19th International Computer Science and Engineering Conference (ICSEC)
Chiang Mai, Thailand, 23-26 November, 2015

Incremental Session Based Collaborative Filtering with Forgetting Mechanisms

Ureerat Suksawatchon*, Sumet Darapisut † and Jakkarin Suksawatchon‡
Faculty of Informatics, Burapha University
Chonburi, 20131 Thailand
*ureerat.w@gmail.com, †dearsumet@gmail.com, ‡jakkarin@informatics.buu.ac.th

***Abstract*—Most of research works in music recommendation systems use Collaborative Filtering (CF) for generating personalized recommendations based on user's previous song ratings or static usage history data. But those researches adapting CF do not consider behavior of listening to songs and are not able to maintain the systems to sensitive to recent user's preferences. Behavior of music listening is continuous and repetitive process, especially, the latest song listening can infer to the favorite song at that moment. In this work, we present *Incremental Session based Collaborative Filtering with forgetting mechanism* or *ISSCF* by adapting Session-based Collaborative Filtering (SSCF), which considers music listened continuously and maintains the recent session. In order to avoid unnecessary memory usage and processing time, we use forgetting mechanism: sliding windows and fading factors incorporating with SSCF. We evaluate our purposed framework by measuring the $HitRatio$. From experimental results, it shows that performance of our purposed approach increases the accuracy of recommendation and low computational time and space when comparing with than SSCF.**

Keywords*—*music recommendation; forgetting mechanism; session; collaborative filtering

I. Introduction

Music Recommender System (MRS) has an important role to help the users to find songs that they really want from a large amount of songs. In provider aspect, MRS is able to filter and to choose appropriated music for the users. Most of MRS uses Collaborative Filtering algorithm (CF) for generating personalized recommended songs by considering users behaviors (such as rating, clicks, history and purchase, etc.) [1], [2], [3], [4]. However, CF requires many users and many ratings and is unable to recommend songs that have a few ratings. This means that users have to well provide their taste if they need effective recommendation.

Characteristic of music domain has different from other domains like movies, books and news. Listening to music is continuous and repetitive process. Especially, users tend to prefer to listen to preference songs repetitively in *session* rather than isolated [4], [5]. Accordance with S. E. Park and et al.[5] tries to capture order and repetitiveness in the playing songs. They proposed Session-based Collaborative Filtering approach (SSCF) for next song prediction with the currently played songs in Bugs Music dataset. SSCF adapt collaborative filtering based on user by taking into account relation of *session profiles* instead of user profiles. The experiments show that SSCF outperforms than traditional collaborative filtering in term of accuracy.

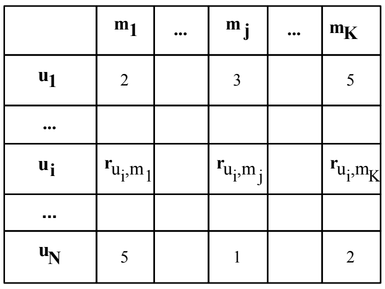
R. Dias and M. Fonseca [6] presented improving music recommendation approach named Temporal Session based Collaborative Filtering approach (TSSCF). This work extracts temporal context including time of day, weekday, a day of month, a month from session profiles, and takes into account the song diversity played in the session. After that, the TSSCF groups sessions according to different of temporal context by using Gaussian Mixture Model via Expectation Maximization algorithm. Finally, the part of recommendation approach is applied with SSCF. Comparing with the traditional session-based CF, the TSSCF can achieve better accuracy values.

However, SSCF and TSSCF approaches are still based on traditional user-based CF. It uses the fully static listening history of users to perform recommendation and requires very expensive computational time and space with the growth of the number of users and music in a database. Thus, both SSCF and TSSCF approaches are not appropriate for on-line manner because the on-line music service always increases new users and new songs. These two algorithms are faced with the scalability problem. This causes the system to become less predictive ability. In order to overcome this problem, we introduce *Incremental Session based Collaborative Filtering with forgetting mechanisms* or *ISSCF* ,in short; by modifications in SSCF [5]. Our approach is capable to accurately recommend the next songs for the active session by considering past sessions in on-line manner. Because of increasing new users and new songs, our approach uses forgetting mechanisms to handle old and obsolete data, and maintain the MRS concerning to recent data. It is possible to reduce memory usage and processing time as well. In this paper, we evaluate the efficiency of two *forgetting mechanisms* – sliding windows and fading factors. In our experiments, we evaluate the accuracy of our purposed algorithm with the $HitRatio$ (HR@n)[5], [6], and also evaluate the time-consuming of our model by comparing with SSCF. The results are shown that our purposed algorithm outperforms in terms of accuracy and computational time.

II. Background Knowledge

A. Collaborative Filtering Algorithm (CF)

Collaborative filtering (CF) is the well-known personalized recommendation technique that widely used in recommender system. The basic idea of CF is to help users to find the items they would like to purchase based on rating of those items by other users with similar taste. CF produces a prediction score or top-N recommendation list of items for an active user. More formally, there are a set of users $U = \{u_1, u_2, \ldots, u_i, \ldots, u_N\}$ and a set of items $M = \{m_1, m_2, \ldots, m_j, \ldots, m_K\}$. Each user has rated a subset of items such as movie, music, book and etc. All available ratings (r_{u_i,m_j}) are collected in user-item rating matrix denoted R matrix as illustrated in Fig 1. In the first step, it finds similarity between active users u_i and users u_v having co-rated as $M_i \cap M_v$. The popular similarity measures are cosine similarity and Pearson correlation similarity as given in (1) and (2). Next, the k most similar users are selected as the k-nearest neighbors of active user. Then, CF calculates the prediction rating (p_{u_i,m_j}) that active user (u_i) would probably prefer item (m_j) based on his/her neighbors using (3). Finally, the top-N recommendation list is generated based on highest prediction scores [2], [7].



	m_1	...	m_j	...	m_K
u_1	2		3		5
...					
u_i	r_{u_i,m_1}		r_{u_i,m_j}		r_{u_i,m_K}
...					
u_N	5		1		2

Fig. 1. The example of user-song rating matrix

Cosine similarity :

$$sim(u_i, u_v) = \frac{\sum_{j \in M_i \cap M_v} (r_{u_i,m_j}, r_{u_v,m_j})}{\sqrt{\sum_{j \in M_i \cap M_v} (r_{u_i,m_j})^2} \sqrt{\sum_{j \in M_i \cap M_v} (r_{u_v,m_j})^2}} \tag{1}$$

Pearson correlation similarity :

$$sim(u_i, u_v) = \frac{\sum_{j \in M_i \cap M_v} \left(r_{u_i,m_j} - \overline{r}_{u_i,\cdot}\right)\left(r_{u_v,m_j} - \overline{r}_{u_v,\cdot}\right)}{\sqrt{\sum_{j \in M_i \cap M_v} \left(r_{u_i,m_j} - \overline{r}_{u_i,\cdot}\right)^2} \sqrt{\sum_{j \in M_i \cap M_v} \left(r_{u_v,m_j} - \overline{r}_{u_v,\cdot}\right)^2}} \tag{2}$$

$$p_{u_i,m_j} = \overline{r}_{u_i,\cdot} + \frac{\sum_{u_v \in U} sim(u_i, u_v) \left[r_{u_v,m_j} - \overline{r}_{u_v,\cdot}\right]}{\sum_{u_v \in U} |sim(u_i, u_v)|} \tag{3}$$

B. Incremental Collaborative Filtering Algorithms

M. Papagelis and et al. [8] presented Incremental Collaborative Filtering (ICF) for handling scalability problem. ICF is based on incremental updates of the user-user similarities. When active user submits a new rating or updates existing rating then similarity between active user and the rest of need to be recalculated in relation to the old similarity values. ICF approach illustrates that it can reduce computation complexity from polynomial time to linear time that gives higher potential than classic CF. C. Miranda and A. M. Jorge [9] proposed the incremental version of item-based CF for binary ratings that regards recommendation approach based on item instead of user. This approach shows that it uses less computational cost and gives predictive accuracy more than user-based CF. In addition, X. Yang and et al. [10] developed scalable item-based collaborative filtering. To deal with scalability problem, incremental update of item-to-item similarity is proposed. In rating prediction process, local link prediction in item similarity graph is used to find implicit neighbor candidates. The experimental results validate that this approach can increase the efficiency in recommendation. Besides, CF should be able to efficiently process data online in order to keep the system up-to-date [11]. J. Vinagre and A. M. Jorge [11] proposed incremental collaborative filtering with forgetting mechanisms approach that maintains recent preference of user. It decreases older importance information with sliding windows and fading factors approaches. The experimental results show that this approach is able to reduce processing time and memory while not significant reducing predictive potentiality of the algorithm.

Although all of these algorithms mentioned before are designed to handle the scalability problem, actually it cannot improve the accuracy of the recommendation system. Because these algorithms consider only user-rating matrix or item-rating matrix and do not take into account behavior of listening to songs or characteristics of songs as the additional information. Moreover, listening to music is continuous *session* and repetitive process [4], [5], [6]. All of those algorithms do not concern about listening behavior. This will lead to increase the accuracy of the recommendation systems.

III. Our approach

The framework of ISSCF system (our proposed system) is described in Fig. 2.

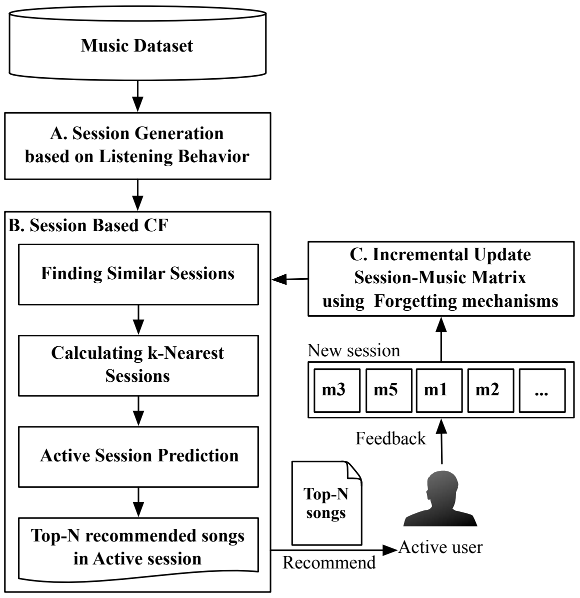

Music Dataset
A. Session Generation based on Listening Behavior
B. Session Based CF
Finding Similar Sessions
Calculating k-Nearest Sessions
Active Session Prediction
Top-N recommended songs in Active session
C. Incremental Update Session-Music Matrix using Forgetting mechanisms
New session
m3
m5
m1
m2
...
Feedback
Top-N songs
Recommend
Active user

Fig. 2. Overall our approach

A. Session Generation based on Listening Behavior

In this work, we use listening dataset from last.fm[1] database as shown in Fig. 3 that collected during 2005 to May 2009. When a user listens to song, one log in database is generated. Last.fm dataset contains 6,741,330 listening logs obtained from 582 users and 613,117 songs. Each user is analyzed the song diversity by using diversity measuring as shown in (4). This song diversity measures the ratio of different songs played in a session and the total songs played [6]. If song diversity value is closed to 1, it can inform that a user played different songs. Otherwise, this value can inform that a user played repeatedly the same songs. In this dataset, the average of song diversity is 0.21. This means that most of users listened to music repeatedly and continuously.

User id	Timestamp	Artist id	Artist	Music id	Music id
user1	2009-01-01 T09:00:10Z	art1	Don Moy	m1	Moy Or Less
user1	2009-01-01 T09:04:40Z	art1	Don Moy	m2	Contest a Moy
user1	2009-01-01 T09:07:47Z	art3	Pogo	m3	Alice
...	...	...	...	...	...

Fig. 3. The example of listening log data obtained from Last.fm

$$Song\ diversity = \frac{\#Different\ songs}{\#All\ songs} \qquad (4)$$

Since our framework is a modification of SSCF (Algorithm 1) [5] designed for on-line handling of on-line music services. First, we define a *session* as the group of songs listened by a user from the moment he/she starts playing songs to the moment they stop it [5]. This work uses continuous time gap of stopping playing songs more than 30 minutes to define as a session. In initial process, we create 100 sessions for a single user to avoid cold start problem as depicted in Fig. 4. Sessions containing less than 2 songs are removed [5], [6].

User id	Timestamp	Artist id	Artist	Music id	Music id
user1	2009-01-01T 09:00:10Z	art1	Don Moy	m1	Moy Or Less
user1	2009-01-01T 09:04:40Z	art1	Don Moy	m2	Contest a Moy
user1	2009-01-01T 09:07:47Z	art3	Pogo	m3	Alice
...	...	...	...	...	...

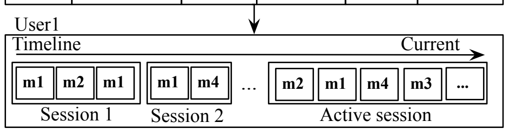


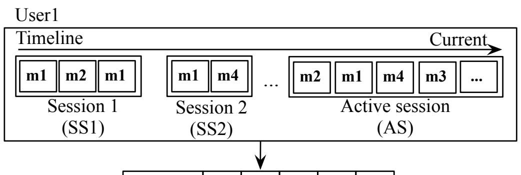
Fig. 4. The example of session generation

Algorithm 1 Traditional SSCF

```
1:  Input : $D, N, k$
2:  Output : Top $N$ recommended songs
3:  Create session-music matrix $Met$ with set of sessions $ss$ $\epsilon$ music datasets $D$
4:  Calculate similarity matrix with $Met$ by using cosine similarity
5:  For each session $ss_u$ :
6:      For each session $ss_v$ :
7:          For each song $m_j$ by $j = M_{ss_u} \cap M_{ss_v}$ :
8:              $sim(ss_u, ss_v) = \frac{\sum_j (r_{ss_u,m_j}, r_{ss_v,m_j})}{\sqrt{\sum_j (r_{ss_u,m_j})^2}\sqrt{\sum_j (r_{ss_v,m_j})^2}}$
9:  Finding $k$ similar session ($S^k$)
10: Prediction score of song $m_j$
11: For each song $m_j$ with active session $as$ :
12:     $p_{as,m_j} = \overline{r}_{as} + \frac{\sum_{ss_v \in S^k} sim(as,ss_v)[r_{ss_v,m_j} - \overline{r}_{ss_v}]}{\sum_{ss_v \in S^k} |sim(as,ss_v)|}$
13: Recommend the top-$N$ songs ($N_{rec}$) based on highest prediction scores
```

B. Session Based CF

Since this work considers session profiles instead of user profiles for generating recommendation, Session-Music matrix is generated for each user as illustrated in Fig. 5. Based on this data matrix rows represent session profiles, and the columns represent songs. Each cell contains the frequency of songs played in that session. In this process, we concentrate on the prediction of next appropriated songs that user requests in the active session (current session). Thus, our approach ranks all candidate songs and recommend top-n songs that are likely to come after songs that are listening to.



	m1	m2	m3	m4	...
ss1	2	1	-	-	...
ss2	1	-	-	1	...
ss3	1	2	4	3	...
...	...	...	...	...	...
ss100	1	-	1	1	...

Fig. 5. The example of Session-Music Matrix for a single user

The first step of this process, we find similar sessions by measuring the similarity between an active session (as) and other sessions (ss_v). To measure the similarity between active session (as) and other sessions (ss_v) denoted as $sim\,(as, ss_v)$, we use cosine similarity [5] as shown in (5), where rating (r_{as,m_j}) is the frequency of listening to song (m_j) in active session (as) and rating (r_{ss_v,m_j}) is the frequency of listen to song (m_j) in the session ss_v.

$$sim\,(as, ss_v) = \frac{\sum_j (r_{as,m_j}, r_{ss_v,m_j})}{\sqrt{\sum_j (r_{as,m_j})^2}\sqrt{\sum_j (r_{ss_v,m_j})^2}} \qquad (5)$$

[1]http://www.dtic.upf.edu/ ocelma/MusicRecommendationDataset/

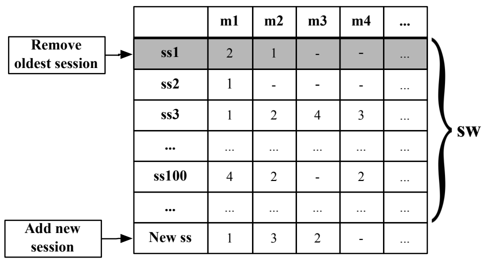
Next, we calculate the k-nearest neighbors is applied to find other sessions which are k most similarity to the active session. Then, the prediction score of song m_j in active session as, defined as (p_{as,m_j}), is determined with k session neighbors (S^k) as shown in (6). Finally, all candidate songs are ranked and recommend top-n songs playing in the active session.

$$p_{as,m_j} = \overline{r}_{as} + \frac{\sum_{ss_v \in S^k} sim\left(as, ss_v\right)\left[r_{ss_v,m_j} - \overline{r}_{ss_v}\right]}{\sum_{ss_v \in S^k} \left|sim\left(as, ss_v\right)\right|} \quad (6)$$

C. Incremental Update Session using Forgetting Mechanisms

Whenever a user listened to songs from the moment he/she starts playing songs to the moment he/she stops it, and the stopping time of playing songs is more than 30 minutes, a new session emerges as to be feedback from a user. Incremental process has been presented, where our approach incrementally updates Session-Music matrix every time new session is available. In order to maintain recent preference of user, it should be decreased older and obsolete sessions. This work uses forgetting mechanisms to re-update Session-Music matrix. Forgetting mechanisms used in this work are sliding windows and fading factors approaches.

1) Sliding windows: The sliding windows approach is performed using a sequence-based sliding window of size sw that holds information about sw most recent sessions in type of first-in-first-out (FIFO) data structure[11]. In this approach, we process by fixed size of most recent sessions. When an incoming new session is added and sw is reached to fixed window size, then oldest session of the user is discarded as shown in Fig. 6. Then, the similarity value corresponding to songs in new session are updated, while other values are kept. Algorithm ISSCF with sliding windows is shown in Algorithm 2.



	m1	m2	m3	m4	...
ss1	2	1	-	-	...
ss2	1	-	-	-	...
ss3	1	2	4	3	...
...	...	...	...	...	...
ss100	4	2	-	2	...
...	...	...	...	...	...
New ss	1	3	2	-	...

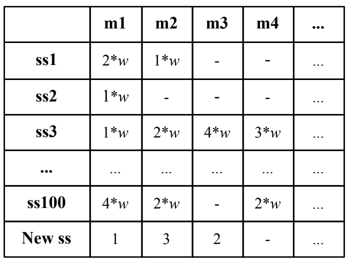
Fig. 6. Sliding windows approach.

2) Fading factors: Since sliding windows provide an effective but abrupt way to forget older data. However, in many cases previous data may contain valuable information and is not necessarily discarded [11]. Fading factors provide another way to gradually forget past data. The fading factors approach is gradual forgetting or full memory approach [12] that uses all sessions of matrix, but decreases importance of old session with small weight as shown in Fig. 7. This approach can be implemented by multiplying the elements of matrix for each session by a factor w using formula (7). Algorithm ISSCF with fading factors is shown in Algorithm 3.

$$w = e^{-\alpha t} \quad (7)$$

Algorithm 2 ISSCF with sliding windows

```
1:  **Input :** $D, N, k, sw$
2:  **Output :** Top $N$ recommended songs
3:  Initialize session-music matrix $Met$ with set of session
    $S$=$\{ss_1, ..., ss_{100}\}$ by $S$ $\epsilon$ music datasets $D$
4:  *For each new active session as* $\epsilon$ *D*:
5:      Hidden last song ($m_{hide}$) of $as$ as test data
6:      Update matrix $Met$ with $as$
7:      *If* length($Met$) > size of window $sw$ *then*
8:          Remove oldest session ($Met_{ss_1}$)
9:      Update session and music (row/column) of $Met$
10:     (Re)calculate similarity matrix with $Met$ by using
        cosine similarity
11:     *For each session* $ss_v$ *with as in Met :*
12:         *For each song* $m_j$ *by* $j = M_{as} \cap M_{ss_v}$ *:*
13:             Calculate equation (5)
14:     Finding $k$ similar session ($S^k$)
15:     Prediction score of song $m_j$
16:     *For each song* $m_j$ *with active session as :*
17:             Calculate equation (6)
18:     Recommend the top-$N$ songs ($N_{rec}$) based on highest
        prediction scores
```



	m1	m2	m3	m4	...
ss1	2*w	1*w	-	-	...
ss2	1*w	-	-	-	...
ss3	1*w	2*w	4*w	3*w	...
...	...	...	...	...	...
ss100	4*w	2*w	-	2*w	...
New ss	1	3	2	-	...

Fig. 7. Fading factors approach

where, w : weight value
α : controlling factor to define how fast the weights decrease
t : session order

IV. Experimental Evaluation

A. Experimental setup

In the experimental process, we would like to know that whether our framework could predict the next song to be played in the current active session based on what user has previously listened in that session. This work removes the last songs to be played in the queried current active session as testing datum [6] as shown in Fig 8. One important test is how to deal with a large amount of data and perform in on-line manner. We consider user sessions as a data stream. From the analysis of last.fm data, there are 564 sessions. These sessions are divided into 100 sessions to be used as initial sessions and 464 sessions to be used as the queried active sessions (testing data) that will continuously feed into system. Then, testing data is executed by our framework (ISSCF) and SSCF algorithm [5] to obtain the top-10 recommendations.

The following parameters are set to conduct the tests. For sliding windows, we set window size (sw) in form of the number of sessions at 200 and 400 latest sessions. For fading factors, we set the weight values (alpha) of exponential time

Algorithm 3 ISSCF with fading factors

```
1:  Input : $D, N, k, \alpha$
2:  Output : Top $N$ recommended songs
3:  Initialize session-music matrix $Met$ with set of session
    $S$={$ss_1, ..., ss_{100}$} by $S \epsilon$ music datasets $D$
4:  For each new active session as $\epsilon$ D
5:      Hidden last song ($m_{hide}$) of $as$ as test data
6:      Update matrix $Met$ with $as$
7:      Assigns weight ($w$) with factor $\alpha$, session order ($t$) :
8:          Calculate equation (7)
9:      (Re)calculate similarity matrix with $Met$ by using
        cosine similarity
10:     For each session $ss_v$ with as in $Met$ :
11:         For each song $m_j$ by $j = M_{as \cap} M_{ss_v}$ :
12:             Calculate equation (5)
13:     Finding $k$ similar session ($S^k$)
14:     Prediction score of song $m_j$
15:     For each song $m_j$ with active session as :
16:         Calculate equation (6)
17:     Recommend the top-$N$ songs ($N_{rec}$) based on highest
        prediction scores
```

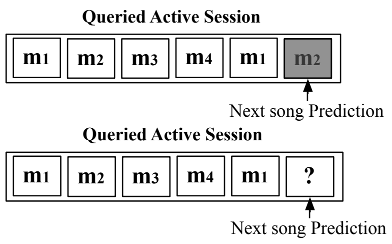


Fig. 8. Evaluation approach in the next song prediction

decay function as controlling how fast the weights decrease at 0.1 and 0.01. The both ISSCF and SSCF approaches use cosine similarity and 30 best neighbor sessions as suggested in [6]. In addition, this work has tested the computational time of ISSCF and SSCF approaches in finding similarity process and prediction process at 10 runs and 200 of the consecutive queried active sessions.

B. Evaluation Metric

In evaluation process, we measure the accuracy of our framework and other methods using $HitRatio$ ($HR@n$) [5]. $HR@n$ indicates that whether the desired songs appear on the top-n recommendation lists for a single user in the queried active session, and how many times they appear [6]. $HR@n$ metric is described in formula (8).

$$HR@n = \frac{\#hit}{k} \tag{8}$$

The average $HR@n$ for one session in all users can be calculated as shown in (9).

$$HitRatio = \frac{\sum_{i=1}^{N} HR@n}{N} \tag{9}$$

Where, hit : if music is listened in the next song appear on recommendation list, so hit is 1 otherwise is 0.

k : the number of hidden songs
N : the number of users

C. Results and Discussions

The result as depicted in Fig 9 shows the $HitRatio$ of ISSCF with sliding window at $sw = 200$ and $sw = 400$ and compared with SSCF for each queried active session of all users. From experimental results, ISSCF with sliding window at $sw = 200$ got the average $HitRatio$ at 0.085 and at $sw = 400$ got the average $HitRatio$ at 0.089. For SSCF approach, the average $HitRatio$ is 0.088. Fig 10 shows the $HitRatio$ ISSCF with fading factors in each queried active session compared with SSCF. The average of $HitRatio$ at $\alpha = 0.01$ is 0.11 and $\alpha = 0.1$ is 0.1146. We have conducted Wilcoxon Tests at $p - value = 0.1$ between ISSCF and SSCF for 50 queried active sessions. We conclude that there was no statistically significant difference between ISSCF with sliding windows and SSCF. Although there was no difference between ISSCF and SSCF, but ISSCF can reduce the computational time that will discuss next. Comparing between ISSCF with fading factors and SSCF, there was a statistically significant. It is clear that ISSCF with fading factors shows a good accuracy than SSCF.

Fig. 11 shows the best result in each algorithms. ISSCF with fading factors at $\alpha = 0.1$ still gives the best average $HitRatio$ than ISSCF with sliding windows and SSCF. We can conclude that ISSCF with fading factor is capable to increase significantly the accuracy of the recommendations.

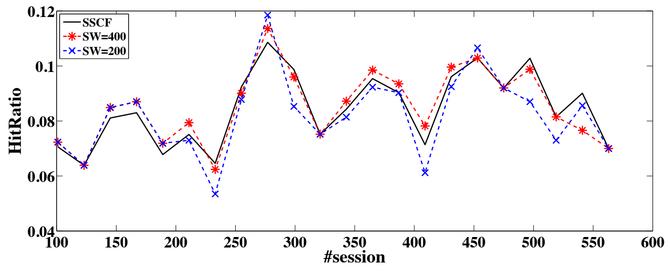


Fig. 9. The $HitRatio$ of ISSCF with sliding windows compared with SSCF

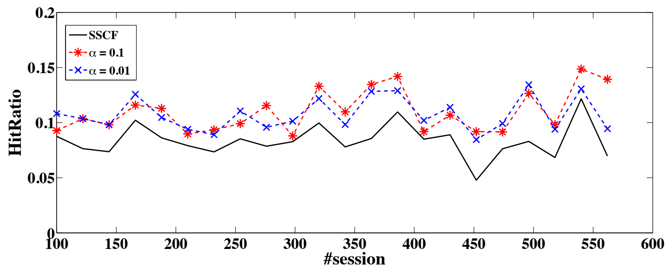


Fig. 10. The $HitRatio$ of ISSCF with fading factors compare with SSCF

This work also evaluated the computational efficient of ISSCF with SSCF. Since SSCF is the algorithm based CF, so it uses the quadratic time, especially, for finding similarity between whole sessions and songs in offline process and uses more time for prediction process. Table I shows the execution time used in those three algorithms at 200 sessions. Time was measured on a PC with CPU speed at 3.20 GHz, 8 GB RAM,

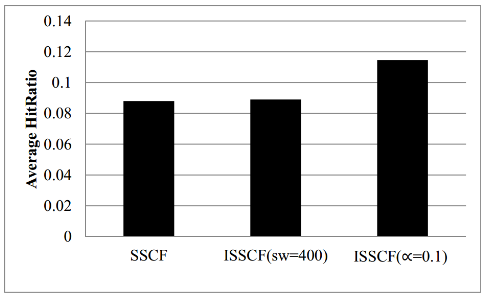


Fig. 11. The average of $HitRatio$ value of SSCF, ISSCF with $sw = 400$, and ISSCF with $\alpha = 0.1$ for all users in all sessions

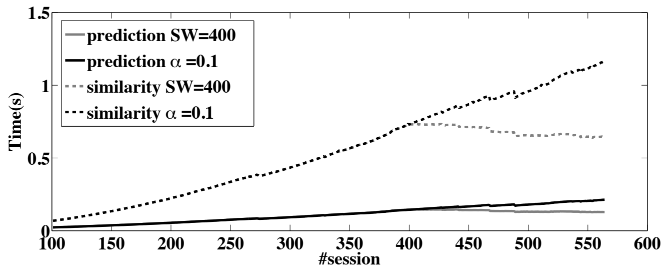


Fig. 12. The execution time of finding session similarity and active session prediction processes of ISSCF

and all algorithms were implemented in Python. The traditional SSCF approach uses $O(m^2n)$ to find similarity (Algorithm 1 lines 5-8) and if we know the session of active user, so time of prediction is $O(n)$ (Algorithm 1 lines 11-12).

For ISSCF approach, it has to find similarity sessions when latest session of active user is updated. This process uses computational time that is $O(mn)$ (Algorithm 2 lines 11-13 and Algorithm 3 lines 10-12), where m is the number of sessions and n is the number of songs in that session. In process of active session prediction, it uses $O(n)$ (Algorithm 2 lines 16-17 and Algorithm 3 lines 15-16). That is appropriate for recommendation systems and helps computational reduction. According to Fig. 12 shows the growth of execution times of finding session similarity process and active session prediction process of ISSCF. The ISSCF with sliding windows presents very efficiency over ISSCF with fading factors approach, hence ISSCF with sliding windows uses the fixed size of windows (sessions). On the other hand, ISSCF with fading factors approach does not discard data. It will affect to computational time that grows with number of sessions in linear time.

TABLE I. EXECUTION TIME OF TWO APPROACHES

Approach	Similarity	Prediciton
Offline SSCF	10.16 sec.	0.02 sec.
ISSCF	**0.09 sec.**	**0.02 sec.**

V. CONCLUSION

Since Session-based Collaborative Filtering (SSCF) requires expensive computations that grow polynomially because of increasing in the number of users and songs, and SSCF is not capable to process data in on-line manner in order to maintain the system up-to-date. Therefore, this paper proposes Incremental Session based Collaborative Filtering with Forgetting Mechanism called ISSCF. It is a new framework that modified in tradition SSCF incorporated with forgetting mechanism. Our approach is suitable for the music domain by taking into account the users listen to songs continuously and repetition in a session. Thus, our purpose is to improve the accurately recommendation for the next songs in active session and to overcome the scalability problem in on-line manner. For supporting on-line process, we adapt incremental algorithm to SSCF by using forgetting mechanisms — sliding windows and fading factors — to deal with old and obsolete data. In our experiments, we evaluate the accuracy of our purposed algorithm compared with SSCF by using $HitRatio$ (HR@n) metric [5], [6], and also evaluate the time-consuming of our model comparing with SSCF. The results are shown that our purposed framework outperforms in terms of accuracy and computational time.

ACKNOWLEDGEMENTS

This research is supported by Faculty of Informatics, Burapha University.

REFERENCES

[1] Y. Shi, M. Larson, and A. Hanjalic, "Collaborative filtering beyond the user-item matrix: A survey of the state of the art and future challenges," *ACM Comput. Surv.*, vol. 47, no. 1, pp. 3:1–3:45, May 2014.

[2] M. Kaminskas and F. Ricci, "Contextual music information retrieval and recommendation: State of the art and challenges," *Computer Science Review*, vol. 6, no. 23, pp. 89 – 119, 2012.

[3] Y. Song, S. Dixon, and M. Pearce, "A survey of music recommendation systems and future perspectives," pp. 395–410, Jul. 2012.

[4] C. H. Park and M. Kahng, "Temporal dynamics in music listening behavior: A case study of online music service," in *Computer and Information Science (ICIS), 2010 IEEE/ACIS 9th International Conference on*, Aug 2010, pp. 573–578.

[5] S. E. Park, S. Lee, and S. goo Lee, "Session-based collaborative filtering for predicting the next song," in *Computers, Networks, Systems and Industrial Engineering (CNSI), 2011 First ACIS/JNU International Conference on*, 2011, pp. 353–358.

[6] R. Dias and M. Fonseca, "Improving music recommendation in session-based collaborative filtering by using temporal context," in *Tools with Artificial Intelligence (ICTAI), 2013 IEEE 25th International Conference on*, Nov 2013, pp. 783–788.

[7] F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, *Recommender Systems Handbook*, 1st ed., New York, NY, USA, 2010.

[8] M. Papagelis, I. Rousidis, D. Plexousakis, and E. Theoharopoulos, "Incremental collaborative filtering for highly-scalable recommendation algorithms," in *Proceedings of the 15th International Conference on Foundations of Intelligent Systems*, ser. ISMIS'05, 2005, pp. 553–561.

[9] C. Miranda and A. Jorge, "Incremental collaborative filtering for binary ratings," in *Web Intelligence and Intelligent Agent Technology, 2008. WI-IAT '08. IEEE/WIC/ACM International Conference on*, vol. 1, Dec 2008, pp. 389–392.

[10] X. Yang, Z. Zhang, and K. Wang, "Scalable collaborative filtering using incremental update and local link prediction," in *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, ser. CIKM '12, 2012, pp. 2371–2374.

[11] J. Vinagre and A. Jorge, "Forgetting mechanisms for scalable collaborative filtering," *Journal of the Brazilian Computer Society*, vol. 18, no. 4, pp. 271–282, 2012.

[12] J. a. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM Comput. Surv.*, vol. 46, no. 4, pp. 44:1–44:37, 2014.