Session-based Collaborative Filtering for Predicting the Next Song

Sung Eun Park, Sangkeun Lee, Sang-goo Lee
School of Computer Science and Engineering
Seoul National University
Seoul, South Korea
{separk1031,liza183,sglee}@europa.snu.ac.kr

Abstract—**Most music recommender systems produce a set of recommendation based on user's previous preference. But the information is not always attainable. Focusing on the fact that music listening behavior is a repetitive action of playing one song at a time, we predict the next item based on user's currently selected items even when user's previous preference is not available. We propose a simple but effective recommendation method for this problem called Session-based Collaborative Filtering (SSCF), and we look into the different parameters that affect the recommendation accuracy. Our evaluation on real-world dataset indicated that SSCF improves recommendation accuracy.**

Internet Technology and Applications, e-Commerce; Music Recommendation

I. INTRODUCTION

The advanced mobile computing technology allows music consumers to access to online music streaming services with their portable devices, such as smart phones. Now that music consumers have the access to virtually all of the songs on the Internet, and it has become more difficult for them to find what they really want. To reduce the burden of choosing items from the large amount of songs, many researchers have focused on recommender systems to produce a set of recommendation by predicting user's preference

The typical music recommenders use Collaborative Filtering(CF) for generating personalized recommendations. The recommender system adopting CF assumes that the system already have enough information of users' preference, such as reasonable number of user's previous item ratings, or usage history data that can be possibly used to infer users' preference, which is a tiresome work for users. For example, this recommendation technique requires users to upload their usage history or at least sign in to collect personal profile. In this scenario, if a user is new or has not signed in to the web service, the recommender system using CF becomes non-reliable. The reason is that it recognizes only the small number of items that the active user has just selected and uses them as a profile of the user. It may cause low recommendation accuracy since the system does not have sufficient users' information to make a fine profile.

Another approach is to exploit the existing music-related knowledge for recommendation, such as music metadata or acoustic features. But these kinds of recommendation assume that each item has descriptive and sufficient number of metadata or features extracted from audio-analysis techniques. However, this requirement cannot always be satisfied. In addition, the recommendation produced by a content-based recommender system tends to share similar style in that they only consider contents similarity. In other words, it cannot generate interesting and brilliant recommendation which is one of the important characteristics for a good recommender systems

In this paper, we study on music recommender system that takes selected item as an input and recommends the next suitable item for the user. Our goal is to accurately recommend the next song to play even when there is not enough information about users. We thought that by repetitively recommending the next song with the currently played songs, it is possible to provide users a list of songs for a timeframe. Thus, we recommend the next song instead of recommending a set of songs at a time. As a better approach for this problem, we introduce a recommendation technique called Session-based Collaborative Filtering (SSCF) that uses preferred items in the similar session. We verify the performance of our method by conducting experiments on a real-world usage log data gathered from Bugs Music, a renowned online music streaming service in Korea. Our experimental results shows an agreement that SSCF improves recommendation accuracy in the cases where the active user's preference is not well-known.

The rest of this paper is organized as follows. We present related work in Section 2. Section 3 describes the problems we are focusing on, and Section 4, we introduce a recommendation technique. Section 5 presents the experimental results on real-world dataset. Finally, the last section concludes the paper including possible future directions in our study.

II. RELATED WORK

In this section, we introduce related work on music recommendation taking items as an input, and then on the next item prediction using the information hidden in session.

A. Music recommendation using items as input

One approach for music recommendation using items is playlist generation. An audio-based playlist generation was proposed in [1]. The authors generate a playlist simply using

the *n* nearest songs from a given seed song. But these approaches assume the existence of rich metadata or audio-processed features of music. The main difference of our work is that our system does not require metadata of music or processing music based on audio signal, since we use the information of co-preferred occurrence in sessions. Also, there is a study on adopting newly selected song to the recommendation. [2] studies on the users' instant feedback by pressing a skip button if the user dislikes the current song, Songs similar to skipped songs are removed, while songs similar to accepted ones are added to the playlist. Although this approach is dealing with a similar problem with this study and plausible application would be more similar, this dynamic playlist generator is based on audio-based features.

While researches introduced above intensively focus on the recommendation of a collection of songs, [3] assumes that there is a meaningful sequence in the order of music playlist and address to suggest playlist of similar songs with an inherent sequential structure. [4] considers sequence of the songs as well by presenting a system that can scale up and handles arbitrarily complex constraints that intrinsically comes from sequence constraints. These researches are also based on metadata or audio signal processing. We also think that it may be an interesting point of view in that sequence matters when recommending music, but we will only focus on the set based recommendation in this study.

Last.fm is an Automatic CF recommender that generates personalized playlists on the basis of usage history of users whose profiles are similar. When there is no profile of an active user, it cannot help using the items user has selected only, so we take this approach as our base line for performance comparison.

B. Predicting Items using Information Hidden in Session

There are many researches that deal with session information in Information Retrieval Researches. [5] intensively studies on the performance heuristics employed to reconstruct sessions from the server log data in the field of Web-usage mining. A URL prediction problem, which predicts next page requests, has been researched in [6,7]. [6] keeps track of what a user has read to collect users' navigation history in a central repository. It applies association rules to discover hidden information from user's navigation history. Each navigation history of a user is considered respectively, and this navigation history is similar concept of session in our work. But this work is different from our model that they only use the information of co-occurrence between items instead of preferred together. [7] showed an approach of predicting the final link the user followed to complete the trail with a already given trail.

[8] focuses on personalized transition graphs over underlying Markov chain, and it predicts the next basket for a purchase. The authors recommend items from sequential set data, in which a set data means a group of items purchased together. This work is different from our work because they focus on finding the relationships of items that possibly exist between the sessions not inside of a session.

[9] shows a methodology of online video recommender system of YouTube. The system finds related videos by defining relatedness score as co-visitation count in session, and the system finds recommendation candidates with items that have been currently watched. This research differs from our work in that our work considers the currently given items a part of the session not an individual items and find the most similar sessions from sessions in training data.

III. Next Song Prediction from Currently Selected Items

In this section, we describe the problem of predicting next item using item usage log. Item usage log data is useful as implicit feedback in that there is no user-side burden for collecting users' preferences. We introduce the notation of data and the model before further describing our approach.

A. Data Model

When a user plays a song, one log in the usage log database is generated, and we call it as an event. A set of events is defined as $E = \{e_1, e_2, ..., e_i, ...e_m\}$. Each event is composed of a user, an item and the occurred time of the event.

We define a session so that each event *e* can belong to one of the sessions. In other words, each session is composed of a set of events made by a user within a given time frame. We formally define session as the group of items listened by a user from the moment he/she starts playing songs to the moment they stop it. Sessions are non-overlapping subsets of events E. All items played at the same session are grouped together. The items and the rating of those items become a profile of the session.

Assuming that there are *n* items and *m* sessions, we can define a set of items $I = \{i_1, i_2, ..., i_j, ...i_n\}$, and a set of sessions $S = \{s_1, s_2, ..., s_i, ...s_m\}$. We view each session profile *s* as pairs of item and its weight.

$$s_k = < (i_{k,1}, w(i_{k,1})), (i_{k,2}, w(i_{k,2})), ..., (i_{k,n}, w(i_{k,n})) > \quad (1)$$

where $w(i_{k,j})$ is the number of occurrence of item i_j in the session s_k.

$$w(i_{k,j}) = \left| \{ e \mid e \in s_k \wedge e.i = i_j \} \right| \quad (2)$$

B. Problem Definition

We denote our problem as next item recommendation. Here we concentrate on the prediction problem, i.e. given a set of items, we ask how well we can predict the next item the user requests.

Definintion 1 Next Item Recommendation. *A query q consists of a set of items in the session. Given a query q, the system ranks all candidates and recommends top-k items that are likely to come after items in query q.*

As Figure 1 intuitively describes the problem, given a few items, we predict the best item that is possibly come to the next of them.

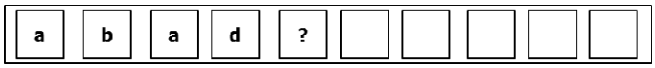

Figure 1 A Discriptive Diagram of Next Item Prediction Problem

IV. RECOMMENING ITEMS FROM SESSIONS

In this section, we propose a technique that predicts the next song request by looking at what past sessions have shown. A person may have a different taste on music depending on the situation. For example, Tom, a Jazz lover, may listen to some energetic jazz music when he feels good while he chooses some misty jazz before he goes to sleep. From this intuition, we thought that by adopting this more detailed relationships shown in the sessions, we can more accurately recommend items for an active session. In short, our approach is that songs frequently preferred together in sessions rather than a user can benefit predicting songs in the active sessions. In the following sections, we describe our recommendation model in detail.

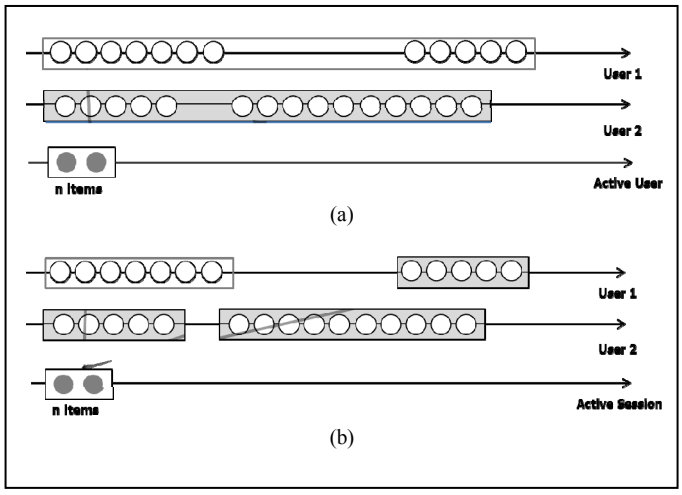

Figure 2. A Comparison between CF and Session-based Approach

A. Session-based Collaboratve Filtering

The underlying assumption on CF is that referencing items of users with similar musical taste will benefit the performance of recommender system. By adopting and modifying this assumption, Session-based Collaborative Filtering (SSCF) finds similar sessions with an active session from given partial information and reference items in the similar sessions. Consequently, instead of making a profile for each user, we make a profile for each session as noted in section III.A above. Figure 2 describes the intuition. (a) depicts a typical CF approach that makes profile of a user with all items a user has listened for entire listening history while (b) depicts a session CF approach which makes many session profiles per user. Given only *n* items of the moment without knowing uses' identification or preferences in advance, the system makes profiles with current items and finds similar users in case of (a), but similar sessions in (b).

We consider the items user currently selected as a part of the current session. Thus, given *p* currently selected items, we make session profile described above, and find the most similar sessions from those in the training sessions. By weighting the similarity to the ratings on the items in the session, the system predicts the rating of an item for the active session. You may have already noticed that we use CF formula for predicting items preference and references top-k similar sessions instead of similar users. Also notice that we use the simplified CF formula shown below. It is because we do not need to predict items' preference exactly, but we only need to rank items. The predicted rating $p_{i,a}$ of item i_i on active session s_a is

$$p_{i,a} = \sum_{j=1}^{topk} r_{i,j} \cdot sim(s_a, s_j) \tag{3}$$

To measure the similarity between sessions, we use cosine similarity shown below.

$$sim(s_a, s_j) = \frac{\sum_{i=1}^{m} r_{i,a} \times r_{i,j}}{\|s_a\| \cdot \|s_j\|} \tag{4}$$

$$\text{Where } \|s_j\| = \sqrt{\sum_{i=1}^{m} r_{i,j}^2}$$

V. EXPERIMENTS

In this section, we discuss the experimental dataset as well as present our evaluation metric for recommendation accuracy, and the results of our study based on these metrics. We empirically compare the recommender quality of proposed SSCF to CF. In addition, we look into the impact of occurrence order in the session and the window size.

A. Experiment Setting

1) Datasets

a) Data source

We use usage log data achieved from Bugs Music service for the experimental evaluation. Bugs Music is one of the largest commercial online music services in Korea. This data is collected for a month from April 1st, 2008 to April 28th, 2009. For this experiment, we split the dataset S

into two non-overlapping sets: a training set and a testing set. This split is depending on the service time. A month of records for training data from April 1st to April 27th and a randomly chosen 140 sessions from the next day of the month as test set are generated.

b) Data Preprocessing

The recommender references training data to generate recommendation and the performance is measured on test sessions. We set the experimental settings with an intention to make similar situations in the industries when they use recommendation techniques. Each log record in the log table contains user identification, song identification, timestamp, length of service in seconds.

We removed the log records in which the length listened is less than 80% of whole length. We consider when a user listens to just a small portion of the whole track, that user did not want the music. Thus, by removing these logs, we ignore those data when calculating the rating of items.

After data preprocessing, the training data set contains a total of 8,239,977 logs and 199,223 sessions. The detailed statistical information of training dataset is described in Table 1. We evaluated it with 140 sessions on this training dataset.

TABLE I. STATISTICAL INFORMATION OF DATASETS

Description	Number of Records
Number of logs	8,239,977
Number of items	170,069
Number of sessions	199,223
Number of users	17,881
Average number of items in a session	25.213
Average length of a session	41.361
Average number of items a user has rated	109.402

c) Session Generation

For segmenting sessions, we consider the elapsed time. There can be various sophisticated method for slicing sessions. But as [7] follows the time-based reconstruction heuristics which is heavily studied on [5], we focus on the fact that if a long time elapses between one request and the next, it is most likely that the latter request is the start of a new session. So we cut off a sequence if no songs are played for more than 30 minutes. Every session is constructed by a user and consisted of sequentially played songs.

2) Experimental Setup

Our evaluation methodologies are as follows. The *w* items starting from *l*th item in a test session are used for generating recommendations, and the next item of the seen items in the test session, i.e. *(w+l+1)th* item, is used to evaluate the generated recommendations. The value *w* reflects the window size and *l* the item start position of window in the session.

3) Evaluation Measure and Parameters

Even though evaluation measures such as MAE, RMSE, and recommendation list precision are popular for evaluating CF-based recommender systems, it is not plausible in our scenario in that we only have an item for each situation. So we believed that Hit Ratio is more suitable for measuring the performance of recommender system of predicting next item since we only focus on finding the one right item. Thus, we use Hit Ratio to evaluate the performance of our method and see the effect of parameters. We adopt HR@*n*, which is proposed in [10] for measuring how many real usages of each recommendation were found within top-*n* recommended items. We believe that Hit Ratio is most suitable measure for the performance of recommender system when predicting next item. It is due to the fact that we only need to focus on finding the one right item. The specific evaluation method is that we first generate a list of songs using items in the test session, and check if the recommending list contains the song found in the log. We find how many real usages of each recommendation were found within top-n recommended items.

The parameters for the experiments are shown in table II. The number of similar neighbor *k* means the constraint number of similar objects that is found using cosine similarity. In case of normal CF, a similar object is a similar user, but in the case of SSCF, it is a similar session. Since we cannot consider all of the objects, we find the most similar neighbors at most the number *i*. The other two parameter w and *l* are about what items we should consider when we generate recommendations. We could see all the items that user has selected, but there may be a more relevance on the items that is temporally close to the current item. Thus, we set the parameter *w* and *l*, which indicate the number of items we reference for recommendation and starting position of window, respectively. Detailed information and the meaning of these parameters are explained in V.A.2

TABLE II. PARAMETES USED IN EVALUATION

Parameters		Value
Similar neighbor # (k)	CF	30,50,100,200,300,400, 500,600
	SSCF	300,500,1000,2000
Number of items used for generating recommendation(window size,w)		1,2,3,5,10
Number of Recommending items (n)		1,2,3
Starting position of item of window(l)		1,2,3,4,5,6,7,8,9,10

4) Baseline Method

We have used the traditional user-item approach as a baseline. When there are only currently selected items, CF will use those items to make unknown user's profile and

find the similar users from the unknown user to make the recommendation.

$$p_{i,a} = \sum_{j=1}^{topk} r_{i,j} \cdot sim(u_a, u_j) \quad (5)$$

We use the cosine similarity for measuring similarity between users as well.

B. *Experimental Result*

Now we discuss the experimental results. To summarize, SSCF outperforms CF when it predicts the next item with items appeared in advance, and it is desirable to include items appeared in the front of the session with appropriate window size, which is 3 items.

1) *Overall accuracy comparison*

For the accuracy comparison, table 3 shows the Hit ratios of CF and SSCF on Bugs Music data. Results show that the optimal number of neighbor has is 50 for CF and 300 for SSCF.

In this dataset, there is a gain of 65% over CF from SSCF on test set 1, and 71% on test set 2 in average. For the optimal setting for each method, SSCF shows 50% improvement on test set 1 and 54 % improvement on test set 2, compared to CF.

TABLE III. THE ACCURACY COMPARISON FOR SSCF AND CF

Methods	Neighbor Num	Hit Ratio	
		Test set 1	Test set 2
CF	30	7.14	6.3
	50	7.65	6.62
	100	6.71	**7.5**
	200	8.23	6.34
	300	8.07	6.5
	400	**8.81**	6.38
	500	8.45	6.23
	600	6	4.39
	average	**7.63**	**6.28**
	max	**8.81**	**7.5**
SSCF	300	**13.23**	**11.56**
	500	12.61	10.74
	1000	13.03	10.72
	2000	11.54	10.11
	average	**12.6**	**10.78**
	max	**13.23**	**11.56**

2) *StartItemNumber*

The aim of this experiment is address the influences of the item position in the session. By adjusting which position of item to start considering to generate recommendation, we desire to see the impact of the position item appears in the session.

Figure 3 depicts the impact of item position on SSCF and CF. Figure 3 shows the impact of starting position of window on Hit Ratio. They show that Hit Ratio decreases as starting position moves forward. Especially, as we see a sharp drop at the beginning, the very first item has the highest importance and the second item moderate but still strong importance to the performance. But after 3 items, they start to show similar importance until it starts to fall a little more. We can say that in general, to get more accurate recommendations, the items appeared in the beginning of a session need to be considered with more importance when generating recommendation, and especially first a few items.

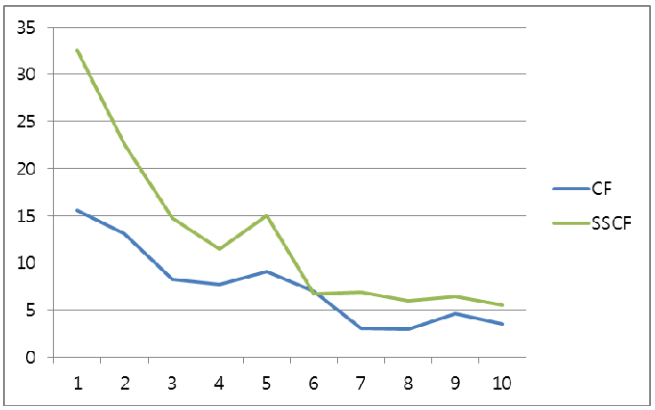

Figure 3 The impact of starting position of a window

3) *Window size*

Window means the range of items we consider to generate the recommendation, and the size of window indicates the number of items in the range. In this experiment, we are interested in analyzing how size of the window affects the performance of recommendation beyond the kinds of methods. In order to see the impact of the window size on recommendation accuracy, we did experiments on both methods, and results are shown in Figure 4.

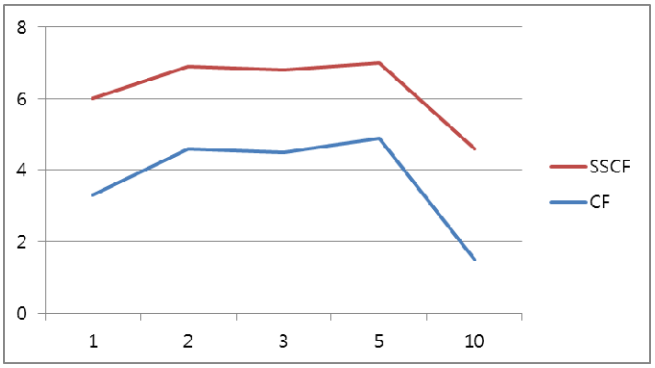

Figure 4 The impact of window size

Figure 4 depicts the influence of window size when the length of the recommendation list is 1 and the optimal neighbor number, 50 and 300, for each method respectively. The results show us that the optimal length of window is 3. This result indicates that it is generally desirable to use 3 current items appeared in advance for recommending the next item for both CF and SSCF. As expected, including too many current items becomes a noise for recommendation.

VI. DISCUSSION

Typical CF formula considers the fact that users may rate item in different basis. For example, Michael tend to give generate score to items whereas Bob gives stingy score to items. Thus, the system subtracts each user's average ratings from individual ratings made by the user. However, as you may have already noticed, we use CF formula that does not consider different rating average of different users. We conducted experiments on the formula considering different users' rating level as well, but for both methods, CF and SSCF, it showed poor performances than using the formula we used in this paper. Even though the formula considering different rating style reflects the conceptual assumptions behind the CF more accurately, it may be better to use formula that does not consider different rating style, especially when we use usage log data for extracting preference instead of using explicit ratings. Since we make a virtual profile of users and sessions by counting listening events, the high average score of items does not mean a bias depending on user characteristic anymore, but the significance of the profile.

VII. CONCLUSION

In this study, we have discussed different methods and parameters that affect the predictions on the next item a user requests. We also proposed a Session-based CF(SSCF) which makes currently selected items as a part of a session profile and finds the most similar sessions in the training set to generate recommendation. In this experiment, we compared the results obtained from two types of recommendation techniques, CF and SSCF, and showed that SSCF outperforms the basic CF. We have also compared different settings using the SSCF approach in search of the optimal window size and impact of the item occurrence position in session.

The current proposal leaves another future task of reducing query time for the recommendation by reducing training dataset with the least performance costs. Moreover, we desire to compose a manual, including practical scenarios and guidelines for music recommendation in stream by predicting the most suitable next songs. For example, a user can give the feedback by either skipping or listening to the music files indicating the user's likes and dislikes. Subsequently, the system can instantly interpret the listener's feedback and respond with a certain music files to be recommended using SSCF.

ACKNOWLEDGEMENT

This research was supported by Seoul R&BD Program (WR080951).

REFERENCES

[1] B. Logan. "Content-based playlist generation: Exploratory experiments". In Proc ISMIR., Paris, 2002

[2] E. Pampalk, T.Pohle, and G.Widmer, "Dynamic Playlist Generation based on Skipping Behavior" .In Proc. ISMIR Conference,2005

[3] C. Baccigalupo, and E. Plaza,"Case-Based Sequential Ordering of Songs for Plalist Recommendation". In Proc. European Conference on Case-based Reasoning(ECCBR06), LNAI,2006, pp. 286-300

[4] J-J. Aucouturier and F. Pachet. "Scaling up music playlist generation." In Proc. of IEEE International Conference on Multimedia and Expo (ICME). Lausanne, Switzerland, August 2002.

[5] M.Siliopoulou, B.Mobasher, B.Berendt, and M.Nakagawa, "A Framework for the Evaluation of Session Reconstruction Heuristics in Web-Usage Anaysist", In Proc. Informs Journal on Computing (INFORMS), pp.171-190, 2003 .

[6] X. Fu, J. Budzik, and K. J. Hammond. Mining Navigation History for Recommendation. In Proc. of the 5th International Conference on Intelligent User Interfaces (IUI 2000), pages 106–112, 2000.

[7] J. Borges and M. Levene, "Mining Users' Web Navigation Patterns and Predicting Their Next Step", In Proc. Security Informatics and Terrorism:Patrolling the Web. Volume 15., 2008

[8] S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme, "Factorizing personalized markov chains for next-basket recommendation," In Proc. of the 19th international conference on World Wide Web(WWW). New York, NY, USA: ACM, 2010, pp. 811–820.

[9] J. Davidson, B. Liebald, J. Liu, P. Nandy, T. Van Vleet, U . Gargi, S . Gupta, Y . He, M . Lambert, B. Livingston, D . Sampath, " The youtube video recommendation system." In Proc. of the fourth ACM conference on recommender systems(Recsys) ACM, 2010, pp 293–296

[10] Lee, S. E. Park, M. Kahng, S. Lee, and S. goo Lee,"Exploiting contextual information from event logs for personalized recommendation," in Proc. 9th IEEE/ACIS International Conference on Computer and Information Science, Studies in Computational Intelligence(ICIS),Springer,2010.