ECTI Transactions on Computer and Information Technology
Journal homepage: https://ph01.tci-thaijo.org/index.php/ecticit/
Published by the ECTI Association, Thailand, ISSN: 2286-9131

An Adaptive Temporal-Concept Drift Model for Sequential Recommendation

Tipajin Thaipisutikul[1]

ABSTRACT

Recently, owing to the great advances in Web 2.0 and mobile devices, various online commercial services have emerged. Recommendation systems play an important role in dealing with abundant product information from massive numbers of online e-commerce transactions. Providing an accurate recommendation at the correct time to customers can contribute to a surge in business success. In this paper, an adaptive temporal-concept drift learning-based recommendation system, ATCRec, is developed for precisely tackling the sequential recommendation problem. We embed sequences of items into the latent spaces and learn both general preferences and sequential patterns concurrently via a recurrent neural network. Specifically, ATCRec captures dynamic changes in the temporal and concept drift contexts by modifying the gate units in a traditional recurrent neural network. The proposed model provides a unified and flexible network structure to learn and reveal the opaque variation of user preferences over time. We evaluate the robustness and performance of ATCRec on two real-world datasets, and the experimental results demonstrate that ATCRec consistently outperforms existing sequential recommendation approaches on various metrics. This indicates that integrating users' temporal information and concept drift variation through time are indispensable in improving the performance of recommendation systems.

Article information:

Keywords: Sequential Recommendation , Recurrent Neural Network, Temporal Analysis, Concept Drift Analysis

Article history:
Received: April 18, 2021
Revised: August 28, 2021
Accepted: March 26, 2022
Published: June 11, 2022
(Online)

DOI: 10.37936/ecti-cit.2022162.248019

1. INTRODUCTION

In the last decade, due to the great advancs in Web 2.0 and the popularity of mobile devices, the abundant information available has abruptly changed the world as we know it. For example, some retailers are providing a million products at one time in one store. Several businesses have achieved extraordinary success in making a profit while maintaining customer loyalty via their recommendation systems (RS). Actually, recommendation systems are decision support systems which often add more value to both consumers and commercial sites. Users can easily find the right items to consume at the right time, and it also reduces the time required to find the specific items.

Developing an adaptive model to learn the dynamic changes of tremendous amounts of customer data is a challenging task since the proposed model should incorporate the asymmetric transition of time between interactions and the dynamic changes of user preferences. Traditional methods, such as collaborative filtering (CF) and matrix factorization (MF), have been used for forecasting unobserved ratings in the user-item matrix. The CF-based techniques generally derive the similarity among users and then predict the item rating based on similar users. The MF-based approach decomposes the user-item rating matrix into user and item latent factor spaces to predict future item ratings. Though CF- and MF-based models are successfully applied in some applications, both techniques still have a limitation in terms of capturing the temporal dynamics of user preferences.

Some previous sequential recommendation studies focused on sequential recommendation which typically relies on either the Markov Chain (MC) or a recurrent neural network (RNN) to capture sequential patterns and forecast the target user's next action based on prior actions. MC-based models extract sequential patterns by learning the transition graph

[1]The author is with Faculty of Information and Communication Technology Mahidol University Thailand, E-mail: tipajin.tha@mahidol.ac.th

over items, and derive the probability of the next item to consume based on users' historical transit traces. Various studies, such as FPMC [1], and FPMC-LR [2], discuss the variations or extensions of MC.

Recently, the RNN-based models have attracted much attention due to their high accuracy and generalization. Examples of these are GRU4Rec [3] and Time-LSTM [4]. However, these approaches still suffer from overlooking dynamic changes in users' preferences. Also, several prior models learned the evolution of auxiliary contexts and the actual interactions separately, which results in lower accuracy. These learned models are intended to process users' sequence data with the regular elapsed time between successive actions. However, they are not appropriate for real-life applications since the elapsed time between actions can vary from seconds to years. The irregularity of time transitions between users' records and the complex hidden patterns are fatal problems for current RNN-based models.

To demonstrate the challenges of modeling the sequential data, we present the real scenarios as examples in Fig.1. We use movie-watching history as an example. Assume there are 10 movies denoted by m_1, m_2, ..., m_{10}. The movies m_1 to m_4 are associated with an action genre. The movies m_5 to m_7 are associated with horror movies. Movies m_8 to m_{10} are associated with the fantasy genre. Each user has historical watching records associated with timestamps in order. For simplicity, the watching history sequence of user u_i is denoted as S_{ui}.

Case Scenario 1 (The effect of Sequential patterns): We assume that not only the user's general preference benefits the recommendation results, but the sequential patterns are also another key to improve the top-N performance. General preferences symbolize static behaviors or long-term interests, while sequential patterns symbolize the dynamic behaviors or short-term interests where the next action more likely depends on the actions recently taken. For example, users u_1 and u_3 have their historical actions denoted as $S_{u1} =< m_1, m_2, m_3, m_4, m_8, m_9 >$ and $S_{u3} =< m_2, m_3, m_4, \ldots, m_5, m_6 >$. From the existing dataset, our goal is to recommend the movies that user u_2 is going to watch in the future. The sequential relationships are analyzed and used in the recommendation process. From watching sequence $S_{u2} =< m_2, m_3 >$, user u_2 will have a higher chance of getting m_4 as the next movie to watch due to the sequential patterns mined from S_{u1} and S_{u3} $\{m_2, m_3 \rightarrow m_4\}$. In this case, a recommendation system which only considers the general preferences may miss the chance to recommend m_4 to u_2 after he watched m_2 and m_3, since there are many possible action movies to recommend.

Case Scenario 2 (The effect of temporal irregularity in user transitions): We assume that the latest movies a user has watched will have a higher impact on the future movies watched than the historical records. For example, suppose u_3 watched $< m_2, m_3, m_4 >$ in the last year. Later, u_3 has just watched $< m_5, m_6 >$. This results in a long elapsed time (Δt) between m_4 and m_5. Therefore, the recommendation system should place more emphasis on the recent records such as m_5 and m_6 than on m_2, m_3 and m_4 in the prediction process. As a result, a movie m_7 which has similar content to m_5 and m_6 is returned as a prediction. In this case, a recommendation system that only considers the general preferences and sequential patterns will probably miss the chance to recommend m_7 to u_3 after he watched $< m_5, m_6 >$, since the system may recommend the wrong movies to the users due to the ignorance of the impact of recent movies.

Case Scenario 3 (The effect of concept drifts in user sequences): We assume that the characteristic of movies is stationary, while there is a concept drift in a particular user since some people might have changed their preferences over time. In most sequential recommendation applications, patterns and relations often evolve over time. Thus, we refer to concept drift as a non-stationary learning problem over time that could capture the gradual change, sudden change, or lack of change in the user's preference [31, 32]. In this manuscript, the concept refers to the movie genre. So, if a user has watched many action movies in the past, but he started to watch fantasy movies recently, the proposed model should have the capability to capture the change in the user's preference for fantasy movies and recommend more films related to the fantasy genre. For example, u_1 was previously interested in watching action movies. However, u_1 was recently interested in fantasy movies. We should recommend m_{10} to u_1 since he most recently interested in fantasy movies more than action movies. In this case, a recommendation system that only considers the general preferences, sequential patterns, and temporal irregularity will probably miss the chance to recommend m_{10} to u_1 after he watched $< m_8, m_9 >$, since u_1 has regular elapsed time (Δt) but he has changed his preferences at some point in time.

Motivated by the above challenges and the unique characteristics of the sequential data, in this study, we develop a novel recommendation system named the **A**daptive **T**emporal-**C**oncept drift model learning for sequential **Rec**ommendation (abbreviated as ATCRec) to handle irregular time intervals and user preference driven sequential data. Two design schemes are considered to build such an applicable learning module. First, the time irregularities between adjacent inputs in a sequence are transformed into a proper weight. Second, the change in users' preferences, known as concept drift, is captured by a similarity calculation between successive inputs and included in the model learning process.

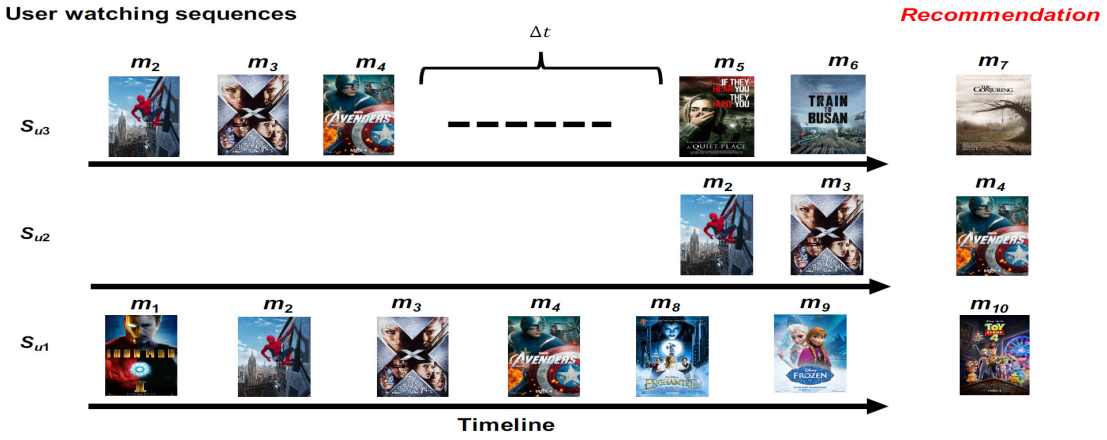


Fig.1: *An example of recommendation results made for random users.*

In this paper, we denote the context values describing the granularity of the adaptive temporal-concept drift as ATContext. We incorporate ATContext into the subspace decomposition of the short- and long-term memory units in the Long Short-Term Memory (LSTM) model. More specifically, we discount the memory cell in a way so that the more time elapses or concept drifts, the smaller the effect of the previous memory on the current output.

The contributions of our work are as follows:

- We define and formalize a significant factor, adaptive temporal-concept context (ATContext), from user sequence data which is unobserved by previous studies on the sequential recommendation. The drift of concept and the influence of temporal dynamics are quantized by ATContext with similarity and temporal calculations between successive items in sequence.
- The proposed ATCRec not only minimizes the inefficiency of traditional RNN, but also constructs a framework to handle concept drift and temporal dynamic contexts between consecutive records. In particular, we decompose the memory cell into short- and long-term effects in the proposed learning model. The short-term counterpart is modified by the discounted weight of ATContext and then combined with the long-term counterpart before entering the recurrent neural network units.
- Extensive experiments were conducted on two real datasets from GroupLens [30] to show the applicability of the proposed system. Our experimental results shows that ATCRec has greater improvement on the performance compared to the existing state-of-the-art models. Also, we demonstrate the superiority and generalization of our proposed model via various standard metrics.

The remainder of this paper is organized as follows. Section 2 provides details about some related work. Sections 3 and 4 present the preliminaries and proposed model, ATCRec, respectively. Section 5 provides the experimental settings, evaluations, and results. Finally, we deliver some concluding remarks in Section 6.

2. RELATED WORK

In this section, we first review the literature related to recommendation tasks including several CF-based and MF-based methods. Then, we introduce the prior context-based and sequential recommendation models and present the differences between our proposed model and existing work.

Collaborative filtering (CF) [21, 25] is a common and successful recommendation technique based on user interests. Sarwar et al. [5] proposed an item-based CF that calculates the similarity between items, and then all missing ratings are returned and used for recommendation. Ricci et al. [6] proposed a user-based CF that computed the similarity of users based on provided ratings, and returned recommendation lists which are similar to user's friends. Zhang et al. [7] employed availability evaluation and trust evaluation modules to improve CF performance. Gupta et al. [8] utilized a weighted scheme to combine CF with a user-side information for item rating prediction to solve the user cold start problem. Melville et al. [9] combined the benefit of content and collaboration-based models into a unified framework to enhance the personalized prediction. Zhao et al. [10] introduced a MapReduce environment as a main implementation for large data processing using item-based CF recommendation.

With the impressive achievements of latent factor models (LFM) in many domains, various approaches based on Matrix Factorization (MF) [11] have been used to leverage user-item rating recommendation systems. Actually, the rating patterns can be inferred through the factor vectors extracted from matrix decomposition. Koren et al. [12] utilized the inner product of user and item low-dimensional matrices for rating prediction. Thai-Nghe et al. [13] modeled an effective recommendation approach implicitly including the latent factors, and proposed a tensor factorization method with temporal effect. Abdi et al. [14]

included contextual information in MF approaches to improve recommendation performance. Xiong et al. [15] analyzed the deviation degree of ratings of users and items, and proposed the concept of user- and item-centrality. Based on the rating centrality, the authors measure the reliability of each user rating and provide an optimized MF recommendation algorithm. Jamali et al. [17] incorporated the mechanism of trust propagation into a model-based approach for social network recommendation. Liang et al. [18] jointly modeled a co-factorization model of the user-item interaction matrix and the item-item co-occurrence matrix to improve next item recommendation.

Temporal context recommendation attempts to improve general recommendation by incorporating a timestamp associated with data records. Ding et al. [19] utilized the time weight function to assign decaying weights to earlier rated items when calculating similarities. The proposed algorithm extends clustering to discriminate between different kinds of items for tracing the change of user-purchased item interest. The authors also discuss a personalized decay factor according to the user's own purchase behaviour. Koren et al. [20] achieved rating prediction on Netflix data by integrating temporal signals, and proposed a model to track the time changing behaviour throughout the life span of the data. This enables the proposed model to exploit the relevant components of all data instances, while discarding only what is modelled as being irrelevant. Shi et al. [21] utilized temporal information such as timestamp associated with ratings of the user-item to put more emphasis on recent ratings. The authors assume that the most recent actions will have more impact on future actions than actions which occurred in the past.

Since the user's preference is dynamic and varies over time, many previous studies on preference context recommendation have tried to detect concept drift to capture the change point in user preferences along the timeline. Most studies capture user preferences by using a clustering-based method. Lo et al. [22] developed a temporal approach for tracking concept drift in each user latent vector. Cheng et al. [23] proposed a time aware-based user interest model to improve document recommendation by distinguishing the main interests from the minor user interests. Jiang et al. [24] recommended target items to target users by forming user-rated target items at a nearby current time as a group. The authors utilized a predefined time function for different items to assign different weights. Then, all items belonging to the users in this group are ranked for prediction.

Dynamically sequential recommendation models always rely on Markov Chains (MC) to capture sequential signals. FPMC [1] is a combination of MF and MC to achieve prediction in the next-basket recommendation by modeling the third-order as a cube to represent the transitions among items made by users. Later, Yu et al. [26] extended FPMC by exploiting aggregation operations such as mean pooling to process complicated interactions. He et al. [27] modeled the interaction between users and items by embedding items into a transition space while users are modelled as translation vectors. The prediction results are calculated by operating the transition space with translation vectors to get the vector direction of the next item. Hidasi et al. [3] applied a parallel recurrent neural network to better model the sequential input data. Fossil [28] simplified the high-order Markov chains by exploiting a weighted sum aggregation function over preceding item latent representations. Zhu et al. [4] proposed an extension of LSTM to include an additional gate to process the time intervals between consecutive inputs and to automatically decay the incoming inputs based on the time gap.

Recently, a number of studies have proposed models for time-aware recommendation based on user preference driven models. For example, Roveri et al. [33] proposed discrete-time Markov Chains (DTMCs) under concept drift to detect three different change mechanisms in data. Neammanee et al. [34] introduced a Time-Aware Recommender System (TARS) to detect user preferences with a Fuzzy C-Mean algorithm and entropy to find the preference change in the rating timeline. Xu et al. [35] offered an anomaly detection problem of smart city services and distinguished different anomalies of communication to protect data privacy of users by integrating the concept-drift concept into the proposed method. Santos et al. [36] presented an approach to capture temporal novelties in social networks and to indicate change points driven by real-world incidents that influence public opinion.

However, our proposed model (ATCRec) differs from the above prior research since our learning model simultaneously learns the temporal and concept-drift contexts and adjusts the recommendation results based on the adaptive time and concept-drift via a modified LSTM unit. Consequently, our proposed model adaptively captures the changes during the longitudinal user behaviors, and provides promising recommendation results.

3. PRELIMINARIES

In this section, we introduce the general definition of sequential recommendation and formulate the problem. The notations used throughout this study are summarized in Table 1.

Let $U = \{u_1, u_2, \ldots, u_i, \ldots, u_m\}$ be a set of users, where $1 \leqslant i \leqslant m$. Let $M = \{m_1, m_2, \ldots, m_j, \ldots, m_n\}$ be a set of items, where $1 \leqslant \mathrm{j} \leqslant \mathrm{n}$. Each m_j is associated with its genres which are represented by a one-hot encoding vector x_{mj}. For example, the genre vector $x_{mj} = [0, 1, 0, 1]$ of m_j means that there are a total of four genres

Table 1: *Notation used in the ATCRec model.*

Symbol	Gross
U, M	A set of users, A set of items
u_i	A user in U
m_j	An item in M
Δt	Elapsed time between successive inputs
Δc	Similarity score between successive inputs
Δct	ATContext between successive inputs
$S_u^{\Delta ct}$	A user ATContext sequence
x_{mj}	The genre vector describing the movie j m_j
a_k	An action pair (m_k, t_k)
S_u	A user action sequence
$E(m_j)$	A latent vector of item m_j
$E(S_u)$	A user sequence of all movies' latent vectors

in the database and m_j belongs to genres 2 and 4. The genres for the movies are indicated with 1 in the genre vector.

The actions of user u are recorded as a user sequence $S_u = <a_1, \ldots, a_k, \ldots, a_{|S_u|}>$. The action a_k in S_u represents a pair (m_k, t_k) which denotes u_i selected item m_k at time t_k and $t_{|S_u|} \leqslant t_{|S_{u-1}|} \leqslant \ldots \leqslant t_1$. For example, suppose S_u is a movie watching sequence of user u, then action a_k is watching movie m_k at time t_k.

4. THE PROPOSED MODEL: ATCREC

We aimed to answer the following questions: (i) How can we capture the temporal and concept-drift context of user preferences? (ii) What is the principle learning model to be developed to enhance the top-N prediction performance? and (iii) How can we incorporate the temporal and concept-drift context into the learning model? In this section, we describe the proposed Adaptive Temporal-Concept drift learning model for Sequential Recommendation (ATCRec) that aims to include the temporal-concept drift context (ATContext) for dynamically predicting possible next items. To facilitate the explanation of the proposed method, in the remaining part of this paper, we will use movie watching data to discuss the ATCRec architecture and related components. The user action sequence will be the movie watching sequence and the selected items will be movies watched. Fig. 2 presents an overview of the architecture of ATCRec which consists of four major components: 1) ATContext derivation, 2) Model embedding, 3) Model learning and training, and 4) Recommendation.

4.1 ATContext derivation

In this section, we first describe how we can extract the temporal context (Δt) and concept-drift context (Δc). Then introduce the ATContext (Δct). Since the user action sequence (S_u) is normally longitudinal, the relationship and dependencies between items in the data sequence must be captured in order to learn a more effective representation. Note that in the movie watching dataset, as shown in Table 2,

Table 2: *Example of attributes used in ATCRec (Genre A denotes Action, C denotes Crime, and H denotes Horror).*

u_id	*m_id*	*rating*	*time_stamp*	*title*	*genre*
1	8125	4	08/31/16	Heat	A|*C*
1	6680	3	11/25/17	Flipper	C|*H*

we consider six attributes: *user_id*, *movie_id*, *rating*, *time_stamp*, *title*, and *genre*.

The definition of the temporal context (Δt) is given in Eq. 1.

$$\Delta t(m_i, m_{i+1}) = \frac{1}{log(e + (t_i - t_{i+1}))} \tag{1}$$

Obviously, the temporal context will be framed in the same unit of time within the same dataset, either in seconds, minutes, or hours, while processing. This important process can prevent large numerical values when there is a very long time between two consecutive items, e.g. many years. For example, if the recent temporal context of a particular user is around 0.99, this indicates that the user continuously interacts with the system. Hence, these selected items provide a source to allow studying the progression of the condition. On the contrary, if the recent temporal context of a particular user is around 0.01, this indicates that there are years between two successive items. In this case, dependency on the earlier information should not play an active role in predicting the current recommendation output. In summary, we construct a temporal weight function by giving high weight to the items selected near the current time. The Δt is transformed into a proper weight for further discounting short-term memory content described in the later section.

On the other hand, since users normally change their preference over time due to the shifting interests or the fluctuation of item popularities, it results in the inconsistencies in user behavior. Different users might have drifted in different ways at each point in time. It is essential to keep track of multiple changing points in a longitudinal sequence. In this study, we extend the content-based and collaborative filtering approaches to dynamically weight the concept-drift Δc transition between two successive items. In order to track the transition of concept drift over time, the genre of each item is encoded as a one-hot encoding genre vector.

First, the content similarity sim_c between two items is derived from two genre vectors input into Eq. 2. The $x^i =< x_1, x_2, \ldots, x_g >$ and $x^j =< x_1^{'}, x_2^{'}, \ldots, x_g^{'} >$ denote the genre vectors of items m_i and m_j, respectively. Notice that the size (dimension) of a one-hot vector is set to g.

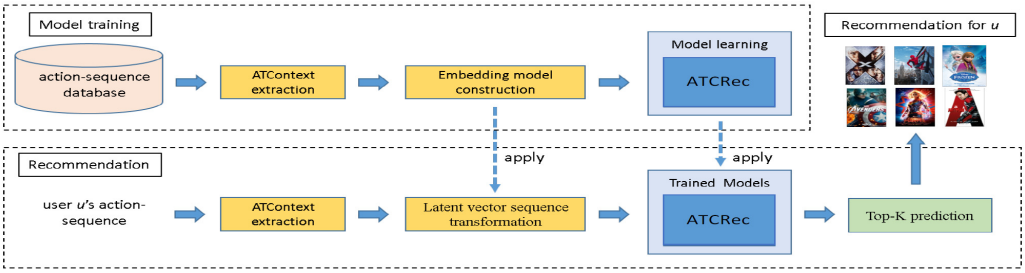


Fig.2: *The architecture of the ATCRec system.*

$$sim_C(m_i, m_j) = x^i x^j = \frac{\sum_{i=1}^{g} x_i x_i'}{\sqrt{\sum_{i=1}^{g} x_i^2}\sqrt{\sum_{i=1}^{g} x_i'^2}} \quad (2)$$

Second, we also find the similarity between two items in the user rating aspect. When all users have rated two items together, we can utilize the rating value to estimate the similarity of these two items. Suppose U_{ij} is a set of users which have rated m_i and m_j together, and r_{ui} and r_{uj} are the values that user u rated to m_i and m_j respectively. The rating similarity sim_R between two items can be derived by Eq. 3.

$$sim_R(m_i, m_j) = \frac{\sum_{u \in U_{ij}} (|r_{ui} - \mu_i|)(|r_{uj} - \mu_j|)}{\sqrt{\sum_{u \in U_{ij}} (r_{ui} - \mu_i)^2}\sqrt{\sum_{u \in U_{ij}} (r_{uj} - \mu_j)^2}} \quad (3)$$

μ_i and μ_j are the mean rating value of m_i and m_j, respectively.

Finally, we combine sim_C and sim_R similarity values to derive the transition value of the concept-drift context (Δc) with Eq. 4 where α is specified by the user.

$$\Delta c(m_i, m_{i+1}) = \alpha sim_C(\kappa) + (1 - \alpha) sim_R(\kappa) \quad (4)$$

κ denotes (m_i, m_{i+1}).

ATContext (ct): For a user $u \in U$, given his/her action sequence S_u=$\{a_1, \ldots, a_k, \ldots, a_{|S_u|}\}$ where $a_k = (m_k, t_k)$, by calculating all temporal and concept-drift contexts, i.e., $\forall \Delta t_k, \forall \Delta c_k$ where 1 <= k <= $|S_u|$, we can derive the temporal sequences $\{S_u^{\Delta t} = \Delta t_1, \ldots, \Delta t_k, \ldots, \Delta t_{S_u - 1}\}$ and concept-drift $S_u^{\Delta c} = \{\Delta c_1, \ldots, \Delta c_k, \ldots, \Delta c_{S_u - 1}\}$ directly. The ATContext is defined as in Eq. 5.

$$\Delta ct(m_i, m_{i+1}) = min(\Delta c(m_i, m_{i+1}, \Delta t(m_i, m_{i+1}))) \quad (5)$$

The ATContext sequence of user u is defined as $S_u^{\Delta c}$=$\{\Delta ct_1, \ldots, \Delta ct_k, \ldots, \Delta ct_{|S_u|}\}$. The reasons why we construct Δct from the minimum values between Δc and Δt are twofold as follows. First, we aim to capture the nature of each sequence dynamically. Since each user's behavior varies across the whole dataset, we need to combine the context weights Δt and Δc properly and differently for a particular user. For example, user A continuously watches movies everyday. At some point in time, his preference changed from action movies to romance movies. His $S_A^{\Delta t}$ can be denoted as $\{1.0, 1.0, 1.0, \ldots, 1.0\}$ and his $S_A^{\Delta c}$ can be denoted as $\{1.0, 1.0, 0.3, \ldots, 1.0\}$.

According to Eq. 5, we can derive $S_A^{\Delta ct}$ = $\{1.0, 1.0, 0.3, \ldots, 1.0\}$. Another example is user B who irregularly watched movies may have $S_B^{\Delta t}$ = $\{1.0, 1.0, 0.2, \ldots, 1.0\}$ and $S_B^{\Delta c}$= $\{1.0, 1.0, 1.0, \ldots, 0.2\}$. According to Eq. 5, we know that $S_B^{\Delta ct}$ = $\{1.0, 1.0, 0.2, \ldots, 0.2\}$.

This indicates that we consider both the impact of long elapsed times between consecutive items and concept drift to construct the ATContext. The ATContext sequence $S_u^{\Delta ct}$ is later used for the model learning process to decay the short-term counterpart memory, while the long-term one still maintains the global profile. The proposed approach not only captures the dynamic of the temporal context but also captures the preference driven for each sequence. Secondly, unlike previous studies that used the same constant values to weight the importance of each context for all data sequences, we dynamically compute the weight for each sequence differently. This important difference helps the proposed method capture the changes in each context at a very fine-grained level for further model learning.

4.2 MODEL EMBEDDING

Due to the sparsity problem in real-life datasets, we use an embedding technique to encode each item semantic instead of using a one-hot encoding technique. We utilized the word2vec technique [29] to model the successive transitions among movies in sequence. In this way, the movie's dense vector representation which is embedded in closer proximity will have more common patterns or semantic relationships than those having farther distance. In summary, in this process, given the action sequence of each user S_u, we can get the sequence of the movie embedding vector denoted as $E(S_u)$.

4.3 MODEL LEARNING AND TRAINING

Traditional recurrent neural network (RNN) is designed to handle data with constant elapsed times between successive items of a sequence. In general, the time-lapse between consecutive actions varies from seconds to years. Obviously, the design of a traditional RNN may lead to suboptimal performance. To the best of our knowledge, currently, there are no prior works that consider the concept drift as an additional context in RNN for tracking the dynamics of user preferences at a fine-grained level. Therefore, in this study, ATCRec is proposed to capture the complicated sequential data in a longitudinal fashion. The proposed models take the sequence of item latent vectors $E(S_u)$ embedded by the previous module and ATContext sequence $S_u^{\Delta ct}$ as the input.

We propose a learning concept to decompose the subspace in the memory cells of the recurrent neural network architecture into short- and long-term effects. The short-term effect is adjusted proportionally to the values from the ATContext sequence. If the ATContext value is large, this means either a user did not have an interaction with the system for a long time or a user suddenly changed his preferences to select another type of item. In this case, the dependence on short memory should not play a significant role in the prediction of current output. After we adjust the short-term effect, it is added to the long-term effect to combine the static and dynamic behavior analysis.

With this concept, ATCRec can not only handle the timing irregularity between interactions, but it can also dynamically capture the changes in user interests during the model learning process.

4.3.1 Model Learning

The design of ATCRec is extended from the traditional LSTM model that consists of one cell of memory and three controlled gates to forget, keep, and update the cell memory. The context of temporal-concept drift (Δct) is incorporated as another input into the proposed model in such a way that the user's short-term interest and long-term interest are concurrently learned. ATCRec overcomes the difficulties of concept drift and irregular time intervals between items by not only updating the predictive model with new information but also forgetting old information.

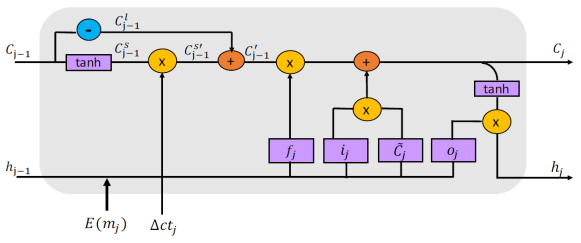


Fig.3: *The design of the ATCRec Learning model.*

We take movie watching data as an example. Intuitively, a movie watched a long time ago is likely to have little influence on the next prediction, and vice versa. Meanwhile, if a user suddenly changes his preferences, recently watched movies should have more influence on the next recommendation. The concept of ATCRec is presented in Fig. 3. When using ATCRec for making the next movie recommendation, $E(m_j)$ represents the latent vector of the user's most recently watched movie m_j, which can be exploited to learn the user's short-term interest. The short-term interest will be adjusted heavily depending on ATContext weight between the last movie and the current movie. The memory cell C_{j-1} represents the the user's historical watching records, which reflect users' long-term interest. The h_{j-1} stores the previous output result. The objective functions are given here:

$$
\begin{aligned}
f_j &= \phi(W_f E(m_j) + U_f h_{j-1} + b_j) \\
i_j &= \phi(W_f E(m_j) + U_f h_{j-1} + b_j) \\
o_j &= \phi(W_f E(m_j) + U_f h_{j-1} + b_j) \\
C^s_{j-1} &= tanh(W_s C_{j-1} + b_s) \\
C^l_{j-1} &= C_{j-1} - C^s_{j-1} \\
C^{s'}_{j-1} &= C^s_{j-1} \times (\Delta ct_j) \\
C'_{j-1} &= C^l_{j-1} + C^{s'}_{j-1} \\
\tilde{C}_j &= tanh(W_c E(m_j) + U_c h_{j-1} + b_c) \\
C_j &= f_j \circ C'_{j-1} + i_j \circ \tilde{C}_j \\
h_j &= o_j \circ tanh(C_j)
\end{aligned}
\tag{6}
$$

$E(m_j)$ represents the current movie embedding vector at time step j. C_{j-1} is the memory cell at timestep j-1. h_{j-1} is the previous output result. We first derive the forget gate f_j, input gate i_j, and output gate o_j to decide how much information will be forgotten, input, and output. ϕ represents a sigmoid function to map the values between 0 and 1, where 0 represents completely ignoring the content and 1 represents completely keeping this content. W and U are the learning weights matrix, and b is the bias vector of each gate. The memory cell C_{j-1} is decomposed into short- and long-term effects. It decomposes the memory of the previous time step into two counterparts, a short-term effect (C^s_{j-1}) and a long-term effect (C^l_{j-1}).

Therefore, we still preserve the long-term effects while adjusting for the short-term effects based on the user's dynamic behavior for prediction. The short-term memory is calibrated properly based on the amount of ATContext (Δct). A heuristic decaying scheme works in such a way that the larger the value of ATContext, the less effect it has on the short-term memory ($C^{s'}_{j-1}$). For example, if the ATContext

weight is large, it infers that there has been no interaction with the system for a long time. Hence, the dependence on short-term memory should play a significant role in the prediction of the current output.

The complement subspace of the long-term effect is combined with the discounted short-term effect to compose the adjusted previous memory as C'_{j-1} is derived from the learning of the input $E(m_j)$ and previous result h_{j-1}. Finally, we derive the new memory cell C_j and output h_j. Note that the parameters of the model for learning (i.e., W, U and b) are all adjusted concurrently with the rest of the network parameters by back-propagation. Also, the operation $\circ$ denotes the Hadamard elementwise product.

4.3.2 Model Training

First of all, we construct an input instance of the proposed learning models for training. For each user u's action sequence S_u in the training dataset, we compute the ATContext (Δct) and derive the ATContext sequence $S_u^{\Delta ct}$. Then, we transform each item m_j to an embedded latent vector $E(m_j)$. Note that the input to ATCRec model is the sequence tuple, $E(S_u) = \{E(m_1), \ldots, E(m_j), \ldots, E(m_{|S_u|})\}$ and the ATContext sequence $S_u^{\Delta ct}$. We use mini-batch learning to train the model on each user's sequence tuple until convergence. According to Eq. 6, h_j could be considered as the prediction latent vector for the next timestamp. Hence, the error function is calculated by root mean square error (RMSE) between the predicted embedding vector h_j and the actual embedding vector $E(m_j)$ as shown in Eq. 7.

$$Loss = \sum_{S_u \in Z} \sum_{j=1}^{|S_u|} (h_j - E(m_j))^2 \tag{7}$$

During the training process, in each epoch, all users' sequences are randomly selected from the training dataset T for each mini-batch Z. Each training batch consists of various lengths of user action sequences for different time steps as described in Eq. 7. We padded the sequence length with maximum values so all T can be processed properly for model training. We use Adam, a variant of gradient descent to optimize the parameters in the proposed model. For the most optimized settings of ATCRec, the cell size and the hidden state size are set as 128 in our experiments. Also, a number of epochs is set as 15 and the batch size is set as 64 for our proposed model.

4.4 ATCRec Recommendation

The goal of ATCRec is to predict the next target items given a sequence of a user selected items (history) and ATContext. When recommending items to user u, we first input u's embedded sequence $E(S_u) = \{E(m_1), E(m_2), \ldots, \ldots, E(m_{|S_u|})\}$ and ATContext sequence $S_u^{\Delta ct}$ into the well trained ATCRec model to derive a prediction vector $\hat{E}(m_p)$. We compare all items in $M = m_1, m_2, \ldots, m_j, \ldots, m_n$ with the predicted output to recommend the top-N similar items to user u. The similarity function is defined in Eq. 8.

$$Similarity(\hat{E}(m_p, E(m_j))) = \frac{\hat{E}(m_p)E(m_j)}{||\hat{E}(m_p)||||E(m_j)||} \tag{8}$$

Notice that both the embedding vectors of $\hat{E}(m_p)$ and $E(m_j)$ are in the same latent space. This allows us to effectively compute the semantic similarity between the two items, and to find the items most similar to an output embedding. Finally, the top-N similar items are returned as the final recommendation results.

5. EXPERIMENTS

In this section, we conduct an experimental study to evaluate the performance of our proposed model on real datasets according to the following research questions: **Q1**: Does the proposed ATCRec outperform the state-of-the-art recommendation approaches on real datasets? **Q2**: Can we reinforce the effectiveness of the proposed recurrent architecture by incorporating the ordinary sequential data with the transition of temporal contexts? **Q3**: Can we reinforce the effectiveness of the proposed recurrent architecture by incorporating the ordinary sequential data with the transition of user preference contexts? **Q4**: Can we improve the performance of the traditional recurrent model by incorporating the transition of both time dynamic and concept drift between consecutive records into the subspace decomposition of the memory cell in LSTM? We first present our experimental settings. After that, the answers to **Q1, Q2, Q3,** and **Q4** are provided respectively in Secs. 5.2, 5.3, and 5.4.

5.1 Experimental Setting

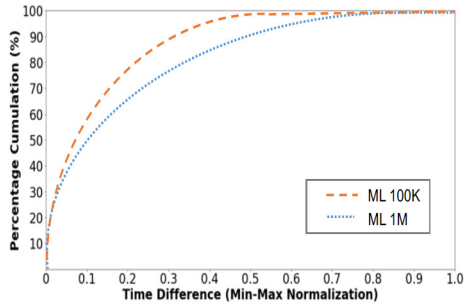
In this section, we discuss how we conducted our experiment on the public dataset provided by Grouplens including ML100K and ML1M. The statistics of the dataset are shown in Table 3. After data-preprocessing, the MovieLens datasets ML100K and ML1M contain 592 and 6,034 users, and 4,569 and 3,260 movies, respectively. The time spans for ML100K and ML1M are from 03/1996 to 09/2018 and 01/2000 to 12/2000, respectively. The range of system rating of all datasets is 1 to 5 (i.e., the best and the worst scores that a user gives to a movie are 5 stars and 1 star, respectively). Every record in the dataset consists of user ID, movie ID, timestamp, rating, genre, title. Since the dataset is usually sparse, all datasets with the users who interact with the system less than 20 times are filtered out. Also, movies with less than 10 ratings are excluded. Note that in

Table 3: *Movielen [30] datasets. MinSL denotes the minimum sequence length while MaxSL denotes the maximum sequence length.*

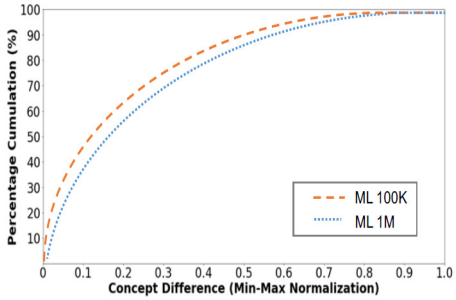
dataset	#user	#movie	#type	period	MinSL	MaxSL
100K	592	4569	18	03/1996-09/2018	20	1634
1M	6034	3260	18	01/2000-12/2000	20	2234

this study, all sequences are sorted in chronological order [37-40].

Fig. 4 shows more descriptions of the irregularities of time and concept differences in our datasets. We normalize all Δt and Δc in both datasets with min-max normalization. Obviously, traditional RNN units such as LSTM and GRU are designed to handle data with constant elapsed times and concepts between successive elements of a sequence. In reality, the time and concept lapse between items can vary from low to high values. Therefore, the design of traditional RNN may lead to suboptimal performance. Figs. 4(a) and (b) show the irregularities of Δt and Δc in our datasets. We can observe that the Δt and Δc between consecutive items vary from small to large units. Since the proposed models consider the dynamics of time and concept differences between successive items, in this study, we have shown the benefits brought by incorporating the irregularities of time and concept differences from the performance differences among LSTM, GRU, ATCRec in terms of H@N, P@N, and R@N in Table 4, Fig. 5., and Fig. 6.



(a) The cumulative distribution of the time differences between two items



(b) The cumulative distribution of the concept differences between two items

Fig.4: *The cumulative distribution of time and concept differences in both datasets.*

5.1.1 Evaluation Metrics

In this study, we discuss three metrics, hit ratio, precision rate, and recall rate, for the top-N recommendations to show the performance of the proposed ATCRec. We take 80% of the dataset as a training set and the remaining 20% for a test set to discuss the three metrics. In advance, for evaluating the three metrics, we take the first 70% of the test data as an input and the remaining 30% is used to compute the hit ratio, precision rate, and recall rate. The definition of hit ratio for the top-N recommendations (H@N) is given in Eq. 9.

$$H@N = \frac{1}{|U|} \sum_{u_i \in U} equal(m_p^N, m^N) \tag{9}$$

equal(m_p^N,m^N) = 1 if $\exists m \in (m_p^N \cap m^N)$, otherwise it is 0. The m_p^N denotes the set of predicted movies in the top-N recommendations for u_i. m^N denotes the set of movies at the first N time steps in the remaining 30% of the test data (i.e., the real N movies that u_i watched after recommendation).

If two sets have overlapping movies, i.e., correct predictions, the function equal(.) will output 1, otherwise it outputs 0. The definition of precision rate (P@N) and recall rate (R@N) at the top-N recommendations are given in Equations 10 and 11.

$$P@N = \frac{1}{|U|} \sum_{u_i \in U} \frac{|m_p^N \cap m|}{|N|} \tag{10}$$

$$R@N = \frac{1}{|U|} \sum_{u_i \in U} \frac{|m_p^N \cap m|}{|m|} \tag{11}$$

m_p^N represents the set of predicted movies in the top-N recommendations. However, m depicts the set of all movies in the remaining 30% of the test data (i.e., all movies that u_i watched after recommendation). For discussing the recommendation quality, N is set to 10, 20, and 50 to show the different results of the metrics.

5.1.2 Baselines

We compare the performance of the proposed ATCRec with several prior methods, including factorization-based and RNN-based approaches. For each existing approach, a grid search is applied to find the optimal setting of hyperparameters using the validation set. All models are implemented in Keras and Tensorflow including:

• MF [11]: A traditional matrix factorization that utilizes the latent factor model to profile user and item latent characteristics based on historical data. Although MF can successfully process the relational rating data, the capability of capturing the temporal dynamic of users' preference is quite limited.
• FPMC [1]: FPMC models user preferences by a personalized Markov chain for the next basket item recommendation based on previous selected items.
• FOSSIL [28]: FOSSIL models the general user preferences by fusing the similarity-based model with Markov Chains to provide a personalized sequential recommendation.
• RNN: A traditional recurrent unit that processes the time-series data but without the concept of contextual information associated with the longitudinal sequence record.
• GRU: The light version of the LSTM model is equipped with update and input gates to control the information flow.
• LSTM: The variant RNN model contains a memory cell for long-term content and three multiplicative gates in input, update, and forget units for dependency learning
• GRU4REC [3]: GRU4REC models the sequential dependencies by using the RNN-based concept, which captures the sequential signal and makes a prediction based on the input session data.
• Time-LSTM [4]: This is a variant version of LSTM. It employs the time gates to model the temporal differences between continuous inputs.

5.2 ATCRec Performance Analysis

In this section, we first investigate the overall performance in terms of the H@N to evaluate the efficiency of predicting the next watched movie. The results of the baselines and the proposed models are summarized in Table 4 and 5.

Clearly, the proposed two learning models have the best performance at all different top-N number settings in all datasets. The results show that integrating adaptive temporal and concept-drift contexts into the recurrent neural models can significantly improve the performance since we discover and integrate the complex and unexpected patterns from the sequence data. Notice that among the baselines, sequential models like FPMC, RNN, LSTM, FOSSIL, and GRU4Rec usually perform much better than MF, indicating the importance of considering sequential information in the model. Also, RNN, GRU, and LSTM outperform FPMC across datasets since FPMC fails to capture the long sequential patterns. Among all baselines, Time-LSTM shows the best recommendation performance due to the capability of the time gate to manage the past information based on the time decay irregularity. In this light, we conclude for research question Q1 that the proposed ATCRec outperforms the state-of-the-art recommendations.

Furthermore, to verify the effectiveness of incorporating the concept of temporal and concept-drift context information, we also implemented some variations of the proposed model ATCRec. The Δcc means that when calculating the concept-drift context (c), we only consider the content-based similarity, i.e., $\alpha = 1$ in Eq. 4. In contrarst, the Δrc means considering the rating similarity only, i.e., $\alpha = 0$ in Eq. 4.

First, we discuss the impact of time granularity on our proposed models to explore the performance improvement when including the time context. The Δtm and Δth infer the calculation of Δt with units of minutes and hours, respectively. From the results in Table 5, we find that the performance of H@N is almost comparable between the variation with Δtm and Δth. However, there is a gradual increase in performance using the time unit of a minute for model learning in all datasets. Also, when we compare the performance of either ATCRec-Δtm or ATCRec-Δth with baselines such as RNN, GRU, and LSTM in Table 4, we see that the variants of our proposed model (ATCRec-Δtm, ATCRec-Δth) performed better in all metrics. Thus, we conclude for research question Q2 that integrating the sequential data with the transition of temporal contexts improves the performance of the proposed recurrent architecture.

Second, we discuss the effectiveness of integrating the concept-drift transition context into the proposed model. We observe in Table 5 that the content-based cosine similarity is more significant than rating-based person similarity in terms of H@N. This is partly because the real-world dataset is usually sparse. The similarity measurement between movies by how much they diverge from the average rating cannot express the overall user behaviors of the whole dataset. Furthermore, when we compared the performance of either ATCRec-Δcc or ATCRec-Δrc with baselines such as RNN, GRU, and LSTM in Table 4, we observe that the variants of our proposed model (ATCRec-Δcc, ATCRec-Δrc) perform better in all metrics. Hence, we conclude for research question Q3 that integrating the sequential data with the transition of user preference contexts improves the performance of the proposed recurrent architecture.

In summary, we conclude that the combination of Δc from both content-based and rating-based and Δt are indispensable for our model. This leads to the conclusion for Q4 that integrating the transition of both time and concept-drift between items into the subspace decomposition of the memory cell in LSTM enhances the recommendation performance.

5.3 Precision and Recall rate on ATCRec

In this section, we investigate more details of the ability of the proposed models to capture both general preference and sequential patterns with the precision (P@N) metric. We vary the Top-N from 10 to 50 to

Table 4: *The comparison of H@N performance on different models.*

Model	ML100K			ML1M		
	H@10	**H@20**	**H@50**	**H@10**	**H@20**	**H@50**
MF	0.0299	0.0312	0.0567	0.0310	0.0421	0.0598
FPMC	0.0412	0.0721	0.0853	0.0358	0.0479	0.0893
FOSSIL	0.0601	0.0798	0.0991	0.0693	0.1094	0.1212
RNN	0.0672	0.0844	0.1008	0.0745	0.1000	0.1293
GRU	0.0820	0.0924	0.1580	0.0791	0.1262	0.1699
LSTM	0.0888	0.1056	0.1760	0.0988	0.1376	0.1812
GRU4Rec	0.1021	0.1214	0.1687	0.1021	0.1329	0.1824
Time-LSTM	0.1136	0.1426	0.1893	0.1103	0.1510	0.1911
ATCRec	0.1220	0.1572	0.2066	0.1351	0.1810	0.2246

Table 5: *The comparison of H@N performance on different variations of the proposed learning models.*

Model	ML100K			ML1M		
	H@10	**H@20**	**H@50**	**H@10**	**H@20**	**H@50**
ATCRec-Δcc	0.0995	0.1261	0.1846	0.1151	0.1509	0.1917
ATCRec-Δrc	0.0975	0.1150	0.1789	0.0899	0.1383	0.1819
ATCRec-Δtm	0.1001	0.1187	0.1890	0.1112	0.1455	0.1811
ATCRec-Δth	0.0913	0.1190	0.1894	0.1012	0.1376	0.1756
ATCRec	0.1220	0.1572	0.2066	0.1351	0.1810	0.2246

observe the trend and variation of the precision rate in a different setting.

As shown in Fig. 5, the precision rate decreases when we increase the N number. This phenomenon is quite legitimate since the recommended result becomes larger as we increase N. Compared with all methods, ATCRec achieves the highest improvement as shown in Fig. 5. This demonstrates the advantage of incorporating Δct into a traditional long-short term sequential model. ATCRec gains more than 10 percent improvement as compared with Time-LSTM, LSTM, FOSSIL, GRU, GRU4Rec, and FPMC respectively. This indicates that the proposed models could model the complex sequential pattern effectively. In addition, we found that LSTM-based methods outperformed the GRU-based methods. One reason for that is that LSTM-based methods have more capability to learn and remember the complicated and long sequential data. In summary, the experimental results show that ATCRec models both user general preferences and sequential patterns via the long-term and short-term counterparts of the recurrent unit. Moreover, the proposed models are robust for capturing the timing irregularity and dynamic concept-drift in the longitudinal sequential data.

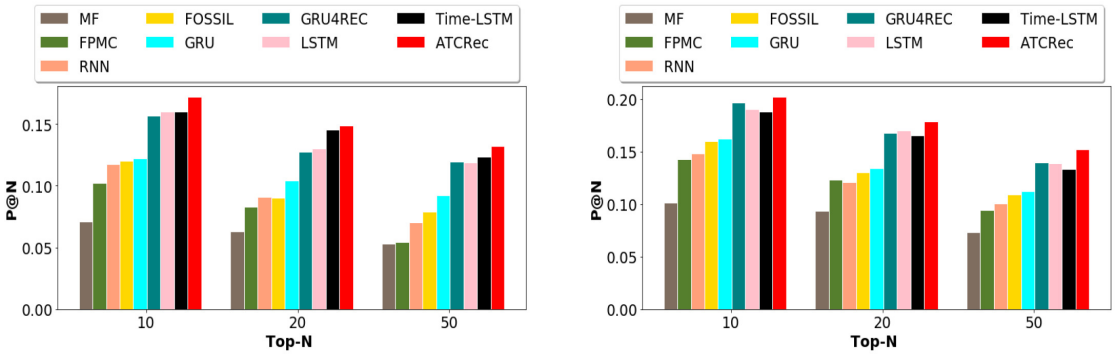
Next, we discuss the proposed models in capturing both general preference and sequential patterns with the recall (R@N) metric. Varying the Top-N setting to show the trend and variation of recall rate is considered. Different from precision, as in Fig.6, the recall rate increases with increasing N. This is quite reasonable since the more movies in the recommendation list for the user, the more probability of bringing out the real desired movies of the user. We could observe that ATCRec outperforms the traditional approaches, i.e., MF and FPMC, as well as the neural-based approaches, i.e., FOSSIL and GRU4Rec, in terms of Recall. When the size of the recommendation list varies from 10, 20, to 50, the improvement of ATCRec over the best baseline also increases. Specifically, the enhancement is 10%, 13.33%, and 20% in terms of Recall@10, Recall@20, and Recall@50 on average on the three datasets. This shows the ability to boost the number of relevant movies of proposed models.

5.4 User Case Study

Fig. 7 shows the top-N ranked movies recommended by ATCRec after the model is well trained for a specific user. Notice that the particular user sequence is randomly chosen from the testing dataset. For user id = 2490, when using the latest five movies in watching sequence as the input consisting of $\{m_1$ (Heavenly Creatures), m_2 (Just Cause), m_3 (Star wars: Episode IV - A New Hope), m_4 (Star Wars: Episode V - The Empire Strikes Back), m_5 (Star Wars: Episode VI - Return of the Jedi)$\}$, the top-three recommendation result ranks as $\{R_1$ (Star Trek: Generations), R_2 (Star Wars: Episode I - The Phantom Menace), and R_3 (Star Wars: Episode II - Attack of the Clones)$\}$.

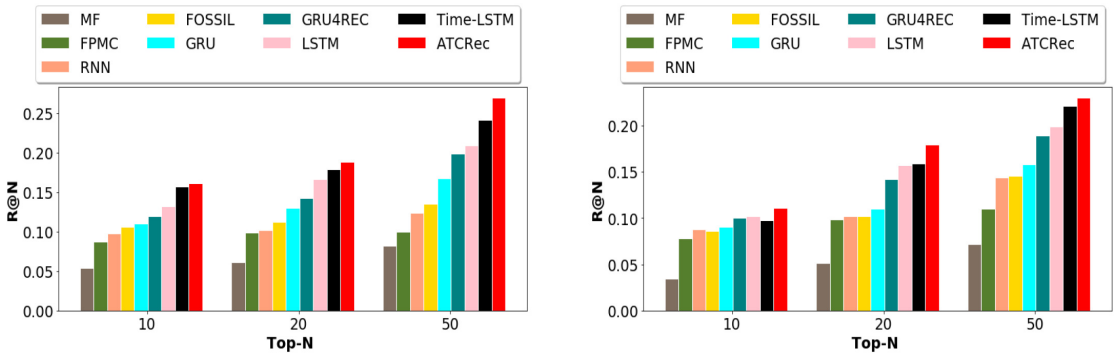
We could find that the actual succeeding movie, i.e., the ground truth, is $\tilde{m}_6$ = (Star Wars: Episode I - The Phantom Menace). Obviously, all movies in the recommendation list are very similar. R_1, R_2 and R_3 belong to the action, adventure, sci-fi, and drama genres. Because of the similarity score of item embedding in Eq. 8 in the recommendation process, the items that are closely embedded into a vector space with the predicted vector could be recommended with the user-specified top-N setting. The recommended results show the ability and performance of the proposed ATCRec on the sequential recommendation.

Table 6 shows the new rank of R_2 after masking some of the previous movies by setting their item



(a)P@N on ML100K dataset (b)P@N on ML1M dataset

Fig.5: *The comparison of P@N performances with varying Top-N on different datasets.*



(a)R@N on ML100K dataset (b)R@N on ML1M dataset

Fig.6: *The comparison of R@N performances with varying Top-N on different datasets.*

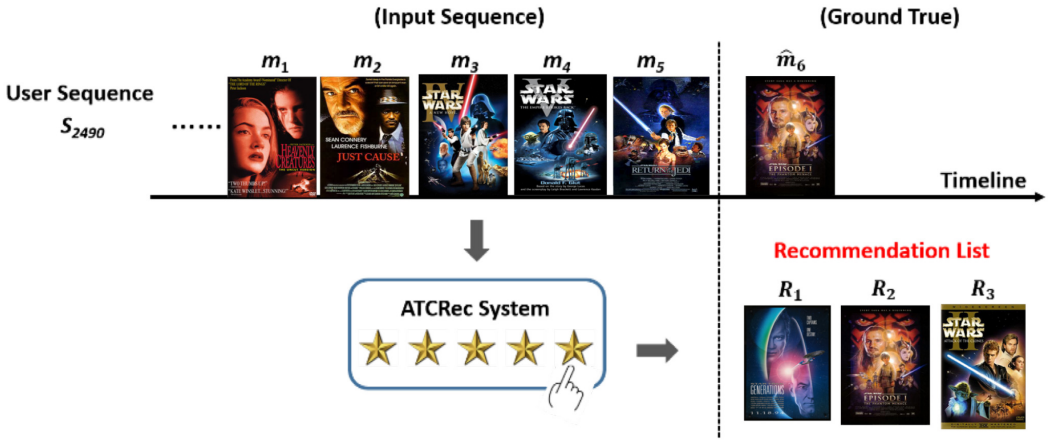


Fig.7: *The use case of ATCRec in capturing the sequential feature of the ML1M dataset.*

Table 6: *The use case of ATCRec in capturing the temporal and concept-drift context of the ML1M dataset.*

Masking Movies	The ranking order of R_2 after masking
m_1 and m_2	1
m_3	45
m_4	123
m_5	180
m_3, m_4, and m_5	289

embeddings to zeros in the trained network. When m_1 and m_2 are masked, the ranking of R_2 increases from top-2 to top-1. Since m_1 and m_2 belong to the thriller, horror, and mystery genres, respectively, both of them act like noise for recommending R_2. When masking each of m_3, m_4 and m_5, the rank of R_2 decreases because all of these movies are in the same genre and are similar to R_2. This phenomenon confirms the assumption in the case scenarios 1 and 2 stated in the Introduction. We observe that the proposed ATCRec detects the variation in the user preference and also contributes a greater influence based on the recently watched movies. Finally, when masking m_3, m_4 and m_5 all together, a significant decrease occurs in the ranking. This effect clearly indicates that the learning model properly captures the dependence of R_2 on the related movies $\{m_3, m_4, m_5\}$ as a sequential feature for recommending R_2 with the user specified top-N setting.

6. CONCLUSIONS

Recommendation systems play an important role in dealing with the massive amount of information from everywhere nowadays. Providing an accurate recommendation at the correct time to customers can contribute to a surge in business success. In this paper, an adaptive temporal-concept drift learning-based system, ATCRec, is developed for precisely tackling the sequential recommendation problem. We embed sequences of items into the latent spaces and propose a novel model to capture both general preferences and sequential patterns concurrently with a recurrent neural network. Temporal and concept-drift intervals between records, called ATContext, are also modelled in the memory unit to enhance the performance of the proposed ATCRec training algorithms. ACTRec provides a unified and flexible network structure to learn and reveal the opaque variation of user preferences over time. Finally, we performed experiments on two real-world datasets to evaluate the robustness and performance. The experimental results demonstrate that ATCRec consistently outperforms existing sequential recommendation approaches on various metrics. We also demonstrate a use case study to show the applicability and expressiveness of the ACTRec recommendation system.

References

[1] S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme, Factorizing personalized Markov chains for next-basket recommendation, *Proceedings of the 19th international conference on World wide web*, pp.811–820, 2010.

[2] S. Zhao, M. R. Lyu, and I. King, STELLAR: Spatial-Temporal Latent Ranking Model for Successive POI Recommendation, *SpringerBriefs in Computer Science Point-of-Interest Recommendation in Location-Based Social Networks*, pp.79–94, 2018.

[3] B. Hidasi and A. Karatzoglou, Recurrent Neural Networks with Top-k Gains for Session-based Recommendations, *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pp.843–852, 2018.

[4] Z. Yu, L. Hao, L. Yikang, W. Beidou, G. Ziyu, L. Haifeng and C. Deng, What to Do Next: Modeling User Behaviors by Time-LSTM, *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*, pp.3602–3608, 2017.

[5] B. Sarwar, G, Karypis, J. Konstan, and J. Reidl, Item-based collaborative filtering recommendation algorithms, *Proceedings of the tenth international conference on World Wide Web*, pp.285–295, 2001.

[6] F. Ricci, L. Rokach, and B. Shapira, Introduction to Recommender Systems Handbook, *Recommender Systems Handbook*, pp.1–35, 2010.

[7] Z. Zhang, Y. Liu, Z. Jin, and R. Zhang, A dynamic trust based two-layer neighbor selection scheme towards online recommender systems, *Neurocomputing*, pp.94–103, 2018.

[8] J. Gupta and J. Gadge, Performance analysis of recommendation system based on collaborative filtering and demographics, *International Conference on Communication, Information & Computing Technology (ICCICT)*, pp.1–6, 2015.

[9] P. Melville, R. J. Mooney, and R. Nagarajan, Content-boosted collaborative filtering for improved recommendations, *American Association for Artificial Intelligence*, pp.187–192, 2002.

[10] Z.L. Zhao, C.D. Wang, Y. Wan, Z. Huang, and J. Lai, Pipeline Item-Based Collaborative Filtering Based on MapReduce, *IEEE Fifth International Conference on Big Data and Cloud Computing*, pp.9–14, 2015.

[11] R. Mehta and K. Rana, A review on matrix factorization techniques in recommender systems, *International Conference on Communication Systems, Computing and IT Applications (CSCITA)*, pp.269–274, 2017.

[12] Y. Koren, R. Bell, and C. Volinsky, Matrix Factorization Techniques for Recommender Systems. Computer, *IEEE Transactions on Automatic Control*, pp.30–37, 2009.

[13] N. Thai-Nghe, L. Drumond, T. Horváth, A. Krohn-Grimberghe, A. Nanopoulos, and L. Schmidt-Thieme, Factorization Techniques for Predicting Student Performance, *Educational Recommender Systems and, Technologies*, pp.129–153, 2012.

[14] M. Abdi, G. Okeyo, and R. Mwangi, Matrix Factorization Techniques for Context-Aware Collaborative Filtering Recommender Systems: A Sur-

vey, *Computer and Information Science*, pp.2–11, 2018.

[15] L. Xiong, X. Chen, T. Huang, J. Schneider, and J. G. Carbonell, Temporal Collaborative Filtering with Bayesian Probabilistic Tensor Factorizatio, *Proceedings of the 2010 SIAM International Conference on Data Mining*, pp.211–222, 2010.

[16] Y. Shi, M. Larson, and A. Hanjalic, Collaborative Filtering beyond the User-Item Matrix, *ACM Computing Surveys*, pp.1–45, 2014.

[17] M. Jamali and M. Ester, A matrix factorization technique with trust propagation for recommendation in social networks, *Proceedings of the fourth ACM conference on Recommender systems*, Vol.29, pp.831–832, 2010.

[18] D. Liang, J. Altosaar, L. Charlin, and D. M. Blei, Factorization Meets the Item Embedding, *Proceedings of the 10th ACM Conference on Recommender Systems*, pp.59–66, 2016.

[19] Y. Ding and X. Li, Time weight collaborative filtering, *Proceedings of the 14th ACM international conference on Information and knowledge management*, pp.485–492, 2005.

[20] Y. Koren, Collaborative filtering with temporal dynamics, *Communications of the ACM*, pp.89, 2010.

[21] J. Tang and K. Wang, Personalized Top-N Sequential Recommendation via Convolutional Sequence Embedding, *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pp.565–573, 2018.

[22] Y. Lo, W. Liao, C. Chang, and Y. Lee, Temporal Matrix Factorization for Tracking Concept Drift in Individual User Preferences, *IEEE Transactions on Computational Social Systems*, pp.156–168, 2018.

[23] S. Cheng and Y. Liu, Time-Aware and Grey Incidence Theory Based User Interest Modeling for Document Recommendation, *Cybernetics and Information Technologies*, pp.36–52, 2015.

[24] T. Jiang and W. Lu, Improved Slope One Algorithm Based on Time Weight, *Applied Mechanics and Materials*, pp.2365–2368, 2013.

[25] Y. Cai, H. Leung, Q. Li, J. Tang, and J. Li, TyCo: Towards Typicality-based Collaborative Filtering Recommendation, *IEEE International Conference on Tools with Artificial Intelligence*, pp.766–779, 2010.

[26] F. Yu, Q. Liu, S. Wu, L. Wang, and T. Tan, A Dynamic Recurrent Model for Next Basket Recommendation, *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, pp.729–732, 2016.

[27] R. He, W. Kang, and J. Mcauley, Translation-based Recommendation: A Scalable Method for Modeling Sequential Behavior, *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*, pp.161–169, 2018.

[28] R. He and J. Mcauley, Fusing Similarity Models with Markov Chains for Sparse Sequential Recommendation, *IEEE 16th International Conference on Data Mining (ICDM)*, pp.300–311, 2016.

[29] X. Rong, word2vec Parameter Learning Explained, *arXiv*, 2016.

[30] F. Harper and J.A. Konstan, The MovieLens Datasets, *ACM Transactions on Interactive Intelligent Systems*, pp.1–19, 2015.

[31] V. Akila and G. Zayaraz, A brief survey on concept drift, *Advances in Intelligent Systems and Computing*, pp. 293–302, 2014.

[32] I. zliobaite, M. Pechenizkiy, and J. Gama, An overview of concept drift applications, *Studies in Big Data*, pp. 91–114, 2015.

[33] M. Roveri, Learning discrete-time markov chains under concept drift, *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 9, pp. 2570–2582, 2019.

[34] T. Neammanee and S. Maneeroj, Time-Aware recommendation based on user Preference driven, *2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC)*, 2018.

[35] R. Xu, Y. Cheng, Z. Liu, Y. Xie, and Y. Yang, Improved long short-term memory based anomaly detection with concept drift adaptive method for supporting iot services", *Future Generation Computer Systems*, vol. 112, pp. 228–242, 2020.

[36] V. M. dos Santos, R. F. de Mello, T. Nogueira, and R. A. Rios, Quantifying temporal novelty in social networks Using TIME-VARYING graphs and Concept Drift detection, *Intelligent Systems*, pp. 650–664, 2020.

[37] Y. Zhu, H. Li, Y. Liao, B. Wang, Z. Guan, H. Liu, and D. Cai, What to do Next: Modeling user behaviors by time-lstm, *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*, 2017.

[38] J. Tang and K. Wang, Personalized top-n Sequential recommendation via CONVOLUTIONAL Sequence Embedding, *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, 2018.

[39] W.-C. Kang and J. McAuley, Self-Attentive sequential Recommendation, *2018 IEEE International Conference on Data Mining (ICDM)*, 2018.

[40] T. Silveira, M. Zhang, X. Lin, Y. Liu, and S. Ma, How good your recommender system is? A survey on evaluations in recommendation, *International Journal of Machine Learning and Cybernetics*, vol. 10, no. 5, pp. 813–831, 2017.

Tipajin Thaipisutikul received the master's degree (Hons.) in the research path from The University of Sydney (USYD), Sydney, NSW, Australia in 2012 and received the Ph.D. degree from the Department of Computer Science and Information Engineering, National Central University, Chung-Li, Taiwan in 2021. She is currently an instructor with the Faculty of Information and Communication Technology (ICT), Mahidol University, Salaya, Thailand. Her research mainly focuses on machine learning, applied intelligence, data mining, and social network analysis.