# Collaborative Filtering by Analyzing Dynamic User Interests Modeled by Taxonomy

Makoto Nakatsuji[1], Yasuhiro Fujiwara[2],
Toshio Uchiyama[1], and Hiroyuki Toda[1]

NTT Service Evolution Laboratories, NTT Corporation[1],
NTT Software Innovation Center, NTT Corporation[2],
1-1 Hikari-no-oka, Yokosuka-Shi, Kanagawa, 239-0847 Japan
{nakatsuji.makoto, fujiwara.yasuhiro,
uchiyama.toshio, and toda.hiroyuki}@lab.ntt.co.jp

**Abstract.** Tracking user interests over time is important for making accurate recommendations. However, the widely-used time-decay-based approach worsens the sparsity problem because it deemphasizes old item transactions. We introduce two ideas to solve the sparsity problem. First, we divide the users' transactions into epochs i.e. time periods, and identify epochs that are dominated by interests similar to the current interests of the active user. Thus, it can eliminate dissimilar transactions while making use of similar transactions that exist in prior epochs. Second, we use a taxonomy of items to model user item transactions in each epoch. This well captures the interests of users in each epoch even if there are few transactions. It suits the situations in which the items transacted by users dynamically change over time; the semantics behind classes do not change so often while individual items often appear and disappear. Fortunately, many taxonomies are now available on the web because of the spread of the Linked Open Data vision. We can now use those to understand dynamic user interests semantically. We evaluate our method using a dataset, a music listening history, extracted from users' tweets and one containing a restaurant visit history gathered from a gourmet guide site. The results show that our method predicts user interests much more accurately than the previous time-decay-based method.

## 1 Introduction

User interests can switch rapidly. For example, some one who listens to "avant garde" music may switch to "relax" music according to his/her mood at that time. The user's transaction history may thus contain several dissimilar strings, each of which is identified by the similarity of contiguous item selections. This creates a significant problem in collaborative filtering.

Accurately identifying and handling changes in user interests over time is an active research challenge in recommender systems, and is the target of many studies [6, 8, 14, 15, 22, 24, 29, 31]. One major research approach is to use a time decay function that decreases the item weight with the item's age [6,14,15]. Time decay methods assume that the recent item transactions of the active user, the one who is to receive the recommendation, reflect his/her future interests more than old transactions. Thus, they gradually decay the influence of old data.

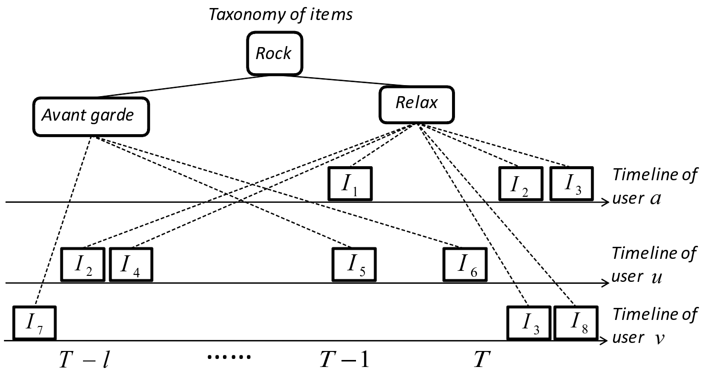

**Fig. 1.** Transaction timelines of users. $T$ is the current epoch. Boxes indicate artists (items) and the subscript identifies the artist; the dotted lines indicate music classes according to the music taxonomy. For example, in the current epoch, user $a$ listened to songs of artist $I_2$ and artist $I_3$ (both in class "Relax").

Previous time-decay-based methods, however, worsen the sparsity problem in collaborative filtering, which is well known to produce low recommendation accuracy when the population of the dataset used to measure the similarity of users is not sufficient [27]. The transaction timing of items is usually different for each user and for each item as described in the diffusion of innovations [26]. We consider that the sparsity problem occurs due to such *time offsets* against the item transactions of users in the real world. We illustrate the problem using Figure 1. Previous time-decay-based methods compute the interests of user $v$ as being similar with those of active user $a$ because they transact the same item, $I_3$, in the same current epoch $T$. Previous methods also indicate that the interests of user $u$ are dissimilar to those of user $a$ because they share no items in recent epochs. This implies that users are seen as similar only if they transact the same items in recent epochs. However, there are few users who have transacted the same items with user $a$ in recent epochs (like user $v$), thus the time-decay-based methods suffer badly from the sparsity problem.

This paper proposes a novel method that overcomes the sparsity problem; it avoids the problems that occur when using temporal information of item transactions to improve recommendation accuracy. Our method has two ideas. First, it extracts, in a per epoch manner, transactions that are similar to the transactions of the current epoch of the active user from all transactions of other users regardless of age. This has the effect of eliminating dissimilar transactions in epochs while it can make use of similar transactions regardless of age. For example, in Figure 1, it can eliminate the transactions made by user $u$ in epoch $T$ while it can make use of those made by $u$ in epoch $T-l$ because they transact the same item, $I_2$. Second, it models transactions of items based on a taxonomy of items, which is sometimes called the "simple ontology" [17]; it is a collection of human-defined classes usually with hierarchical structure. Our method makes use of the class structure in the taxonomy and thus can measure the similarity of transactions in epochs from both transacted items and their classes. As described in [18], this avoids the sparsity problem since more data is available for similarity matching. Moreover, it suits the computation of recommendations under the situation that the items transacted by users dynamically change over time. This is because the semantic meaning behind classes does not change so often while items transacted by users dynamically change over time. For example, active user $a$ recently transacted item $I_2$ that was selected by user $u$ who also selected item $I_4$ in the same epoch as $I_2$ (that is, epoch $T$–$l$), and $I_4$ shares the class of "Relax" with $I_2$. Thus, our method measures transactions by user $a$ in epoch $T$ and those by user $u$ in epoch $T-l$ as similar because those transactions are dominated by class "Relax" in addition to the fact that those transactions include item $I_2$. The semantic meaning behind class "Relax" does not change over time, so our method can identify $I_4$, which is missing in epoch $T$, as of potential interest to user $a$. Those two ideas enable our method to overcome the sparsity problem while suppressing noisy transactions, and thus achieve high accuracy.

Taxonomies provide another attractive effect. They enrich semantics behind the recommendations. The users can understand the recommended items as belonging to the same classes as the items that the user has transacted recently. Those taxonomies are becoming available on the Web due to the spread of the Linked Open Data (LOD) vision [1]. For example, Freebase[1] and DBPedia [2] have detailed taxonomies against several domains such as music, movie, and books. As an example, music genre "Electronic dance music" in FreeBase is identified by the unique resource identifier (URI)[2] and is available in RDF format. By referring to this URI, the computer can acquire the information that "electronic_dance_music" has "electronic_music" as parent_genre, "house_music" as one of its subgenres, and "pizzicato_five" as one of its artists as well as having the owl:sameAs relationship with "Electronic_Dance_Music" in DBPedia. Why don't we use those taxonomies for semantically analyzing dynamic user interests in the era of the "Web of Data"?

To the best of our knowledge, this is the first study that employs a taxonomy of items and uses discrete time periods to isolate changes in interest over time. We consider that analyzing dynamic user interests over taxonomies (or simple ontologies) is very important since taxonomy is a core Semantic Web technology. Our idea is simple but provides accurate recommendations. It provides a new theoretical alternative to collaborative filtering techniques that use temporal information in making recommendations.

We applied our ideas to the widely used service of neighborhood-based collaborative filtering. We evaluated our method using the following two datasets: (1) a dataset of music listening history extracted from users' tweets at twitter[3] with a taxonomy created from last.fm[4] tags[5] and (2) one containing restaurant

[1] http://www.freebase.com
[2] http://rdf.freebase.com/rdf/en/electronic_dance_music
[3] http://twitter.com/
[4] http://www.last.fm/
[5] The music taxonomy can be acquired, only for academic use, by mailing the authors.

visit histories gleaned from a popular Japanese gourmet guide site, Tabelog[6] with an expert created taxonomy. Those taxonomies have the same structures as the LOD dataset in Freebase as explained above. The results show that our method outperforms previous time-decay-based methods. They also indicate that our taxonomy-based method is superior to the typical topic model, LDA (Latent Dirichlet Allocation) method [3], which estimates user interests from data-driven topics, and is also successful in overcoming the sparsity problem of collaborative filtering. This is because topics estimated from users' item transactions dramatically change over time whereas the semantics behind human-defined classes do not change so often.

The paper is organized as follows: we describe related works in the next section. Section 3 describes the background of this paper. Section 4 explains our method in detail; how to measure the similarity of current transactions of the active user and the epoch-based transactions of another user by using a taxonomy of items. Section 5 evaluates our method in detail. Finally, Section 6 concludes the paper.

## 2 Related work

Recently, several works have attempted to integrate temporal dynamics into collaborative filtering methods [6, 8, 14, 15, 22, 24, 29, 31]. One of the major research approaches in this field uses a time decay function and computes the time weights for different items by decreasing the weights according to data age [6, 14, 15]. Recently, Liu et al. proposed an incremental algorithm for updating neighborhood similarities given new data [15]. While our method is not a time-decay-based method, it can be combined with those to catch the trends in item transactions. To this end, it is necessary to apply time decay functions to item transactions according to their transacted epochs, after extracting transactions similar to those in the current epoch of the active user.

Other research studies use state-based models. Markov chain models have been widely applied to the next-page prediction problem [24, 31]. Topic Tracking Model [8], which extends LDA [3], uses state space models on the natural parameters of the multinominal distributions that represent the topics. Recently, [24] introduced a novel personalized Markov chain method that models a transition cube, where each slice is a user-specific transition matrix of an underlying Markov chain on the users' basket history. They introduce a factorization model that gives a low-rank approximation of the transition cube. The quality of the final transition graph, and thus the accuracy of item prediction, is much improved since the influence of transitions of similar users, similar items, and similar transitions is considered. However, those factorization models can not provide semantics under the recommendation results such as the semantic relationships between recommended items and items that the user has transacted recently. [14] proposed a method that applies a time-decay method to a factorization model with complicated parameter learning from explicit rating datasets. It learns parameters for the biases of each user and each item, both of which

[6] http://tabelog.com/

change over time. However, generalizing factorization models to handle implicit feedback data only achieves a slight improvement [7]. Thus, it is not well suited for the implicit ratings used in our evaluation.

There are several alternatives that use temporal information in making recommendations. [22] proposed a preceding mining model that exploits the mined precedence information to recommend the top-k choices that could follow the past choices of a particular user. It models the user's history as a set of items that occurred in the past instead of a strict sequence of items, and predicts the set of items most likely to follow in no particular order. [29] proposed a graph model that captures users' long-term and short-term interests over time. It balances the impacts of long-term and short-term interests for accurate recommendations.

We note that taxonomies are becoming available on the Web in the format of LOD such as those published by DBPedia [2] and Freebase. Though most of the data published in the format of LOD is instance data, there are projects that link and build taxonomies by merging the data in several domains by using already published taxonomies like those in Wikipedia[7] [9, 23]. Such merged taxonomies enable us analyze user transactions distributed in multiple service domains comprehensively. Thus, recommendation methods that use taxonomies to understand user interests semantically are becoming more important [4, 18–21, 28, 30]. For example, [18] measures the similarity of users based on items rated by users as well as the classes that include those items. Thus, it accurately predicts user interests even when the rating dataset is sparse. They recently proposed a method that analyzes user interests more in detail by linking multiple taxonomies [19]. However, they fail to handle temporal information against item transactions.

This paper differs from the methods that apply a time-decay function against users' item transactions, methods that use state-based models, and techniques that focus on the sequence of item transactions. Our method employs the taxonomy of items and extracts transactions that are similar to the transactions of the current epoch of the active user. Our ideas give a new vision of collaborative filtering by better utilizing the temporal information of item transactions.

## 3 Background

Collaborative filtering methods can be classified into two approaches: memory-based (or neighborhood-based) collaborative filtering [6, 15, 25] and model-based collaborative filtering [8, 24, 31]. Previous time-weighted collaborative filtering methods [6,15] are examples of memory-based collaborative filtering. Our method, however, can also be applied to model-based collaborative filtering[8].

Traditional memory-based collaborative filtering methods assume that each user belongs to a larger group of users with similar behavior [25]. In computing user similarity, they often use cosine similarity.

[7] http://en.wikipedia.org

[8] For example, we can create an order-three tensor from transactions per epoch by users, items, and their classes. By applying tensor decomposition [12], we can compute recommendations against the current transactions of the active user.

**Table 1.** Definition of main symbols.

| Symbols | Definitions |
|---|---|
| $t$ | An epoch |
| $T$ | A current epoch |
| $\mathcal{C}$ | A class set in the taxonomy |
| $I_j$ | An item |
| $C_j$ | A class in the class set $\mathcal{C}$ |
| $\mathbf{i}^t_u$ | A vector of item transactions of user $u$ in epoch $t$ |
| $\mathbf{c}^t_u$ | A vector of class transactions of user $u$ in epoch $t$ |

If $\mathcal{I}$ is the set of items transacted by users $a$ and $u$, and $i_{u,j}$ is the transaction frequency of user $u$ for item $I_j$, the similarity between active user $a$ and user $u$, $S(a,u)$, is determined as follows:

$$S(a,u) = \frac{\sum_{I_j \in \mathcal{I}} (i_{a,j} \cdot i_{u,j})}{\sqrt{\sum_{I_j \in \mathcal{I}} (i^2_{a,j})}\sqrt{\sum_{j \in \mathcal{I}} (i^2_{u,j})}}. \tag{1}$$

If $\mathcal{N}$ is the set of users that are most similar to user $u$, the predicted value of user $a$ on item $I_j$, $p_{a,j}$, is given by the following equation:

$$p_{a,j} = \frac{\sum_{u \in \mathcal{N}} (i_{u,j} \cdot S(a,u))}{\sum_{u \in \mathcal{N}} S(a,u)}. \tag{2}$$

This equation implies that the methods recommend items based on user similarities. Therefore, the effective assessment of user similarities is important in improving recommendation accuracy. Our method, explained in the next section, extracts transactions that are similar to the transactions of the current epoch of the active user. Thus it can produce more accurate recommendations than the ordinary cosine based method as shown in the evaluation section.

## 4 Method

We first explain our model of item transactions in an epoch according to a taxonomy of items. Next, we show how to extract transactions that are similar to the transactions in the current epoch of the active user. We then introduce recommendation computation by analyzing extracted transactions.

### 4.1 Modeling transactions of a user in an epoch

We explain how to model transactions by a user in an epoch. Please refer to the symbol definitions of Table 1.

We assume that epoch $t$ is a discrete variable, a time period, and we can set the time period for an epoch arbitrarily at, for example, one day or one week as [8] did. An epoch can overlap adjacent epochs by offsetting the starting time of epochs. This is useful because we try to analyze the change in user interests in detail. We also denote $T$ as the current epoch.

Our model is based on two observations. First is that a user who transacts an item within an epoch, tends to like items of the same class in that epoch. In other words, users tend to be interested in the same types of items in the same period of time. For example, users who are interested in Opera music item in a certain epoch, tend to transact several Opera items in the same epoch. Second is that the semantics behind the classes do not change so often while the items transacted by users appear and disappear in each epoch as influenced by the trends in the epoch. For example, the class "Metal Rock" is used with almost the same meaning for a long period while particular artists in "Metal Rock" often appear and disappear. Users who like "Metal Rock" tend to like items that were classified into "Metal Rock" even if those items were transacted by other users in different epochs. Taxonomy-based modeling is thus useful in analyzing temporal user interests because it lets our method measure the similarity of transactions in different epochs by using classes. Thus, we propose to use a taxonomy of items to model a user's temporal interests from his/her item transactions in epochs. As a result, our method can identify similar transactions in epochs by using both items and their classes as described in Section 4.2. Thus it can overcome the sparsity problem that occurs when there are few item transactions in each epoch even if the items transacted by users dynamically change over time.

Formally, our method models the temporal interests of a user by using two vectors; a vector of item transactions of user $u$ in epoch $t$, $\mathbf{i^t_u}$, and a vector of class transactions of user $u$ in epoch $t$, $\mathbf{c^t_u}$. The $j$-th element of vector $\mathbf{i^t_u}$, $i^t_{u,j}$, represents the frequency of transactions of item $I_j$ in epoch $t$. The $k$-th element of vector $\mathbf{c^t_u}$, $c^t_{u,k}$, represents the frequency of transactions of class $C_k$ in epoch $t$. $c^t_{u,k}$ is computed by the following equation:

$$c^t_{u,k} = \sum_{I_j \in f(k)} i^t_{u,j}. \tag{3}$$

Here, function $f(k)$ returns an item set whose members belong to descendant classes of class $C_k$ and have unique names (no names are shared)[9]. Equation (3) reflects the transaction frequencies of items in their ascendant classes. Thus, we can measure the similarity of transactions of users in epochs from classes as well as from items, which overcomes the sparsity problem.

## 4.2 Measuring similarities of transactions in epochs

We explain here how to compute the similarity between the transactions in current epoch $T$ of active user $a$ and those in epoch $t$ of user $u$. To avoid the sparsity problem, we measure the similarity of transactions in epochs using both transacted items and their classes. We also apply set theory [11] to analyze the class structure, which is composed by class/sub-class relationships and class/item relationships in the taxonomy, to assess user similarities in detail. We first give the notations for the algorithm and then explain it in detail.

[9] Some items in the taxonomy may have the same name, however, they should be identified by URIs. For example, some artists may be placed into several classes, however, each artist in a different class should be identified by a URI, which is assigned to that artist regardless of its name.

**Algorithm 1** Measuring transaction similarity in the current epoch of user $a$ and transactions in all epochs of user $u$.

**Input:** Transaction vectors of user $a$ in epoch $T$, ($\mathbf{i_a^T}$ and $\mathbf{c_a^T}$) and transaction vectors in epoch $t$ of user $u$, ($\mathbf{i_u^t}$: $0 \leq t \leq T$) and ($\mathbf{c_u^t}$: $0 \leq t \leq T$)).
**Output:** Similarity score between transactions in the current epoch of user $a$ and those in epoch $t$ of user $u$.

```
1: for each epoch ($t : 0 \leq t \leq T$) do
2:   for each user $u$ do
3:     for each class $C_j$ in $\mathcal{C}$ do
4:       for each sub-class $C_k$ in $\{\mathcal{C}_{j,T}(a) \cup \mathcal{C}_{j,t}(u)\}$ do
5:         compute $S(a, u, C_k, t)$;
6:       end for
7:       for each item $I_k$ in $\{\mathcal{I}_{j,T}(a) \cup \mathcal{I}_{j,t}(u)\}$ do
8:         compute $S(a, u, I_k, t)$;
9:       end for
10:     end for
11:     Compute similarity score $S_C(a, u, t)$;
12:     Compute similarity score $S_I(a, u, t)$;
13:   end for
14:   for each user $u$ do
15:     Normalize $S_C(a, u, t)$ and $S_I(a, u, t)$;
16:     Compute $S(a, u, t)$ as $S'_C(a, u, t) + S'_I(a, u, t)$;
17:   end for
18: end for
```

**Notation** Our algorithm applies set theory to assess the similarity of users' interests in epochs according to the class structure of the taxonomy, which is composed by class/sub-class and class/item relationships in the taxonomy. Thus, we introduce notations to represent those relationships. We denote $\mathcal{C}_{j,t}(u)$ as a sub-class set whose members belong to class $C_j$ and that have been transacted by user $u$ in epoch $t$. Thus, $\{\mathcal{C}_{j,T}(a) \cup \mathcal{C}_{j,t}(u)\}$ and $\{\mathcal{C}_{j,T}(a) \cap \mathcal{C}_{j,t}(u)\}$ are a union set and an intersection set of sub-classes of class $C_j$ transacted by user $a$ in epoch $T$ and those by user $u$ in epoch $t$, respectively. We also denote $\mathcal{I}_{j,t}(u)$ as an item set whose items belong to class $C_j$ and that have been transacted by user $u$ in epoch $t$. Thus, $\{\mathcal{I}_{j,T}(a) \cup \mathcal{I}_{j,t}(u)\}$ and $\{\mathcal{I}_{j,T}(a) \cap \mathcal{I}_{j,t}(u)\}$ are a union set and an intersection set of items in class $C_j$ transacted by user $a$ in epoch $T$ and those by $u$ in epoch $t$, respectively.

**Algorithm** Our method measures the similarity of interests in epochs using both transacted items and their classes. Please also see Algorithm 1. The algorithm proceeds in the following steps:

1. Our method measures the similarity between the transactions in the current epoch of the active user and those in each epoch of the other users. Thus, it repeats steps from 2 to 6 by changing time $t$ from 0 to $T$ (line 1) and by setting target user $u$ as all users (line 2).
2. It also picks up class $C_j$ among the class set $\mathcal{C}$ in the taxonomy (line 3).
3. For each sub-class $C_k$ in class $C_j$, our method computes the similarity between interests of user $a$ in current epoch $T$ and those of user $u$ in epoch $t$. We assume that if users transact the same amount of transactions against a class in an epoch, they have similar interests to the class. Thus, the similarity

of users' interests against sub-class $C_k$ in epochs is not high if the transactions of user $u$ have too many or too few transactions against sub-class $C_k$ in epoch $t$ compared with transactions against $C_k$ by user $a$ in epoch $T$. Thus, we design this similarity, denoted as $S(a, u, C_k, t)$, to filter such transaction noise and take the smaller of the class transaction frequencies between users in epochs as follows (line 4-6):

$$S(a, u, C_k, t) = min(c^T_{a,k}, c^t_{u,k}). \tag{4}$$

4. As is the case with $S(a, u, C_k, t)$, for each item $I_k$, our method computes the similarity between the interests in epoch $T$ of user $a$ and the interests in epoch $t$ of user $u$. This similarity, denoted as $S(a, u, I_k, t)$, is formally computed as follows (line 7-9):

$$S(a, u, I_k, t) = min(i^T_{a,k}, i^t_{u,k}). \tag{5}$$

5. Next, it computes the similarity between the class transactions in the current epoch $T$ of active user $a$ and the class transactions in epoch $t$ of user $u$. This similarity, $S_C(a, u, t)$, is computed as follows (line 11):

$$S_C(a,u,t) = \sum_{C_j \in \mathcal{C}} \frac{\sum_{C_k \in \{\mathcal{C}_{j,T}(a) \cap \mathcal{C}_{j,t}(u)\}} S(a, u, C_k, t)}{|\{\mathcal{C}_{j,T}(a) \cup \mathcal{C}_{j,t}(u)\}|}. \tag{6}$$

The numerator sums the similarity of users' interests in epochs against each sub-class $C_k$ in class $C_j$ computed in step 3. The denominator, which represents the number of members of a set $\{\mathcal{C}_{j,T}(a) \cup \mathcal{C}_{j,t}(u)\}$, lets our method measure the similarity of users in epochs against class $C_j$ considering the overlap between the sub-classes of a class $C_j$ transacted by user $a$ and those by user $u$. By picking up each class $C_j$ in $\mathcal{C}$ and by investigating the similarity of users' interests in each epoch against each class/sub-class relationship, we can make use of the structure of the class hierarchy for measuring the similarity. We consider that the denominator should be added when the taxonomy of items is not so detailed, that is, each of the members of $\mathcal{C}$ has many sub-classes, like our evaluation dataset of music listening history. This is because our method can compute the similarity of users' interests against a class by investigating the overlap rate of its sub-classes owned by users.

6. As is the case with $S_C(a,u,t)$, it computes the similarity between the current item transactions of active user $a$ and the item transactions in epoch $t$ of user $u$. This similarity, $S_I(a, u, t)$, is computed as follows (line 12):

$$S_I(a,u,t) = \sum_{C_j \in \mathcal{C}} \frac{\sum_{I_k \in \{\mathcal{I}_{j,T}(a) \cap \mathcal{I}_{j,t}(u)\}} S(a, u, I_k, t)}{|\{\mathcal{I}_{j,T}(a) \cup \mathcal{I}_{j,t}(u)\}|}. \tag{7}$$

We can also use cosine similarity, see Equation (1) in computing the similarity between the current item transactions of active user $a$ and the item transactions in epoch $t$ of user $u$. In our evaluation, we compared those similarity measurements against item transactions.

7. Our method normalizes the similarity against classes and that against items. Thus, the variance and the average of similarity scores among all epochs of users equal one and zero, respectively. We denote those normalized similarities as $S_C^{'}(a,u,t)$ and $S_I^{'}(a,u,t)$. Next, our method computes the similarity between the transactions in current epoch $T$ of user $a$ and those in epoch $t$ of user $u$. This similarity, $S(a,u,t)$, is computed as follows (line 14-17):

$$S(a,u,t) = S_C^{'}(a,u,t) + S_I^{'}(a,u,t). \tag{8}$$

Note that our algorithm is fast even though it includes several loops because typically there are few user transactions in each instance or each class in each epoch.

### 4.3 Computing recommendation

Our method uses the similarity values computed by Equation (8) to compute a prediction value against item $I_j$ for active user $a$. The predicted value of user $a$ on item $I_j$, $p_{a,j}$, is obtained by the following equation:

$$p_{a,j} = \frac{\sum_{u,t\in\mathcal{N}}(i^t_{u,j} \cdot S(a,u,t))}{\sum_{u,t\in\mathcal{N}} S(a,u,t)}, \tag{9}$$

where $\mathcal{N}$ is the set of transactions in epochs of users that are most similar to the current transactions of user $a$.

This equation is similar to Equation (2) used in traditional memory-based collaborative filtering. However, note that our method computes item prediction from the most similar transactions in epochs not from the most similar users. Similar users, computed by traditional memory-based collaborative filtering, may transact different types of items at different times. Our method can filter out such transaction noise for the active user, and so achieves high accuracy.

## 5 Evaluation

We conduct an evaluation to confirm the method's accuracy.

### 5.1 Datasets

Our evaluation used the following two datasets:

*Music listening history* We crawled users' tweets against music artists (items) from Twitter from 6th July to 20th September, 2011. To extract users' listening history from tweets, we first extracted artist names from the tweets submitted through last.fm twitter client[10]. Tweets submitted via the client have a fixed format allowing us extract music artist name without error. We also extracted tweets other than those submitted from the client to increase the number of music tweets. For this, we pulled the time-line of tweets of crawled users and checked if those tweets included both music artist name and hashtag "#nowlistening".

[10] http://tweetmlyast.fm/

Users' listening history has been used for evaluating recommendations [13]. By analyzing tweets, we can attach temporal information to the listening history.

The taxonomy of music artists was gathered from last.fm API according to the following procedure: (1) We first crawled the top tags (descending popularity) for each artist. We also crawled 26 genre tags as classes that were located at the top of the "music page" of last.fm to categorize artists. (2) We next crawled the top similar tags for each genre tag and classified those top similar tags as sub-classes of the genre class[11]. (3) We then classified an artist as an instance under sub-classes if their tags were the same as the tags crawled for the artist. For example, if "Beatles" has a tag "Classical rock" and the genre class "Rock" has a similar tag "Classical rock", we create sub-class "Classical rock" under class "Rock" and classify "Beatles" into "Classical rock" as an instance. (4) We resolved the ambiguity caused by tags with the same name. We checked whether an artist had ambiguous tags (i.e. were classified in several genres). If so, we checked the most popular tag for the artist whose name was the same as the genre tag. We also checked whether the ambiguous tag was one of the tags similar to that genre tag. If so, we classified these ambiguous tags into that genre class as sub-classes. The other ambiguous tags were eliminated. For example, if the artist "Beatles" had a tag "classic" but the most popular tag of this artist whose name was the same as the genre tag was "Rock" and "Rock" had "classic" as similar tag, we classified "Beatles" into sub-class "classic" under class "Rock". Finally, we permitted an artist to be classified into not more than three sub-classes in the taxonomy according to the tag popularity for the artist.

As a result, we could extract 62,527 tweets of 14,884 users against 6,886 artists (items) in creating the evaluation dataset. The taxonomy of artists has 1,223 classes and has three hierarchy levels; first hierarchy level is root class. The classes (tags) in the lowest hierarchy in the taxonomy are themselves concrete, however, the parent classes of those classes are not so detailed[12]. Thus, we need the denominator of Equation (6) as explained in the method section.

*Restaurant visiting history* We also used a restaurant visiting history gathered from the popular Japanese gourmet guide site, Tabelog with its expert-created taxonomy. This dataset was also used for the evaluation of recommendation methods [18,19]. Users submit reviews against restaurants that they liked, along with the date of dining.

We focused on restaurants in Tokyo, and extracted 63,885 reviews of 13,633 users against 44,321 restaurants (items) posted from 9th March to 20th June, 2010. The taxonomy of restaurants is quite deep; it has 318 genres as item classes, and has four or five hierarchy levels. For example, the end classes of this taxonomy have genres such as "Wine bar" and "Beer garden".

### 5.2 Compared methods

We compared our method to the following methods:

[11] Thus, the music taxonomy is really a taxonomy expanded with similar tags created by statistical analysis against tagging activities of last.fm users.

[12] The music taxonomy has 26 genre-classes. Each genre-class has, on average, about 46 sub-classes. We consider that each genre-class can be categorized in more detail.

- *Cosine*: This is the most commonly used memory-based collaborative filtering method; user similarity is based on cosine similarity. The prediction values of items are computed by using Equation (2).
- *Time*: This is the time-weighted collaborative filtering method [6] that uses a time decay function that computes the time weights for different items by decreasing the weights of old data. Time decay is determined by the exponential function $\mathrm{e}^{-(\lambda \cdot (T-t))}$. We selected this method because we consider that our method, which is implemented to realize simple neighborhood-based collaborative filtering should be compared with the simplest and closest methods.
- *Taxonomy*: this taxonomy-based method was proposed by [18]. It computes the similarity of users from the transactions of users against both items and their classes; it does not use time information.
- *LDA*: this method is based on LDA [3], which is a representative topic model, however, it does not use time information. Method *LDA* is model-based collaborative filtering. We selected this method for investigating the characteristics of human-defined classes and those of data-driven topics when applying the methods to the datasets that contain time information.
- *Without classes*: This is the proposed method with the taxonomy of items omitted [13]. This method was chosen for investigating the effect of using taxonomy of items in measuring similarity between transactions in epochs.

### 5.3 Methodology and parameter setup

We divided each dataset into a training dataset and a prediction dataset. The latter contains the data gathered over the last week and former contains the remainder. We then used the training dataset to measure the similarity of transactions in epochs. The starting time of epochs was shifted in units of one week (this means each adjacent epoch has overlap of two weeks) and the length of each epoch was varied from one week to three weeks. The results shown later are those achieved with the length of epoch set to three weeks because our method achieves the most accurate results at this setting. In concrete, the accuracy degrades if epoch length differs from 3 weeks. Basically, epochs $< 3$ weeks yield sparse transactions and epochs $> 3$ weeks yield transactions that include several different classes. We also confirmed that accuracy degrades if the overlap between epochs equals zero because it is difficult to catch the dynamic change of users' interests in detail. We next computed the prediction values of items for the active user by using Equation (9). We focused on users who had both item transactions in the current (last) epoch in the training dataset and those in the prediction dataset as active users. As a result, the music dataset and the restaurant dataset have 1,555 and 2,034 users to be evaluated, respectively.

We used Average Precision (AP) [16] to evaluate our method. If we let the number of ranked items be $k$, the number of correct answers among the top-j ranked items be $N_j$, and the number of all correct answers be $A$ (defined as items the user is interested in), AP is defined as $\frac{1}{A}\sum_{1\leq j\leq k}\frac{N_j}{j}$.

[13] This is equivalent to not summing the first term in Eq. (8) when computing $S(a,u,t)$.

We checked AP against the top-k ranked items. We set k to 5, 10, 20, 40, and 60 in our evaluation, and the corresponding results are denoted as AP@5, AP@10, AP@20, AP@40, and AP@60. The active user ordinarily checks the top ranked items in the list, thus typical recommender systems only use those items. We did not check the accuracy against lower ranked items because we consider that the highly ranked items are more important as did a previous work [10]. Some readers may consider that top-40 or top-60 items are too many for the active user. We, however, consider that, in some cases, the recommendation diversity, and hence user satisfaction, is improved by randomly showing about 5 of the 40 or more top ranked items in the recommendation [30].

We set the number of $\mathcal{N}$ (number of most similar *transactions per epoch* used in methods other than *LDA*) in both datasets to 20. We also set parameter $\lambda$ used in method *Time* to 0.2 for the music dataset and to 0.1 for the restaurant dataset. The number of topics in method *LDA* was set to 20 for both datasets. Those settings maximize the accuracy for each method.

### 5.4 Results

We evaluated the accuracy of our methods by changing the number of items recommended to the active user. AP@k results against the music dataset are shown in Table 2 and those against the restaurant dataset are shown in Table 3. Our methods (*Proposed* or *Without Classes*) offer better AP@k than the other methods in all cases other than AP@60 of *LDA* method against the restaurant dataset. Bold typeset indicates statistical significance at $p < 0.05$ (t-test was used) compared to *Time* and *LDA*. This indicates that *Proposed* well employs the taxonomy of items and collects the most similar transactions in epochs with the current transactions of the active user while eliminating noisy transactions. We also applied cosine similarity to compute the similarity of item transactions in Equation (5) and (7) in our methods and confirmed that it also has higher accuracy than *Time*. Due to the space limitation, we omit the results of this.

Interestingly, *Proposed* improves the accuracy of higher ranked items more in the restaurant dataset than in the music dataset since the restaurant taxonomy is more detailed than the music one. We also investigated the effect of procedure (4) in Section 5.1 on the recommendation accuracy. This procedure resolves the ambiguity caused by tags with the same name and thus avoids the classification mistakes when expanding the taxonomy with user-generated tags. As a result (we omit the result due to space limitation), we found that the accuracy of the method *Proposed* becomes much worse if we omit procedure (4). Thus, a detailed taxonomy increases the recommendation accuracy of our method.

Our methods also offer higher accuracy than the *Taxonomy* method. This is because *Taxonomy* simply reflects users' transaction frequency against items to their classes and does not consider the time-line of users' transactions. Thus, interests of a user are constructed by the many classes transacted over the entire time-line of the user. This approach is useful when recommending highly novel items that are located in classes that the active user has not yet transacted while maintaining high recommendation accuracy (in terms of Mean Absolute Error)

**Table 2.** Results (x$10^{-2}$) against music dataset.

| | AP@5 | AP@10 | AP@20 | AP@40 | AP@60 |
|---|---|---|---|---|---|
| *Cosine* | 1.10 | 1.35 | 1.59 | 1.85 | 2.00 |
| *Time* | 1.16 | 1.45 | 1.63 | 1.91 | 2.01 |
| *Taxonomy* | 0.92 | 1.10 | 1.52 | 1.74 | 1.87 |
| *LDA* | 0.31 | 0.39 | 0.48 | 0.54 | 0.58 |
| *Without classes* | *1.18* | *1.52* | *1.67* | 1.91 | 2.01 |
| *Proposed* | 1.14 | 1.48 | 1.66 | *1.99* | ***2.13*** |

**Table 3.** Results (x$10^{-3}$) against restaurant dataset.

| | AP@5 | AP@10 | AP@20 | AP@40 | AP@60 |
|---|---|---|---|---|---|
| *Cosine* | 1.32 | 1.44 | 1.53 | 1.69 | 1.77 |
| *Time* | 1.46 | 1.52 | 1.63 | 1.91 | 2.01 |
| *Taxonomy* | 1.06 | 1.19 | 1.27 | 1.38 | 1.46 |
| *LDA* | 1.60 | 1.69 | 1.82 | 2.03 | *2.14* |
| *Without classes* | 1.54 | 1.61 | 1.74 | 1.80 | 1.86 |
| *Proposed* | *1.63* | ***1.78*** | ***1.90*** | *2.08* | 2.12 |

as [18] described. On the other hand, *Proposed* focuses on epochs to improve accuracy. The effect is to suppress the mixing of classes that are not applicable to the current transactions of the active user, which improves accuracy.

*Proposed* also offers much higher accuracy than the *LDA* method against the music dataset, and it offers higher accuracy than the *LDA* method against the restaurant dataset method except for AP@60. Items transacted by users in the music dataset appear and disappear over short time cycles according to the trends in each epoch. On the other hand, restaurants visited by users do not change so often [14]. From those results, topics estimated by the *LDA* method are not useful when the item transactions are more dynamic. On the other hand, human-defined classes do not change so often, thus *Proposed* achieves higher accuracy in this situation. Variants of the LDA method have appeared that consider time information [8]. An LDA method that employs the taxonomy of words (items) has been proposed [5]. Thus, we consider that the combination of topics and taxonomies for estimating user interests is promising for future recommendation methods in dynamic situations.

We then investigated the effect of using a taxonomy of items on measuring the similarity between transactions in epochs. To this end, we compared *Proposed* and *Without classes*. For the music dataset, *Without classes* achieves the highest accuracy against AP@5, AP@10, and AP@20 because the music taxonomy is not so detailed, however, it can not achieve higher accuracy against AP@40 and AP@60 than *Proposed* due to the sparsity problem caused by lack of the taxonomy. Another interesting finding is that *Without classes* can not achieve higher accuracy against AP@40 and AP@60 than *Time*. This is because *Without classes* uses only current transactions of the active user to compute recommendations for the active user while *Time* uses all transactions of the active user.

[14] The high accuracy of the *LDA* method for the restaurant dataset has another reason. After investigation, we found that *LDA* could estimate topics based on restaurant location. Users tend to go dining in the same area even in different epochs.

Thus, *Without classes* suffers from the sparsity problem. On the other hand, *Proposed* achieves higher accuracy than *Time* in all cases though it also uses only current transactions of the active user. The above results confirm that our method, *Proposed*, avoids the sparsity problem and improves recommendation accuracy by using available taxonomies.

Note that our methods use only item transactions in the current epoch of the active user for comparison purposes. Considering this fact, the above results confirm that our two ideas have great potential for future recommendation techniques that use the temporal information of the users' item transactions; utilizing the transactions of the active user in older epochs can greatly enhance the recommendation accuracy.

Finally, we show examples of recommendations made by only our method, *Proposed*. It can recommend item "Sam Sparro" (in class "Electronic"/"Electronic pop") to the user who recently transacted items, "Gym Class Heroes" (in class "Pop"/"Dance") and "Blue Foundation" (in class "Electronic"/"Dance"). The number of recent transactions against "Sam Sparro" is not so many in the training dataset, however, our method can recommend it to the active user. This is because there are several transactions that have classes "Electronic" and "Pop"/"Dance", and item "Sam Sparro". On the other hand, *Time* tends to recommend items that are frequently transacted in the current epoch such as items "NE-YO" and "Blink 182", which were new releases at that time. *Taxonomy* tends to recommend items in the classes that are transacted frequently such as item "Maroon" in class "Alternative rock" and item "Nicki Minaj" in class "R&B", even if they are not in the same classes that the active user has transacted in current epoch. Those results decrease the accuracy of the methods.

## 6 Conclusion

This paper proposed a novel method that accurately predicts user interests by dividing the historical data into discrete time periods, epochs, and identifying those periods that best match the current transactions of the active user. Moreover, it uses the taxonomy-based approach to model transactions by users and so well identifies the similarity of transactions even if there are few transactions in each epoch. It computes recommendations for the active user by analyzing the extracted transactions. We evaluated our method using a music listening history and a restaurant visit history, and confirmed that our method predicts user interests much more accurately than the previous time-weighted collaborative filtering approach. We also confirmed that our method is superior to the typical topic model when applied to the situations in which items transacted by users dynamically change over time. The basic ideas that underlie in our method are quite simple but have great potential for future recommendation techniques that use the temporal information of the users' item transactions.

**Acknowledgements**

I dedicate this research study to my wife Kumiko Nakatsuji. She gave me sincere support and encouragement throughout this study.

## References

1. Bizer, C., Heath, T. and Berners-Lee, T.: Linked Data - The Story So Far, *International Journal on Semantic Web and Information Systems*, Vol. 5, No. 3, pp. 1–22 (2009).
2. Bizer, C., Lehmann, J., Kobilarov, G., Auer, S., Becker, C., Cyganiak, R. and Hellmann, S.: DBpedia - A crystallization point for the Web of Data, *Journal of Web Semantics*, Vol. 7, No. 3, pp. 154–165 (2009).
3. Blei, D. M., Ng, A. Y. and Jordan, M. I.: Latent Dirichlet Allocation, *Journal of Machine Learning Research*, Vol. 3, pp. 993–1022 (2003).
4. Cantador, I., Castells, P. and Bellogín, A.: An Enhanced Semantic Layer for Hybrid Recommender Systems: Application to News Recommendation, *Int. J. Semantic Web Inf. Syst.*, Vol. 7, No. 1, pp. 44–78 (2011).
5. Chemudugunta, C., Holloway, A., Smyth, P. and Steyvers, M.: Modeling Documents by Combining Semantic Concepts with Unsupervised Statistical Learning, *Proc ISWC'08*, pp. 229–244 (2008).
6. Ding, Y. and Li, X.: Time weight collaborative filtering, *Proc. CIKM'05*, pp. 485–492 (2005).
7. Hu, Y., Koren, Y. and Volinsky, C.: Collaborative Filtering for Implicit Feedback Datasets, *Proc. ICDM'08*, pp. 263–272 (2008).
8. Iwata, T., Watanabe, S., Yamada, T. and Ueda, N.: Topic tracking model for analyzing consumer purchase behavior, *Proc. IJCAI'09*, pp. 1427–1432 (2009).
9. Jain, P., Hitzler, P., Sheth, A. P., Verma, K. and Yeh, P. Z.: Ontology Alignment for Linked Open Data, *Proc. ISWC'10*, pp. 402–417 (2010).
10. Jamali, M. and Ester, M.: Using a trust network to improve top-N recommendation, *Proc. RecSys'09*, pp. 181–188 (2009).
11. Jech, T.: *Set Theory: The Third Millennium Edition, Revised and Expanded*, Springer Monographs in Mathematics, Springer-Verlag (2003).
12. Kolda, T. G. and Bader, B. W.: Tensor Decompositions and Applications, *SIAM Rev.*, Vol. 51, No. 3, pp. 455–500 (2009).
13. Konstas, I., Stathopoulos, V. and Jose, J. M.: On social networks and collaborative recommendation, *Proc. SIGIR'09*, pp. 195–202 (2009).
14. Koren, Y.: Collaborative filtering with temporal dynamics, *Proc. KDD'09*, pp. 447–456 (2009).
15. Liu, N. N., Zhao, M., Xiang, E. and Yang, Q.: Online evolutionary collaborative filtering, *Proc. RecSys'10*, pp. 95–102 (2010).
16. Manning, C. D., Raghavan, P. and Schtze, H.: *Introduction to Information Retrieval*, Cambridge University Press (2008).
17. McGuinness, D. L.: Ontologies Come of Age., *Spinning the Semantic Web*, MIT Press, pp. 171–194 (2003).
18. Nakatsuji, M., Fujiwara, Y., Tanaka, A., Uchiyama, T., Fujimura, K. and Ishida, T.: Classical music for rock fans?: novel recommendations for expanding user interests, *Proc. CIKM'10*, pp. 949–958 (2010).
19. Nakatsuji, M., Fujiwara, Y., Uchiyama, T. and Fujimura, K.: User Similarity from Linked Taxonomies: Subjective Assessments of Items, *Proc. IJCAI'11*, pp. 2305–2311 (2011).
20. Nakatsuji, M., Miyoshi, Y. and Otsuka, Y.: Innovation Detection Based on User-Interest Ontology of Blog Community., *Proc. ISWC'06*, pp. 515–528 (2006).
21. Nakatsuji, M., Yoshida, M. and Ishida, T.: Detecting innovative topics based on user-interest ontology, *Journal of Web Semantics*, Vol. 7, No. 2, pp. 107–120 (2009).
22. Parameswaran, A. G., Koutrika, G., Bercovitz, B. and Garcia-Molina, H.: Recsplorer: recommendation algorithms based on precedence mining, *Proc. SIGMOD'10*, pp. 87–98 (2010).
23. Parundekar, R., Knoblock, C. A. and Ambite, J. L.: Linking and Building Ontologies of Linked Data, *Proc. ISWC'10*, pp. 598–614 (2010).
24. Rendle, S., Freudenthaler, C. and Schmidt-Thieme, L.: Factorizing personalized Markov chains for next-basket recommendation, *Proc. WWW'10*, pp. 811–820 (2010).
25. Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P. and Riedl, J.: GroupLens: An Open Architecture for Collaborative Filtering of Netnews, *Proc. CSCW'94*, pp. 175–186 (1994).
26. Rogers, E. M.: *Diffusion of innovations*, Free Press, 5th edition (2003).
27. Sarwar, B. M., Karypis, G., Konstan, J. A. and Riedl, J.: Analysis of recommendation algorithms for e–commerce, *Proc. EC'00*, pp. 158–167 (2000).
28. Szomszor, M., Alani, H., Cantador, I., O'Hara, K. and Shadbolt, N.: Semantic Modelling of User Interests Based on Cross-Folksonomy Analysis, *Proc. ISWC'08*, pp. 632–648 (2008).
29. Xiang, L., Yuan, Q., Zhao, S., Chen, L., Zhang, X., Yang, Q. and Sun, J.: Temporal recommendation on graphs via long- and short-term preference fusion, *Proc. KDD'10*, pp. 723–732 (2010).
30. Ziegler, C. N. and McNee, S. M.: Improving recommendation lists through topic diversification, *Proc. WWW'05*, pp. 22–32 (2005).
31. Zimdars, A., Chickering, D. M. and Meek, C.: Using Temporal Data for Making Recommendations, *Proc. UAI'01*, pp. 580–588 (2001).