all items; I_u an item list that has been already selected by user u; and L_u, an item list that user u has not yet selected. $L_u = I - I_u$ and $I_u \cap L_u = \varnothing$. Given the two items, i and j, $i \in L_u$ and $j \in L_u$. The item i will be of more interest to user u than the item j if and only if the posterior probability $P_{u,i}$ of item i is higher than that of item j, $P_{u,i} > P_{u,j}$. The top-N recommendation is an ordered set of N items, $TopN_u$, that will be of interest to user u, $|TopN_u| \leqslant N$ and $TopN_u \subseteq L_u$.

Algorithm 3. Top-*N* recommendation

Input: target user u; user–item matrix **R**; user–tag matrix **A**; tag–item matrix **Q**; candidate tag matrix **M**; size of *CTS* **w**; size of *Top-N* **N**; items not selected by user u $\mathbf{L_u}$
Output: a set of N items, $\mathbf{TopN_u}$

generatingTopNItem(u, w, N, L_u, M)
01: // get *CTS* of user u from the matrix M
02: **for** $i \leftarrow 1$ to m // m is column count of matrix M
03: **if** $M_{u,i} = 1$ **then**
04: add tag i to $CTSw(u)$
05: **for** each $i_y \in L_u$
08: $P_{u,y} \leftarrow$ **NaïveBayes**$(u, CTSw(u), i_y, R, A, Q)$
09: **for** each $i_y \in L_u$
10: **if** $P_{u,y}$ is within the N highest values in $P_{u,*}$ **then**
11: add item y to $TopN_u$
12: **return** $TopN_u$

3.3. Computational complexity

This section discusses the scalability of the collaborative filtering algorithm based on the collaborative tagging. The computational complexity of our algorithm is closely connected with time required to build the candidate tag model and time required to generate item recommendation based on the model. The former can be accomplished offline, prior to actual recommendations for a given user, whereas the latter has to be done online and often in real-time.

In the model building phase, we need to identify a set of preference tags, called *CTS*, for each user. Therefore, we need to compute the similarity between a target user and the other users in the *user–tag matrix, A*, and then predict each tag preference of the target user. The upper bound on the complexity of this step is $O(lm + km) \cong O(lm)$, where l is the number of users, m is the number of tags, and k is the size of the nearest users. Finally, the computational complexity of building the candidate tag model for all users becomes $O(l^2m)$, which is independent of the number of items n. Comparing the well known user-based CF (Sarwar et al. 2001) and item-based CF (Deshpande and Karypis 2004), the complexity of the former for building a user–user similarity matrix and that of the latter for building an item–item similarity matrix is $O(ln^2)$ and $O(l^2n)$, respectively.

In the probabilistic recommendation phase, the complexity required to generate the top-N item recommendations for a target user is given as $O(wn) \cong O(n)$ because we need to compute the posterior probability of the user for each items using the w candidate tags.

4. Our experimental results

In this section, we empirically evaluate the recommendation algorithm via collaborative tagging and compare its performance against the performances of the benchmark algorithms. The system prototype was implemented using *JDK* 5.0 and *MySQL* 5.0 and experiments were performed on *Dual Xeon* 3.0 GHz, 2.5 GB RAM computers, running a MS-Window 2003 server.

4.1. The dataset and evaluation metrics

The experimental data comes from *del.icio.us* which is a well-known social bookmark service supporting collaborative tagging. We collected the dataset by crawling the *del.icio.us* site. The dataset contains 27,066 bookmarks from 17,390 Web pages bookmarked by 1544 users (1544 rows and 17,390 columns of a user–item binary matrix, *R*) and 10,077 unique tags were tagged 44,681 times by 1544 users (1544 rows and 10,077 columns of a user–tag frequency matrix, *A*). The dataset was also converted into a tag–item frequency matrix, *Q* that had 10,077 rows (i.e., 10,077 tags) and 17,390 columns (i.e., 17,390 Web pages). The sparsity level, which is defined as 1 − (non-zero entries/total entries) (Sarwar et al. 2001), of the collected dataset is: The user–item matrix *R* is $1 - (27{,}066/1544 \times 17{,}390)$ which is 0.9989 and the user–tag matrix *A* is $1 - (44{,}681/1544 \times 10{,}077)$ which is 0.9971. Table 1 summarizes our dataset.

To evaluate the quality of the recommendation, the dataset was divided into two groups; 80% of the data (21,653 bookmarks) was used as a training set and 20% of the data (5413 bookmarks) was used as a test set.

The performance was measured by looking at the number of items in the test set that were also included in the top-*N* items recommended for a target user *u*, also called *recall* (Sarwar et al. 2000, Deshpande and Karypis 2004). In our research, *recall*, which can be useful in a comparative fashion on the same dataset, is a measure of how often a list of recommendation contains an item that the user has actually bookmarked. The *HitRatio* for each target user *u* is given:

$$HitRatio(u) = \frac{|Test_u \cap TopN_u|}{|Test_u|} \tag{5}$$

where $Test_u$ denotes a set of the item list of a target user u in the test data and. $TopN_u$ is a top-N recommended item list for the user u. Finally, the overall *recall* of all users in the test set is computed:

$$recall = \frac{\sum_{u=1}^{k} HitRatio(u)}{k} \times 100 \tag{6}$$

Although *recall* metric is widely used for evaluating recommender systems, it cannot be treated as the traditional information retrieval *recall* measure because there is no ground truth about which items are good (Herlocker et al. 2004, Miller et al. 2004). For instance, it is infeasible to ask 1544 users to visit and evaluate each of 17,390 items (Web pages).

4.1.1. Benchmark algorithms

In order to compare the performance of our algorithm, a user-based CF algorithm, where the similarity is computed by cosine-based similarity (denoted as *UserCF*) (Sarwar et al. 2000), and the item-based CF approach of Deshpande and Karypis (2004), which employs cosine-based similarity (denoted as *ItemCF*), were implemented. The *top*-N recommendation via collaborative tagging (denoted as *TagCF*) was evaluated in comparison with benchmark algorithms.

4.2. Experimental results

In this section, we present detailed experimental results. The performance evaluation is divided into three dimensions. The performances of the benchmark algorithms are first evaluated, and then our algorithm is evaluated. Finally, we compare our performance to that obtained by benchmark algorithms. In the experiments, we set the number of recommended items *N* to 10 for each user in the test set.

4.2.1. A performance evaluation of the benchmark algorithms

The following experiments are investigations of the effect of the neighborhood size on the performance of benchmark algorithms because the size has significant impact on the quality of the recommendations. Therefore, we evaluated the quality of the benchmark

Table 1
The dataset from *del.icio.us*.

Number of users	Number of items	Number of tags	Number of bookmarks	Number of taggings
1544	17,390	10,077	27,066	44,681

algorithms, *UserCF* and *ItemCF*, based on a user–item matrix, *R*, where we varied the neighborhood size *k* from 10 to 100. In the case of *UserCF*, the parameter *k* denotes the number of the nearest users (*k* nearest neighbors) (Sarwar et al. 2000), whereas it is the size of the most similar items for *ItemCF* (*k* most similar items) (Deshpande and Karypis 2004).

Fig. 8 illustrates the variation of *recall* for each algorithm. We observe that the recommendation quality improves as the number of *k* is increased. *ItemCF* elevates *recall* as the neighborhood size increases from 10 to 70; beyond this point, the curve tends to become flat. Likewise, *UserCF* improves until a neighborhood size of 60. The result demonstrates that, at all neighborhood size levels, *ItemCF* performed better than *UserCF*. This is caused by the sparsity of the dataset being too high to compute user–user similarities. However, because the number of items ($n = 17{,}390$) is far larger than the number of users ($l = 1544$), $n \gg l$, the computational requirements of the item–item similarity matrix, $O(ln^2)$, took much longer than those of the user–user similarity matrix, $O(l^2n)$.

4.2.2. A performance evaluation of our algorithm

The next experiments involved investigations of the effect of the neighbor size and the model size (i.e., the size of the *CTS*) on the performance of our algorithm. We expected that the size of the *CTS*, *w*, could be a significant factor affecting the quality of the recommendation in our work. So, we first evaluated our algorithm by measuring *recall* according to each model size from 10 to 100. In order to compute the tag preference of users from a user–tag matrix, *A*, the size of the neighborhood *k* was set to 60 in consideration of *UserCF*.

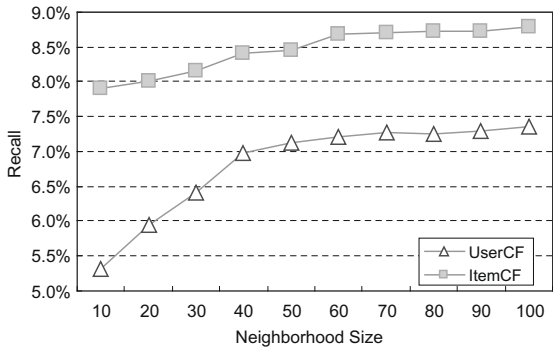

Fig. 8. Recall for the user-based CF and item-based CF as the neighborhood size grows.

Fig. 9 (left) illustrates a variation in *recall* by changing the *w* value. It can be observed that *recall* improved as the *w* value was increased from 10 to 50. After this value, any further increases of model size did not practically affect the performance. Interestingly, the quality of the algorithm was rather worse when *w* was 100 (recall was 8.79%); whereas its recall was 8.96% which was the highest when *w* was 80. This result indicates that superfluous tags which do not represent user's preference can be included in the *CTS*. That is, selecting too many numbers of tags can cause not only a bad impact on representing user's preference but also unnecessary computation cost. For this reason, *w* should be selected within a reasonable level for experimentation.

We continued to examine the effect of the variation of *recalls* according to the neighborhood size *k* which is closely connected with tag preference generation. From the previous experiment, the model size *w* was set 50. Fig. 9 (right) shows a graph of how recall changes as the neighbor size grows. In general, with the growth of the size, *recalls* tend to be elevated. However, after the neighborhood of size 50, increasing the value of *k* did not lead to statistically significant improvements. That is, once the number of nearest neighbors, *k*, is sufficiently large, the *CTS* for each user is not changed by any further increases in the number of nearest neighbors.

4.2.3. Comparisons with other methods

To experimentally compare the performance of our algorithm with those of user-based CF and item-based CF, we selectively varied the number of recommended items *N* from 10 to 50 in an increment of 10. According to the results of the prior experiment in Section 4.2.1, the neighborhood size of *UserCF* and *ItemCF* was set to 60 and 70, respectively. For *TagCF*, considering both trends of the prior experiments we selected 50 as the neighborhood size and model size ($k = 50$, $w = 50$).

Table 2 includes three methods as the value of *N* increases. As we can see from these experiments, overall *recall* for all three algorithms was improved according to the increment of *N*. However, due to the large number of the collected item sets from *del.icio.us*, all three methods did not perform well. These results were affected by the fact we were only looking for a small number of recommended items: 10, 20, 30, 40, and 50. Even though target users had an enormous amount of bookmarks, they were under 5% of the total items (5% of 17,390 items). In other words, the number of items that each target user had not yet bookmarked (i.e., items that the system should consider recommending to the target user) was more than a hundred thousand. Nevertheless, *TagCF* provided considerably improved performance on all occasions

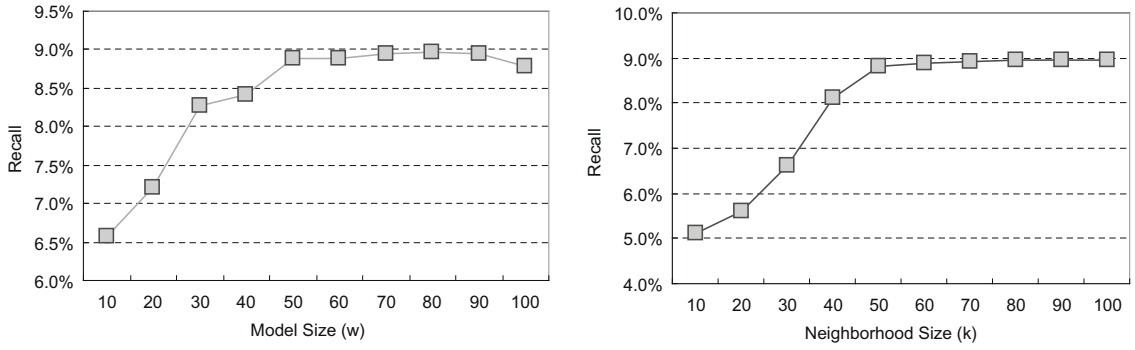

Fig. 9. Recall of collaborative tag-based CF according to the variation of model size (left) and neighborhood size (right).

Table 2
Comparisons of recall as the value of the number of recommended items *N* increases.

	10	20	30	40	50	Average
UserCF (%)	7.22	8.12	8.89	9.16	9.63	8.60
ItemCF (%)	8.71	9.33	9.98	10.72	11.21	9.99
TagCF (%)	8.82	9.90	10.32	10.86	10.97	10.17

compared to *UserCF*. In addition, comparing the results achieved by *TagCF* and *ItemCF*, *recall* of the former was superior to that of *ItemCF* as *N* was increased from 10 to 40, although the performance was diminished slightly in the case that *N* was 50. That is, when a relatively small number of items were recommended, the proposed method provided more suitable items to be at a higher rank in the returned item set, $TopN_u$. So our algorithm can provide better items for a target user than the other algorithms.

After demonstrating the effectiveness of our approach in recommending proper items to a target user, we carried out a further experiment to examine recommendation performance for cold-start users. For the experiment, we considered different subsets of users who had few bookmarks in the training dataset (e.g., users who have less than 3 bookmarks, less than 6 bookmarks, and so forth). And, we set the number of recommended items *N* to 10 for each user. Table 3 summarizes the result of the experiment, showing how *TagCF* outperformed the other methods.

Let us now concentrate on cold-start user groups. Cold-start users are defined as users who have bookmarked less than 5 items (i.e., users who have tagged less than 5 items) in a training dataset (as shown in the 2nd and 3rd column of the table). As our dataset analysis has shown, the cold-start users were also not active to annotate tags when he bookmarked. In the dataset, the average number of different tags used by 409 users who had less than 3 bookmarks was 2.81, whereas those of different tags used by 710 users who had less than 6 bookmarks was 4.79.

As we can see from Table 3, the result demonstrated that recommendation performances were considerably low compared to those for the common group of users (no constraint). Such results were caused by the fact that it was hard to analyze the users' propensity to bookmark Web pages because they did not have enough information (bookmarks or tags). Comparing *recalls* achieved by *UserCF* and *ItemCF*, interesting results were observed. The recommendation quality of *UserCF* is superior to that of *ItemCF* in the case of cold-start users, even though its performances were worse than those of *ItemCF* in the previous experiments. These results may be affected by the fact *ItemCF* basically tries to capture whether the target user had bookmarked the similar items or not. In the case of the cold-start target user, the similar items to a certain item were not almost bookmarked by him, and thus, it was more difficult to predict the preference of the item for him. In the cases of *UserCF*, cold-start users did not share a sufficient number of items selected in common with other users. Similar trends in common tags were observed for *TagCF* as well. Accordingly, available data was not sufficient and reliable for identifying similar users.

Although the recommendation performances were not satisfied, notably *TagCF* provided better quality than the other two CF methods in the event of cold-start users. For example, *TagCF* obtained *recall* of 1.21% for users who had less than 3 bookmarks, whereas *ItemCF* and *UserCF* obtained *recall* of 0.25% and 0.72%, respectively (the column labeled '<3' in Table 3). Similar improvement was obtained for the other user group (the column labeled '<6' in Table 3). This result indicates that the tags can be more helpful to grasp users' preference than only a few bookmarks by alleviating the sparsity of data.

Table 3
A comparison of recall for users according to the number of bookmarks.

Bookmarks	<3	<6	<8	<10	No constraint
Users	409	710	823	934	1544
Avg. tags	2.81	4.79	7.90	9.73	52.37
UserCF (%)	0.72	1.49	2.09	2.38	7.22
ItemCF (%)	0.25	0.71	1.14	1.41	8.71
TagCF (%)	1.21	2.23	2.41	2.42	8.82

We conclude from these comparison experiments that the collaborative tag-based CF method provides better quality than other methods in the event of both sparse data and cold-start users. In addition, we believe that our method can also improve the recommendation quality in the event of a cold-start item which is another notable challenge in recommender systems.

5. Conclusions and future work

As a part of Web 2.0, collaborative tagging is becoming widely used as an important tool to classify dynamic content for searching and sharing. In this paper, we analyzed the potential of collaborative tagging systems, including personalized and biased user preference analysis, and specific and dynamic classification of content for applying the recommendations. We also presented a new and unique recommendation algorithm via collaborative tags of users to provide enhanced recommendation quality and to overcome some of the limitations in CF systems. The experimental results demonstrated that the proposed algorithm offers significant advantages both in terms of improving the recommendation quality for sparse data and in dealing with cold-start users as compared to traditional CF algorithms. Moreover, we also observed that our method can provide more suitable items for user preference even though the number of recommended items is small.

There are several interesting research issues to address in order to successfully apply our algorithm to a practical environment. The empirical result showed that "noise" tags that have a bad influence on analyzing user preference can be included in the *CTS*. Such tags, due to the characteristics of the tag, personalized and content-criticizable (e.g., *bad*, *my work*, and *to read*), should be treated effectively for more valuable and personalized analyses. In addition, there remain common issues that have been mentioned in keyword-based analysis: polysemy, synonymy, and basic level variation (Golder and Huberman 2006). Semantic tagging is one of the interesting issues that we plan to consider for addressing these problems in the future.

Acknowledgement

This work was supported by INHA University Research Grand.

References

Breese, J. S., Heckerman, D. and Kadie, C. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*, 1998, 43–52.

Deshpande, M., and Karypis, G. Item-based Top-N recommendation algorithms. *ACM Transactions on Information Systems (TOIS)*, 22, 1, 2004, 143–177.

Golder, S. A., and Huberman, B. A. Usage patterns of collaborative tagging systems. *Journal of Information Science*, 32, 2, 2006, 198–208.

Herlocker, J. L., Konstan, J. A. and Riedl, J. Explaining collaborative filtering recommendations. In *Proceedings of the 2000 ACM Conference on Computer Supported Cooperative Work*, 2000, 241–250.

Herlocker, J. L., Konstan, J. A., Terveen, L. G., and Riedl, J. T. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems*, 22, 1, 2004, 5–53.

Hofmann, T. Latent semantic models for collaborative filtering. *ACM Transactions on Information Systems*, 22, 1, 2004, 89–115.

Linden, G., Smith, B., and York, J. Amazon.com recommendations: Item-to-Item collaborative filtering. *IEEE Internet Computing*, 7, 1, 2003, 76–80.

Marlow, C., Naaman, M., Boyd, D. and Davis, M. HT06, Tagging Paper, Taxonomy, Flickr, Academic Article, To Read. In *Proceedings of the17th Conference on Hypertext and Hypermedia*, 2006, 31–40.

Massa, P. and Avesani, P. Trust-aware collaborative filtering for recommender systems. In *Proceedings of International Conference on Cooperative Information Systems*, 2004, 492–508.

McCallum, A. and Nigam, K. A comparison of event models for na Bayes text classification. In *Proceedings of AAAI-98 Workshop on Learning for Text Categorization*, 1998.

Miller, B. N., Konstan, J. A., and Riedl, J. PocketLens: towards a personal recommender system. *ACM Transactions on Information Systems*, 22, 3, 2004, 437–476.

Papagelis, M., Plexousakis, D. and Kutsuras, T. Alleviation the sparsity problem of collaborative filtering using trust inferences. In *Proceedings of the Third International Conference on Trust Management*, 2005, 224–239.

Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P. and Riedl, J. GroupLens: an open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work*, 1994, 175–186.

Sarwar, B., Karypis, G., Konstan, J. and Riedl, J. Analysis of recommendation algorithms for E-commerce. In *Proceedings of the Second ACM Conference on Electronic Commerce*, 2000, 158–167.

Sarwar, B., Karypis, G., Konstan, J. and Riedl, J. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th International Conference on World Wide Web*, 2001, 285–295.

Schein, A. I., Popescul, A. and Ungar, L. H. Methods and metrics for cold-start recommendations. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2002, 253–260.

Shardanand, U. and Maes, P. Social information filtering: algorithms for automating ‘‘word of mouth”. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 1995, 210–217.

Wal, T. V. *Explaining and Showing Broad and Narrow Folksonomies*, 2005. <http://www.personalinfocloud.com/2005/02/explaining_and_.html> (last accessed 11.05.2009).