Egyptian Informatics Journal xxx (xxxx) xxx

Contents lists available at ScienceDirect

Egyptian Informatics Journal

journal homepage: www.sciencedirect.com

Exploiting dynamic changes from latent features to improve recommendation using temporal matrix factorization

Idris Rabiu [a,b,*], Naomie Salim [a], Aminu Da'u [a], Akram Osman [a], Maged Nasser [a]

[a] Faculty of Engineering, School of Computing, Universiti Teknologi Malaysia, 81310 Johor Bahru, Malaysia
[b] Computer Science Department, Ibrahim Badamasi Babangida University, Lapai P.M.B. 11, Niger State, Nigeria

ARTICLE INFO

Article history:
Received 7 April 2020
Accepted 11 October 2020
Available online xxxx

Keywords:
Recommender system
Collaborative filtering
Concept drift
Temporal models
Temporal matrix factorization

ABSTRACT

Recommending sustainable products to the target users in a timely manner is the key drive for consumer purchases in online stores and served as the most effective means of user engagement in online services. In recent times, recommender systems are incorporated with different mechanisms, such as sliding windows or fading factors to make them adaptive to dynamic change of user preferences. Those techniques have been investigated and proved to increase recommendation accuracy despite the very volatile nature of users' behaviors they deal with. However, the previous approaches only considered the dynamics of user preferences but ignored the dynamic change of item properties. In this paper, we present a novel Temporal Matrix Factorization method that can capture not only the common users' behaviours and important item properties but also the change of users' interests and the change of item properties that occur over time. Experimental results on a various real-world datasets show that our model significantly outperforms all the baseline methods.

1. Introduction

The need for research in the area of recommender system in modern society is activated in response to address the potential challenge of information overload which hindered the users timely access to items of interest over the internet [1]. This however, calls for a need to find the solution to the information overload by filtering, prioritizing and efficiently delivering relevant information to users, hence the birth of recommender system [1]. Recommender systems are regarded as tools that generate recommendations to the target users by leveraging various kinds of knowledge and information about users, items and past records of user-item interactions. During the user interactions with systems, his actions and feedbacks (such as, ratings, like/dislike, purchase, browse histories etc.) are usually collected and stored in a database which can be used to generate new recommendation for next user-system interactions.

Although recommender systems suggest the items that appear most likely to be useful to a particular user based on his actions and feedbacks, it is important to note that user interests are sometimes dynamic and change over time. Also, as more data about new users and items are being constantly generated, this causes a lot of changes in fundamental relationship between users and recommended items [2]. These complex and dynamic characteristics accompanied with stream data posed a great challenge in making effective recommendations as a result of these changes in relationship between users and recommended items [3]. On this note, several studies have considered the problem of modelling user behavioural patterns and shown that a user's preferences are likely to change from time to time as they become more aware of new products [2,4–7]. To satisfy the users' current need, the appropriate way to build a holistic recommender system is to properly model the user interest and preferences as they evolve over time [8]. It was shown in these studies that modeling such user preference dynamisms over time can improve the prediction accuracy.

Recently, a Temporal Matrix Factorization (TMF) was proposed by [9], which only takes into account the dynamic change of user preferences by monitoring the evolution of user latent vectors, but it neglects the changed item features. However, the existing temporal models for capturing users' changing interests are insufficient and in most cases ignored the changed item properties. For

* Corresponding author at: Faculty of Engineering, School of Computing, Universiti Teknologi Malaysia, 81310 Johor Bahru, Malaysia.
E-mail addresses: idrisrabiu43@gmail.com (I. Rabiu), naomie@utm.my (N. Salim).

Peer review under responsibility of Faculty of Computers and Information, Cairo University.

https://doi.org/10.1016/j.eij.2020.10.003

Please cite this article as: I. Rabiu, N. Salim, A. Da'u et al., Exploiting dynamic changes from latent features to improve recommendation using temporal matrix factorization, Egyptian Informatics Journal, https://doi.org/10.1016/j.eij.2020.10.003

example, a user who is fancy of latest documentary films gave a good rating to the recommended films at first, because the recommendation is either up-to-date or gain more popularity among others. But, when the same user requests a recommendation the second time, and he is recommended a documentary film which is much older and unsatisfying than the previous one, then his rating for this recommendation will be low reflecting his dissatisfaction with the recommendation. This implies that the user's perception for documentary films have changed for only the latest and up-to-date recommendations. Different from previous studies, we propose a novel TMF method that considers the dynamic change of user preferences and the changed item features. Generally, the intuition behind temporal models is that, when a new set of ratings is received at a time point t, we gain more information about a user preference which may distort the precision of the prediction model. This, therefore, warrants for the adjustment of the user preference model (the latent matrix P) [10]. Similarly, items go in and out of popularity over time such as the *recency* effects or periodic effects, which also contributes to dynamic changes in the item preference model (the latent matrix Q). This, however, nullifies the claim that the item latent factors are time invariants.

To capture the dynamic changes in each individual user latent vector and item latent vector, we propose to model the user and item latent features separately: the user specific model is built using the dynamic user latent vectors and the item specific model is built using the dynamic item latent vectors. To this end, the major contributions of this paper can be summarized as follows:

i. We propose a *dui*-TMF that leverage the dynamics of user and item latent vectors to capture changes in each user preferences and item features. This method does not only break the limit of the basic Matrix Factorization (MF) method, it also offers a tool for recommenders to accurately recommend items that meet the user's current needs.
ii. We performed experiments using the weighting values estimated based on the changes in latent vectors of each individual users and items.
iii. We further compare our *dui*-TMF model with state-of-the-art methods to demonstrate its effectiveness based on four real-world datasets.

The remainder of this paper is as follow. In section 2, a review of related works is presented. In section 3, we define the general rating prediction problem, and the propose method for capturing changes in user and item latent features, as well as the rating prediction are presented in Section 4. Section 5 presents the experimental results on various real-world datasets to validate the proposed method. Lastly, the paper is concluded by recommending the future research directions in section 6. The list of notations used in this paper is provided in Table 1.

Table 1
List of notation.

m	number of users
n	number of items
T	$T$is the prediction time, while $T-1$ is the training periods
K	number of latent factors
$R=(R_{ij})$	$m \times n$rating matrix
$\widehat{R}_{ij}$	user i rating prediction for item j
P	$K \times m$user latent matrix
P_i	user i^{th} latent vector
$P_i(t)$	i^{th}user latent vector learned at time$t=1,2,\cdots T-1$
$P_i(T)$	i^{th}user latent vector predicted at time$T$
Q	K $\times$ n item latent matrix
Q_j	item j^{th} latent vector
$Q_j(t)$	j^{th}item latent vector learned at time $t=1,2,\cdots T-1$
$Q_j(T)$	j^{th}user latent vector predicted at time$T$
I	identity matrix
I_{ij}	indicator function which is equal to 1 if user i rated item j and equal to 0 otherwise
e_{ij}	prediction error
$e_{ij}(t)$	prediction error at time $t=1,2,\cdots T-1$
α	learning rate
λ	regulator parameter
$\Delta P(t)$	changes between user latent vectors at time$t=1,2,\cdots T-1$
$\Delta Q(t)$	changes between item latent vectors at time$t=1,2,\cdots T-1$
SD	standard deviation of changes in the latent vectors.

2. Related work

This section presents a brief review of the MF method and several recent approaches on temporal aspects that integrate time information with MF for recommender systems, including time-dependent and time-independent MF methods [37].

2.1. Matrix Factorization

In the context recommender system, MF has attracted a considerable attention for their superior performance in the rating prediction task [11,12]. The motivation behind the MF method is the ease to represent individual user and item by latent factors that characterize the user and item interactions from the historical rating matrix R. Specifically, the basic idea is to decompose the rating matrix R into the product of a user feature matrix P and item feature matrix Q, where both P and Q are $m \times K$ and $n \times K$ matrices for rank $k \ll \min\{m,n\}$ respectively. Each row p_i in P is vector of user i preference towards the $k$features and the row $q_j$in Q represent the affinity of j^{th} item towards k features.

Based on this assumption, it follows that each rating is approximated by the dot product of each user feature vector and item feature vector as in Eq. (1):

$$R_{ij} = P_i.Q_j^T = \sum_{s=1}^{k} P_{is}.Q_{js} \tag{1}$$

The MF approach considers the optimization problem in Eq. (2) to learn P and Q:

$$P,QMin\ \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{n} I_{ij}\left(R_{ij} - P_i.Q_j^T\right)^2 + \frac{1}{2}\left(P^2 + Q^2\right) \tag{2}$$

where the optimization parameters P and Q can be learn through a SGD algorithm [10], which loops through all ratings in the training set. For each training example R_{ij}, the associated prediction error is first computed in equation in (3) as:

$$e_{ij} = R_{ij} - P_i.Q_j^T \tag{3}$$

Then for each loops through the ratings, the P_i and Q_j matrices can be updated in the opposite direction of the gradient in Eqs. (4) and (5) as follows:

$$P_i \leftarrow P_i + \alpha\left(e_{ij}.Q_j - \lambda.P_i\right) \tag{4}$$

$$Q_j \leftarrow Q_j + \alpha\left(e_{ij}.P_i - \lambda.Q_j\right) \tag{5}$$

where α and λ are the learning parameter and the regulator parameter respectively. This iterative and incremental learning method provides a key building block to extend MF method to a large scale and more dynamic scenarios.

2.2. Time-dependent collaborative filtering (TDCF)

Research on TDCF that takes the temporal changes into account has become more triggered after the Netflix competition worn by

the Koren team in the year 2009. It has since then emerged as an appropriate tool to aid the process of recommendations, considering the dynamic nature of data and to address the limitations of the content-based approach [4]. In the same manner, as the classical CF system, DCFM uses a collaborative approach that leverages the user information, such as the age, country, city, and items purchased when processing recommendations [13]. Based on this information, the system looks for users that share the same preference, and then suggest relevant items purchased by similar users. DCFM approaches are basically of two types, namely, the temporal memory-based [14] and the temporal model-based methods [15].

There are several prior works on the temporal CF approaches based on memory models such as [16–19], which focused on Neighborhood-based CF and build the extended techniques that incorporates the dynamic user behaviors based on their interaction sequences to produce a personalized recommendation. In this approach, window-based [20] and weighting functions [21] are the most widely employed to address the dynamic change of user preferences. These two approaches are based on the assumption that most recent ratings reflect more of users' current interests than the old ratings. Specifically, the window-based approach discards the old data instances that might be less relevant to the current state of the situation and considers only the recent instances [22]. In this model, all the ratings within a given time-window are assigned equal importance, which is most suitable when the change is an abrupt type, but less so when change is a gradual type [4]. The weighting functions, on the other hand, penalize the old the preferences by applying a decreasing weight to old ratings and attach more importance to the recent ratings. Although these techniques improve the prediction accuracy to some extent compared to their baseline neighborhood-based methods, several issues were raised. The possible drawback of these methods is that, by way of penalizing or discarding the old ratings that could provide better information about general preferences of the user, they tend to lose too much data and these make them more susceptible to data sparsity problem [23].

In addition to memory-based temporal models, several attempts were also made on the temporal factor-based CF models to address the issue of dynamic changes of user interest, which also consider the sparsity limitation of the previous techniques [20,24–27]. One of the famous models in this category is the temporal matrix factorization (TMF) model [20,28,29]. Thanks to the flexibility of interpreting the latent factors in matrix factorization (MF) methods, different grounds for user preferences and item attractiveness in recommender systems can be seemingly represented in a way that the dynamic changes in user behaviors or item features are captured. These can be achieved either by employing the *recency*-based methods [20,30,31] or by incorporating a time variable when modeling user behaviors [25].

2.3. Time-independent model

Pairwise Interaction Tensor Factorization (PITF) is one of the basic approach in which time information is duly incorporated. PITF is a classic tensor model that has a linear runtime for both the learning and prediction tasks [32]. In this case, the pairwise interactions among users, items and the tags are jointly learned by this model [32]. Given the user-item-time rating matrix, the tag dimension is replaced in a time-aware approach where the resulting matrix can be factorized to get the corresponding feature models. In this regard, [33] proposed a time-aware approach based on Base-Level Learning (BLL) which utilized the time feature to capture temporal dynamic of user preferences and the items popular tags to handle the cold start of new users. However, the BLL approach works only on individual and the target resource levels, and therefore fails to capture users' potential interests. In recent contributions [15], proposed a unified tag recommendation model which extends PITF method by integrating both time awareness and personalization aspects and adding more weight to user-tag and item-tag interactions, respectively [32]. Unlike the common power-form functions employed by the existing temporal models [33], the authors utilized a temporal weight in the form of exponential intensity function which was proven more efficient in modeling user-item interactions. [34] provided a novel approach which focused mainly on identifying the semantic attributes extracted from user-side information, including the user sentiments, the number of interactions, and objectivity among others. To achieve that, the authors employed a 3-dimensional MF method that takes the temporal alterations into account.

However, one of the draw backs of tensor model is sparsity problem. That is, the higher the order of tensor model is, the more severe the sparsity problem is [34]. This leads to a high space complexity, time-consuming and slow convergence rates in the learning process. Moreover, the previous temporal model [20] is based on a strong assumption that the concept drift of user preferences is captured by the transition matrix. With this approach, it is very difficult, if not impossible, to measure the quantitative drift rate. Secondly, it is not sufficient for the tractability of item changing properties as it assumes that items features are invariant over time, and thus focused only on the preference changes of individual users.

In this paper, we want to remove the assumption that the item's features are time-invariant and tracking changes from item features can lead to an improved solution. By doing so, we measure the concept drift of individual users based on the user latent vectors and the concept drift in item features based on the item latent vectors. Furthermore, we also verified that individual users and items preference vectors changes in different ways. The result shows that some of the users and likewise the items have considerable changes in their latent vectors, and the inclusion of items dynamicity in our work is the major factor for the improvement of our proposed temporal model.

3. Problem definition

Before introducing our novel dynamic user and item features for TMF model (*dui*-TMF), we first define the problem of rating prediction in general based on MF, which has been modified as TMF by using time information in rating prediction [9]. Given a time-stamped rating matrix R, the basic idea is to decompose the rating matrix R into the product of a user feature matrix P and item feature matrix Q, where both P and Q are $m \times k$ and $n \times k$ matrices for rank $k \ll \min\{m, n\}$ respectively. However, considering the associated time information for user-item interactions, the user rating matrix is represented in four tuples:

$$(\mathrm{i}, \mathrm{j}, \mathrm{r}, \mathrm{t})$$

It is assumed that each rating may be a real number and each item is often rated at most once by a user. When timestamps of the rating matrix are dropped, then the ratings can be represented in the form of $m \times n$ matrix with element (R_{ij}), where R_{ij} denotes the user i rating on an item j if the user$i$ actually rated item j. Conversely, R_{ij} is said to be missing rating if the item j is not been rated by the user i. In real sense, the matrix R may be a sparse matrix with many missing values. Therefore, given a sparse matrix, the rating prediction goal is to predict the missing ratings in the sparse matrix$R$.

To demonstrate the effectiveness of recommender systems, the performance of the prediction algorithm is often evaluated by partitioning the rating matrix is into two sets, which comprises of the training and the test sets. The training set is provided to a learning

algorithm to "learn" the desired parameters for the prediction model. The test set on the other hand is not presented to the learning model but only used to test the accuracy of the prediction model.

In literature, several performance measures have been used to evaluate the temporal recommendation algorithms. In this paper, we adopt the Time-averaged RMSE to measure the performance of our proposed method. *TA_RMSE*is a temporal accuracy metric based on RMSE, which is computed on ratings made until a particular point of time as follows:

$$TA\_RMSE = \sqrt{\frac{1}{|Testset_t|}\sum_{i=1}^{Testset_t}\left(r_{ij}-\hat{r}_{ij}\right)^2} \quad (5)$$

where $\hat{r}_{ij}$ and r_{ij} are the predicted and actual rating values, and $Testset_t$ is the set of test ratings made until time t. In this case, the lower values of *TA_RMSE* indicates better accuracy.

4. Proposed temporal matrix Factorization method

In view of the prediction problem discussed in Section 3, we propose a *dui*-TMF method for tracking not only the changes in user latent vectors but also the changes in item latent vectors. Fig. 1 shows the proposed model for adoption of the temporal matrix factorization based on dynamics of user and item latent features. Our method follows the same assumptions with the previously used temporal methods in the literature [10,20]. These include:

1) As defined in the original MF, there are m number of users, indexed as i = 1, 2, …, m, and$n$ number of items, indexed as j = 1, 2, …, n, such that the rating matrix R can be decomposed into the product of a user feature matrix P and item feature matrix Q, where both P and Q are $m \times k$ and $n \times k$ matrices for rank $k \ll \min\{m, n\}$ respectively. The latent vector of user i, denoted by P_i, is the ith column of the user latent matrix P. Likewise, the latent vector of item j, denoted by Q_j, is the jth column of the item latent matrix Q. Hence, the user i rating for item j can be predicted as the inner product of both P_i and Q_j
2) As people changed their preferences for items over time, the user preference model at one time may not be valid for predicting user preference at a future time. Likewise, the item latent vector changes as item popularity change over time. To this end, we represent the user feature matrix P at time t by P(t), and the user latent vector at time$t$, is represented by P_i(t), for i = 1, 2, …, m. Similarly, $Q(t)$ and $Q_j(t)$ are also defined in the same manner.

Based on these assumptions, the key factor of our work is to train the proposed method on the training data set to study the possible drifts in each of the user latent vector, which signifies the user preferences towards the $k$features and item latent vectors, which represents the affinity of individual items towards k features. To this end, the major steps involved in our approach are as follows:

Step 1: In this step, the time series of $m \times n$ rating matrices, {R (t), t = 1, 2, …, T − 1} were constructed by using the rating feedbacks in the training data set.
Step 2: In this step, using the time series of rating matrices {R(t), t = 1, 2, …, T − 1} constructed in step 1, the time series of both the user latent vector P_i (t), and item latent vector Q_j(t), {t = 1, 2, … , T − 1} were learned.
Step 3: The dynamic concept drift in the user latent vectors for each user i, and the item latent vector for each item j are computed using the time series of user latent vectors, {P_i (t), t = 1, 2, …, T − 1}, and the time series of item latent vectors, {Q_j (t), t = 1, 2, …, T − 1}.
Step 4: In this step, the weighted user and item latent vectors based on the dynamics of the concept drift obtained in the step 3 is computed to update the previous models.
Step 5: In this step, the missing values in the rating matrix is predicted by using the product of P (T) and Q(T) predicted for the user latent matrix and item latent matrix at time T

I. Constructing the time series of rating matrices

One of the possible ways for constructing the time series of rating matrices $\{R(t), t = 1, 2, \cdots T\}$ is either by sorting the original rating matrix such that items appear chronologically according to users' ratings submission or by equally partitioning the rating matrix on the same scale based on their timestamps. Here we choose the latter approach where the original rating matrix is partitioned according to their stamps. To avoid the possibility of creating sparser data which is guaranteed using partitioned based method, we use a sliding window method that combines the ratings in a number of consecutive slices into a single time step, as adopted in [21].

II. Learning the time series of user and item latent vectors

The intuition behind time-evolving user latent factors is that, when a new set of ratings is received at time t, we gain more information about a user preference which may distort the precision of the prediction model. This, therefore, warrants for the adjustment of the reference model (the user latent vector P_i). The temporal methods proposed by previous authors [9] assumed the user latent vector always reflects the user preferences not only associated with the ratings of the user for that specific time step but also reflect the overall behavior. Based on this claim, they proposed a method that updates the user preference model at a regular time interval, and keep the items' feature vectors fixed based on the claim that item features are time-invariant and does not change. We argued that items feature undergoes several changes at a regular time interval, and therefore we adopt a procedure different from [9].

To learn the time series of user latent vectors for individual users, we begin by decomposing the rating matrix at t = 1 and determine the user feature matrix P and the item feature matrix Q related to this time step. As such, the user-item interactions for each time period is modeled using the proposed *dui*-TMF, defined as follows:

$$\hat{R}_{ij}(t) = P_i(t).Q_j^T(t) \quad (6)$$

In view of this, we first set $P_i(t)$ and $Q_j(t)$ as the original user and item latent vectors P_i and Q_j. Then, the ratings observed for that time step is used to learn $P_i(t)$ and $Q_j(t)$ respectively. When the new batch of rating is received, we then incrementally train the model to obtain an updated latent user vector $P_i(t+1)$ and $Q_j(t+1)$, without the making changes permanently. To solve the optimization problem, we adopt a stochastic gradient descent (SGD) method in order to obtain the optimized learning parameters:

$$e_{ij}(t) = R_{ij}(t) - P_i(t).Q_j^T(t) \quad (7)$$

$$P_i(t) \leftarrow P_i(t) + \alpha\left(e_{ij}(t).Q_j(t) - \lambda.P_i(t)\right) \quad (8)$$

$$Q_j(t) \leftarrow Q_j(t) + \alpha\left(e_{ij}(t).P_i(t) - \lambda.Q_j(t)\right) \quad (9)$$

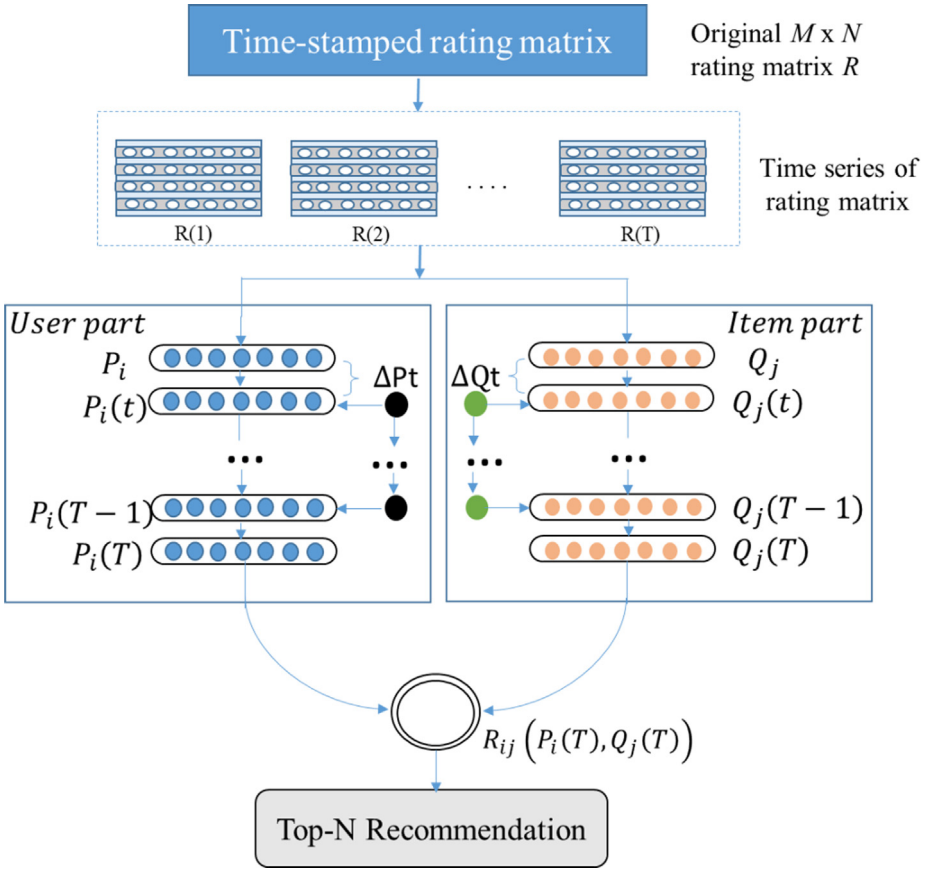


Fig. 1. Proposed temporal matrix factorization model.

This implies that, for every new rating we gain more information about a user preference in addition to the last preferences and thus update the latent vectors related to users and items based on these new ratings. This approach maintains a balance between the old preference and the new preferences and as well ensure that the drifts in the latent vectors are timely addressed.

III. Learn the dynamics of concept drift in $\boldsymbol{P_i(t)}$ *and* $\boldsymbol{Q_j(t)}$

In real life recommender systems, individual users experience drift of interests at different time points or rates. For some users, the change rate occurs frequently, for instance when users buy items as gifts for others during the festive periods, or when several individuals share one account. While other users change their preferences at more slow pace. In either case, learning the model with respect to these dynamic changes distorts the precision of the prediction model. In this section, we assume that the latent vectors P_i and $Q_j$changes over time and analyze how much of these changes occurred in the latent vectors.

In order to track the perceived changed preferences, we first store the latent vectors at time point t, denoted as $P_i(t)$ and $Q_j(t)$. Then, we perform an update training and obtain a new latent user vector $P_i(t+1)$ and the corresponding item vector $Q_j(t+1)$ on the new batch of the ratings. There are different methods for calculating dissimilarity between two concepts. Matuszyk strategy to measure how much the latent factors have changed compared to the previous time point was by calculating the squared difference between $P_i(t+1)$ and $P_i(t)$ [10]. This strategy however, proves to be sufficient only when the change involved is a gradual change type. Therefore, we proposed a Hellinger distance measure to compute the similarity scores in this study. The Hellinger distance measure is a feature based drift detection method that have the benefit of adapting to gradual or abrupt changes in a data distribution [35]. This strategy is formally expressed by the following formula:

$$h(P_i(t+1), P_i(t)) = \frac{1}{\sqrt{2}} \cdot \sqrt{\sum_{i=1}^{k} \left(\sqrt{P_i(t+1)} - \sqrt{P_i(t)}\right)^2}$$

The drift score reveals a piece of important evidence as to whether the user has a consistent taste or if item features are time-invariant. The lower the score value, the more the possibility that the concept-drift has occurred. To test if the change has occurred, we adopt the same principle with [10] by computing the standard deviation $SD(h(P_i(t+1), P_i(t)))$ of how much the latent features have changed compared to the previous time point. In this case, the change is decided if the following inequality is satisfied, else no change has occurred:

$$h(P_i(t+1), P_i(t)) > \alpha . SD(h(P_i(t+1), P_i(t))) \quad (11)$$

where the parameter α controls the sensitivity of the forgetting the old preferences. In every case, the choice of α is specifically chosen for each dataset and has to be determined experimentally. The same strategy is also applied to measure the concept drift in $Q_i(t+1)$ and $Q_i(t)$.

IV. Computing weighted user and item latent factors

After analyzing how much the latent user vector P_i and the item latent vector Q_j changes between two-time points, then we adjust the preference models (the latent vector P_i and the item latent vector Q_j) appropriately. If the new ratings are consistent with this user's preferences, then the change in the latent vectors between $P_i(t)$ and $P_i(t+1)$ is expected to be minimal. However, if the latent vectors changed dramatically after training on the new batch, this

shows that the user preference has drifted away from past preferences. Hence, we update the model of a user preference (Latent item vectors) by multiplying with an exponential function whose value depends on the rate of change of user's preference [36]. This is expressed in the equations (12) and (13) as follows.

$$\overrightarrow{P_i(t+1)} = \alpha^{-SD(h(P_i(t+1),P_i(t)))}.\overrightarrow{P_i(t)} \quad (12)$$

$$\overrightarrow{Q_j(t+1)} = \alpha^{-SD(h(P_i(t+1),P_i(t)))}.\overrightarrow{Q_j(t)} \quad (13)$$

The exponential function controls the extent of the penalty. Specifically, the function takes the values in the range [0, 1) for $\alpha > 1$. For a high value of drift score, the exponential function yields a lower value and as a result, penalize the old preferences the more.

V. Rating prediction

Once the weighted users and items latent vectors have been obtained for the time period $T-1$, i.e., both $P_i(T-1)$ and $Q_j(T-1)$, then we can predict the latent vectors at time T using the equation (14) and (15) as follows:

$$\overrightarrow{P_i(T)} = \alpha^{-SD(h(P_i(t+1),P_i(t)))}.\overrightarrow{P_i(T-1)} \quad (14)$$

$$\overrightarrow{Q_j(T)} = \alpha^{-SD(h(P_i(t+1),P_i(t)))}.\overrightarrow{Q_j(T-1)} \quad (15)$$

After learning the latent factors P and Q at time T, the future prediction of the unseen items can be predicted using the inner product of both $P_i(T)$and $Q_j(T)$ latent vectors in equation (6) that is:

$$\widehat{R}_{ij}(T) = P_i(T).Q_j^T(T) \quad (16)$$

5. Experiments

In this section, we performed a series of experiments to evaluate the performance of the proposed method using the four real-world datasets. These comprises of MovieLens 1 M, Flixster, Ciao, and Epinions datasets. These datasets were carefully chosen based on the fact that they have timestamped information and are broadly used in collaborative filtering research projects that involve tracking the dynamics of concept drift in the recommender system. Specifically, the MovieLens 1 M dataset contains ratings for over a period of three years (from the year 2000 to 2003) for the user who joined ML during the year 2000. As such, the ratings in this category correspond to three years of a follow-up by these users, therefore the dataset has enough users, items, and ratings to detect the trends of users' preferences. Specifically, the dataset has 1, 000, 0054 ratings on 3,900 items provided by 6040 users respectively. Flixster is an American-based social movie site that provides applications for customers to share the ratings or reviews about movies. The dataset contains the rating of 7, 837, 76, by 114,747 users on 44,439 items. Ciao is a European-based online site with claims, which have reached an audience of 28.4 million monthly unique users in Europe. In the Ciao dataset, we sample a subset of the dataset with 22,894 ratings on 5,004 ratings by 1,947 users respectively. Epinions is one of the largest users' review site, established in 1999 comprises of thousands of users and product records in the world. These datasets provide services for users to rate movies based on a 5-point Likert scale. It thus provides information about the user, item, users' preference ratings, and the associated timestamp information. The statistics about these datasets is provided in Table 2.

To enable proper tracking of change detection, new users and new items are removed, that is, the users or items that appear in less than 20 times in the dataset and focus on the existing users and items for tracking changes in their latent vectors. The parameters that we adopted for our experiments are: the number of k factors = 40, the learning rate α = 0.003, and the regulator λ = 0.01. To obtain the latent factors for each time period, we perform 50 iterations of SGD. The parameters for computing the user and item latent vectors for each time period in our method are set to be the same as those for the baselines. We compare the results against three baseline methods in different aspects to show the benefit of tracking concept drift in the latent vectors. These baselines include:

MF. Matrix Factorization method is the most widely and successful method for rating prediction in the context of recommendation tasks [31,32]. Here, we assume that user behaviours as well as items features are time invariants and therefore ignore tracking their changing situations.

timeSVD++. This method models the temporal dynamics of users by a simple sum of time-changing rating biases for each user and each item with the estimate from the traditional MF [4].

TMF. This is a state-of-the-art method for tracking the concept drift of each individual user's preferences over time [20]. It is based on a linear system model with a user-specific transition matrix which represent the concept drift of user preference in each time steps.

We implemented the proposed model in python 3.5 and tune the parameters based on time-aware RMSE and report the results based on the test dataset. All the models are trained with the stochastic gradient decent algorithm until converge with at least 50 iterations. Table 3 shows the performance of the three baselines and the improved methods when exploiting the dynamic changes in a user and item features. Fig. 2 shows the experimental results on the four datasets with explicit rating feedbacks for better explanations.

First, in comparison with the previous approaches, the proposed temporal model improves the performance in terms of RMSE across all the four datasets. However, the performances of different baselines varies across datasets. As could be seen from Fig. 2, timeSVD++ and TMF performed best compare to MF across all the datasets, which shows the benefits of temporal models over the non-temporal models. Also, we noticed that the temporal models shows different capabilities in various scenarios. Specifically, the timeSVD++ and TMF perform quite well in Ciao and Epinion datasets, which may be the result of longer time span and the sensitivity to dynamic preferences in these datasets. This explains the fact that the performance gain of tracking concept drift effects depend on the domain dataset as a result of some intrinsic properties. As for our *dui*_TMF model, we achieve the best performance improvements compare to all baselines methods. Specifically, on the Ciao, we obtain 14% improvement, while in Epinion dataset, we obtained 10% improvement, which implies that these datasets are more beneficial to concept drifts both in terms of user latent vectors and item latent vectors as well. But in other datasets, users' interactions tend to be in more consistent direction, therefore, mainly modelling temporal dynamics in these datasets result in lower performance gain.

Second, in order to check the validity of tracking the dynamic changes from both the latent vectors of the user and item models, we also examine the effect of changes in user latent vectors as well as item latent vectors over time (see Fig. 3). Here, the opinions vary about the importance of tracking changes in item latent vectors. The previous studies assume that item vectors are invariant over time and thus ignored the dynamic properties of items [20]. Therefore, our study verifies the impact of dynamic changes in items properties and the results showing the evolution of both the user latent vectors as well as the item features are provided in the Tables 3-6. However, as a result of space constraints and for an easier comparison, we report only the results on the Ciao and

Table 2
Datasets statistics.

Datasets	Number of users	Number of items	Number of ratings	Density	Earlier date	End date
ML 1 M	6040	3,900	10,000,054	0.80%	April 2000	Feb. 2003
Flixster	114,747	44,439	7,837,765	0.14%	Dec. 2005	Nov. 2009
Ciao	1,942	5.004	22,894	0.23%	Jan 2000	April 2011
Epinions	21,752	242,842	853,664	0.02%	Jul. 199	May 2011

Table 3
Experimental result based on RMSE on the four datasets.

	MovieLens M1	Flixster	Ciao	Epinions
MF	0.8393	1.1489	1.1099	1.1598
timeSVD++	0.7957	1.2006	1.1124	1.0341
TMF	0.8040	1.1102	1.0540	1.1089
dui-TMF	0.7931	1.0843	0.9131	1.0043
Improvement	1.09%	2.59%	14.09%	10.46%

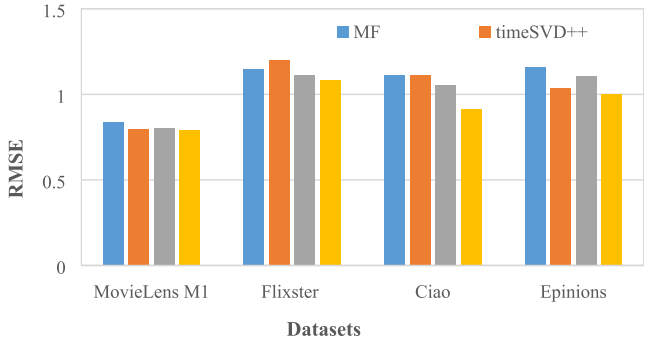
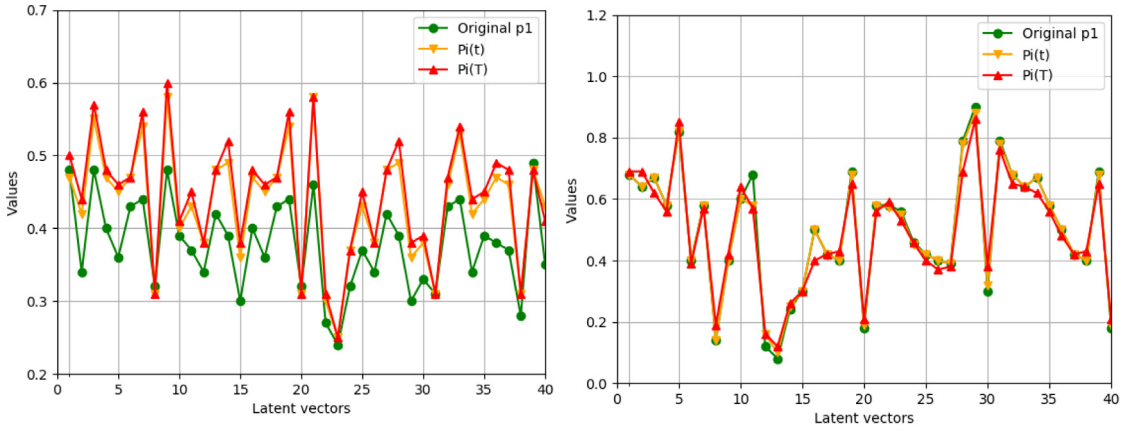


Fig. 2. Comparison of results for MF, timeSVD++, TMF, and *dui*-TMF on all the datasets.

MovieLens datasets. From the user preference perspective, our finding reveals that users tend to change their interest in different ways which require a more robust approach to account for the differences in user's preference drift. The implication is that, even if two users share the same view about same items for some time, they will likely to have different views at some point in the future. Also, the rate at which user's preference changes varies among different users. Some users basically tend to change their interests more rapidly than others. To gain a better understanding of these concepts, we conducted a qualitative analysis. First of all, we consider the changes in the first five latent factors which represent the user preferences for user 63 and user 155 latent vectors based on the MovieLens dataset as well as Ciao dataset. In Table 4, we recorded the first five factors of the user latent vectors for the different time periods t, and the predicted factors of the user latent vector $Pi\ (T)$ where $T = 6$. In the case of user 63, the predicted latent vector at time period 6 using the proposed method is significantly different from the initial latent vector at time 1, and the changes occurred more rapidly after the fifth period. However, in the case of user 154 as reported in Table 5, the rate of change is relatively slow. The predicted latent vector for this user at time period 6 is very close to that obtained at the origin.

Third, in the perspective of item latent factors, Table 6 analyses the advantage of computing the dynamic changes in item latent vectors. As can be seen from the table, the results show that items affinity values for each of the latent factors also change from time to time. The changes occur for several reasons such as when a particular item became popular as a result of the presence of a particular actor in a movie or the arrival of the festive periods. These factors have the tendency to shift the preferences for the item latent factors. For example, we show the evolution of the first five latent factors of the latent vector for item 80 and item 218 in the Ciao and the MoveLens dataset. It is also observed that the changes in items features vary among items. In the case of item 80, the pre-



a) Dynamic changes in user latent vectors for user 63 b) Dynamic changes in user latent vectors for user 154

Fig. 3. Result on the Cio and MovieLens datasets for dynamic changes in user latent vectors. The left figure shows the changes in user latent vectors for user 63 in Cio dataset. While the right figure shows the changes in user latent vectors for user 154 in MovieLens dataset.

Table 4
User 63 latent vectors for the first five factors in the Ciao dataset.

Factors	1	2	3	4	5
P_i	0.4878	0.3468	0.4869	0.4085	0.3670
$P_i(1)$	0.5078	0.4468	0.5769	0.4885	0.4670
$P_i(2)$	0.4678	0.3802	0.5791	0.4043	0.3935
$P_i(3)$	0.4835	0.3934	0.4455	0.3813	0.4236
$P_i(4)$	0.4948	0.3447	0.4748	0.3699	0.3887
$P_i(5)$	0.4344	0.3785	0.5693	0.3895	0.3703
$P_i(6)$	0.4740	0.4207	0.5580	0.4711	0.4575

Table 5
User 154 latent vectors for the first five factors in the MovieLens dataset.

Factors	1	2	3	4	5
P_i	0.6878	0.6443	0.6787	0.5832	0.8241
$P_i(1)$	0.6758	0.6353	0.6787	0.5832	0.8241
$P_i(2)$	0.6419	0.6445	0.6144	0.9932	0.6241
$P_i(3)$	0.5582	0.7354	0.6480	0.5837	0.7241
$P_i(4)$	0.5437	0.7543	0.5675	0.5432	0.7133
$P_i(5)$	0.5418	0.7312	0.5819	0.5985	0.7345
$P_i(6)$	0.6966	0.6923	0.6231	0.5625	0.8584

Table 6
Item 80 latent vectors for the first five factors in the Ciao dataset.

Factors	1	2	3	4	5
q_j	0.5235	0.7034	0.7255	0.7113	0.5236
$q_j(1)$	0.5835	0.8027	0.8515	0.8613	0.5036
$q_j(2)$	0.5748	0.7174	0.7284	0.7869	0.4987
$q_j(3)$	0.4085	0.6461	0.6163	0.7075	0.4311
$q_j(4)$	0.4023	0.6178	0.7305	0.7239	0.4856
$q_j(5)$	0.4158	0.6554	0.7559	0.7605	0.4721
$q_j(6)$	0.5748	0.7174	0.8284	0.8669	0.5087

dicted latent vector at time period 6 using the proposed method is significantly different from the initial latent vector at time 1, and the changes occurred more rapidly after the fourth period. However, in the case of item 218, the rate of change is relatively slow. The predicted latent vector for this user at time period 6 is very close to that obtained at the origin as shown in Table 7.

Fourth, to further analyze the advantage of computing the dynamic changes for each user and item separately, we ran the *dui*-TMF with the following configurations, adopted from [29]: (a) *dui*-TMF: where the dynamic drift score is calculated for each individual user and item latent vectors as in equation (11) and (12); (b) *dui*-TMF -c: where a common drift score (c) is used for all users and items. We chose three values for s ranging from small to high: c $\in$ {0.1,0.5,1}. The results are shown in Table 8. The TA_RMSE for this experiments is shown in Fig. 4. The interesting observations from the result include: (1) Using *dui*-TMF -c with adapted concept drift scores for both the user and item latent vectors is better than the baselines that do not consider item dynamicity. (2) *dui*-TMF significantly improves the performance of the recommendation task by computing the individual concept drift scores for both the user and item vectors on all the datasets as shown in Fig. 5.

To better understand the performance of *dui*-TMF -c for individual latent factors using different values of c, we conducted a qualitative analysis on the test set of the Ciao dataset. We found that when c is set to 0.1, the result shows better performance for a number of 41 users but found to be worst for 347 users. When increasing c to 0.5, it shows to be improved for a numbers of 104 users but worsened for 284 users. This implies that, for 104 users, c = 0.5 is a better choice than c = 0.1 but for 284 users, reverse is the case. The same observation is made when increasing c to 1, related to c = 0.5. The results show to be better for 218 users but worst for 170 users. A notable observation is the significant changes in results when different values of c are used. For c = 0.5, we observed that not all users who showed improved results for this value also showed improvement for c = 1. When moving to c = 1, additional 171 of users showed better results, while about 57 of users got worst results. This analysis proves our argument that each individual user or item has distinct drift scores that do not necessarily fit for others.

In summary, the results demonstrate that *dui*-TMF clearly captures the temporal dynamics of user and item features, and suggested that computing drift scores for each individual user and item latent features yields a better result than assigning a common drift value for each dimension.

6. Conclusion and future work

In this paper, we introduced *dui*-TMF, a time-aware approach that combined the concept drift effects of both the user and item latent vectors to improved recommendation tasks. Based on the preference vectors of the user and item latent vectors derived with the MF method, we compute a user-specific and item-specific drift scores. Consequently, these scores are used to weigh down the importance of old features and make the recommender system more adaptive to dynamic changes in user's preferences and item specific features. These weights control the contribution of the latent vectors in future predictions. The interesting findings from this work are: (1) in order to achieve realistic recommendation, it is essential to consider a dynamic changes of user preferences and dynamic changes in item features when making predictions; and (2) as individual users and items drift in different directions,

Table 7
Item 218 latent vectors for the first five factors in the MoveLens dataset.

Factors	1	2	3	4	5
q_j	0.2998	0.4365	0.2460	0.2156	0.2907
$q_j(1)$	0.2845	0.5665	0.2860	0.2356	0.3407
$q_j(2)$	0.2817	0.4194	0.3098	0.3294	0.2800
$q_j(3)$	0.3249	0.4065	0.3178	0.3246	0.2727
$q_j(4)$	0.2793	0.4482	0.3440	0.3219	0.2489
$q_j(5)$	0.2810	0.4645	0.3789	0.2969	0.2551
$q_j(6)$	0.2854	0.5517	0.2864	0.2235	0.3412

Table 8
RMSE comparisons for dui-TMF and dui-TMF – c on Ciao dataset.

	ML 1M	Flixster	Ciao	Epinions
dui-TMF-0.1	0.8006	1.0114	0.8283	0.9997
dui-TMF-0.5	0.7931	1.0843	0.9131	1.0043
dui-TMF-1	0.8083	0.9837	0.9623	1.2025
dui-TMF	0.7207	0.8743	0.8031	0.9473

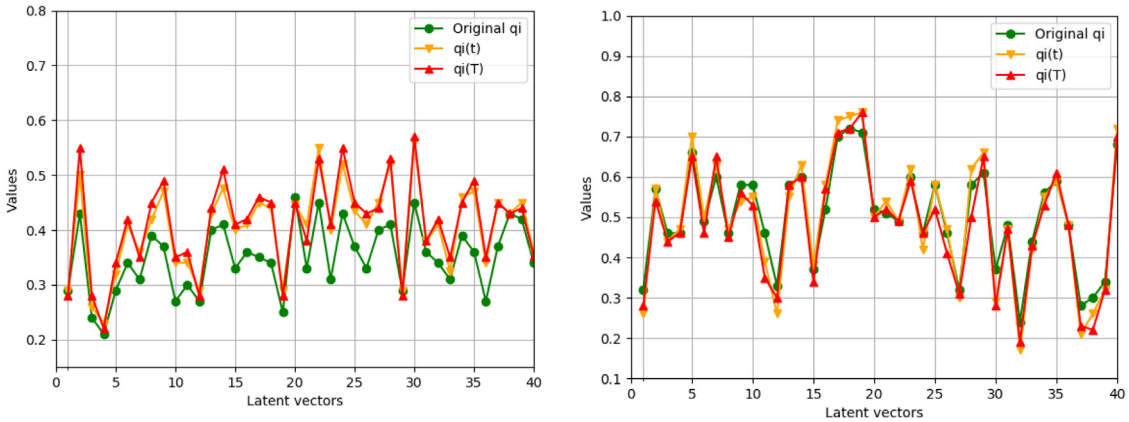


Fig. 4. Result on the Ciao and MovieLens datasets for dynamic changes in item latent vectors. The left figure shows the changes in item latent vectors for item 80 in the Ciao dataset. While the right figure shows the changes in user latent vectors for item 218 in MovieLens dataset.

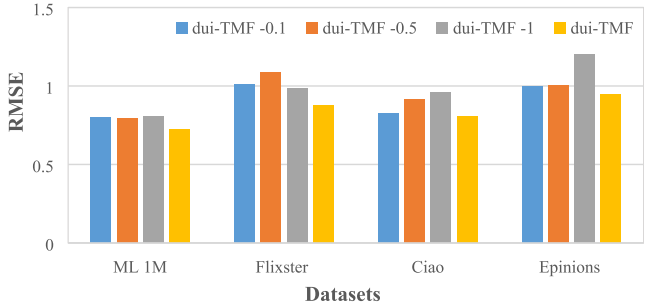


Fig. 5. RMSE comparisons for *dui*-TMF and *dui*-TMF –c on Ciao dataset.

drift scores should be computed for each user and item separately. Our approach to improving the adaptivity of recommenders is by continuously updating the obsolete information. We extended a state-of-the-art Matrix Factorization method for the implementation of our proposed method. We performed several experiments on four datasets with rating feedback. The main aspect that we plan to investigate in our future work is to explore several drift detection methods that allow learning the concept-drift by performing rigorous change detection analysis instead of relying on the heuristic approach of calculating the Hellinger distance of the previous user or item features.

Funding

This research was funded by the Universiti Teknologi Malaysia, through the grant number: Q.J13000.2551.21H38-Novel Deep Learning, Concept Drift, and Hybrid Models Research Funding Program.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

The authors wish to express their gratitude to Universiti Teknologi Malaysia (UTM), the Ministry of Higher Education Malaysia (MOHE) and Ibram Badamasi Babangida University, Lapai, Niger State, Nigeria for their financial support.

References

[1] Isinkaye FO, Folajimi YO, Ojokoh BA. Recommendation systems: Principles, methods and evaluation. Egypt Inform J 2015;16(3):261–73.
[2] Sun B, Dong L. Dynamic model adaptive to user interest drift based on cluster and nearest neighbors. IEEE Access 2017;5:1682–91.
[3] Zhang Q, Wu D, Lu J, Liu F, Zhang G. A cross-domain recommender system with consistent information transfer. Decis Support Syst 2017;104:49–63.
[4] Koren Y. Collaborative filtering with temporal dynamics. Commun ACM 2010;53(4):89–97.
[5] N. Hariri, B. Mobasher, and R. Burke, "Context adaptation in interactive recommender systems," pp. 41–48, 2014.
[6] P. M. Gonçalves, S. G. T. De Carvalho, R. S. M. Barros, and D. C. L. Vieira, "Expert Systems with Applications A comparative study on concept drift detectors," vol. 41, pp. 8144–8156, 2014.
[7] Yuan Q, Cong G, Ma Z, Sun A, Magnenat-Thalmann N. Time-aware point-of-interest recommendation. SIGIR 2013 - Proceedings of the 36th international ACM SIGIR conference on research and development in information retrieval, 2013.
[8] McAuley J, Leskovec J. From amateurs to connoisseurs: Modeling the evolution of user expertise through online reviews. WWW 2013 – proceedings of the 22nd international conference on world wide web, 2013.
[9] Y. Lo, W. Liao, C. Chang, and Y. Lee, Temporal matrix factorization for tracking concept drift in individual user preferences, pp. 1–13.
[10] Matuszyk P, Vinagre J, Spiliopoulou M, Jorge AM, Gama J. In: Proceedings of the ACM symposium on applied computing. p. 947–53.
[11] Abdollahi B, Nasraoui O. Using explainability for constrained matrix factorization. RecSys 2017 – proceedings of the 11th ACM conference on recommender systems, 2017.
[12] Kumar B. A novel latent factor model for recommender system. J Inf Syst Technol Manag 2016.
[13] Portugal I, Alencar P, Cowan D. The use of machine learning algorithms in recommender systems: A systematic review. Expert Syst Appl 2018.
[14] L. Zhang, P. Liu, and J. A. Gulla, "Dynamic attention-integrated neural network for session-based news recommendation," Machine Learning, Springer New York LLC, 14-Oct-2019.
[15] Wang K, Jin Y, Wang H, Peng H, Wang X. Personalized time-aware tag recommendation. In: 32nd AAAI conference on artificial intelligence. AAAI; 2018. p. 2018.
[16] Bellogín A, Sánchez P. Revisiting neighbourhood-based recommenders for temporal scenarios. CEUR workshop proceedings, 2017.
[17] Koren Y, Bell R. Advances in collaborative filtering. Recommender systems handbook, 2015.
[18] Liu NN, Zhao M, Xiang E, Yang Q. Online evolutionary collaborative filtering. In: RecSys'10 – proceedings of the 4th ACM conference on recommender systems. p. 95–102.
[19] Wei S, Ye N, Zhang Q. Time-aware collaborative filtering for recommender systems. Communications in computer and information science, 2012.
[20] Lo YY, Liao W, Chang CS, Lee YC. Temporal matrix factorization for tracking concept drift in individual user preferences. IEEE Trans Comput Soc Syst Mar. 2018;5(1):156–68.
[21] Zhu Z, Li D, Liang JIE, Liu G, Yu HAI. A dynamic personalized news recommendation system based on BAP user profiling method. IEEE Access 2018;6:41068–78.
[22] Escovedo T, Koshiyama A, da Cruz AA, Vellasco M. DetectA: abrupt concept drift detection in non-stationary environments. Appl Soft Comput J 2018;62:119–33.
[23] Al-Hadi IAAQ, Sharef NM, Sulaiman MN, Mustapha N. Review of the temporal recommendation system with matrix factorization. Int J Innov Comput Inf Control 2017.
[24] A. Lommatzsch, B. Kille, Incorporating context and trends in news recommender systems, 2017.
[25] A.B. Othmane A. Tettamanzi S. Villata N.L. Thanh in ICAART 2018 - Proceedings of the 10th international conference on agents and artificial intelligence 2018 48 57
[26] Song D, Li Z, Jiang M, Qin L, Liao L. A novel temporal and topic-aware recommender model. World Wide Web 2019;22(5):2105–27.
[27] Zafari F, Moser I, Baarslag T. Modelling and analysis of temporal preference drifts using a component-based factorised latent approach. Expert Syst Appl Feb. 2019;116:186–208.
[28] Al-Ghossein M, Murena PA, Abdessalem T, Barré A, Cornuéjols A. Adaptive collaborative topic modeling for online recommendation. In: RecSys 2018–12th ACM conference on recommender systems. p. 338–46.
[29] Alzogbi A. Time-aware collaborative topic regression: Towards higher relevance in textual item recommendation. CEUR Workshop Proc 2018;2132:10–23.
[30] Liu Y, Liu C, Liu B, Qu M, Xiong H. Unified point-of-interest recommendation with temporal interval assessment. Proceedings of the ACM SIGKDD international conference on knowledge discovery and data mining, 2016.
[31] R. Pálovics, A. A. Benczúr, L. Kocsis, T. Kiss, and E. Frigó, Exploiting temporal influence in online recommendation, RecSys 2014 - Proc. 8th ACM Conf. Recomm. Syst., pp. 2–9, 2014.
[32] Rendle S, Schmidt-Thieme L. Pairwise interaction tensor factorization for personalized tag recommendation. In: WSDM 2010 – Proceedings of the 3rd ACM international conference on web search and data mining. p. 81–90.
[33] Kowald D, Seitlinger P, Kopeinik S, Ley T, Trattner C. Forgetting the words but remembering the meaning: Modeling forgetting in a verbal and semantic tag recommender. Lecture notes in computer science (including subseries lecture notes in artificial intelligence and lecture notes in bioinformatics, 2015.
[34] Feltoni Gurini D, Gasparetti F, Micarelli A, Sansonetti G. Temporal people-to-people recommendation on social networks with sentiment-based matrix factorization. Futur Gener Comput Syst 2018;78:430–9.
[35] Ditzler G, Polikar R. Hellinger distance based drift detection for nonstationary environments. IEEE SSCI 2011: symposium series on computational intelligence - CIDUE 2011: 2011 IEEE symposium on computational intelligence in dynamic and uncertain environments, 2011.
[36] Matuszyk P, Vinagre J, Spiliopoulou M, Jorge AM, Gama J. Forgetting techniques for stream-based matrix factorization in recommender systems. Knowl Inf Syst 2018;55(2):275–304.
[37] Idris Rabiu, Naomie Salim, Aminu Dau, Akram Osman. Recommender System Based on Temporal Models: A Systematic Review 2020;10(7)(2204):1–27. doi: https://doi.org/10.3390/app10072204.