Session-based Recommendation with User Cold-Start Problem Using Markov Chain Model & Incremental Learning

by

Zhengru Li

**A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science in Engineering
(Electrical Engineering)
in the University of Michigan-Dearborn
2021**

Master's Thesis Committee:

**Professor Yi Lu Murphey, Chair
Professor Weidong Xiang
Associate Professor Paul Watta**

Acknowledgements

I appreciate Professor Yi Lu Murphey for her supervision and support for this thesis. I also appreciate for Ford Motor Company for the dataset they provided to do this study. Special thanks to Dr. Yongquan Xie who teaches machine learning techniques, the methods of reading papers and programing. His knowledge and patient help me a lot to enter the field of machine learning and deep learning. Big thanks for Tian Qin to study the algorithm with me. As a partner in the same research team, she gave me a lot of help and inspire me with new ideas. I also need to thank my parents who gave me support on both mentally and financially. Thanks to my roommates Jiaming Liu, Feng Han, Yuxiao Zhang and Yuchi Guo, for support me in daily life as good friends.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

RS Recommendation Systems

SR Session-based Recommendation

IVI In-Vehicle Infotainment

SE-FPMC Statistics Enhanced FPMC

HR@K Hit Rate at K

MRR@K Mean Reciprocal Rank at K

Abstract

Session-based recommendation has become a hot topic of intelligent system in recent years. As a sub-field of Recommending System, the session-based recommendation studies the sequential relationship of data in user's usage sessions. In some applications, the recommending system should focus more on the personalized usage feature in order to make better recommendations. This thesis analyzed the statistics of user's usage sessions and proposed the Statistics Enhanced FPMC algorithm to enhance the personalized usage pattern of users to improve the recommending performance of recommender system for in-vehicle infotainment system and APP manage system application. The proposed algorithm also addressed the user cold-start problem by incremental learning with a knowledge distillation method to alleviate the catastrophic forgetting problem. The user cold-start problem is defined as making recommendations to new users under cold-start conditions. While the usage data becomes available for new users, the model can continue to be updated to improve recommending performance.

Chapter 1
Introduction

1.1 Overview

In recent years, the recommendation systems (RS) have gained a lot of attention from researchers. Nowadays, the recommendation system has been applied to a lot of real-world fields such as e-commerce, social APPs and In-vehicle Infotainment systems and brought a big convenience for people. Session-based recommendation (SR) is a subfield of recommendation system which deals with sequential data in users' usage sessions. The SR system does not view user's behaviors as independent incidences but considers the actions within a certain period as a sequence which the former action will have effects on latter decisions of what action the user will take. The SR system has become a core technology in recommendation system.

There are a lot of state-of-the-art algorithms which leverage neural networks[2,4,5,7,8,11,15] to extract session features to make recommendation. Models are made based on CNN or RNN structures combining with attention mechanism or extra memory part in order to enhance the models with a stronger ability to analyze the dependency of items used in the same session. Those algorithms need to learn the user's latent representation and item's latent representation and generate usage probability for all the candidate items. But those models are unable to deal with user cold-start problem which when new users come, their latent representation are unknow so that the models will lose the ability to make recommendations for those cold-start users. Another limitation of algorithms implemented with neural networks is those models have poor interpretability. Generally, neural networks are regarded as a black box model. The reason why the new structure can improve the model performance is unknown, which makes it hard for researchers to do error cases study.

Despite researchers made a great effort to cope with the user cold-start recommendation, they still need a lot of content information of users which will result in a privacy issue. Algorithms which used content information to solve cold-start problem is called content-based SR. Xu et al. proposed a method to generate a latent representation for the content info of users, then learn a matching function between the content info representation and the user latent representation[29]. Another way of solving cold-start problem is to use cross-domain recommendation. They use data from multiple domains to learn the user latent representation. For example, some researchers use social media data to learn a user latent representation so then apply them to deal with user cold-start problem. Those kinds of methods will acquire a lot of user privacy information which is unavailable most of the time. An algorithm which does not use users' content information but can make recommendation for users at cold-start condition will be more suitable for real-world applications. The algorithm should also have the ability to improve the recommendation accuracy once there are training data available. In other words, the recommending algorithm should learn the general usage patterns from a group of users while it can keep the personalized usage patterns so that it can combine general and personalized usage patterns together to improve the recommendation performance.

Session-based recommender systems based on Markov chain model utilized the sequential data by analyzing each pair of two adjacent items in the same session. Figure 1 is an example of leveraging Markov Chain to make movies recommendation. Therefore, a transition matrix will be generated to estimate the probability of what next item a user wants to use based on the last item a user used. Although the Markov chain model is a naïve mathematical model which just considers the dependency between two adjacent items, it is easy to extend to high order Markov chain to combine more sequential information. The Markov chain model also has good interpretability. The

FPMC[1] algorithm is a model which decomposed the transition matrix into item latent representation and user latent representation. The user latent representation encodes the personalized usage patterns, and the item latent representation encodes the usage patterns learned from the group of users. It is suitable for solving the cold-start problem. But due to the FPMC algorithm can learn the group usage patterns, it may weaken the personalized usage pattern and result in a performance reduction. Based on different demands, a trade-off between the personalized usage feature and the generalized usage feature should be made to enhance the FPMC algorithm. For an In-vehicle Infotainment system (IVI) or phone APP arrangement system, the personalized pattern is more important. For the online shopping and place of interest recommending system, the general pattern is more important. This thesis will focus on the IVI and the phone APP arrangement system.

Besides, most of the existing recommendation algorithms are train by an offline method which is one-time learning. But in real-world application, the new data will arrive continuedly. In other words, the recommendation system should not be trained only once but it should have the ability to be updated continuedly. The major challenge for continue learning is the catastrophic forgetting which is a tendency to forget the previously learned knowledge among new data. The easiest way to deal with the forgetting problem is to mix the new data with old data and retrain the model. But this method is very time-consuming, and the data comes continuedly which will result in a storage overhead problem. So, it is impossible to always mix all the data together. A state-of-the-art model should use an incremental method to continue updating the model parameters on newly received data while it keeps the knowledge it learned before. Although it may not be able to eliminate the forgetting problem, it should alleviate it and make the overall recommending performance to improve. Some researchers used the knowledge distillation [31] method combined with exemplar replay[30] to do the incremental learning algorithm so that the model can keep the old knowledge learned among the old data while the model parameters is been updated among newly received data. The knowledge distillation method showed a promising result on the neural network models which use SoftMax function as the activation function. In order to apply the knowledge distillation to the FPMC algorithm, a modified knowledge distillation method should be designed to enhance the FPMC algorithm to an online learning algorithm.

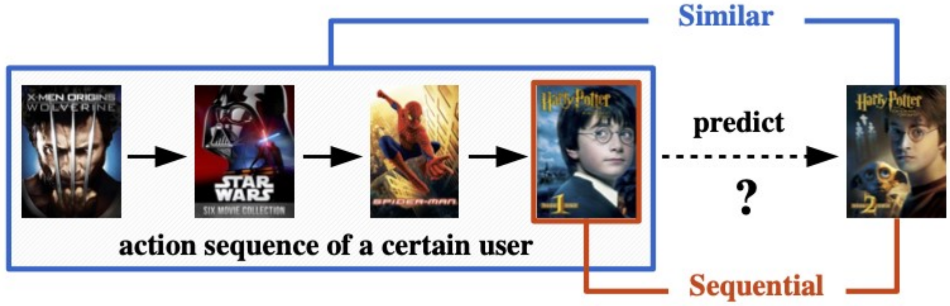


Figure 1: The session-based recommendation on movie recommendation. Image credit: R. He et al. "Fusing Similarity Models with Markov Chains for Sparse Sequential Recommendation." In International Conference on Data Mining. IEEE.

1.2 Contributions and Thesis Outline

A new Session-based recommendation algorithm will be proposed in this thesis. It is inspired by the FPMC algorithm[1] but it is enhanced by applying statistics extracted from the user usage history. The new recommendation algorithm will also engage in the incremental technique to deal with user cold-start problems. The main contributions of this newly proposed SR algorithm are summarized as follows:

- This thesis utilized the daily, weekly, monthly item usage statistics to enhance the FPMC algorithm so that the recommending accuracy can be improved. The statistics keep the user's personalized preference to balance the general usage pattern of a group of users with the target user's own taste.
- The newly proposed SR algorithm can make recommendation for cold-start users who do not have any history data recorded. Different from other existing algorithms[29] which aims at solving user cold-start problem, the SR algorithm proposed by this thesis does not

need any user's content information that may not be provided by users due to privacy concerns.

- The SR algorithm proposed in this thesis is designed for the real-world application which requires the model can be continuedly updated rather than one-time training. It leverages the incremental learning technique to facilitate the continue learning demand. The newly received data do not need to mix with old data to prevent the storage overhead. Meanwhile, the model updated on new data can forget the knowledge learned from old data as less as possible.

Contents for each chapter are illustrated below:

In Chapter 2, the related work of this thesis will be introduced. Section 2.1 introduces the state-of-the-art SR algorithms. The Markov Chain based SR models will be compared with other deep learning models which are built with different neural network architectures. The reason why Markov Chain is selected as the mathematical model to make recommendations will be illustrated then. Section 2.2 discusses the user cold start problem. It points out the existing methods to deal with user cold start problem are not suitable for user's privacy protection. In Section 2.3, the incremental learning techniques will be introduced. In Section 2.3.1, the catastrophic forgetting problem will be discussed. Then the knowledge distillation method will be introduced in Section 2.3.2. The knowledge distillation method is the key technique of the algorithm proposed in this thesis to deal with catastrophic forgetting problem on incremental learning.

Chapter 3 introduce the methodology of the proposed algorithm in detail. In Section 3.1 the details of the Statistics Enhanced FPMC(SE-FPMC) algorithm will be presented. This section includes the basic Markov Chain theory, the matrix factorization method, the statistics enhancement method and the loss function for the SE-FPMC algorithm. Section 3.2 presents how to make recommendation for cold-start users and how to update the model when the user's history usage data is available. The technique about leveraging knowledge distillation method to enhance the SE-FPMC for incremental learning is also illustrated in this section. Finally, in Section 3.3, the details about how to train the model to get the model parameters are presented.

In Chapter 4, the experiment processes are illustrated. The Ford SYNC screen dataset and the APP Usage dataset will be introduced in Section 4.1. Then the Section 4.2-4.5 will introduce the evaluation metrics, baseline methods, experiment results and ablation study separately.

Finally, In Chapter 5, the conclusion of this thesis is summarized, and the potential future researching direction is pointed out.

Chapter 2
Related Works

In this section, we will talk about some related works about Session-based Recommendation, the User Cold-start problem in Recommendation and the incremental learning.

This thesis mainly discussed the FPMC algorithm and some enhancements to improve its ability to make more accurate recommendation and deal with user cold-start problem. The Section 2.2 will introduce some existing algorithm which dealing with user cold-start problem. But it points out that the existing methods require user's content information that may not be provided by users. This thesis defined the user cold-start problem as: the algorithm should make recommendation for cold-start users without any history usage data and content information, but it can improve recommendation performance after training data is available.

Section 2.3 introduces the incremental learning. It illustrates the catastrophic forgetting which is the main challenge in incremental learning. Then it introduces the knowledge distillation method which is a popular technique to alleviate the catastrophic forgetting problem.

2.1 Session-based Recommendation

The study on the session-based recommender system (SR) is a subfield of modern Recommender System (RS). Compared with RS, SR look at the user's recent user-item interactions higher than all other historical actions. SR assumes that the recent user-item interactions can reflect the recent preference of a user[2].

The Markov Chain[1,3,4,5] based model is an early strategy to deal with the sequential recommendation problem. The first-order Markov chain model assumes the user-item interaction is a memoryless process so that it can be characterized by transition matrixes. Rendle et al. leveraged the matrix factorization method to proposed FPMC model[1]. He et al. uses high-order Markov chains using a weighted sum aggregation over previous items' latent representations to generate the FOSSIL model [3]. It uses L-order Markov Chain to make recommendation base on L-previous actions. Shani et al.[3] made use of a Markov Decision Process (MDP) to compute the probability of recommendation with the transition probability between items.

Deep learning models are popular recently which use RNN, CNN, GNN to model the users' usage sequences and making recommendation. Recurrent neural networks, which is naturally designed for processing sequential data. The GRU4REC [8], proposed by Hidasi et al., applies a multi-layer GRU[7] to simply treat the data as time series. Some work makes improvements by selecting the architecture based on RNN[9]. Jannach and Ludewig [10] combined the neighborhood-based method with RNN to capture co-occurrence signals. Tang et al. proposed the Caser model to capture the union level sequential pattern and deal with the skipping behavior by utilizing Convolutional neural networks. Some recent work uses the attention mechanism to avoid the time order. Memory networks [12,13,14] are also widely used architecture that build an external memory unit to save the users' history usage pattern. It can be used together with RNN model to improve model performance. The Graph Neural Network (GNN) is gaining researchers' attention recently. It has been explored for learning representation of graph structure data such as a social network. Chen et al. proposed a graph collaborate filtering method[15] which builds a bipartite graph for user-item interactions. SR-GNN [16] applies a gated GNN to learn item and session embeddings which can combine the long-term usage patterns with recent preference to make recommendations. The Full Graph Neural Network (FGNN) [17] leverage the weighted attention mechanism to learn the inherent order of the item transition patterns.

Although the deep learning models built with different neural network architectures perform great results on SR systems, the large scale of parameters makes the training time of deep learning model to be very long. It is hard for them to deal with big data problem. Also, most of the deep learning models are black box models which have poor interpretability. The Markov Chain based models have better interpretability. The architecture for Markov Chain based models are simply because Markov Chain models the sequence data as memoryless transitions. It is easy to extend the Markov Chain model to high order Markov Chain to extract more session features. So, this thesis will focus on Markov Chain model especially the FPMC to study the SR algorithm and make enhancements to it.

2.2 User Cold-start Problem

The term "Cold-Start Problem" is derived from cars. When the outside temperature is low, it's hard for car engines to warm up and run smoothly. For a recommending system, the cold-start problem[34] means that there are no data for the system to learn a good usage patter to make recommendation. There are mainly two sub-fields of cold-start problem: the user cold-start problem and the item cold-start problem. In this thesis, only the user cold-start problem will be discussed.

The user cold-start problem denotes the situation that to make recommendation when a new user start to use the recommending system. Much research has been done for improving the user cold-start problem. There are mainly three ways of strategies which are the Content-based recommendation, the cross-domain recommendation and Meta learning approaches.

The *content-based* recommendation leverages the user's content information to find similar users to make recommendation. This approach is more suitable for those recommending tasks which the items were ratted by users before. From the user content information (e.g., age, gender, comments to items, etc.) and item content information (e.g., comments from users, price, movie director, etc.) the similar users or similar items can be found. The system will generate recommending items from similar users' history used items and items similar to those history used items.

The *cross-domain* recommendation[32,33] utilizes all the available data collected from multiple domains to transfer usage patterns across different domains. Hu et al.[35] proposed a cross-domain collaborative filtering model to alleviate the cold-start issues in the target domain. The cross-domain collaborative filtering method is further developed to tag-based[36] or review-based[37] cross-domain factorization models so that they can improve the recommendation performance of cold-start problem in the target domain. The cross-domain collaborative filtering for cold-start problem can also leverage social network data to improve the recommendation performance[38,39]. Zhou et al.[40] studied the cold-start user problem in video recommending system by comparing the user's social media's relevance. Lin et al.[41] made Apps recommendation for cold-start users with the follower information in Twitter. A star-structured graph of user's social networks was studied by Jiang et al.[42] to transfer knowledge across different item domains and improve the recommendation accuracy for cold-start users in single domains.

Meta learning [18], which aims to learn a learner that is capable of well adapting or generalizing to new tasks and new environments that have never been encountered. Some few-shot learning or cold start learning problems have gained considerable improvement when applying meta learning [19, 20, 21, 22, 23, 24]. Generally, the common approach of meta learning can be divided into three groups: metric-based, model-based and optimization based. The metric-based method is similar to the nearest neighbor algorithm which project the real-world data into high-dimensional vector and compares the similarity between new instances and instances in the training set. Typical metric-based models are Matching Network [20], Siamese Network [21], Relation Network [24], and Prototypical Network [23]. The model-based meta-learning is mainly applied to models with RNN or memory networks. It depends on a model designed for fast learning. Typical model-based models are Meta-MANN [25], Meta Network [26]. The optimization-based meta-learning trying to find a way to adjust the optimization algorithm so that the model parameters can converge within a small number of training steps. Typical optimization-based models are MAML [27], Meta-LSTM [21].

The methods illustrated above all improve the recommendation performance for cold-start users from different aspects. But they still have different defects. The content-based methods rely a lot on users' prior rating history to different items. Although it can find similar users for cold-start users, the improvement of performance is limited. It also does not have a general standard to calculate the similarity. The cross-domain recommendation requires large data of different domains which is hard to acquire in real-world application. Both of the content-based and cross-domain recommendation need users to provide their privacy information. In real life, most users are not willing to provide that information for their privacy concerns. The Meta learning algorithms won great success on few-shot learning and can make recommendations for cold-start users with very few usage histories. But meta learning methods are mostly not designed for SR systems. After users' data is available, it needs to retrain the whole meta learning parameters which is a very time-consuming process. In this thesis, it will deal with cold-start problem without any user or item content information. After users' data is available, it can keep updating the model to improve recommending performance.

2.3 Incremental Learning

Incremental learning is an important field of machine learning. It allows algorithms to continue update on new datasets. It has been an important learning technique for the SR systems in real-world applications where new users, new items will continuing come. For those SR algorithms described in the above sections, they all trained offline which means the model was trained for just one time and the model parameters are unchangeable in the feature evaluation. That does not match the real-world situations. This thesis will add the incremental learning techniques to equip the SR algorithm with the ability to continually update its model parameters on newly received data.

The major challenge of the incremental learning is catastrophic forgetting [43,44] which is defined as the tendency of a machine learning model to forget the old knowledge after its parameters are updated on new datasets. For example, if the model parameters are updated on cold-start users' usage dataset, the recommending accuracy for old users may reduce. Although the model may not totally forget the old knowledge, it is better to keep the model as good as before.

The best way to solve the catastrophic forgetting problem is to mix the new dataset with old datasets. But the memory space for a computer has a limitation. If the new dataset is mixed with old dataset, it will result in a storage overhead problem which means the computer memory cannot store the big size of data. Also, if the new and old datasets are combined together, the training time for SR model will also increase. The incremental learning requires the model parameters are only updated on newly received dataset without mixing it with old data.

Strategies of alleviating the catastrophic forgetting in incremental learning can be mainly divided into three categories: regularization [45,46], exemplar replay [47] and dynamic architecture [48]. The dynamic architecture is facilitated by fixing part of the neural network’s structure and building new brunch structures for new tasks. It will continually increase the model parameters. The exemplar replays method samples examples from the old datasets, keep them as an exemplar to mix them with new dataset for updating parameters. There are many ways for sampling examples from old dataset. The easiest way is random sampling which just selects data examples randomly. Each example has the same probability to be selected. Another way is called the herding technique

[49]. It selects exemplars dynamically by dividing the old data into several groups and finding the group "center examples". The exemplar replay method does not mix the entire old dataset with the new data but selects a subset with a fixed size of old data to combine with new data for parameters updating. This method can prevent the storage overhead while preventing the model from forgetting previous knowledge. The regularization method adds an extra term to the loss function to enhance the previously learned knowledge. The most famous regularization method in incremental learning is knowledge distillation [31]. Knowledge distillation was first to be proposed in [31] for model compression. It uses the knowledge learned in large model to guide the training process of small models. The incremental learning borrows this concept to consolidate the previous learned knowledge while parameters are updated on new data. Li et al.[46] added a penalty term to the loss function of SR model to penalize model logit change. It is further developed to penalize the changes on parameters which is important to remember the old knowledge. The knowledge distillation is easy to apply to different SR models so that it becomes popular in the incremental learning of recommending systems. In this thesis, the knowledge distillation method will also be leveraged to adjust the loss function of the proposed SE-FPMC algorithm to learn new user's usage patterns without forgetting the old user's usage features.

Chapter 3
Methodology

This Chapter first describes the mathematical modeling of the SR problem and some basics of Markov Chain theory that underpin the work in this thesis. Then it presents the proposed model in section 3.3: SE-FPMC. It leverages statistics extracted from user's daily/weekly/monthly usage data to enhance the recommending performance of FPMC[1] algorithm. In section 3.4, the procedures of how to use SE-FPMC algorithm to deal with users cold-start problem will be presented. It will introduce in detail how to use the knowledge distillation method with the SE-FPMC model to make the SE-FPMC algorithm can be used in doing online learning.

3.1 Problem Description

Let $U = \{u_1, u_2, u_3 \dots u_{|U|}\}$ denotes a set of users and $I = \{i_1, i_2, i_3, \dots, i_{|I|}\}$ denotes a set of items. For each user u, all its item usage history is known as $B^u = (B_1^u, B_2^u, \cdots, B_{|B^u|}^u)$, $B_i^u \in I$. Some of the items in B^u may be used together or be used within a short time period so that we can further divide B^u into several sessions $B^u = (S_1^u, S_2^u, \cdots, S_{|S^u|}^u)$, where $S_i^u = \left(S_{i_1}^u, S_{i_2}^u, \cdots, S_{i_{|S_i^u|}}^u\right), S_{i_i}^u \in I$. For two adjacent items B_i^u, B_{i+1}^u, if they are in the same session S_i^u, we think they are related and the usage of B_{i+1}^u may result from the usage of B_i^u. Otherwise, we think the two adjacent items are independent. My objective is to predict the next item the user may want to use based on the last item it used and thus make recommendations accordingly.

$$S_{i_i}^u \rightarrow S_{i_{i+1}}^u \tag{3.1}$$

3.2 Markov Chain Basics

A Markov chain is a mathematical system that experiences transitions from one state to another according to certain probabilistic rules. The probability of transitioning to any particular state is dependent solely on the current state and time elapsed.

Markov Chain has a memoryless property. For any positive integer n and possible states $i_0, i_1, i_2, \ldots, i_n$ of the random variables $P(X_n = i_n | X_0 = i_0, X_1 = i_1 \ldots, X_{n-1} = i_{n-1}) = P(X_n = i_n | X_{n-1} = i_{n-1})$. This function means the next state is only dependent on the current state. A transition matrix P_t for Markov chain at time t is a matrix containing information on the probability of transitioning between states:

$$(P_t)_{i,j} = P(X_{t+1} = j | X_t = i) \tag{3.2}$$

3.3 SE-FPMC: Statistics Enhanced Factorize Personalize Markov Chain Method

3.3.1 Overall Framework of SE-FPMC

The Overall framework of SE-FPMC algorithm mainly contains 3 parts: The Decomposition Matrixes Learning, The Usage Frequency Score Calculation, and the Item Preference Score Generation.

The Decomposition Matrixes Learning is based on Tuck decomposition theory to decompose the transition matrixes of users. It is also known as Matrix Factorization technique. In real world, users will not interact will all items which makes the transition matrixes have sparsity problem. The goal of Decomposition Matrixes Learning is to use factorized dense matrix to approximate the sparse transition matrixes so that every item can have the probability to be recommended.

The Usage Frequency Score is a method to use statistics extracted from user history usage data to enhance the personalized usage pattern in recommendation. Because this thesis aims at improving the recommending performance for IVI and APP arrangement SR system, the personalized usage feature should have a higher weight when conducting recommendation.

The Item Preference Score Generation combines the results of decomposition matrixes learning and usage frequency score calculation to generate the final score for each candidate item. The top K items with the largest item preference scores will be recommended to users. The overall framework can be described in Figure 2

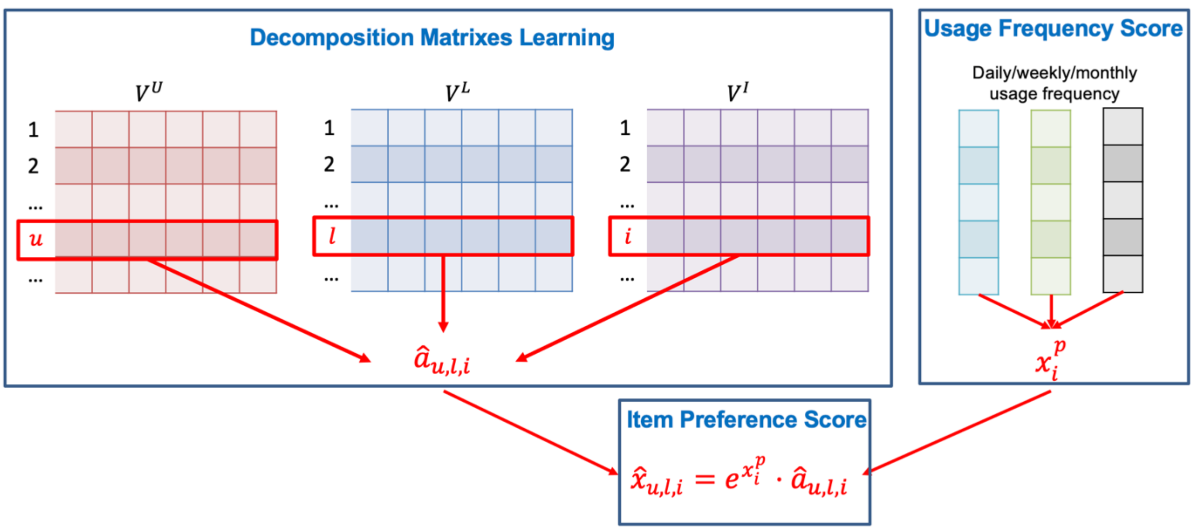


Figure 2: The overall framework of SE-FPMC.

3.3.2 Decomposition Matrixes Learning

The core theory that pins up the FPMA algorithm is a 3-d tensor that can be decomposed into 3 matrices using the Tuker decomposition method shown in Figure 3. For a 3-D tensor $\in R^{|U|\times|I|\times|I|}$, it can be decomposed into 3 matrices, $V^U \in R^{|U|\times P}, V^L \in R^{|I|\times Q}, V^I \in R^{|I|\times L}$, and a core tensor $\mathcal{g} \in R^{P\times Q\times L}$. P, Q, L, are predefined matrix dimensions, such as 32, 64 (usually much smaller than the number of items).

Generally, we use an identity tensor as core tensor $\mathcal{g}$. With the decomposition theory, we can factorize the transition tensor A into 3 decomposed matrices V^U, V^L, V^I

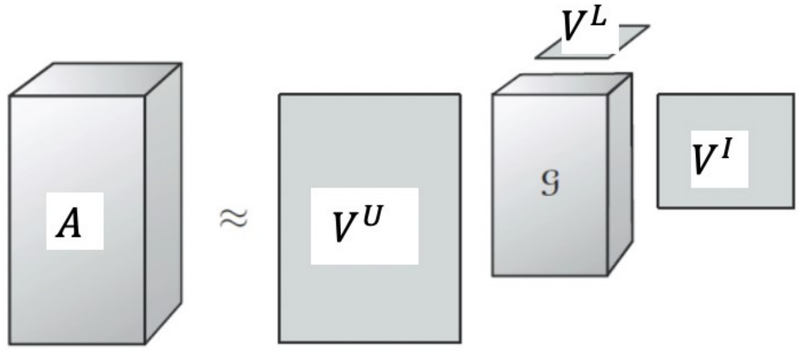


Figure 3: Tuker decomposition theory.

V^U, V^L, V^I are the trainable parameters of model, and they can be updated by learning from the training data. V^U is regarded as the feature matrix for users, V^L is regarded as the feature matrix for the items in the last transition, V^I is regarded as the feature matrix for the items to predict. Let$\theta = \{V^U, V^L, V^I\}$ denotes the model parameters for brevity.

The element of the tensor A can be approximated by:

$$\hat{a}_{u,l,i} = < v_u, v_i > + < v_i, v_l > + < v_u, v_l > \tag{3.3}$$

v_u is the u-th row of matrix V^U, v_i is the i-th row of matrix V^I, v_l is the l-th row of matrix V^L, $<,>$ is the inner product operator between two vectors.

All approximated elements compose a matrix $\hat{A} : \hat{A}_{u,l,i} = \hat{a}_{u,l,i}$. The matrix $\hat{A}$ is used to approximate A. The goal of this algorithm is to obtain the approximated $\hat{A}$ for calculating item rankings, instead of using A for the calculation because of its sparsity.

Let $i >_{u,l} j$ denote the user u ranked item i higher than item j in choosing the next item to the item l. We randomly selected an item j ($j \neq i$) from the item space which represents the item that is not chosen by the user as the next item to l. The items l,i,j together contain the sequential information that, after using l, the user transits to i, instead of the item j, or in other words, the user u prefers to choose item i to item j after using item l. To model the ranking, an estimated score $\hat{x}$ is calculated so items can be sorted according to their scores:

$$\hat{x}_{u,l,i} = \hat{a}_{u,l,i} =< v_u, v_i > + < v_i, v_l > + < v_u, v_l > \tag{3.4}$$

To estimate the model parameters θ, it needs to optimize the posterior probability $p(\theta | i >_{u,l} j)$ to have a maximum value. According to Bayesian theorem,

$$p(\theta | i >_{u,l} j) = \frac{p(i >_{u,l} j | \theta) \times p(\theta)}{p(i >_{u,l} j)} \tag{3.5}$$

Assume independence of users and sessions, the objective function is generated as:

$$\arg\max \prod_{u \in U} \prod_{S_t^u \in S^u} \prod_{i \in S_t^u} \prod_{j \notin S_t^u} p(i >_{u,l} j | \theta) \tag{3.6}$$

Here we use sigmoid function to model the likelihood probability $p(i >_{u,l} j | \theta) = \sigma(\hat{x}_{u,l,i} - \hat{x}_{u,l,j})$, $\sigma(z) = \frac{1}{1+e^{-z}}$.

After applying the function $\ln(\cdot)$, the objective function is converted to the loss function:

$$l = -\sum_{u \in U} \sum_{S_t^u \in S^u} \sum_{i \in S_t^u} \sum_{j \notin S_t^u} ln\sigma(\hat{x}_{u,l,i} - \hat{x}_{u,l,j}) \tag{3.7}$$

By optimizing the loss function with gradient descend method, the model parameters $\theta = \{V^U, V^L, V^I\}$ can be trained from data.

3.3.3 Statistics Enhancement: Usage Frequency Score Calculation

We integrated the historical screen usage frequency information on day/week/month into the FPMC algorithm. For each user and each session, the statistics include 3 lists that represent daily, weekly and monthly preferences. Each list contains all the screens that are ranked according to the usage frequencies in a day/week/month prior to a session. If an item i was used more frequently by a user, it is ranked higher in the list.

For a session S^u, let P^d, P^w, P^m denote 3 lists of the 239 screens in descending order according to daily/weekly/monthly usage frequency separately. P_i^d, P_i^w, P_i^m denote the ranking of item S_i^u in P^d, P^w, P^m separately. We define the screen usage frequency score x_i^p:

$$x_i^p = \left(\frac{1}{P_i^d} + \frac{1}{P_i^w} + \frac{1}{P_i^m}\right) \div 3 \tag{3.8}$$

The usage frequency score combines users' monthly, weekly and daily usage preferences. If a screen was used more frequently in the day, week, month, its preference score will have a larger value in the range of [0, 1]

3.3.4 Item Preference Score Generation

In the SE-FPMC algorithm, the item preference score function will be calculated as:

$$\hat{x}_{u,l,i} = e^{x_i^p} \cdot (< v_u, v_i > + < v_i, v_l > + < v_u, v_l >) \tag{3.9}$$

Here the usage frequency score is used as a coefficient to enlarge the estimate scores. If a screen is used more frequently in the past day/week/month, the coefficient $e^{x_i^p}$ will have a larger value. As a result, the corresponding screen will be estimated to have a relatively larger preference score than the FPMC algorithm. When making recommendation, the item preference scores will be generated for every candidate item. And the system will recommend top K items with the largest item preference scores.

The FPMC algorithm uses all users' usage data to train a single model which can encode personalized usage feature while extracting a general usage pattern from the group of users. But when it is extracting usage patterns of the group of users, some personalized usage taste will be weakened.

That will make the performance of recommendation for some specific users reduce. The usage frequency score encodes the user's long and short usage habit which is a personalized usage feature. When the usage frequency score is added to the score function, it will enhance the personalized usage feature for the target user we need to make recommendation to. It is a trade-off between the personalized usage pattern with the general usage pattern. The general usage pattern can help users to explore more items they may be interested in while the personalized usage pattern can help users to reach the item they prefer to use in their daily life. For the in-vehicle infotainment system (IVI) or the APP usage recommending system, users do not need to explore too many items but just use several frequently used items. The personalized patterns are more important for these SR systems. The online shopping or video website SR systems hope users can explore more items to buy or to view. So, the general usage pattern for those SR systems is more important. The SE-FPMC algorithm proposed in this thesis aiming at improving recommending performance for IVI or APP users, that's why it uses the usage frequency score to enhance the item preference score.

3.3.5 Loss Function

In section 3.3.1 the loss function of FPMC algorithm is presented. The SE-FPMC algorithm proposed in this thesis modified the item preference score to

$$\hat{x}_{u,l,i} = e^{x_i^p} \cdot (< v_u, v_i > + < v_i, v_l > + < v_u, v_l >) \tag{3.10}$$

The inputs to the SE-FPMC algorithm are tuples (u, l, i, j), which denotes the user u prefer item i to item j after interacting with item l. The probability such that user u will prefer to use item i more than item j after using item l can be calculated by $p\left(i >_{u,t} j \middle| \theta\right) = \sigma(\hat{x}_{u,l,i} - \hat{x}_{u,l,j})$, $\sigma(z) = \frac{1}{1+e^{-z}}$. The form of loss function will be the same as the FPMC loss function which is equation (3.7). But the scores in the function are modified scores which are enhanced by the usage frequency scores.

3.4 Deal with Cold-start Problem with SE-FPMC

3.4.1 Introduction

Traditionally, the cold-start problem is solved usually by using users' content information (e.g., gender, age, data from social media). In this thesis, such users' content information is unavailable. The cold-start problem in this thesis is to make recommendation for new users under cold-start conditions. But when the usage data for new users become available, the model should have the ability to update model parameter to improve the recommendation performance. The proposed SE-FPMC algorithm can learn a general usage pattern and make recommendation for cold-start users. In Section 3.4.2, the method about how to make recommendation for new users under cold-start conditions will be introduced. Section 3.4.3 will illustrate how to do incremental learning to update model parameters and how to alleviate the catastrophic forgetting problem.

3.4.2 Make Recommendation for Cold-start Users

In SE-FPMC algorithm, we use decomposition method to decompose the transition tensor into 3 matrices V^U, V^L, V^I. The matrix V^U encodes users' usage patterns, in which each row is a user representation. The matrixes V^L and V^I encode the relations between the current screen and the next predict screens. Because V^L, V^I encode the general usage pattern of the group of users, we can use them to make recommendation for cold-start users, whose V^U was not available. When a new user comes, one row will be added to V^U, but V^L, V^I will keep the same. The process is shown in Figure 4. If we don't have data for the new user, we do not know the value of newly added row in V^U, we can just use V^L, V^I to make recommendation. The new row will be set as a zero vector. Under the cold-start condition, the user's statistics information cannot be acquired, so the usage frequency score cannot be used to make recommendation too. In other words, the usage frequency score is equal to zero. Recalling the score function (3.10), the x_i^p equal to zero so that $e^{x_i^p}$ equal to 1. The v_u under cold-start condition will be set to a zero vector, the first term and the third term of the function will be zero. So, the score function for new users under cold-start condition will be: $\hat{x}_{u,l,i} = < v_i, v_l >$. Just the V^L, V^I are used for recommendation. From this score function we actually using the similarity of current using item with candidate item to make recommendation. Because we do not have information from the new users, we can just use the similarity between items to make recommendation.

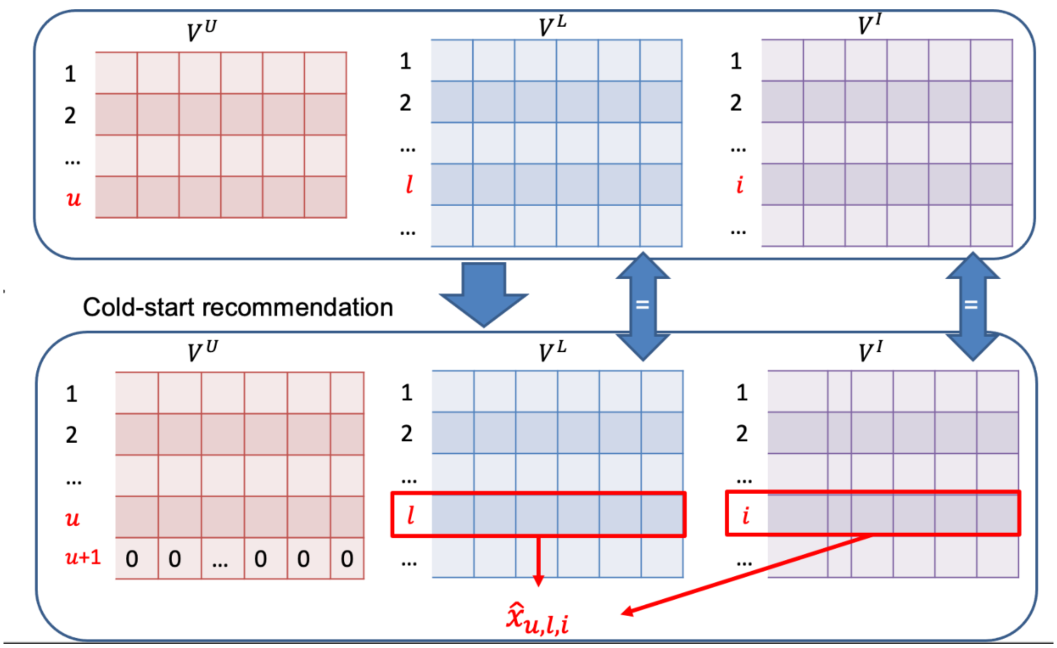


Figure 4: Recommendation for cold-start user.

3.4.3 Incremental Learning with Knowledge Distillation

Incremental learning denotes the learning process is conducted on newly received data without mixing them with old data. In Section 3.4.3, we presented how to make recommendation for cold-start users. Once the data for the new user is available, we can update V^U, V^L, V^I with the new data until the loss function converge. For matrix V^U only the newly added row will be update, other rows which represent old users will not be changed. V^L, V^I will be updated on new user's data. The updating processes are shown in Figure 5.

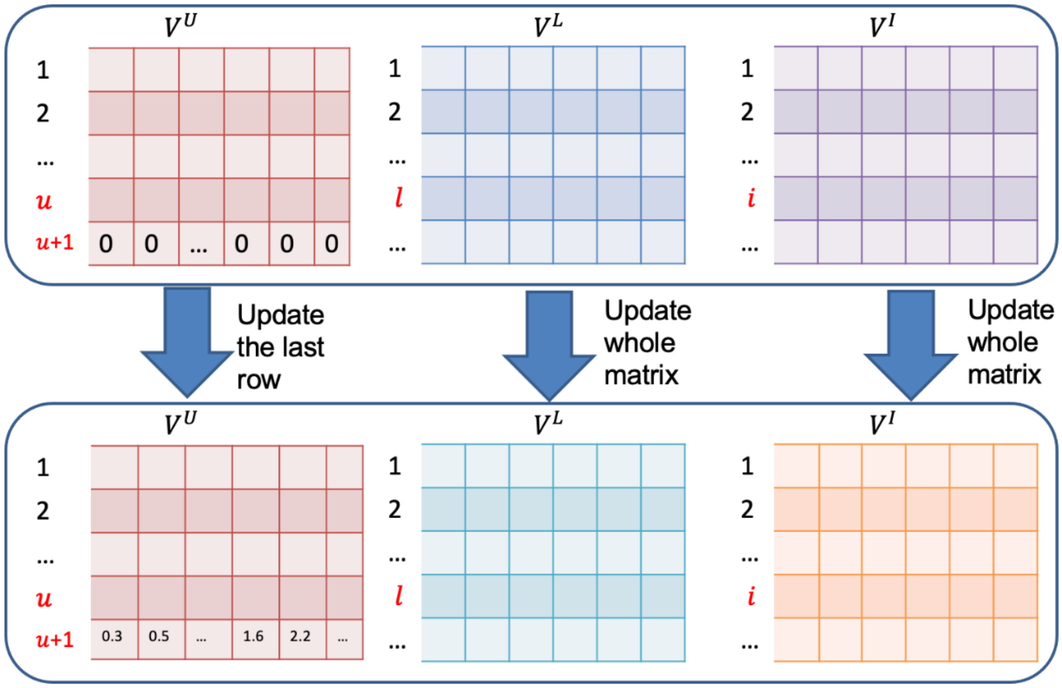


Figure 5: Parameters updated on new user's data.

In real-world applications, the SR system should have the ability to conduct continual learning. The new data will continually arrive in a different time period. For the old users in newly received data, the SR system should make recommendation with the decomposed matrixes and the usage frequency scores. For the new users, the SR system should make recommendation under cold-start conditions. Then, the SR system should use the newly received data to update the model with incremental learning.

Assume at time period t, the newly received data is D_t. The SR model trained on last time period is $model\ \ t-1$ and its parameters are $V_{t-1}^U, V_{t-1}^L, V_{t-1}^I$. For old users in D_t, the recommendation will be made by $V_{t-1}^U, V_{t-1}^L, V_{t-1}^I$ and usage frequency scores. For new users in D_t, the recommendation will be made by V_{t-1}^L, V_{t-1}^I. After that, the whole dataset D_t will be used for update $model\ \ t-1$ into $model\ \ t$ with incremental learning. The parameters for $model\ \ t$ are V_t^U, V_t^L, V_t^I, which will be used for making recommendation at time $t+1$. At time period 1, there is no previous model that can be used for recommendation. So, the dataset D_1 will only be used for training $model$ 1. The following chart of the incremental learning processes is shown in Figure 6.

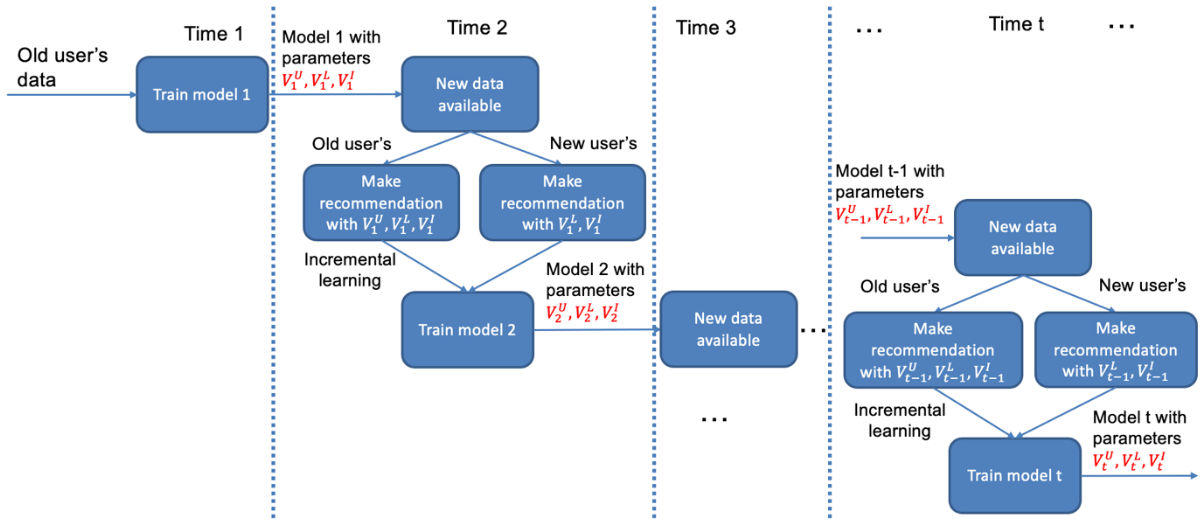


Figure 6: Follow chart of incremental learning processes.

At time period t, the incremental learning will be conducted to update the V_{t-1}^L, V_{t-1}^I to V_t^L, V_t^I. If we still use the loss function presented in Section 3.3.5, the recommending performance for old users at time period $t + 1$ may reduce. Because we make the model to learn new knowledge on new dataset, it may make it to forget old knowledge it learned before. The tendency of a machine learning model to forget previously learned information completely upon learning new information is called catastrophic forgetting. Although the real-world SR system may not forget the previous knowledge completely, but it will cause the recommending performance for old users to reduce. There may not be a way to prevent the model from forgetting old knowledge, but we hope the model can remember old knowledge as many as possible.

The knowledge distillation is a popular method to regularize the loss function so that catastrophic forgetting problem can be alleviated. For user u in dataset D_t, after using item l, the item i is used rather than item j. We think the user u prefer item i to item j after using item l. So, the ground truth is $p_t(i >_{u,l} j) = 1$. In the SE-FPMC algorithm, before the training loss converge, the

$model\ \ t$ will generate a probability such that user u will prefer to use screen i more than screen j after using screen l. Assume the probability generated by $model\ \ t$ is $\widehat{p_t}(i >_{u,l} j)$. Assume the probability generated by $model\ \ t-1$ is $\hat{p}_{t-1}(i >_{u,l} j)$. Comparing $\widehat{p_t}(i >_{u,l} j)$ with the ground truth $p_t(i >_{u,l} j)$, the loss calculated by loss function in Section 3.3.5 can make the model to learn new knowledge. We denote it as $l_{incremental}$. Comparing $\widehat{p_t}(i >_{u,l} j)$ with $\hat{p}_{t-1}(i >_{u,l} j)$, the model can consolidate the knowledge learned before time period t. Here we define the distillation loss as:

$$l_{distillation} = -\sum_{u \in U} \sum_{S_t^u \in S^u} \sum_{i \in S_t^u} \sum_{j \notin S_t^u} \hat{p}_{t-1}(i >_{u,l} j) \ln\left(\widehat{p_t}(i >_{u,l} j)\right) + \left(1 - \hat{p}_{t-1}(i >_{u,l} j)\right) \ln(1 - \widehat{p_t}(i >_{u,l} j)) \tag{3.11}$$

Combine the two terms of loss, the total loss function is:

$$l_{total} = \lambda * l_{distillation} + (1 - \lambda) * l_{incremental} \tag{3.12}$$

The coefficient λ is a hyper-parameter which is used to adjust the weights of distillation loss and incremental loss. Optimize the total loss with backward propagation method until it is converged, we can get the updated parameter V_t^U, V_t^L, V_t^I for $model\ \ t$. The training processes can be conducted with mini-batch stochastic gradient descend method to maintain training efficiency. For each adjacent item in the same session, 10-20 negative example will be randomly selected from all the candidate items. They will form tuples to feed into the SE-FPMC system for training. The follow chart of how to conduct incremental learning with knowledge distillation is shown in Figure 7.

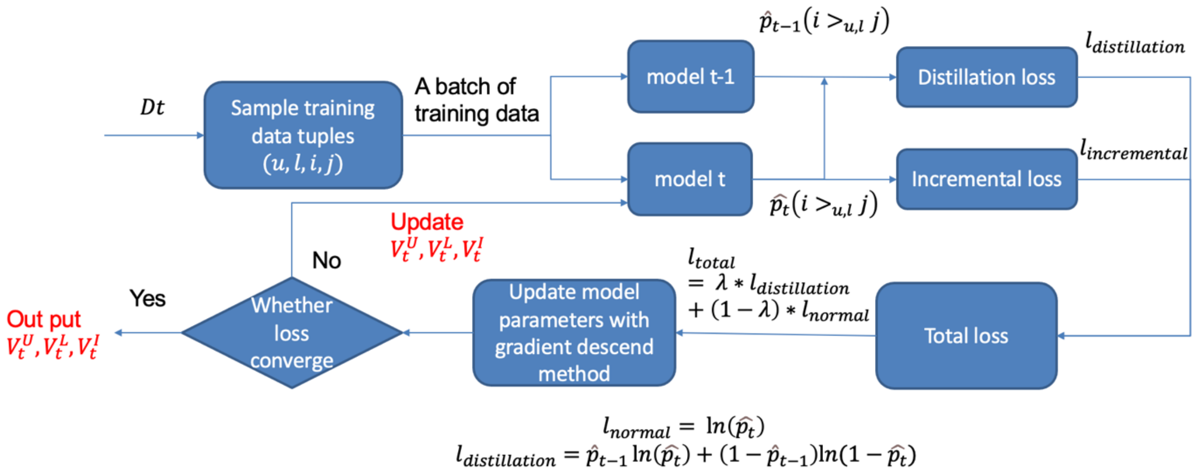


Figure 7: Follow chart of how to conduct incremental learning with knowledge distillation.

Chapter 4
Experiments

This Chapter firstly describes the SYNC dataset and APP usage dataset which are used for running experiments. Then it will introduce the baseline SR algorithms and the evaluation metrics. For the proposed SE-FPMC algorithm, the recommending performance will be compared with the baseline SR models. Based on our knowledge, there are no existing algorithms to deal with the same user cold-start problem as we did. So, we only did ablation study on the experiments of dealing with user cold-start problem using SE-FPMC algorithm.

4.1 Datasets

This thesis mainly deals with the SR system which looks high on the personalized usage features such as IVI and APP manage systems. The dataset selection should base on the following rules: (1) the datasets must provide distinct user IDs so that the recommending system can extract personalized usage features and also separate cold-start users; (2) the datasets must provide clear timestamps for extracting the usage frequency scores; (3) the dataset must be recorded for several months to make sure there are enough data for training and testing. Another reason for requiring datasets to be recorded for months is that the usage frequency scores are extracted from daily, weekly and monthly usage. Based on the rules described above, we selected the following datasets for conducting experiments:

- *SYNC screen usage data*, provided by Ford Motor Company. This dataset recorded user's interaction with the IVI system during the last quarter of 2019. It contains 192 user's usage data and sessions can be divided based on specific events such as engine starting. Sessions with a length less than 2 screens will be removed. Averagely, each user has 125 sessions and the average session length is 12. This dataset is not released in public domain.

- *App Usage dataset,* provided by LiveLab of Rice University in a public platform. This dataset recorded the usage of 2292 APPs on iPhone. It records 34 user's usage data in 14 months. A predefined time threshold with 10 minutes is used for session division. Sessions with a length shorter than 2 are removed. On average, each user has 1684 usage sessions. The average length of the session is 5.

4.2 Evaluation Metrics

There are 2 metrics are used in this thesis to assess the recommending performance. Which are Hit-Rate at K (HR@K) and Mean Reciprocal Rank at K (MRR@K). These two metrics are also used in many other studies. For every recommending task, K items with the highest item preference scores will be recommended to the target user.

The HR@K evaluates the ability of a model to include a ground truth item in the recommended items. HR@K $= \frac{\#\, successful\ hits}{\#\, total\ recommendations}$. A higher HR@K value indicates a better performance. The MRR@K evaluates the ability of a model to rank the ground truth item higher than other recommended items. We use r to denote the rank of ground truth among the K recommended items. $r = +\infty$ in the case that the ground truth item is not included in the K recommended items. MRR@K$= \frac{\sum \frac{1}{r}}{\#\, total\ recommendations}$. A larger MRR@K value indicates a better performance.

4.3 Benchmarks

In this thesis, 3 widely used SR algorithms are compared with the proposed SE-FPMC algorithm:

- ***FPMC***, the backbone algorithm of the proposed SE-FPMC model. It decomposed the transition matrixes into 3 decomposed matrixes to deal with data sparsity. It facilitates the Markov Chain model with the ability to extract both personalized usage patterns and general usage patterns of a group.

- ***SKNN***, the Session-based K Nearest Neighbors algorithm. It searches neighbor sessions for the target session base on similarity metrics. Then the recommendation is made by items in neighbor sessions.
- ***DQN***, the Deep Q-Learning Network, is a popular method of Reinforcement Learning. It generates Q-table of each item by Deep Neural Network. The learning processes are made with a simulation environment to get the reword of recommending each item. Then it recommends top K items with the largest values in Q-table.

Those benchmarks are used to compare the performance with SE-FPMC algorithm for users without cold-start problem. Due to the lack of existing algorithms to deal with the same cold-start problem as our work, we only did the ablation study for cold-start problems.

4.4 Experiment Settings and Results

For every user in each dataset, 20% of recent sessions are reserved for evaluation and 80% of history sessions are used for model training. The recommending system can only learn the history usage habits to predict the future behaviors of users. We can not learn future usage features to predict the history behaviors. So, we did not do cross validation on evaluation. Different recommending values K are set to K=1, 3, 6, 9, 12, 15. The larger K value is not considered because for IVI or APP usage, users will not be likely to view too many recommended items. For each training session, one couple of adjacent items will be sampled randomly from the session. And 10 negative examples will be randomly select from all candidate items as long as they are different from the ground truth. The sampled items will form tuples as training data to train the SE-FPMC model. The hidden dimensions for all the 3 decomposed matrixes are set to 64. The elements of the decomposed matrixes are optimized by mini-batch Adam optimizer with learning rate $1e^{-3}$. The maximal training epoch is set to 150 for the 2 datasets. For benchmarks FPMC and DQN, the experiments settings are similar. For SKNN algorithm, it does not have any trainable parameters but doing search. The neighbor quantity N is set to 10.

To compare the recommending performance of the proposed SE-FPMC model with the benchmarks, the results are the average performance of all 192 users in SYNC dataset. Same for the APP usage dataset. We used the average performance of all 34 users for comparison. The evaluation is made to predict the last item for each testing session. Results are shown in Table 1, Figure 10, and Figure 11.

To evaluate the performance under the cold-start condition, we selected 13 users as the cold-start users for SYNC dataset. The 13 users used the SYNC system across all the 3 months of data collection. The other 179 users are used for pretraining the SE-FPMC model. Because we want the model parameters can be updated while the new user's history data become available, the data size for cold-start users should be large enough. If cold-start users have very few sessions, it will not be able to update model parameters. The updated model with very few sessions is still under cold-start condition. For APP usage dataset, we also selected 4 users with the longest usage time as the cold-start users to make sure that their history sessions are enough for updating model parameters. The rest of the 30 users' sessions are used for conducting the pretraining process. For each user, including the cold-start users and the old users whose data are used for pretraining the model, its 20% of the most recent sessions are used for evaluation. 80% of the history sessions are used for model pretraining or model updating. The maximal training epoch for model updating is 150. Figure 8 and Figure 9 shows how the SYNC dataset is divided to conduct the experiments of cold-start recommendation and evaluation of updated model. The APP usage dataset is processed with the same method.

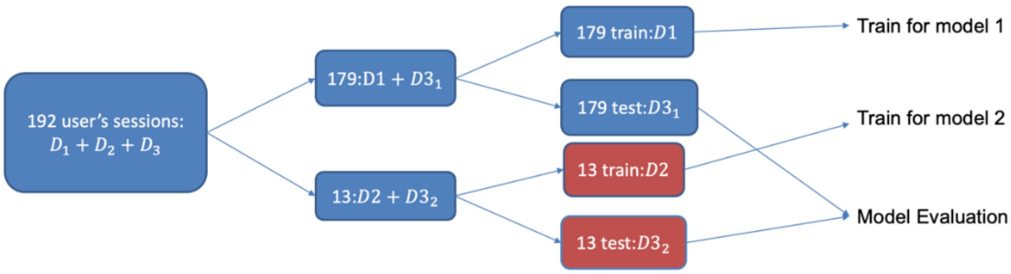


Figure 8: Divide SYNC dataset into 3 parts.

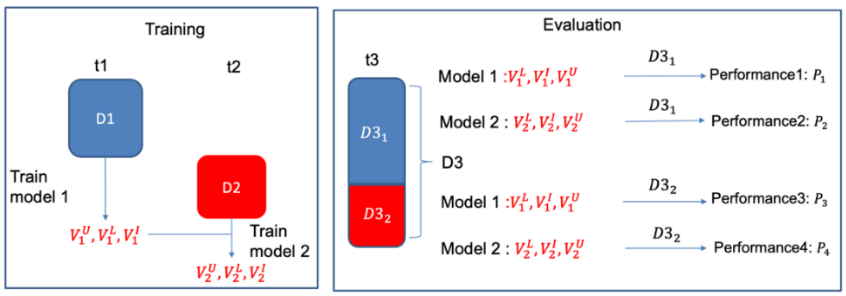


Figure 9: Training, updating and evaluation processes.

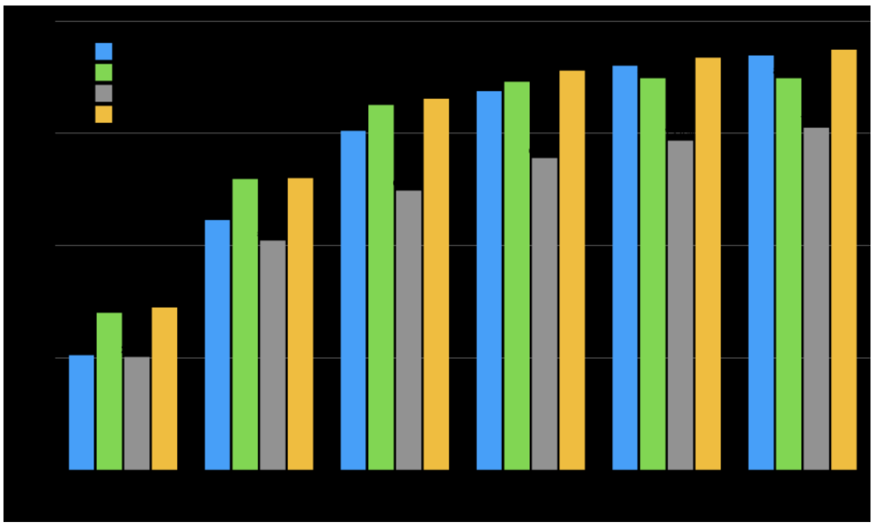
The 3 parts of data are assumed to be available at different time periods. The first part which contains old users' sessions is used for training the model 1. The second part which contains sessions of cold-start users is used for training model 2. The third part is testing data of both old users and cold-start users which are used for evaluation model 1 and model 2. When updating the model parameters with new users' data, the λ value of the weight of distillation loss is $\lambda = 0.7$. Results are shown in Table 2, Figure 12 and Figure 13.

Table 1: Performance (%) comparison over the two datasets.

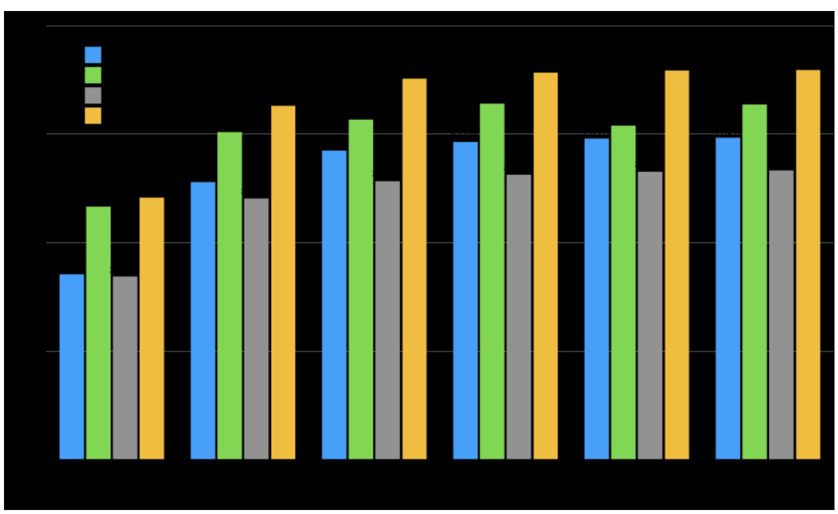
Dataset	Model	K=1		K=3		K=6		K=9		K=12		K=15	
		HR@K	MRR@K	HR@K	MRR@K	HR@K	MRR@K	HR@K	MRR@K	HR@K	MRR@K	HR@K	MRR@K
SYNC	FPMC	25.6	25.6	55.7	38.4	75.6	42.7	84.4	43.9	90.1	44.4	92.4	44.5
	SKNN	35.0	35.0	64.9	45.3	81.4	47.0	86.5	49.2	87.3	46.2	87.3	49.1
	DQN	25.2	25.2	51.1	36.1	62.3	38.5	69.5	39.4	73.4	39.8	76.3	40.0
	SE-FPMC	**36.2**	**36.2**	**65.1**	**48.9**	**82.7**	**52.7**	**89.0**	**53.5**	**91.9**	**53.8**	**93.7**	**53.9**
APP Usage	FPMC	20.1	20.1	44.4	30.6	63.0	34.6	72.0	35.8	77.3	35.8	81.5	36.6
	SKNN	8.4	8.4	43.2	26.0	62.3	29.9	68.9	27.2	73.1	27.2	70.2	21.0
	DQN	8.7	8.7	25.8	15.6	41.2	18.8	55.4	20.6	66.4	20.6	74.8	22.2
	SE-FPMC	**25.5**	**25.5**	**51.7**	**36.6**	**68.2**	**40.2**	**76.5**	**41.2**	**81.9**	**41.2**	**85.6**	**42.0**

Table 2: Performance (%) comparison of cold-start users.

Dataset	Model	K=1		K=3		K=6		K=9		K=12		K=15	
		HR@K	MRR@K	HR@K	MRR@K	HR@K	MRR@K	HR@K	MRR@K	HR@K	MRR@K	HR@K	MRR@K
SYNC	Model 1	30.0	30.0	52.5	39.7	73.8	44.2	86.5	45.8	92.8	46.4	93.8	46.4
	Model 2	47.2	47.2	71.2	57.3	87.2	60.7	93.2	61.6	95.2	61.8	96.3	61.9
APP	Model 1	2.9	2.9	14.1	7.5	22.0	9.4	29.8	10.3	32.2	10.5	38.8	11.0
	Model 2	21.5	21.5	42.1	30.1	59.9	33.9	68.8	35.0	74.1	35.5	77.2	35.8

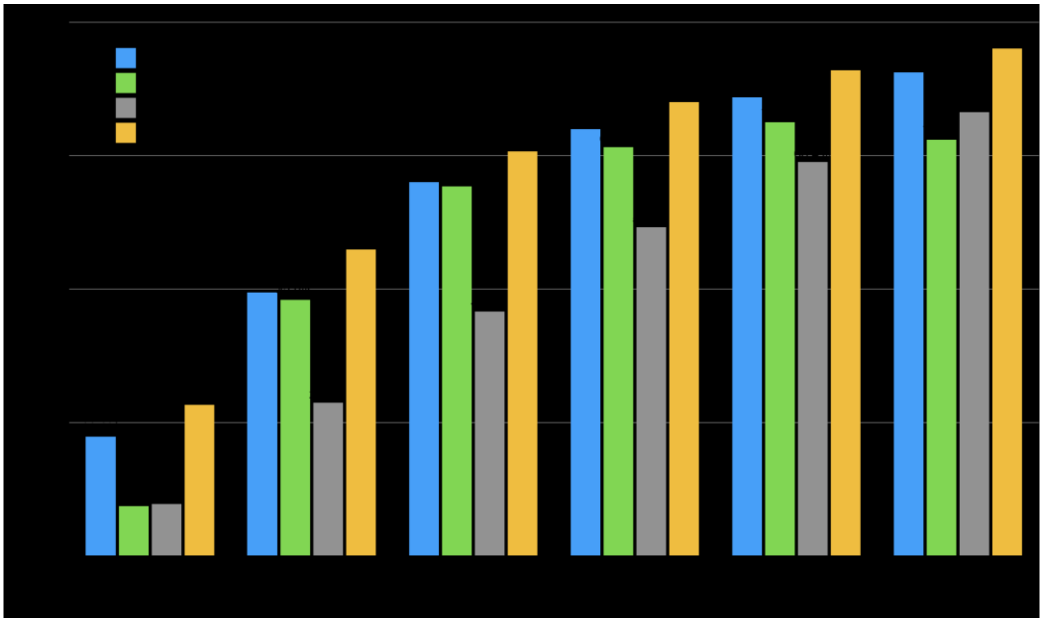


(a) HR@K

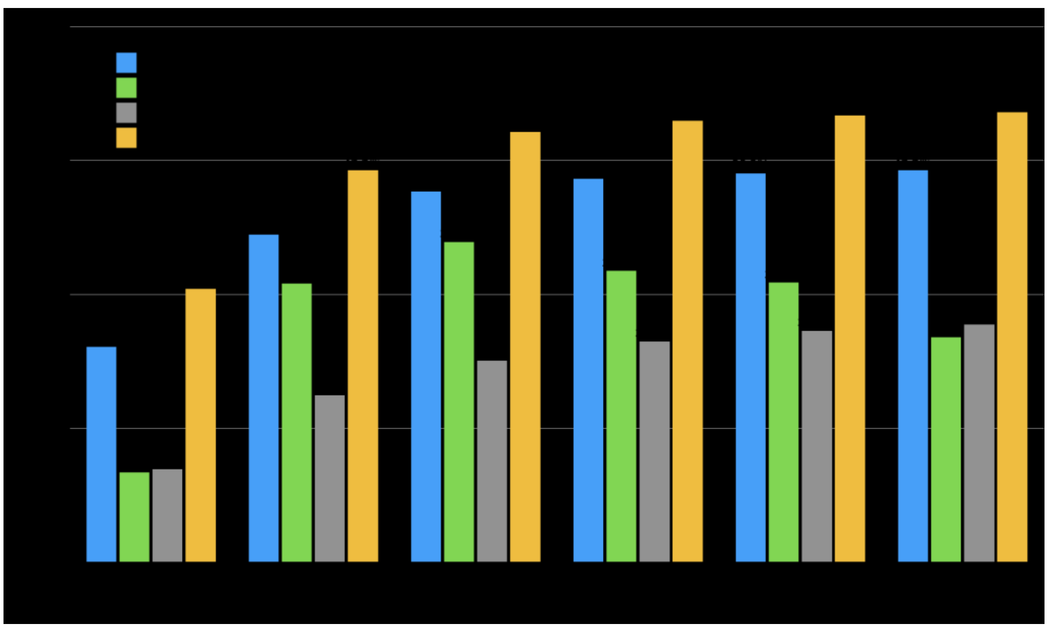


(b) MRR@K

Figure 10: Results of recommending performance on SYNC Screen dataset.

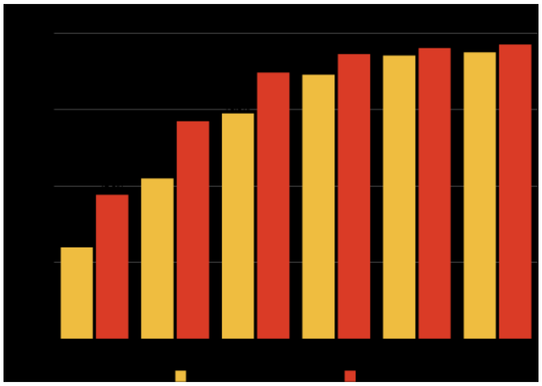


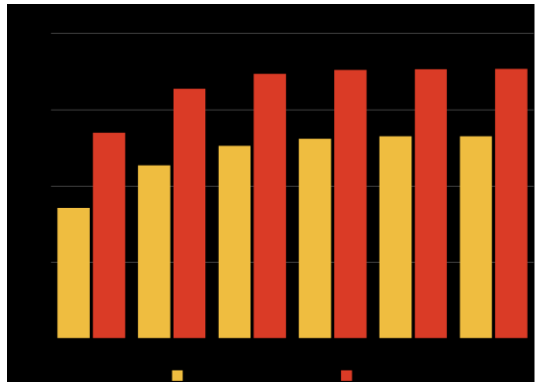
(a) HR@K



(b) MRR@K

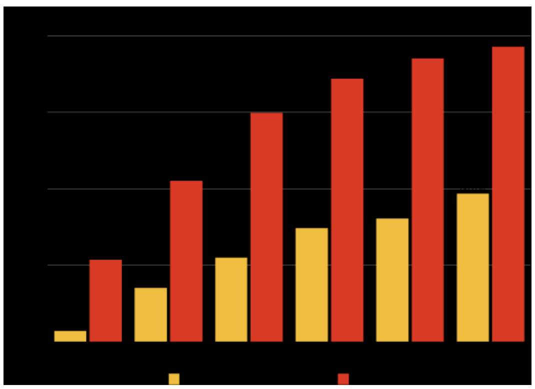
Figure 11: Results of recommending performance on APP Usage dataset.

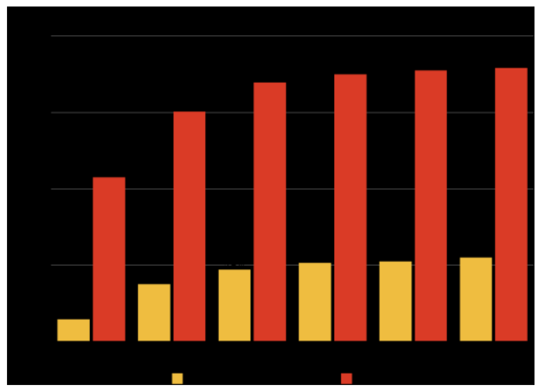




(a) HR@K of cold-start users (b) MRR@K of cold-start users

Figure 12: Results of performance of cold-start users in SYNC Screen dataset.





(a) HR@K of cold-start users (b) MRR@K of cold-start users

Figure 13: Results of performance of cold-start users in APP Usage dataset.

The tables and graphs of results show that the SE-FPMC model generated better recommending performances than the FPMC, SKNN and DQN model. On SYNC Screen dataset, the FPMC and SKNN models are also performing well. But on APP usage dataset, the SE-FPMC model outperformed a lot better than other models. For the cold-start users, the SE-FPMC model can generate recommendation on cold-start condition. On the SYNC Screen dataset, it generated 30% of successful recommendation when K=1. And the performance also increased to 47.2% when K=1 after the model is updated with available data. The cold-start performance on APP usage dataset is not as good as that on SYNC Screen dataset, it got 2.9% of successful recommendation. It still much better than randomly select an item among all the 2292 items to make recommendation. The performance also increased a lot after the model is updated with available data.

4.5 In-depth Study

4.5.1 Case Study

To better understand how the proposed SE-FPMC algorithm improves the recommending performance, we did a case study. We found a case from the SYNC Screen dataset to compare the recommending list of the FPMC algorithm and the SE-FPMC algorithm. The target user id is 127. The current using screen is ‘AUDIO/SiriusScreen’ and the ground truth is ‘AUDIO/AmScreen’. Generally, for most other users, they prefer to use ‘AUDIO/FmScreen’ and seldom use the ‘AUDIO/AmScreen’. But for the target user, the frequency of using ‘AUDIO/AmScreen’ is higher than the frequency of using ‘AUDIO/FmScreen’. The usage frequency score of ‘AUDIO/AmScreen’ is 0.5 while the usage frequency score of ‘AUDIO/FmScreen’ is 0.037. The recommend items list generated by FPMC and SE-FPMC are in Table 3.

From the table we can see, the ground truth, ‘AUDIO/AmScreen’ screen, is ranked at 7th place by FPMC algorithm but it is ranked at first place by SE-FPMC algorithm. The FPMC ranked the ‘AUDIO/FmScreen’ in the first place because most other users prefer to use ‘AUDIO/FmScreen’ to ‘AUDIO/AmScreen’. It is the general usage pattern. The usage frequency score helps the system to keep the special taste of the target user to rank the ‘AUDIO/AmScreen’ higher than ‘AUDIO/FmScreen’ so that the SE-FPMC algorithm can make a successful recommendation with K=1. The personalized usage features play a more important role in making recommendation. That’s the reason why the SE-FPMC algorithm can perform better than the FPMC algorithm in IVI or APP recommending system.

Table 3: Case study of SE-FPMC algorithm.

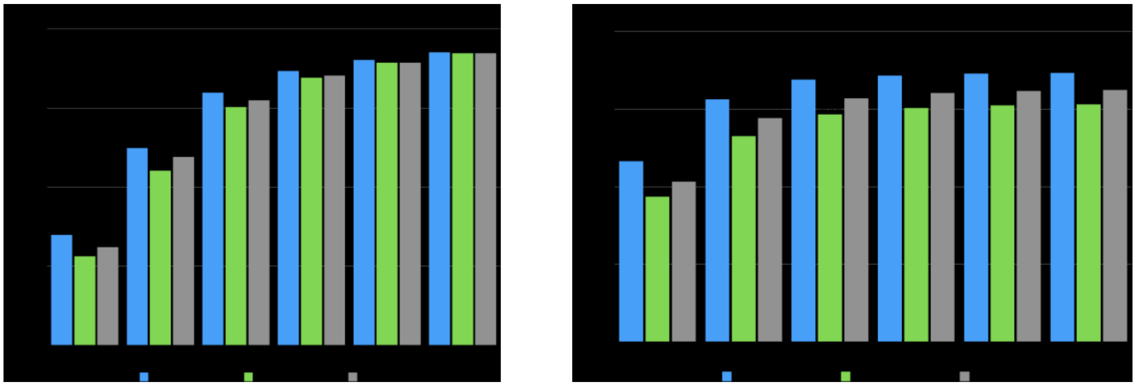
	FPMC	**SE-FPMC**
K=1	'AUDIO/FmScreen'	***'AUDIO/AmScreen'***
K=3	'AUDIO/FmScreen', 'AUDIO/SiriusErrorScreen', 'PHONE/CallInProgress'	***'AUDIO/AmScreen'***, 'AUDIO/SiriusErrorScreen’ 'ROOTSTATE/VHMView',
K=6	'AUDIO/FmScreen', 'AUDIO/SiriusErrorScreen', 'PHONE/CallInProgress', 'AUDIO/ChangeSource', 'ROOTSTATE/VHMView', 'AUDIO/SiriusBrowse'	***'AUDIO/AmScreen'***, 'AUDIO/SiriusErrorScreen', 'ROOTSTATE/VHMView', 'PHONE/PhoneMenu', 'PROJECTION/ProjectionView’, 'AUDIO/ChangeSource',
K=9	'AUDIO/FmScreen', 'AUDIO/SiriusErrorScreen', 'PHONE/CallInProgress', 'AUDIO/ChangeSource', 'ROOTSTATE/VHMView', 'AUDIO/SiriusBrowse’, ***'AUDIO/AmScreen'***, 'SETTINGS/HmiBezelDiagnosticsPropertyList’, 'NAV/MapView_Full'	***'AUDIO/AmScreen'***, 'AUDIO/SiriusErrorScreen', 'ROOTSTATE/VHMView', 'PHONE/PhoneMenu', 'PROJECTION/ProjectionView', 'AUDIO/ChangeSource', 'SETTINGS/SettingsRootView', 'PHONE/CallInProgress’, 'NAV/MapView_Full'

4.5.2 Ablation Study

When updating the model with available new users’ data, we used the knowledge distillation method to consolidate the old knowledge. In this experiment, we compare the performance of updating model with or without distillation loss to justify our design. We did the experiment on

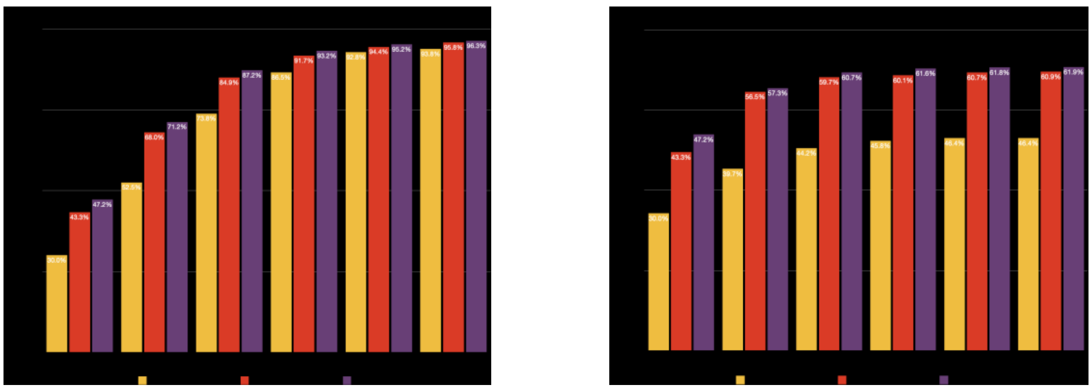
SYNC Screen dataset which contains more users. In this experiment, recommending performance for old users made by model 1 is denoted as P1, performance for old users made by model 2 which trained without distillation loss is denoted as P2, performance for old users made by model 2 which trained with distillation loss is denoted as P2_distill. Recommending performance for cold-start users made by model 1 is denoted as P3, performance for cold-start users made by model 2 which trained without distillation loss is denoted as P4, performance for cold-start users made by model 2 which trained with distillation loss is denoted as P4_distill.

The performance comparison is shown in Figure 14. and Figure 15. From the figure we can see, without distillation loss, the performance for model dropped a lot. It is a result from the catastrophic forgetting problem in incremental learning. Using the knowledge distillation loss to update the model parameters help to alleviate the catastrophic forgetting problem and also improve the performance of making recommendation for new users.



(a) HR@K of old users (b) MRR@K of old users

Figure 14: Ablation study on SYNC Screen dataset of old users.



(a) HR@K of cold-start users (b) MRR@K of cold-start users

Figure 15: Ablation study on SYNC Screen dataset of cold-start users.

Chapter 5
Conclusion and Future Works

5.1 Conclusions

In this thesis, we studied the SR systems for IVI and APP recommendation. It requires a higher weight of personalized usage patterns in making prediction. The usage frequency score proposed in this thesis help to extract personalized usage patterns from the statistics of user's history usage sessions. The proposed SE-FPMC algorithm can help to make recommendation for users under cold-start condition. Once the usage data of cold-start users become available, the model can be updated to continually learn from new data. The knowledge distillation method is used to alleviate the catastrophic forgetting problem in the incremental learning. Experiment on 2 dataset shows that the proposed SE-FPMC algorithm makes a better recommending performance for both old users and cold-start users.

5.2 Limitations

Although the proposed SE-FPMC algorithm performs better than other existing SR algorithms, it still not making recommendation based on the entire session. The SE-FPMC algorithm still using the Markov Chain model which is a memoryless mathematical model. Some of the information encoded in the session will be lost. The newly proposed model also requires enough data to train the model parameters. If the target user only has a small number of sessions, the model cannot make full use of the existing data but will still treat the user as a cold-start user. The recommending performance for the user will not be as good as other users who have more history usage sessions.

5.3 Future Works

The SE-FPMC algorithm can be further improved by using the high order Markov Chain which can leverage the entire session's information. And the few-shot learning techniques can be used to deal with users who only have a small number of sessions. The SE-FPMC should leverage more incremental learning techniques to further alleviate the catastrophic forgetting problem and make the SR system to be a continual learning system.

References

[1]Rendle, Steffen, Christoph Freudenthaler, and Lars Schmidt-Thieme. "Factorizing personalized markov chains for next-basket recommendation." In *Proceedings of the 19th International Conference on World Wide Web*, pp. 811-820. 2010.

[2]Qiu, Ruihong, Jingjing Li, Zi Huang, and Hongzhi Yin. "Rethinking the item order in session-based recommendation with graph neural networks." In *Proceeding of the 28th ACM International Conference on Information and Knowledge Management*, pp. 579-588. 2019.

[3]He, Ruining, and Julian McAuley. "Fusing similarity models with markov chains for sparse sequential recommendation." In *2016 IEEE 16th International Conference on Data Mining (ICDM)*, pp. 191-200. IEEE, 2016.

[4]Cheng, Chen, Haiqin Yang, Michael R. Lyu, and Irwin King. "Where you like to go next: Successive point-of-interest recommendation." In *Twenty-Third International Joint Conference on Artificial Intelligence*. 2013.

[5]Pengfei, Jiafeng Guo, Yanyan Lan, Jun Xu, Shengxian Wan, and Xueqi Cheng. "Learning hierarchical representation model for next basket recommendation." In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 403-412. 2015.

[6]Shani, Guy, David Heckerman, Ronen I. Brafman, and Craig Boutilier. "An MDP-based recommender system." *Journal of Machine Learning Research 6*, no. 9 (2005).

[7]Chung, Junyoung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. "Empirical evaluation of gated recurrent neural networks on sequence modeling." *arXiv preprint arXiv:1412.3555* (2014).

[8]Hidasi, Balázs, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. "Session-based recommendations with recurrent neural networks." *arXiv preprint arXiv:1511.06939 (2015)*.

[9]Tan, Yong Kiam, Xinxing Xu, and Yong Liu. "Improved recurrent neural networks for session-based recommendations." In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*, pp. 17-22. 2016.

[10].Jannach, Dietmar, and Malte Ludewig. "When recurrent neural networks meet the neighborhood for session-based recommendation." In *Proceedings of the Eleventh ACM Conference on Recommender Systems*, pp. 306-310. 2017.

[11]Tang, Jiaxi, and Ke Wang. "Personalized top-n sequential recommendation via convolutional sequence embedding." In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pp. 565-573. 2018.

[12]Chen, Xu, Hongteng Xu, Yongfeng Zhang, Jiaxi Tang, Yixin Cao, Zheng Qin, and Hongyuan Zha. "Sequential recommendation with user memory networks." *In Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pp. 108-116. 2018.

[13]Huang, Jin, Wayne Xin Zhao, Hongjian Dou, Ji-Rong Wen, and Edward Y. Chang. "Improving sequential recommendation with knowledge-enhanced memory networks." In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pp. 505-514. 2018.

[14]Ebesu, Travis, Bin Shen, and Yi Fang. "Collaborative memory network for recommendation systems." In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pp. 515-524. 2018.

[15]Chen, Lei, Le Wu, Richang Hong, Kun Zhang, and Meng Wang. "Revisiting graph based collaborative filtering: A linear residual graph convolutional network approach." In *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 01, pp. 27-34. 2020.

[16]Wu, Shu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan. "Session-based recommendation with graph neural networks." In *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, pp. 346-353. 2019.

[17]Lim, Wei Yang Bryan, Nguyen Cong Luong, Dinh Thai Hoang, Yutao Jiao, Ying-Chang Liang, Qiang Yang, Dusit Niyato, and Chunyan Miao. "Federated learning in mobile edge networks: A comprehensive survey." *IEEE Communications Surveys & Tutorials 22*, no. 3 (2020): 2031-2063.

[18]Brazdil, Pavel, Christophe Giraud Carrier, Carlos Soares, and Ricardo Vilalta. *Metalearning:*

Applications to Data Mining. Springer Science & Business Media, 2008.

[19]Finn, Chelsea, Pieter Abbeel, and Sergey Levine. "Model-agnostic meta-learning for fast adaptation of deep networks." In *International Conference on Machine Learning*, pp. 1126-1135. PMLR, 2017.

[20]Vinyals, Oriol, Charles Blundell, Timothy Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. "Matching networks for one shot learning." *arXiv preprint arXiv:1606.04080* (2016).

[21]Ravi, Sachin and Larochelle, Hugo. Optimization as a model for few-shot learning. In *International Conference on Learning Representations*, 2017.

[22]Du, Zhengxiao, Xiaowei Wang, Hongxia Yang, Jingren Zhou, and Jie Tang. "Sequential scenario-specific meta learner for online recommendation." In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 2895-2904. 2019.

[23]Snell, Jake, Kevin Swersky, and Richard S. Zemel. "Prototypical networks for few-shot learning." *arXiv preprint arXiv:1703.05175* (2017).

[24]Sung, Flood, Yongxin Yang, Li Zhang, Tao Xiang, Philip HS Torr, and Timothy M. Hospedales. "Learning to compare: Relation network for few-shot learning." In *Proceedings of the IEEE Conference on Computer Cision and Pattern Recognition*, pp. 1199-1208. 2018.

[25]Santoro, Adam, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. "Meta-learning with memory-augmented neural networks." In *International Conference on Machine Learning*, pp. 1842-1850. PMLR, 2016.

[26]Munkhdalai, Tsendsuren, and Hong Yu. "Meta networks." In *International Conference on Machine Learning*, pp. 2554-2563. PMLR, 2017.

[27]Finn, Chelsea, Pieter Abbeel, and Sergey Levine. "Model-agnostic meta-learning for fast adaptation of deep networks." In *International Conference on Machine Learning*, pp. 1126-1135. PMLR, 2017.

[28]Yu, Rong, and Peichun Li. "Toward Resource-Efficient Federated Learning in Mobile Edge Computing." *IEEE Network 35*, no. 1 (2021): 148-155.

[29]Xu, Jingwei, Yuan Yao, Hanghang Tong, Xianping Tao, and Jian Lu. "Ice-breaking: Mitigating

cold-start recommendation problem by rating comparison." In *Twenty-Fourth International Joint Conference on Artificial Intelligence*. 2015.

[30]Mi, Fei, Xiaoyu Lin, and Boi Faltings. "Ader: Adaptively distilled exemplar replay towards continual learning for session-based recommendation." In *Fourteenth ACM Conference on Recommender Systems*, pp. 408-413. 2020.

[31]Hinton, Geoffrey, Oriol Vinyals, and Jeff Dean. "Distilling the knowledge in a neural network." *arXiv preprint arXiv:1503.02531* (2015).

[32]Cantador, Iván, Ignacio Fernández-Tobías, Shlomo Berkovsky, and Paolo Cremonesi. "Cross-domain recommender systems." In *Recommender Systems Handbook*, pp. 919-959. Springer, Boston, MA, 2015.

[33]Fernández-Tobías, Ignacio, Iván Cantador, Marius Kaminskas, and Francesco Ricci. "Cross-domain recommender systems: A survey of the state of the art." In *Spanish Conference on Information Retrieval*, pp. 1-12. sn, 2012.

[34]Schein, Andrew I., Alexandrin Popescul, Lyle H. Ungar, and David M. Pennock. "Methods and metrics for cold-start recommendations." In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 253-260. 2002.

[35]Hu, Liang, Jian Cao, Guandong Xu, Longbing Cao, Zhiping Gu, and Can Zhu. "Personalized recommendation via cross-domain triadic factorization." In *Proceedings of the 22nd International Conference on World Wide Web*, pp. 595-606. 2013.

[36]Fernández-Tobías, Ignacio, and Iván Cantador. "Exploiting Social Tags in Matrix Factorization Models for Cross-domain Collaborative Filtering." In *CBRecSys@ RecSys*, pp. 34-41. 2014.

[37]Song, Tianhang, Zhaohui Peng, Senzhang Wang, Wenjing Fu, Xiaoguang Hong, and S. Yu Philip. "Based cross-domain recommendation through joint tensor factorization." In *International Conference on Database Systems for Advanced Applications*, pp. 525-540. Springer, Cham, 2017.

[38]Wang, Senzhang, Xia Hu, Philip S. Yu, and Zhoujun Li. "MMRate: inferring multi-aspect diffusion networks with multi-pattern cascades." In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1246-1255. 2014.

[39]Chen, Wei, Wynne Hsu, and Mong Li Lee. "Making recommendations from multiple domains." In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 892-900. 2013.

[40]Zhou, Xiangmin, Lei Chen, Yanchun Zhang, Dong Qin, Longbing Cao, Guangyan Huang, and Chen Wang. "Enhancing online video recommendation using social user interactions." *The VLDB Journal 26,* no. 5 (2017): 637-656.

[41]Lin, Jovian, Kazunari Sugiyama, Min-Yen Kan, and Tat-Seng Chua. "Addressing cold-start in app recommendation: latent user models constructed from twitter followers." In *Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 283-292. 2013.

[42]Jiang, Meng, Peng Cui, Xumin Chen, Fei Wang, Wenwu Zhu, and Shiqiang Yang. "Social recommendation with cross-domain transferable knowledge." *IEEE Transactions on Knowledge and Data Engineering 27,* no. 11 (2015): 3084-3097.

[43]French, Robert M. "Catastrophic forgetting in connectionist networks." *Trends in Cognitive Sciences 3,* no. 4 (1999): 128-135.

[44]McCloskey, Michael, and Neal J. Cohen. "Catastrophic interference in connectionist networks: The sequential learning problem." In *Psychology of Learning and Motivation,* vol. 24, pp. 109-165. Academic Press, 1989.

[45]Kirkpatrick, James, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan et al. "Overcoming catastrophic forgetting in neural networks." In *Proceedings of the National Academy of Sciences 114,* no. 13 (2017): 3521-3526.

[46]Li, Zhizhong, and Derek Hoiem. "Learning without forgetting." *IEEE Transactions on Pattern Snalysis and Machine Intelligence 40,* no. 12 (2017): 2935-2947.

[47]Rebuffi, Sylvestre-Alvise, Alexander Kolesnikov, Georg Sperl, and Christoph H. Lampert. "icarl: Incremental classifier and representation learning." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2001-2010. 2017.

[48]Maltoni, Davide, and Vincenzo Lomonaco. "Continuous learning in single-incremental-task scenarios." *Neural Networks* 116 (2019): 56-73.

[49]Welling, Max. "Herding dynamical weights to learn." In *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 1121-1128. 2009.