Exploiting Contextual Information from Event Logs for Personalized Recommendation

Dongjoo Lee, Sung Eun Park, Minsuk Kahng, Sangkeun Lee, and Sang-goo Lee

Abstract. Nowadays, recommender systems are widely used in various domains to help customers access to more satisfying products or services. It is expected that exploiting customers' contextual information can improve the quality of recommendation results. Most earlier researchers assume that they already have customers' explicit ratings on items and each rating has customer's *abstracted context* (e.g. summer, morning). However, in practical applications, it is not easy to obtain customers' explicit ratings and their abstract-level contexts. We aim to acquire customers' preferences and their context by exploiting the information implied in the customers' previous event logs and to adopt them into a well known recommendation technique, Collaborative Filtering (CF). In this paper, we show how to obtain customers' implicit preferences from event logs and present a strategy to abstract context information from event logs considering fuzziness in context. In addition, we present several methods to cooperate achieved contextual information and preferences into CF. To evaluate and compare our methods, we conducted several empirical experiments using a set of music listening logs obtained from last.fm, and the results indicate that our methods can improve the quality of recommendation.

1 Introduction

Large numbers of various products and services have given customers more freedom of choices. However, at the same time, having too many choices causes difficulties to customers to find out and choose ones that are more suitable for them. There have been many researches in the area of recommender systems to filter out the items

Dongjoo Lee · Sung Eun Park · Minsuk Kahng · Sangkeun Lee · Sang-goo Lee
School of Computer Science and Engineering,
Seoul National University,
Seoul 151-742, Korea
e-mail: {therocks,separk1031,minsuk,liza183}@europa.snu.ac.kr,
sglee@europa.snu.ac.kr

R. Lee (Ed.): Computer and Information Science 2010, SCI 317, pp. 121–139.
springerlink.com

that customers may not be interested in and provide proper ones that may satisfy customers [13, 15, 9].

Recently, as we get more opportunities to acquire customers contextual information (e.g. current location) due to the advance of mobile computing technologies, a lot of researchers have made effort to incorporate contextual information in recommender systems [1, 5, 16, 12, 17, 2, 14]. Some systems adopt rule based framework to generate recommendation for varying context. In this case, recommendation rules for different situations should be manually defined by application developers [14]. However, this approach requires time-consuming effort of application developers or service providers.

As collaborative filtering is acknowledged as one of the most widely used recommendation techniques whose performance in quality and execution time are proved to be reasonable by lots of applications on commercial sites like Amazon.com[1] and CF systems do not require manual definition of rules, so the burden of application developers can be reduced, several researches have been introduced to incorporate contextual information into CF systems.

Adomavicius et al. [1] adopt the multidimensional data model to incorporate contextual information in CF systems and apply the reduction-based approach, which uses ratings only related to the current context. The approach of Weng et al. [17] is similar to [1], but they focus more on solving the contradicting problems among hierarchical ratings. Chen et al. [5] propose a design for a context-aware CF system where ratings of users are weighted according to context similarity.

All of these CF-based context-aware recommendation approaches are based on the assumption that there are available ratings of items and each rating has customer's abstract level context information. However, it is hard to obtain sufficient feedback from users in practice and generally the contextual information exists as numerical or continuous values (e.g. timestamp, GPS code, temperature) rather than abstracted context data (e.g. summer, morning). As results, the approaches are not able to be easily applied to real world applications and hard to be evaluated, so they are not evaluated [5] or present evaluation through survey rather than performing experiments using real world data set.

To tackle these problems, we get the insights from the fact that many real world applications and services continuously produce a lot of their customers' activity logs, and there have been many works to record a persons various activities [10] such as Nike+, Nokia's LifeBlog, MyLifeBits [7] and so on. Event logs include a lot of information that implies customers contextual information and their different preferences depending on contexts. However, most of context related data, such as timestamp, GPS code, and temperature, recorded in event logs also cannot be directly used as contextual information. Thus, we need a context abstraction strategy to obtain abstract context data from the event logs. In this paper, we present a context abstraction method that considers fuzziness in context and show how to extract different preferences of users according to each context in several ways.

[1] Refer http://en.wikipedia.org/wiki/Collaborative_filtering#Applications to see list of commercial sites adopting collaborative filtering.

We conducted several experiments using users' music listening logs gathered from last.fm[2] to see the effects of our approach and compare them. We defined a novel evaluation measure applicable to context-aware cases, *HR@n*. It measures how many real usages of each recommendation were found within top-*n* recommended items. In general, experimental results show that incorporating contextual information presented in the paper leads to higher *HR@n* compared to traditional CF and exploiting implicit feedback and abstract contextual information achieved from event logs can improve the quality of recommendation.

The remainder of the paper is organized as follows. In Sect. 2, we explain how to use implicit feedback from logs to recommend items to users. Then, we present how we incorporate contextual information into traditional CF systems in Sect. 3. Empirical evaluation performed with users' music listening logs is shown in Sect. 4. Finally, Sect. 5 concludes the paper.

2 Exploiting Implicit Feedback in Event Logs

When we do not have explicit feedback from users on items, we may use *event logs* related to the users and items to obtain users' preferences on items. In this section, we describe how to exploit implicit feedback involved in event logs in traditional recommendation technique.

2.1 Recommendation Space

Let us assume that there are *ns* items and *nu* users. Then, we can define a set of items $S = \{s_1, s_2, \ldots, s_i, \ldots, s_{ns}\}$ and a set of users $U = \{u_1, u_2, \ldots, u_j, \ldots, u_{nu}\}$, where s_i and u_j mean i-th item and j-th user respectively. Suppose we have obtained event logs related to the items and the users. For simplicity, we assume that there are only one type of events. Then, we can define a set of event logs $L = \{l_1, l_2, \ldots, l_{nl}\}$. Each log l is a tuple $(u, s, t, \ldots)$, where u is a user, s is an item, and t is a timestamp for the event log. In addition to the timestamp, there are various types of data such as IP address, GPS code, and so on, which can be the sources to obtain the context related to the event. Figure 1 shows an example of music listening logs of users.

On the basis of this recommendation space, we describe how to recommend items to an *active user*.

2.2 Popularity and Individual Preference

Simple way of recommending items to the active user is to give the most popular items. For example, in music domains, most frequently listened top-*n* songs can be a simple recommendation list. In such a notion, popularity r_i of an item s_i can be measured by counting how many times the item appears in the logs.

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

User	Song	Timestamp
1432	White winter hymnal - Fleet foxes	08/07/19 16:55
1941	Let's get out of this country - Camera obscura	08/08/03 22:14
1941	White winter hymnal - Fleet foxes	08/08/10 22:12
1432	White winter hymnal - Fleet foxes	08/08/31 12:01
2133	Let's get out of this country - Camera obscura	08/09/04 01:04
2133	My Moon My Man - Feist	08/09/06 14:21

Fig. 1 An example of event logs: music listening logs from last.fm.

$$r_i = |\{l | l \in L \wedge l.s = s_i\}| \tag{1}$$

However, the popularity-based recommendation method will generate the same list of items to all users without considering individual preferences.

If we obtained the explicit feedback, such as rating scores, from users, we can use them as individual preferences on items and generate different item lists for each user. However, getting rating scores from users is very hard even if a user actually had used or bought them in practice. Therefore, we use implicit feedback involved in event logs to obtain individual preferences in the same manner as the popularity. For example, in music domain, we assume that a user prefers songs that he or she has listened many times. In this way, preference $r_{i,j}$ of user u_j on item s_i is measured by counting how many times u_j and s_i appears together in the logs.

$$r_{i,j} = |\{l | l \in L \wedge l.s = s_i \wedge l.u = u_j\}| \tag{2}$$

Once these initial preferences are specified, preference function P that maps all users and items to preference scores can be estimated.

$$P : Users \times Items \rightarrow Preferences$$

Then, items on which the user has high preference can be recommended to the active user.

2.3 Collaborative Filtering

In collaborative filtering systems, predicted preference score $p_{i,a}$ for active user u_a on item s_i is calculated by weighted sum of similar users' rating score on s_i. Usually only k most similar users are considered to predict the preferences.

$$p_{i,a} = \bar{r}_{u_a} + \frac{1}{\alpha} \cdot \sum_{j=1}^{topk} (r_{i,j} - \bar{r}_{u_j}) \cdot \mathrm{sim}(u_a, u_j), \tag{3}$$

where $\alpha = \sum_{j=1}^{topk} \mathrm{sim}(u_a, u_j)$, $\bar{r}_{u_j}$ is the average preference score of comparing user u_j, and $\mathrm{sim}(u_a, u_j)$ is the similarity between active user u_a and comparing user u_j, i.e., the more u_j similar to u_a, the more weight rating $r_{i,j}$ will carry in predicting $p_{i,a}$. There are many ways to measure the similarity between two users. In this paper, we use cosine similarity between two users.

$$\mathrm{sim}(u_a, u_j) = \frac{\sum_{i=1}^{ns} r_{i,a} \times r_{i,j}}{\|u_a\| \cdot \|u_j\|}, \tag{4}$$

where $\|u_j\| = \sqrt{\sum_{i=1}^{ns} {r_{i,j}}^2}$.

3 Exploiting Contextual Information

3.1 Context in Event Logs

Dey et al. [6] noted that *context is any information that can be used to characterize the situation where a user or an entity is*. In this paper, we focus on exploiting contextual information obtainable from event logs, such as timestamp, GPS code, temperature, and so on, which are usually sensed by various sensors and recorded in event logs, in recommendation.

Previously, Adomavicius et al. [1] proposed an approach to incorporate contextual information in traditional recommendation techniques. In their approach, only ratings related to the *current context* of the active user were used to make recommendation and recommendation space was reduced to the user-item space. We call this approach the *reduction-based approach*. The reduction-based approach is useful because we can easily adopt traditional recommendation techniques used in the user-item space. In addition, we can easily use existing analytical techniques used in the areas of *data warehouse* (DW) and *online analytical processing* (OLAP). However, we cannot directly adopt this approach because our situation is different from their situation where they assumed that 1) they had records of rating and 2) each record had *abstracted context*, such as 'weekend', 'at home', 'with friend'.

If we can select logs related to the current context ctx, we may solve the first problem by obtaining context dependent preferences. The context dependent preference $r_{i,j,ctx}$ of user u_j on item s_i is measured by counting how many logs related to the user and the item appear in that context.

$$r_{i,j,ctx} = |\{l | l \in L \wedge l.s = s_i \wedge l.u = u_j \wedge l.ctx = ctx\}|, \tag{5}$$

where $l.ctx$ is the context in which the event recorded in log l had happened. In our case, the context of a log is a complex of raw level values recorded in each log. Therefore, a context is defined as a tuple of nv values from different types of sensors.

$$\begin{aligned} l.ctx &= (timestamp, GPScode, temperature, \ldots) \\ &= (v_1, v_2, \ldots, v_{nv}) \end{aligned} \tag{6}$$

In this context model, it is not natural to compare contexts exactly in the raw level to obtain context-dependent preferences. For example, two temperature values 29°C and 30°C is not equal but have the same context, 'hot'. Instead of considering logs whose context is equal to the current context, we may use logs whose context is *similar* to the current context. Then, context-dependent preference $r_{i,j,ctx}$ of user u_j on item s_i can be measured by the sum of similarity values between the current context and the context of each event log.

$$r_{s_i,u_j,ctx} = \sum_{\forall l \in L \wedge l.s = s_i \wedge l.u = u_j} sim(ctx, l.ctx) \tag{7}$$

Chen et al. [5] proposed an idea of measuring similarity between contexts based on this context model. The similarity between the current context ctx and the context of a log $l.ctx$ is measured by the sum of similarities between values of each attribute.

$$\text{sim}(ctx, l.ctx) = \sum_{k=1}^{nv} \text{sim}_k(ctx.v_k, l.ctx.v_k), \tag{8}$$

where $\text{sim}_k()$ is the comparator for the k-th context type and returns the value between 0 and 1. Values obtained from sensors can be compared in various manners according to its type. *Categorical* values can be considered as similar if they are the same. *Continuous* values can be considered as similar if they are close each other.

The most data obtained from sensors are continuous, and there are several problems in defining comparators for them. First, how closely do they have to be to be similar? Obviously, 29°C is similar to 30°C but 5°C is not. Then, how about 22°C? Second, how do we treat periodic values such as time? For example, timestamp itself cannot reflect the periodicity of our life. It needs to be subdivided into several dimensions, such as year, month, and day. Third, is closeness between values the only semantic to compare contexts? For example, two GPS points indicating highways near Seoul and Busan respectively are far, but they should be considered similar because both of them is on 'highway' for some applications. Besides, there is another critical problem in this approach. We need to access whole logs to obtain preferences of users at the current context at every time we make a recommendation. This is not feasible when we deal with very large amount of logs.

We abstract context of logs in conceptual level to solve these problems and access the merits of the multidimensional approach, where preferences can be materialized in DW and existing techniques used in DW and OLAP are easily adoptable.

3.2 Context Abstraction

We exploit the cognitive aspects of contexts to abstract context. Let us consider sentences below.

- "I listen joyful music in the *morning* to help waking me out."
- "I like listening to dance music when it is *hot summer*."

User	Song	Season	Day-of-week	Time-of-day
1432	S1001	summer	weekend	evening
1941	S2130	summer	weekday	night
1941	S1001	summer	weekend	night
1432	S1001	summer	weekend	noon
2133	S2130	autumn	weekday	afternoon
2133	S1010	autumn	weekend	afternoon

Fig. 2 Context of each log is abstracted with contextual concepts. Songs are represented by their ID for simplicity.

In these sentences, a user expresses her or his preferences on music. In particular, the preferences are only valid at the specific contexts, which are expressed with imprecise words such as 'morning', 'summer', and 'hot'. As Whorf [18] stated that *the language we speak, to some degree at least, forces us to orient our view of the world*, we also think that *words* that we use in daily life can specify contexts that affect our life. We can find this easily in 'Eskimo words for snow'[3]. This notion gives a foundation to consider raw level values in conceptual level. We can map raw values into several groups that share common characteristics in the semantic level. For example, we can categorize temperature into groups expressed by words such as {'warm', 'cool', 'cold', 'hot', ...}. Since these words have been used for a long time, they categorize raw values well in the semantic level. We call these words as *contextual concepts*.

Context can be re-defined as a tuple of contextual concepts, each of which describes a certain attribute of context, such as season, day-of-week, time-of-day, location, temperature, and so on.

$$\begin{aligned} l.ctx &= (\text{timestamp}, \text{GPScode}, \text{temperature}, \ldots) \\ &= (\text{season}, \text{dayofweek}, \text{timeofday}, \text{location}, \text{temperature}, \ldots) \\ &= (c_1, c_2, \ldots, c_k, \ldots, c_{nd}) \end{aligned} \tag{9}$$

The raw values can be used to determine several attributes of context. For example, a timestamp can be used to determine temporal contexts such as season, day-of-week, and time-of-day. The periodic characteristic of time can be reflected in this manner. Figure 2 shows the extended form of the logs shown in Fig. 1.

3.2.1 Fuzziness in Context Abstraction

Some contextual concepts do not have the clear boundary. For example, there is no clear boundary between 'winter' and 'spring'. Sometimes a day in late 'winter'

[3] http://en.wikipedia.org/wiki/Eskimo_words_for_snow

User	Song	Context
1432	S1001	(summer,1.0), (weekend,1.0), (evening,1.0)
1941	S2130	(summer,1.0), (weekday,1.0), (night,1.0)
1941	S1001	(summer,1.0), (weekend,1.0), (night,1.0)
1432	S1001	(summer,0.7), (autumn,0.3), (weekend,1.0), (noon,1.0)
2133	S2130	(summer,0.4), (autumn,0.6), (weekday,1.0), (afternoon,1.0)
2133	S1010	(summer,0.3), (autumn,0.7), (weekend,1.0), (afternoon,1.0)

Fig. 3 Context of each log is represented as a set of fuzzy sets corresponding to each contextual concept.

can be thought as early 'spring'. A value of temperature interpreted as 'cold' can be interpreted as 'cool'. Fuzzy set theory [4] can be a foundation to resolve this *unclear boundary problem*. We consider each contextual concept as a fuzzy set. Then, context of each log can be represented as a set of fuzzy sets corresponding to contextual concepts. Let us assume that there is a universe of contextual concepts $C = \{c_1, c_2, \ldots, c_k, \ldots, c_{nc}\}$. Then, a context ctx can be described by a set of pairs of contextual concept c_k and the membership degree m_k indicating how strongly the context is described by c_k.

$$l.ctx = \{(c_k, m_k) | c_k \in C \wedge m_k = f_{c_k}(l)\}, \tag{10}$$

where f_{c_k} is a membership function that measures how much a log belongs to the concept c_k and returns a value between 0 and 1. In this way, logs can be extended to reflect fuzziness of contextual concepts as shown in Fig. 3.

3.3 Context-Aware Recommendation Algorithm

If the context of logs is represented in the abstracted form, logs can be easily materialized and stored in multidimensional *data cubes* to capture context-dependent preferences of users. Then, we can apply various techniques described in [1] to recommend items. Although the context of logs can be abstracted in two forms as shown in Eq. (9) and Eq. (10), two forms of logs can be easily materialized in data cubes by aggregating the abstracted logs. In this section, we describe several methods of recommending items considering the current context of the active user. We do not describe in detail how data cubes for each method can be constructed and the recommender system is implemented in this paper because we aim to show abstracting and exploiting contextual information in event logs can improve the quality of recommendation results.

3.3.1 Popularity-Based Approach

We can easily obtain the *context-dependent popularity* $r_{i,ctx}$ of item s_i in context ctx by counting how many times s_i appears in the logs having ctx and recommend items that have high context-dependent popularity in the current context.

$$r_{i,ctx} = |\{l | l \in L \wedge l.s = s_i \wedge l.ctx = ctx\}| \tag{11}$$

This is the most simple method of exploiting context to recommend items.

The context-dependent popularity cannot consider individual preferences of each user. We can integrate individual preference shown in Eq. (2) and context-dependent popularity shown in Eq. (11) to recommend items considering individual preferences and context.

$$p_{i,a,ctx} = \lambda \cdot p_{i,a} + (1 - \lambda) \cdot r_{i,ctx} \tag{12}$$

3.3.2 Reduction-Based Approach

In the reduction-based CF, predicted preference score $p_{i,a,ctx}$ of active user u_a on item s_i in context ctx is estimated by the Eq. (13).

$$p_{i,a,ctx} \propto \sum_{j=1}^{topk} (r_{i,j,ctx} - \bar{r}_{u_j}) \cdot \mathrm{sim}_{ctx}(u_a, u_j) \tag{13}$$

Similarity between two users is captured by only ratings related to the current context.

$$\mathrm{sim}_{ctx}(u_a, u_j) = \frac{\sum_{i=1}^{ns} r_{i,a,ctx} \times r_{i,j,ctx}}{\|u_{a,ctx}\| \cdot \|u_{j,ctx}\|}, \tag{14}$$

where $\|u_{j,ctx}\| = \sqrt{\sum_{i=1}^{ns} r_{i,j,ctx}{}^2}$.

Instead of using the context dependent user similarity, we may directly use *global* user similarity shown in Eq. (4) to weigh the preference of a comparing user.

Also, the global user similarity can be measured by considering context dependent preferences in all contexts as shown in Eq. (15).

$$\mathrm{sim}'(u_a, u_j) = \frac{\sum_{\forall ctx} \sum_{i=1}^{ns} r_{i,a,ctx} \times r_{i,j,ctx}}{\|u'_a\| \cdot \|u'_j\|}, \tag{15}$$

where $\|u'_j\| = \sqrt{\sum_{\forall ctx} \sum_{i=1}^{ns} r_{i,j,ctx}{}^2}$.

3.3.3 Disjunction-Based Approach

There is another way of recommending items to the active user considering the current context. Let the current context $ctx = (c_1, c_2, \ldots, c_k, \ldots c_{nc})$. Preference score $p_{i,a,ctx}$ of the active user on item s_i in the current context can be calculated by the disjunctive aggregation of the estimated preferences of the active user in each context described by contextual concept c_k.

$$p_{i,a}(ctx) = \frac{1}{\gamma} \sum_{k=1}^{nc} \alpha_k \cdot p_{i,a,k}, \tag{16}$$

where γ is $\gamma = \sum_{k=1}^{nc} \alpha_k$. The preference $r_{i,j,k}$ of user u_j on item s_i in the context described by c_k is measured by counting how many logs related to u_j, s_i, and c_k appear in the logs.

$$r_{i,j,k} = |\{l | l.u = u_j \wedge l.s = s_i \wedge l.ctx.c_k = c_k\}| \tag{17}$$

Then, $p_{i,a,k}$ can be calculated as:

$$p_{i,a,k} = \bar{r}_{u_a} + \frac{1}{\alpha_k} \cdot \sum_{j=1}^{topk} (r_{i,j,k} - \bar{r}_{u_j}) \cdot \mathrm{sim}_{c_k}(u_a, u_j), \tag{18}$$

where $\alpha_k = \sum_{j=1}^{topk} \mathrm{sim}_{c_k}(u_a, u_j)$ and $\mathrm{sim}_{c_k}(u_a, u_j) = \frac{\sum_{i=1}^{ns} r_{i,a,k} \times r_{i,j,k}}{\sqrt{\sum_{i=1}^{ns} r_{i,a,k}{}^2} \sqrt{\sum_{i=1}^{ns} r_{i,j,k}{}^2}}$.

3.3.4 Incorporating Fuzziness

We presented a way of abstracting context while considering fuzziness of contextual concepts. Let the current context be $ctx = \{(c_1, m_1), \ldots, (c_k, m_k), \ldots, (c_{nc}, m_{nc})\}$. Then, the reduction-based and disjunction-based approach can be easily modified to this context model. In this paper, we only present how disjunction-based approach can be modified so as to consider fuzziness. At first, Eq. (16) can be modified to

$$p_{i,a}(ctx) = \frac{1}{\gamma} \sum_{\forall (c_k, m_k) \in ctx} m_k \cdot \alpha_k \cdot p_{i,a,k}, \tag{19}$$

where γ is $\gamma = \sum_{\forall (c_k, m_k) \in ctx} m_k \cdot \alpha_k$. The preference $r_{i,j,k}$ of user u_j on item s_i in the context described by c_k is measured by the sum of m_k of logs related to the u_j and s_i.

$$r_{i,j,k} = \sum_{\forall l \in L \wedge l.s = s_i \wedge l.u = u_j} f_k(l) \tag{20}$$

Then, $p_{i,a,k}$ can be calculated as:

$$p_{i,a,k} = \bar{r}_{u_a} + \frac{1}{\alpha_k} \cdot \sum_{j=1}^{topk} (r_{i,j,k} - \bar{r}_{u_j}) \cdot \mathrm{sim}_{c_k}(u_a, u_j), \tag{21}$$

where $\alpha_k = \sum_{j=1}^{topk} \mathrm{sim}_{c_k}(u_a, u_j)$ and $\mathrm{sim}_{c_k}(u_a, u_j) = \frac{\sum_{i=1}^{ns} r_{i,a,k} \times r_{i,j,k}}{\sqrt{\sum_{i=1}^{ns} r_{i,a,k}{}^2} \sqrt{\sum_{i=1}^{ns} r_{i,j,k}{}^2}}$.

Table 1 lists the methods of recommending items to the active user and summarize how each method scores items considering individual preference and contextual information. Experimental analysis about the methods with varying several parameters is presented in the next section.

Table 1 Summarization of recommendation algorithms.

Name	Scoring method	Related equations
POP	Popularity	(1)
CF	Generic CF	(3)
CAPOP	Context-dependent popularity	(11)
CA+CF	Generic CF is combined with context-dependent popularity	(12)
CACF-A1	Reduction based CF with global user similarity	(13) and (4)
CACF-A2	Reduction based CF with global user similarity considering context	(13) and (15)
CACF-A	Reduction based CF with context-dependent user similarity	(13) and (14)
CACF-O	Disjunction based CF	(16)
F-CACF-A	Considering fuzzines based on CACF-A	
F-CACF-O	Considering fuzzines based on CACF-O	(19)

4 Experiment and Analysis

We performed several experiments with music listening logs to see the effects of the methods for exploiting contextual information in CF systems.

4.1 Experimental Setup

4.1.1 Dataset

Among several domains, such as movie, news, music, and book, we chose music domain for our experiments since the amount of available logs are larger and users' preference on music seems more dependent to context especially for time than other domains [3, 12]. We collected users' music listening logs from the last.fm Web site. It is composed of total 28,809,524 logs of 10,792 users on 2,283,206 songs. By analyzing the dataset, we found out that the large number of songs were listened by small number of users. This sparsity of data brings several problems that are not the major interest in this paper, so we chose the frequently listened songs first and then chose the logs related to those songs. We created two datasets for varying the dataset size: SET1 is composed of 2,421,362 logs of 4,204 users on most frequently listened 10,000 songs; SET2 is composed of 1,356,137 logs of 4,204 users on most frequently listened 3,000 songs. Their statistical information is summarized in Table 2.

4.1.2 Evaluation Measure

Although evaluation measures such as MAE, RMSE, and recomendation list precision are popular for evaluating CF-based recommender systems [8], evaluating the

Table 2 Statistical information of datasets

Property	SET1	SET2
Total number of songs	10,000	3,000
Total number of logs	2,421,362	1,357,137
Average number of logs related to a user	582.6	335.1
Average number of songs a user listened	207.6	113.1
Average number of logs a user has listened a song	2.8	3.0
Average number of users who had listened a song	242.1	452.4

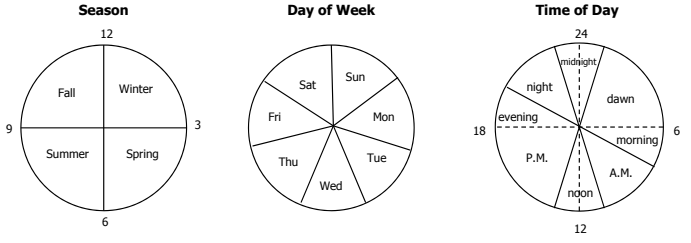


Fig. 4 Concepts for temporal context

recommender system that considers temporal context should be different since the recommended item lists are not same even for the same user according to when the recommendation is made. Therefore, we designed a novel evaluation measure called $HR@n$[4], which indicates how many real usages of each recommendation were found within top-n recommended items. In specific, we first generate a list of songs using the information of a log in the testset, and check if the list contains the song found in the log.

4.1.3 Context Abstraction

As we discussed in Sect. 3.2, abstracting context is possible in various ways depending on the application. In our experiment, we only exploit temporal context since we could not get other data but timestamps. We subdivided temporal context into three dimensions: season, day-of-week, and time-of-day. The contextual concepts and the method to map raw level data into them are described in Fig. 5.

We used the trapezoidal fuzzy membership function to measure membership degree of each log for the temporal context. Several previous works [12, 16] also used trapezoidal function for abstracting timestamp. Fig. 5 presents the membership functions for the seasonal concepts visually.

[4] HR stands for Hit Ratio.

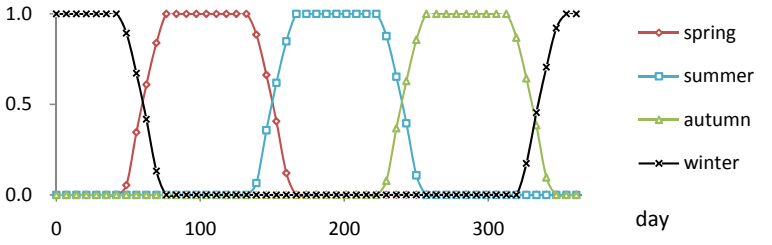


Fig. 5 Fuzzy membership functions for seasonal concepts

4.2 Analyses on Experimental Results

4.2.1 Impact of the Number of Similar Users

In order to see the effect of the number of similar users, we first fixed the ratio of training and test datasets, and observed how the performance of each method varies as the number of similar users changes.

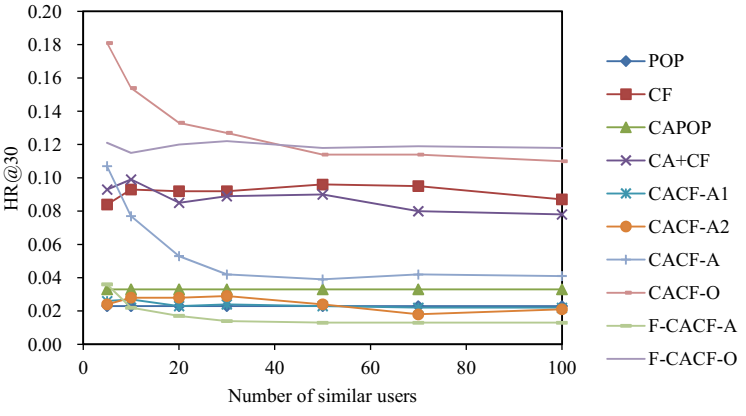
Generally, most methods showed the best performance when the number of similar users is less than 20. When the number of similar users is above 20, the methods does not show big difference on their performance with varying the number of similar users. At both Fig. 6a and 6b, the performance of CACF-O and CACF-A sharply decreases in the range from 5 to 20. It seems that using the small number of similar users is better when using these methods for recommendation. When the number of similar users increases, $HR@30$ for CF and CA-CF increases until they get stable. and then decrease.

Since it shows a general compromise over all methods when the size of neighborhood is 10, we examine the performance at this value and analyze the performances of various algorithms int the next section.

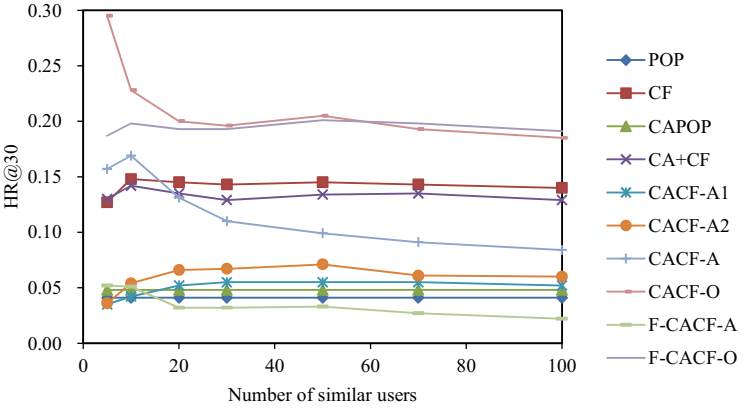
4.2.2 Impact of Dataset

As shown in Fig. 7, changing training set ratio has not shown large influences on performance. From this observation, we can say that 60 percent of our dataset is enough for building recommendation models. Although the graphs show small waves over training set ratio, no consistent patterns is found. It leads us to conclude that these small waves mean nothing more than small difference of distribution on training and test sets.

Furthermore, we see that the larger dataset is not always better when it comes to the recommending items with the proposed methods. The dataset with high density of logs has more effect on the results. To be more specific, as you can see at the Table 2, SET2 has larger average number of logs a user has listened to one song and the average number of users who had listened to a song than SET1 while keeping less number of listening logs per user and smaller data size. Generally all



(a) SET1, 80% training dataset



(b) SET2, 80% training dataset

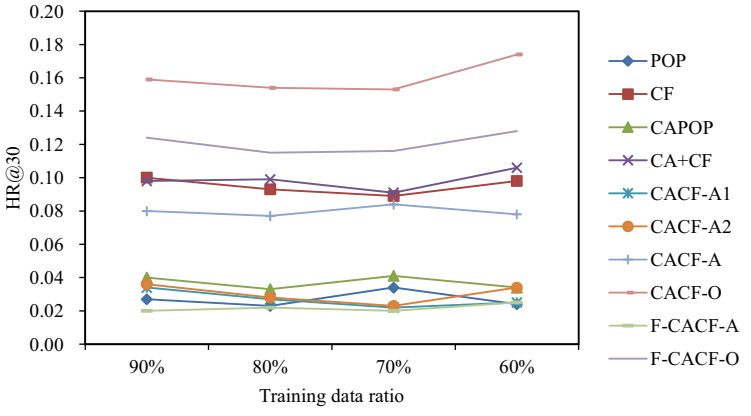
Fig. 6 *HR*@30 with varying the number of similar users(k)

algorithms showed better performance with SET2 than SET1. Figure 7 shows that the maximum *HR*@30 for SET2 is around 0.26 while that of SET1 is around 0.17.

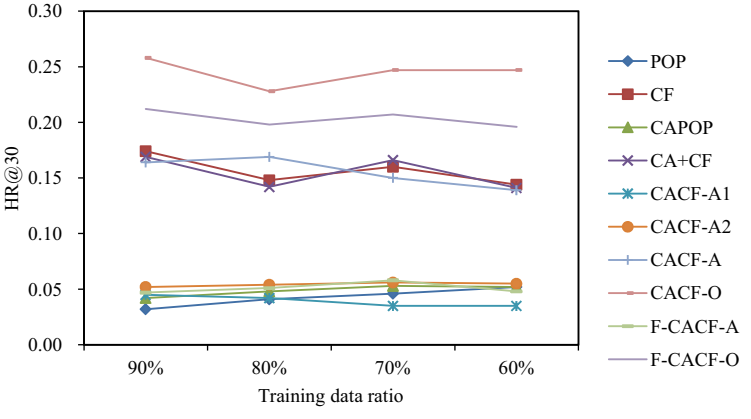
4.2.3 Comparison of Algorithms

We compared several algorithms to see the effect of each factor for exploiting contextual information for recommendation. Figure 8 presents the results.

As we expected, the simple methods, such as POP, CAPOP showed low performances. By observing low performance of CAPOP, which is almost the same as that of simple POP algorithm, we conclude that there is no clear performance improvements when considering only temporal context without any personalization



(a) SET1, 10 similar users



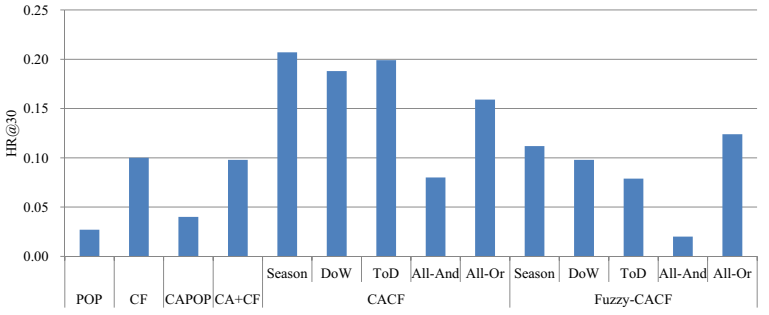
(b) SET2, 10 similar users

Fig. 7 *HR*@30 with varying the ratio of training and test sets.

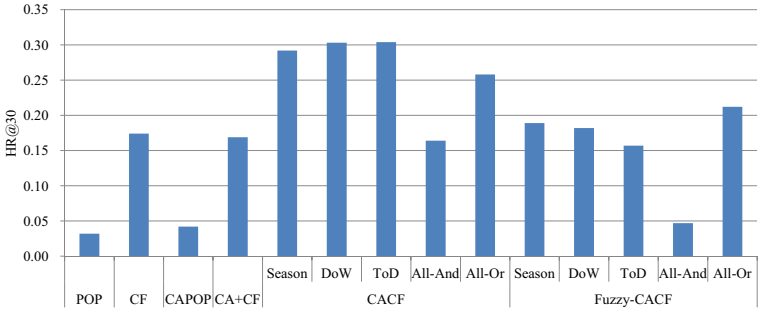
technique. In addition, simple aggregation of CA and CF results show almost no performance improvements.

Combination of CF and Context-awareness

The methods which dynamically find similar users in the current context, and gain their preferences at that context is a better way to combine the CF technique and the context-awareness. CACF-A1 finds similar users who has similar taste in general, and generates recommendation list with those users' preferences at the context related to the current context. On the other hand, CACF-A finds similar users in the current context and recommends items which they preferred at the context to the



(a) SET1, 90% training, 10 similar users



(b) SET2, 90% training, 10 similar users

Fig. 8 *HR*@30 for several algorithms with 90% training dataset and 10 similar users. Season, DoW, ToD means that the recommendation was perfomred considering the season, day-of-week, and time-of-day respectively. All-And stands for CACF-A and All-Or stands for CACF-O.

recommending moment. The result shows CACF-A outperforms CACF-A1 with large differences. Thus, we can conclude that not similar users in overall time, but context-dependent similar users help context-aware recommendation.

Reduction vs. Disjunction

By observing better performance of CACF-O and F-CACF-O than that of CACF-A and F-CACF-A, we can conclude that disjunctive aggregation is a better way for incorporating context in CF than reduction-based aggregation. While CACF-A shows performance which is similar to or even worse than CF, CACF-O shows the results with two times better performance over CACF-A and also better than CF. This result is consistent over the adoptation of fuzziness. The inferior result for reduction-based approach might be due to sparsity problem which happens when many dimensions of context are considered at the same time. As more context dimension is considered, the size of data used for finding similar users is decreased, which diminishes recommendation accuracy.

Fuzziness

The adoptation of fuzziness turned out to degrade the methods as shown in Fig. 8. It might be because we derived the formula in an intuitive way based on previous work. Since there is no clear methodology for defining fuzzy membership function for abstracting timestamp, the fuzzy membership function we used gives wrong weight, and it might degrade recommendation performance. From this result, we strongly believe that using fuzziness for abstracting context should be careful until further research shows the usage and effect of fuzzy functions on various datasets in detail.

Multiple Dimensions

Considering only one dimension for the recommendation showed similar performances (CACF-Season, CACF-DoW, CACF-ToD) although the result for considering only season (CACF-Season) showed a little better performance than other dimensions. The interesting result is that considering one dimension at a time instead of combining all the dimensions at once is better at both CACF and F-CACF. This observation implies that we should be careful when there are many dimensions to consider for context abstraction. It is because all dimensions or concepts are not independent to others, for example, concept 'midnight' might be more related to 'night' than 'noon'. Therefore, when there are many dimensions to consider, simple aggregation may degrade recommendation performance, and some techniques used in [11] can be applied to overcome this issue.

5 Conclusion and Future Work

Although there have been many researches to enable context-awareness in recommender systems, most of them are based on the assumption that customers' explicit ratings and their abstracted context are available, which are not available in many cases. In this paper, we presented a context abstraction strategy and showed how we can obtain implicit users' preference and abtracted context from event logs. In conclusion, since event logs are generally available, our approach is applicable to a wide range of applications that are expected to produce better performance when context information is incorporated.

We chose music domain for evaluation and conducted several experiments using real world data set gathered from last.fm. Through the experimental results, we showed that our methods using implicit feedback in event logs, especially the disjuctive aggregation method, can enhance the recommendation quality. In addition, it showed that using fuzziness on the concept abstraction and combining multiple dimension should be applied carefully.

In future work, we plan to work on choosing fuzzy functions by considering context dimension characteristics and finding out better way of combining

multiple dimensions. Additionally, we are currently working on developing a general context-aware recommendation system which enables empirical experiments of various combination of dimensions that affect recommendation results by using event logs.

Acknowledgments

This research was supported by the MKE(The Ministry of Knowledge Economy), Korea, under the ITRC(Information Technology Research Center) support program supervised by the NIPA(National IT Industry Promotion Agency). (grant number NIPA-2010-C1090-1031-0002)

References

1. Adomavicius, G., Sankaranarayanan, R., Sen, S., Tuzhilin, A.: Incorporating contextual information in recommender systems using a multidimensional approach. ACM Trans. Inf. Syst. 23(1), 103–145 (2005)
2. Baltrunas, L.: Exploiting contextual information in recommender systems. In: RecSys 2008: Proceedings of the 2008 ACM conference on Recommender systems, pp. 295–298. ACM, New York (2008)
3. Baltrunas, L., Amatriain, X.: Towards Time-Dependant Recommendation based on Implicit Feedback. In: Proceedings of Workshop on Context-Aware Recommender Systems (CARS-2009) in conjunction with the 3rd ACM Conference on Recommender Systems (2009)
4. Bojadziev, G., Bojadziev, M.: Fuzzy Sets, Fuzzy Logic, Applications, pp. 113–140. World Scientific Publishing Co. Pte. Ltd., Singapore (1995)
5. Chen, A.: Context-aware collaborative filtering system: predicting the user's preferences in ubiquitous computing. In: CHI 2005: CHI 2005 extended abstracts on Human factors in computing systems, pp. 1110–1111. ACM, New York (2005)
6. Dey, A.K.: Understanding and using context. Personal Ubiquitous Comput. 5(1), 4–7 (2001)
7. Gemmell, J., Bell, G., Lueder, R.: MyLifeBits: a personal database for everything. Communications of the ACM 49(1), 95 (2006)
8. Herlocker, J., Konstan, J., Terveen, L., Riedl, J.: Evaluating collaborative filtering recommender systems. ACM Transactions on Information Systems (TOIS) 22(1), 53 (2004)
9. Hill, W., Stead, L., Rosenstein, M., Furnas, G.: Recommending and evaluating choices in a virtual community of use. In: CHI 1995: Proceedings of the SIGCHI conference on Human factors in computing systems, pp. 194–201. ACM Press/Addison-Wesley Publishing Co. (1995)
10. Lee, S., Gong, G., Lee, S.G.: Lifelogon: Log on to your lifelog ontology! In: Bernstein, A., Karger, D.R., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., Thirunarayan, K. (eds.) ISWC 2009. LNCS, vol. 5823, Springer, Heidelberg (2009)
11. Melucci, M., White, R.: Utilizing a geometry of context for enhanced implicit feedback. In: Proceedings of the sixteenth ACM conference on Conference on information and knowledge management, pp. 273–282. ACM, New York (2007)

12. Park, H.S., Yoo, J.O., Cho, S.B.: A context-aware music recommendation system using fuzzy bayesian networks with utility theory. In: Wang, L., Jiao, L., Shi, G., Li, X., Liu, J. (eds.) FSKD 2006. LNCS (LNAI), vol. 4223, ch. 121, pp. 970–979. Springer, Heidelberg (2006)
13. Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., Riedl, J.: Grouplens: an open architecture for collaborative filtering of netnews. In: CSCW 1994: Proceedings of the 1994 ACM conference on Computer supported cooperative work, pp. 175–186. ACM, New York (1994)
14. van Setten, M., Pokraev, S., Koolwaaij, J.: Context-aware recommendations in the mobile tourist application compass. In: De Bra, P.M.E., Nejdl, W. (eds.) AH 2004. LNCS, vol. 3137, pp. 235–244. Springer, Heidelberg (2004)
15. Shardanand, U., Maes, P.: Social information filtering: algorithms for automating "word of mouth". In: CHI 1995: Proceedings of the SIGCHI conference on Human factors in computing systems, pp. 210–217. ACM Press/Addison-Wesley Publishing Co. (1995)
16. Shin, D., Lee, J.W., Yeon, J., Lee, S.G.: Context-aware recommendation by aggregating user context. In: CEC 2009: Proceedings of the 2009 IEEE Conference on Commerce and Enterprise Computing, pp. 423–430. IEEE Computer Society, Los Alamitos (2009)
17. Weng, S.S., Lin, B., Chen, W.T.: Using contextual information and multidimensional approach for recommendation. Expert Syst. Appl. 36(2), 1268–1279 (2009)
18. Whorf, B.L.: Language, thought and reality: Selected writings of Benjamin Lee Whorf. MIT Press, Cambridge (1956)