Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

Modelling and analysis of temporal preference drifts using a component-based factorised latent approach

F. Zafari[a,*], I. Moser[a], T. Baarslag[b]

[a] Faculty of Science, Engineering and Technology, Swinburne University of Technology, Melbourne, VIC 3122, Australia
[b] Centrum Wiskunde & Informatica, Amsterdam, Netherlands

ARTICLE INFO

Article history:
Received 18 March 2018
Revised 4 September 2018
Accepted 5 September 2018
Available online 13 September 2018

Keywords:
Latent factor models
Bias
Feature preferences
Feature value preferences
Temporal dynamics
Preference drift

ABSTRACT

In recommender systems, human preferences are identified by a number of individual components with complicated interactions and properties. Recently, the dynamicity of preferences has been the focus of several studies. The changes in user preferences can originate from substantial reasons, like personality shift, or transient and circumstantial ones, like seasonal changes in item popularities. Disregarding these temporal drifts in modelling user preferences can result in unhelpful recommendations. Moreover, different temporal patterns can be associated with various preference domains, and preference components and their combinations. These components comprise preferences over features, preferences over feature values, conditional dependencies between features, socially-influenced preferences, and bias. For example, in the movies domain, the user can change his rating behaviour (bias shift), her preference for genre over language (feature preference shift), or start favouring drama over comedy (feature value preference shift). In this paper, we first propose a novel latent factor model to capture the domain-dependent component-specific temporal patterns in preferences. The component-based approach followed in modelling the aspects of preferences and their temporal effects enables us to arbitrarily switch components on and off. We evaluate the proposed method on three popular recommendation datasets and show that it significantly outperforms the most accurate state-of-the-art static models. The experiments also demonstrate the greater robustness and stability of the proposed dynamic model in comparison with the most successful models to date. We also analyse the temporal behaviour of different preference components and their combinations and show that the dynamic behaviour of preference components is highly dependent on the preference dataset and domain. Therefore, the results also highlight the importance of modelling temporal effects but also underline the advantages of a component-based architecture that is better suited to capture domain-specific balances in the contributions of the aspects.

1. Introduction

Recommender systems suggest items (movies, books, music, news, services, etc.) that appear most likely to interest a particular user. Matching users with the most desirable items helps enhance user satisfaction and loyalty. Therefore, many e-commerce leaders such as Amazon and Netflix have made recommender systems a salient part of their services (Koren, Bell, & Volinsky, 2009). Currently, most recommendation techniques leverage user-provided feedback data to infer user preferences (Chen, Chen, & Wang, 2015). Typically, recommender systems are based on collaborative filtering (CF) (Aldrich, 2011; Koren & Bell, 2011), where the preferences of a user are predicted by collecting rating information from other similar users or items (Ma, Yang, Lyu, & King, 2008). Many recent studies have contributed extensions to the basic Probabilistic Matrix Factorisation (PMF) by incorporating additional information. Despite their popularity and good accuracy, recommender systems based on latent factor models encounter some important problems in practical applications (Zafari & Moser, 2016). In these models, it is assumed that all values for item features are equally preferred by all users.

Another major problem with latent factor models based on matrix factorisation is that they do not usually take conditional preferences into consideration (Liu, Wu, Feng, & Liu, 2015). Furthermore, in general, latent factor models do not consider the effect of social relationships on user preferences, which encompasses peer selection (homophily) and social influence (Lewis, Gonzalez, & Kaufman, 2012; Zafarani, Abbasi, & Liu, 2014). In previous work, we addressed the problem of modelling the socially-influenced con-

* Corresponding author.
E-mail addresses: fzafari@swin.edu.au (F. Zafari), imoser@swin.edu.au (I. Moser), T.Baarslag@cwi.nl (T. Baarslag).
URL: http://www.ict.swin.edu.au/personal/imoser/ (I. Moser)

https://doi.org/10.1016/j.eswa.2018.09.010
0957-4174/© 2018 Elsevier Ltd. All rights reserved.

ditional feature value preferences, and proposed CondTrustFVSVD (Zafari & Moser, 2017).

Since data usually changes over time, the models should continuously update to reflect the present state of data (Koren, 2010). A major problem with the most of the recent recommender systems is that they mostly ignore the drifting nature of preferences (Zafari & Moser, 2017). Modelling the time drifting data is a central problem in data mining. Drifting preferences can be considered a particular type of concept drift, which has received much attention from researchers in recent years (Widmer & Kubat, 1996). However, very few recommendation models have considered the drifting nature of preferences (Chatzis, 2014). Changes in user preferences can originate from substantial reasons, or transient and circumstantial ones. For example, the items can undergo *seasonal changes* or some items may experience *periodic changes*, for instance, become popular in the specific holidays.

Apart from the short-term changes, user preferences are also subject to long term drifts. For example, a user may be a fan of romantic or action movies at a younger age, while his/her preference may shift more towards drama movies as gets older. Also, users may change their rating scale over time. For example, a user may be very strict and give 3 out of 5 for the best movie. However, he/she might become less strict with age and be more willing to elect the full rate when fully satisfied. A similar situation may apply for movies. A movie may receive a generally high/low rate at some time period, and lower/higher rates at some other period (Koren, 2010). Therefore, a preference model should be able to distinguish between different types of preference drifting, and model them individually in order to achieve the highest accuracy.

In recommender systems research, six major aspects to the preferences have been identified. These aspects include *feature preferences* (Salakhutdinov & Mnih, 2011; Zafari, Nassiri-Mofakham, & Hamadani, 2015), *feature value preferences* (Zafari & Nassiri-Mofakham, 2016; 2017; Zhang et al., 2014), *socially-influenced preferences* (Jamali & Ester, 2010; Ma et al., 2008; Ma, Zhou, Liu, Lyu, & King, 2011; Zafari & Moser, 2017; Zhao, Wang, Chen, & Cao, 2015), *temporal dynamics* (Koren, 2010), *conditional preferences* (Liu et al., 2015), and *user and item biases* (Koren & Bell, 2011). Feature value preferences refer to the relative favourability of each one of the item feature values, social influence describes the influence of social relationships on the preferences of a user, temporal dynamics means the drift of the preferences over time, conditional preferences refer to the dependencies between item features and their values, and user and item biases pertain to the systematic tendencies for some users to give higher ratings than others, and for some items to receive higher ratings than others (Koren & Bell, 2011). Modelling the temporal properties of these preference aspects is the central theme of this paper.

In this paper, we extend our previous work (Zafari & Moser, 2017), by considering the drifting nature of preferences and their constituting aspects. We assume that the socially-influenced preferences over features and conditional preferences over feature values, as well as user and item rating scales can be subject to temporal drift. Therefore, the two major research questions addressed in this paper are:

- How can we efficiently model the drifting behaviour preferences, and how much improvement would incorporating such information make?
- Which aspects are more subject temporal changes, and how is this related to the domain on which the model is trained?

The current work proposes a novel latent factor model based on matrix factorisation to address these two questions. This paper has two major contributions for the field. In this paper, we make further improvements on the accuracy of, CondTrustFVSVD, a model that we proposed earlier. CondTrustFVSVD proved to be the most accurate model among a large set of state of the art models. The additional improvements were achieved by incorporating the *temporal dynamics of preference aspects*. We also draw conclusions about the dynamicity of preference aspects, by analysing the temporal aspects of the these aspects using a component-based approach, and show which aspects are more subject to drift over time. This research provides useful insights into the *accurate modelling of preferences and their temporal properties*, and helps pave the way for boosting the performance of recommender systems. The findings suggest that the temporal aspects of user preferences can vary from one domain to another. Therefore, *modelling domain-dependent temporal effects of preference aspects* are critical in improving the quality of recommendations.

The rest of the paper is organised as follows: The related work is introduced in Section 2. In Section 3.1, we first briefly introduce probabilistic matrix factorisation, and CondTrustFVSVD. Then in Section 3.2 we introduce Aspect-MF to overcome the challenge of learning drifting conditional socially-influenced preferences over feature values. In Section 4, we first explain the experimental setup, and then report on the results of Aspect-MF using two popular recommendation datasets. Finally we conclude the paper in Section 5, by summarising the main findings and giving the future directions of this work.

2. Related work

Collaborative Filtering models are broadly classified into memory-based and model-based approaches. Memory- or instance-based learning methods predict the user preferences based on the preferences of other users or the similarity of the items. Item-based approaches in memory-based CF (D'Addio & Manzato, 2015) calculate the similarity between the items, and recommend the items similar to the items that the user has liked in the past. User-based approaches recommend items that have been liked by similar users (Ma et al., 2008). The time-dependent collaborative filtering models are also classified into the memory-based time-aware recommenders and model-based time-aware recommenders (Xiang & Yang, 2009).

2.1. Model-based time-aware recommenders

The models in this category usually fall into four classes: (1) models based on Probabilistic Matrix Factorisation, (2) models based on Bayesian Probabilistic Matrix Factorisation, and (3) models based on Probabilistic Tensor Factorisation, and (4) models based on Bayesian Probabilistic Tensor Factorisation.

2.1.1. Models based on probabilistic matrix factorisation

Modelling the drifting preferences using a model-based approach based on PMF has first been considered by Koren (2010) in TimeSVD++. TimeSVD++ builds on the previous model called SVD++ (Koren et al., 2009), in which the user preferences are modelled through a latent factor model that incorporates the user bias, item bias, and also the implicit feedback given by the users. For each one of these preference aspects, Koren (2010) used a time-dependent factor to capture both transient and long-term shifts. They showed TrustSVD++ achieves significant improvements over SVD++ on a daily granularity (Xiang & Yang, 2009).

In TrustFVSVD (Zafari & Moser, 2017), we extended TrustSVD by adding the preferences over feature values and the conditional dependencies between the features. We did this by adding additional matrices that captured the feature value discrepancies, where the values of these matrices were related to the values of the social influence matrix. In TrustFVSVD, the explicit influence of the social relationships on each one of the aspects of preferences were captured. Through comprehensive experiments on three benchmark datasets, we showed that TrustFVSVD significantly outperformed TrustSVD and a large set of state of the art models. However, similar to most of the state of the art models, in TrustFVSVD, we assumed that the preferences are static.

Another model-based time-aware recommendation model was proposed by Koenigstein, Dror, and Koren (2011). In this model, the authors use session factors to model specific user behaviour in music learning sessions. Unlike TimeSVD++ which is domain-independent, was developed especially for the music domain. First, it enhances the bias values in SVD++, by letting the item biases share components for items linked by the taxonomy. For example, the tracks in a good album may all be rated higher than the average, or a popular artist may receive higher ratings than the average for items. Therefore, shared bias parameters are added to different items with a common ancestor in the taxonomy hierarchy of the items. Similarly, the users may also tend to rate artists or genres higher than songs. Therefore, the user bias is also enhanced by adding the type of the items. It is also assumed that unlike in the movies domain, in music it is common for the users to listen to many songs, and rate them consecutively. Such ratings might be rated similarly due to many psychological phenomena. The advantage of the models proposed by Koenigstein et al. (2011) and Koren (2010) that extend SVD++ is that they enable the capturing of dynamicity of the preference aspects with a high granularity for aspects that are assumed to be more subject to temporal drift. Furthermore, as shown by Koenigstein et al. (2011), domain-dependent temporal aspects of the preferences and their individual aspects can also be taken into consideration.

Jahrer, Töscher, and Legenstein (2010) split the rating matrix into several matrices, called bins, based on their time stamps. For each bin, a separate time-unaware model is trained by producing an estimated rating value that is obtained using the ratings of given for that bin. Each one of the bins is assigned a weight value, and the final rating is obtained by combining the ratings that are obtained through the models trained on each bin. Therefore, using this approach, they combine multiple time-unaware models into a single time-aware model. The disadvantage of this model is that the ratings matrix is usually sparse as it is, and it even becomes sparser, when the ratings are split into bins.

A similar approach is followed in the model proposed by Liu and Aberer (2013). They systematically integrated contextual information and social network information into a matrix factorization model to improve the recommendations. To overcome the sparsity problem of training separate models based on their time-stamps, they applied a random decision trees algorithm, and create a hierarchy of the time-stamps. For example, the ratings can be split based on year in the first level, month in the second level, day in the third level, and so on. They argue that the ratings that are given at similar time intervals are better correlated with each other, and therefore such clustering is justified. They also added the influence of the social friends to the model, using a context-aware similarity function. In this function users who give similar ratings to those of their friends in similar contexts get higher similarity values. Consequently, in this model, the role of time on the social influence is also indirectly taken into consideration.

Baltrunas, Ludwig, and Ricci (2011) argued that methods based on tensor factorisation can improve the accuracy when the datasets are large. Tensor factorisation requires the addition of a large number of model parameters that must be learned. When the datasets are small, simpler models with fewer parameters can perform equally well or better. In their method, a matrix is added to capture the influence of contextual factors (e.g. time) on the user preferences by modelling the interaction of contextual conditions with the items. Although the model is quite simple and fast, it does not include the effect of time on individual preference aspect. Unlike the models proposed by Koenigstein et al. (2011) and Koren (2010), it can not capture fine-grained and domain-specific dynamicities.

Another recent model in this category is proposed by Rafailidis (2018). He proposes a multi-latent transition model, in which the items' meta-data are used to better capture the transitions of user preferences over an ongoing period of time. Guo, Zhang, and Yorke-Smith (2013) also propose a time-aware model based on matrix factorisation called PCCF to capture periodic and continual temporal effects. Then they show the effectiveness of capturing both effects on three benchmark datasets, and superiority of this model over some state of the art models.

2.1.2. Models based on Bayesian probabilistic matrix factorisation

BPMF extends the basic matrix factorisation (Salakhutdinov & Mnih, 2008) by assuming Gaussian–Wishart priors on the user and item regularisation parameters and letting the hyper-parameters be trained along with the model parameters. Dynamic BPMF (dBPMF) is a non-parametric Bayesian dynamic relational data modelling approach based on the Bayesian probabilistic matrix (Luo & Cai, 2016). This model imposes a dynamic hierarchical Dirichlet process (dHDP) prior over the space of probabilistic matrix factorisation models to capture the time-evolving statistical properties of modelled sequential relational datasets. The dHDP was developed to model the time-evolving statistical properties of sequential datasets, by linking the statistical properties of data collected at consecutive time points via a random parameter that controls their probabilistic similarity.

2.1.3. Models based on probabilistic tensor factorisation

In tensor factorisation methods, the context variables are modelled in the same way as the users and items are modelled in matrix factorisation techniques, by considering the interaction between users-items-context. In tensor factorisation methods, the three dimensional user-item-context ratings are factorised into three matrices, a user-specific matrix, an item-specific matrix, and a context-specific matrix. A model in this category is proposed by Karatzoglou, Amatriain, Baltrunas, and Oliver (2010), who used Tensor Factorisation with CP-decomposition, and proposed multiverse recommendation, which combines the data pertaining to different contexts into a unified model. Therefore, similar to the model proposed by Baltrunas et al. (2011), other contextual information besides time (e.g. user mode, companionship) can also be taken into consideration. However, unlike Baltrunas et al. (2011), they factorise the rating tensor into four matrices, a user-specific matrix, an item-specific matrix, a context-specific matrix, and a central tensor, which captures the interactions between each user, item, and context value. Then the original ratings tensor, which includes the ratings given by users to items in different contexts (e.g. different times) can be reconstructed by combining the four matrices back into the ratings tensor. Other models in this category are the models proposed by Li, Li, Jin, Xue, and Zhu (2011) and Pan, Ma, Pang, and Yuan (2013).

2.1.4. Models based on Bayesian probabilistic tensor factorisation

There is a class of dynamic models that are based on Bayesian Probabilistic Tensor Factorisation (BPTF) (Xiong, Chen, Huang, Schneider, & Carbonell, 2010). BPTF generalises BPMF by adding tensors to the matrix factorisation process. A tensor extends the two dimensions of the matrix factorisation model to three or more dimensions. Therefore, besides capturing the user-specific and item-specific latent matrices, this model also trains a time-specific latent matrix, which captures the latent feature values in different time periods. The models based on tensor factorisation are similar in introduction of the time-specific matrices into the factorisation process. However, they are different in the way they

Table 1
Summary of key notations and symbols used through the paper.

Symbol	Definition
N	number of users
M	number of items
D	number of latent factors
u, v	indexes to denote users u and v
i, j	indexes to denote items i and j
f, f'	indexes to denote latent features f and f'
t_{uj}	the time at which user u rated item j
$P_{uf}(t)$	dynamic preference of user u over latent feature f
Q_{jf}	value of feature f for item j
$W_{uf}(t)$	dynamic gradient value to capture the preference of user u over value of feature f
$Z_{uf}(t)$	dynamic intercept value to capture the preference of user u over value of feature f
y_{jf}	implicit feedback of the users regarding latent feature f of item j
$Y_{ff'}$	feature-specific dependency matrix entry, to capture conditional preferences
T_{uv}	trust value between user u and user v
$\hat{T}^t_{uv}$	estimated influence of user u on user v's preferences over features
$\hat{S}^t_{uv}$, $\hat{G}^t_{uv}$	estimated influence of user u on user v's preferences over feature values
$\lvert T_u\rvert$	number of users user u trusts
$\lvert T^+_v\rvert$	number of users trusted by user v
I_u	the vector of ratings given by user u
$\lvert I_u\rvert$	number of ratings given by user u
$\lvert U_i\rvert$	number of ratings given to item i
ω	the social influence of user u on the other users according to the latent factor model
μ	the average ratings given by all users to all items
$bu_u(t)$	user u's dynamic rating bias
$bi_j(t)$	item j's dynamic rating bias
R_{uj}	the real rating value given by user u on item j
$R'_{uj}(t)$	the predicted rating value given by user u on item j at time t

factorise the ratings matrix into the user, item, and time matrices, and also the way they train the factorised matrices. Similar to BPMF, BPTF uses Markov Chain Monte Carlo with Gibbs sampling to train the factorised matrices.

2.2. Memory-based time-aware recommenders

Some simple time-dependent collaborative filtering models have been proposed by Lee, Park, and Park (2008). The models use item-based and user-based collaborative filtering, and exploit a pseudo-rating matrix, instead of the real rating matrix. In the pseudo-rating matrix the entries are obtained using a rating function, which is defined as the rating value when an item with launch time l_j was purchased at time p_i. This function was inspired by two observations, that more recent purchases better reflected a user's current preferences, and also recently launched items appealed more to the users. If the users are more sensitive to the item's launch time, the function gives more weight to new items, and if the user's purchase time is more important in estimating their current preference, the function assigns more weight to recent purchases. After obtaining the pseudo-rating matrix, the neighbours are obtained as in the traditional item-based or user-based approaches, and the items are recommended to the users. These models are less related to the proposed model in this paper, so we are not going to review them further.

3. Modelling time-aware preference aspects in CondTrustFVSVD

In this section, we explain how to integrate the time-awareness on different aspects of preferences into CondTrustFVSVD (Zafari & Moser, 2017). The main notations used throughout this paper are summarized in Table 1.

3.1. Brief introduction of PMF and CondTrustFVSVD

In rating-based recommender systems, the observed ratings are represented by the user-item ratings matrix R, in which the element R_{uj} is the rating given by the user u to the item j. Usually, R_{uj} is a 5-point integer, 1 point means very bad, and 5 points means excellent. Let $P \in \mathbb{R}^{N\times D}$ and $Q \in \mathbb{R}^{M\times D}$ be latent user and item feature matrices, with vectors P_u and Q_j representing user-specific and item-specific latent feature vectors respectively (N is the number of users, M is the number of items, and D is the number of item features). In PMF, R_{uj} is estimated by the inner product of the latent user feature vector P_u and latent item feature vector Q_j, that is $\hat{R}_{uj} = P_u Q_j^T$.

PMF maximises the log-posterior over the user and item latent feature matrices with rating matrix and fixed parameters given by Eq. (1).

$$\ln p(P, Q|R, \sigma, \sigma_P, \sigma_Q) = \ln p(R|P, Q, \sigma) + \ln p(P|\sigma_P) + \ln p(Q|\sigma_Q) + C \quad (1)$$

where C is a constant that is not dependent on P and Q. σ_P, σ_Q, and σ are standard deviations of matrix entries in P, Q, and R respectively. Maximising the log-posterior probability in Eq. (1) is equivalent to minimising the error function in Eq. (2).

$$argmin_{U,V} \left[E = \frac{1}{2}\sum_{u=1}^{N}\sum_{j=1}^{M} I_{uj}(R_{uj} - \hat{R}_{uj})^2 + \frac{\lambda_P}{2}\sum_{u=1}^{N}\|P_u\|^2_{Frob} + \frac{\lambda_Q}{2}\sum_{j=1}^{M}\|Q_j\|^2_{Frob} \right] \quad (2)$$

where $\|.\|_{Frob}$ denotes the Frobenius norm, and $\lambda_P = \frac{\sigma^2}{\sigma_P^2}$ and $\lambda_Q = \frac{\sigma^2}{\sigma_Q^2}$ (regularisation parameters). *Stochastic Gradient Descent* and *Alternating Least Squares* are usually employed to solve the optimisation problem in Eq. (2). Using these methods, the accuracy of the method measured on the training set is improved iteratively.

As mentioned in the introduction section, the disadvantage of traditional matrix factorisation methods is that the discrepancies between users in preferring item feature values and conditional dependencies between features are disregarded. CondTrustFVSVD (Zafari & Moser, 2017) addresses these problems by adding matrices W and Z to learn the preferences over item feature values. Suppose that a social network is represented by a graph $\mathbb{G} = (\mathbb{V}, \mathbb{E})$, where $\mathbb{V}$ includes a set of users (nodes) and $\mathbb{E}$ represents the trust relationships among the users (edges). We denote the adjacency matrix by $T \in \mathbb{R}^{N\times N}$, where T_{uv} shows the degree to which user u trusts user v. Accordingly, $|T_u|$ denotes the number of users user u trusts, and $|T_v^+|$ is the number of users trusted by user v. Throughout this paper, we use the indices u and v for the users and indices i and j for items, and indices f and f' for item features. In CondTrustFVSVD, all aspects of preferences are assumed to be subject to change by social interactions, and therefore the explicit influence of social relationships on each of the aspects of the preferences are modelled. In this method, we assume that the user preferences over an item feature can be formulated with a linear function. In this function, matrix W is used to capture the "gradient" values and matrix Z is used to learn the "intercept" values. These matrices have the same dimensions as the user matrix P. According to this figure, the probabilities of the matrices P, Q, W, Z, ω, y and vectors bu and bi are dependent on the hyper-parameters σ_P, σ_Q, σ_W, σ_Z, σ_ω, σ_y, σ_{bu} and σ_{bi} respectively. Likewise, the probability of obtaining the ratings in matrix R is conditional upon the matrices P, Q, W, Z, ω, y and vectors bu and bi. CondTrustFVSVD finds the solution for the optimisation problem formulated by Eq. (3).

$$argmin_{P,Q,W,Z,\omega,y,bu,bi}$$

$$\Bigg[E = \frac{\lambda_t}{2}\sum_{u=1}^{N}\sum_{\forall v\in T_u} I_{uv}\left(T_{uv} - \sum_{f=1}^{D} P_{uf}\omega_{vf}\right)^2$$

$$+ \frac{\lambda_t}{2}\sum_{u=1}^{N}\sum_{\forall v\in T_u}\left(T_{uv} - \sum_{f=1}^{D}(1-W_{uf})\omega_{vf}\right)^2$$

$$+ \frac{\lambda_t}{2}\sum_{u=1}^{N}\sum_{\forall v\in T_u}\left(T_{uv} - \sum_{f=1}^{D}Z_{uf}\omega_{vf}\right)^2 + \frac{1}{2}\sum_{u=1}^{N}\sum_{j=1}^{M}(R_{uj}-\hat{R}_{uj})^2$$

$$+ \sum_{u=1}^{N}\left(\frac{\lambda_P}{2}|I_u|^{-\frac{1}{2}} + \frac{\lambda_T}{2}|T_u|^{-\frac{1}{2}}\right)\|P_u\|^2_{Frob} + \frac{\lambda_Q}{2}\sum_{j=1}^{M}\|Q_j\|^2_{Frob}$$

$$+ \sum_{u=1}^{N}\left(\frac{\lambda_W}{2}|I_u|^{-\frac{1}{2}} + \frac{\lambda_T}{2}|T_u|^{-\frac{1}{2}}\right)\|W_u\|^2_{Frob}$$

$$+ \sum_{u=1}^{N}\left(\frac{\lambda_Z}{2}|I_u|^{-\frac{1}{2}} + \frac{\lambda_T}{2}|T_u|^{-\frac{1}{2}}\right)\|Z_u\|^2_{Frob}$$

$$+ \frac{\lambda}{2}\sum_{i=1}^{M}|U_i|^{-\frac{1}{2}}\|y_i\|^2_{Frob} + \frac{\lambda_\omega}{2}\sum_{v=1}^{N}|T_v^+|^{-\frac{1}{2}}\|\omega_v\|^2_{Frob}$$

$$+ \frac{\lambda_{bu}}{2}\sum_{u=1}^{N}|I_u|^{-\frac{1}{2}}bu_u^2 + \frac{\lambda_{bi}}{2}\sum_{j=1}^{M}|U_j|^{-\frac{1}{2}}bi_j^2 + \frac{\lambda_Y}{2}\sum_{f=1}^{D}\sum_{f'=1}^{D}Y_{ff'}^2\Bigg] \quad (3)$$

where $\lambda_W = \frac{\sigma^2}{\sigma_W^2}$, $\lambda_Z = \frac{\sigma^2}{\sigma_Z^2}$, $\lambda_\omega = \frac{\sigma^2}{\sigma_\omega^2}$, $\lambda_y = \frac{\sigma^2}{\sigma_y^2}$, $\lambda_{bu} = \frac{\sigma^2}{\sigma_{bu}^2}$, $\lambda_{bi} = \frac{\sigma^2}{\sigma_{bi}^2}$, $\lambda_Y = \frac{\sigma^2}{\sigma_Y^2}$. μ denotes the global average of the observed ratings, and bu_i and bi_j denote biases for user i and item j respectively. I_u is the set of items rated by user u and U_j is the set of users who have rated item j. The values of $\hat{R}_{uj}$ in Eq. (3) are obtained using Eq. (4).

$$\hat{R}_{uj} = \mu + bu_u + bi_j + \sum_{f=1}^{D}(P_{uf} + |I_u|^{-\frac{1}{2}}\sum_{\forall i\in I_u} y_{if}$$

$$+ |T_u|^{-\frac{1}{2}}\sum_{\forall v\in T_u}\omega_{vf})(W_{uf}Q_{jf} + Z_{uf}) \quad (4)$$

According to the Eq. (4), the user u's preference value over an item j is defined using different aspects. These aspects are user bias, item bias, the **socially-influenced preferences over features**, and the **socially-influenced preferences over feature values**. Therefore, preferences are defined using different aspects that interact with each other by influencing the values of one another.

3.2. Time-aware CondTrustFVSVD (Aspect-MF)

In the following sections, we first provide a high-level view of Aspect-MF by explaining the interactions between aspects that are captured by the model, and then elaborating how the aspects are trained from the users' ratings and social relationships.

3.2.1. Aspect interactions and high-level view of the model

To address the problem of capturing drifting socially-influenced conditional preferences over feature values, we extend the method CondTrustFVSVD, by adding the dynamicity of each one of the preference aspects that are assumed to be subject to concept drift. The method proposed here is abbreviated to Aspect-MF. A high-level overview of the preference aspects in Aspect-MF are presented in Fig. 1. This figure shows how the preference aspects' effects on each other are captured in Aspect-MF. For example, the social aspect influences feature preferences and feature value preferences, while conditional dependencies exist between feature value preferences. Time aspect also causes changes in feature value preferences and user and item biases. There is also interplay between feature preference and feature value preference aspects.

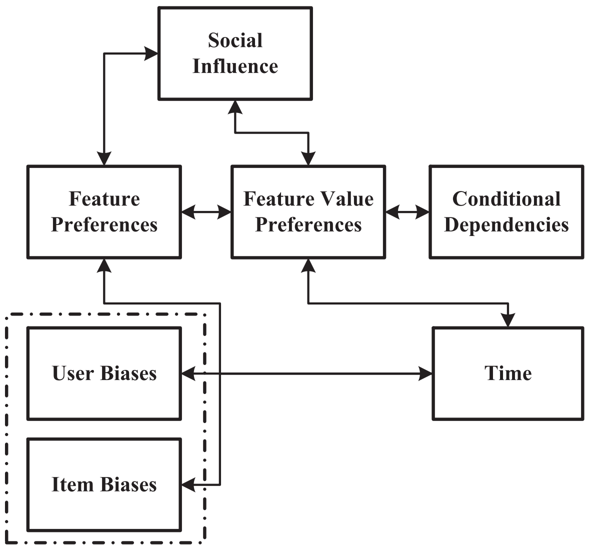


Fig. 1. The preference aspects and their interplay in Aspect-MF.

In Fig. 2b, **FP** represents preferences over features, which is captured by matrix P in the basic matrix factorisation. **F** represents item features captured by matrix Q in the basic matrix factorisation. **CP** represents conditional dependencies, **FVP** represents preferences over feature values, **SI** stands for social influence, and finally **T** is an abbreviation for time. Aspect-MF incorporates additional matrices and vectors into matrix factorisation to capture as many aspects present in the data as possible. As Fig. 2 shows, the model starts by loading the time-stamped user ratings as well as the social network data into the memory. The main loop accounts for the learning iterations over the model. The first loop within the main loop iterates over the time-stamped user-item ratings matrix, while the second loop iterates over the social network adjacency matrix, to train the socially influenced parts of the model. In each loop, one entry of the input matrix is read and used to update the matrices/vectors related to that input data. As can be seen, the user and item bias values are only updated in loop 1, since they are only related to the user-item ratings. Both user-item ratings and users' social relationships include information about the users' preferences over features. Therefore, the new values for FP are calculated in both loops and updated in the main loop, when all new values have been calculated. Similarly, the values for SI and FVP depend on both user-item ratings and social relationships. Consequently, their new values are calculated inside both loops 1 and 2, and are updated in the main loop. In contrast, the values of F as well as CP only need the user-item ratings to be updated. Therefore, they are immediately updated inside loop 1. The time aspect includes parameters that account for the dynamics of user and item biases, feature value preferences, and preferences over features. Since bias values do not depend on the user-item ratings matrix, they are updated immediately in loop 1. However, the new values for the dynamics of feature value preferences, and preferences over features are updated in the main loop. In Aspect-MF, every one of the preference aspects can be arbitrarily switched off and on by setting their respective learning rates and regularisation parameters (hyper-parameters) to zero or a non-zero value respectively.

Although social relationships are likely to be time-dependent, most datasets do not contain this information. Conditional preferences are related to the feature value preferences, since they model

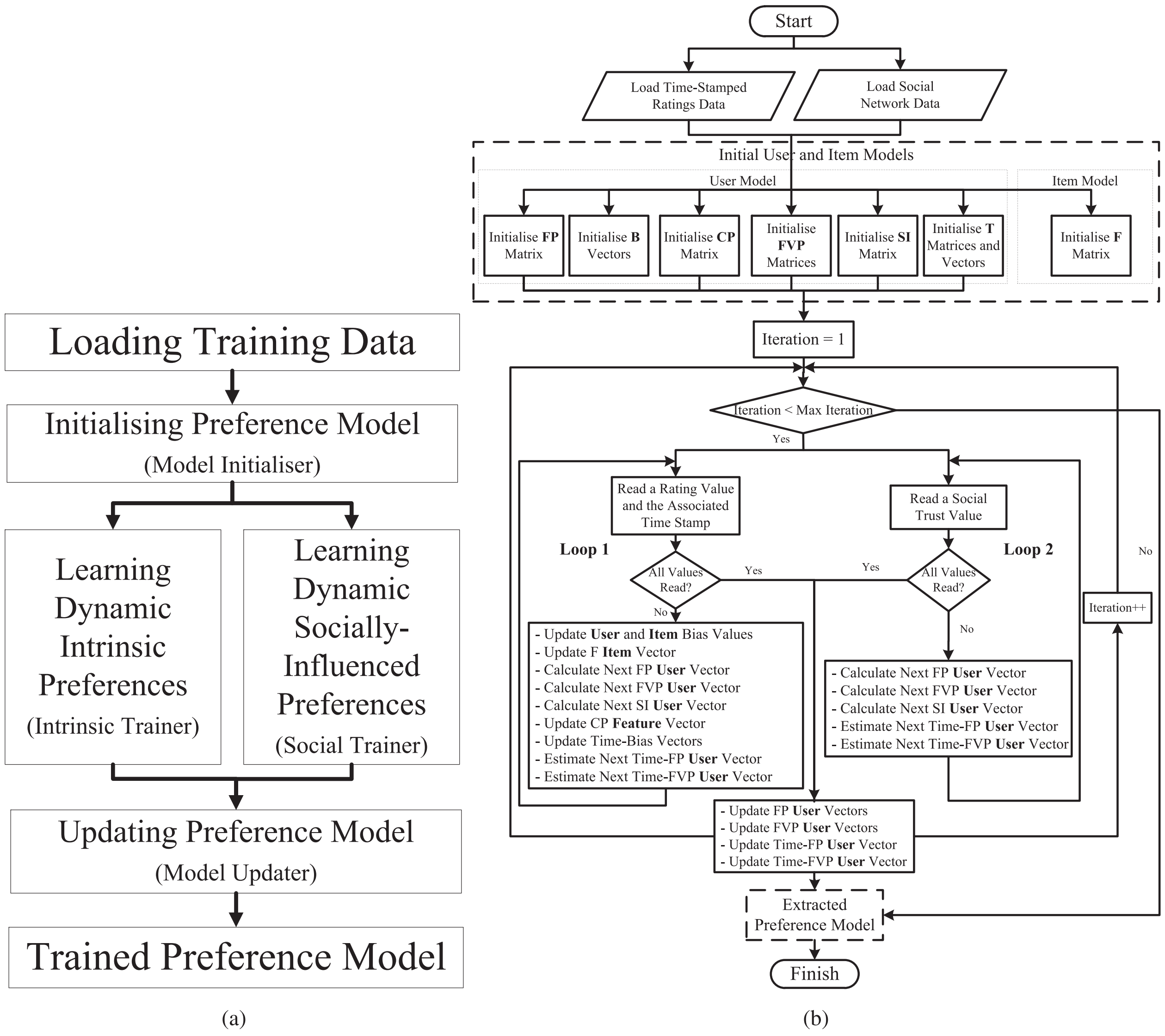

Loading Training Data
Initialising Preference Model
(Model Initialiser)
Learning Dynamic Intrinsic Preferences
(Intrinsic Trainer)
Learning Dynamic Socially-Influenced Preferences
(Social Trainer)
Updating Preference Model
(Model Updater)
Trained Preference Model
(a)
Start
Load Time-Stamped Ratings Data
Load Social Network Data
Initial User and Item Models
User Model
Item Model
Initialise FP Matrix
Initialise B Vectors
Initialise CP Matrix
Initialise FVP Matrices
Initialise SI Matrix
Initialise T Matrices and Vectors
Initialise F Matrix
Iteration = 1
Iteration < Max Iteration
Yes
No
Read a Rating Value and the Associated Time Stamp
Read a Social Trust Value
Loop 1
Loop 2
All Values Read?
Yes
No
- Update User and Item Bias Values
- Update F Item Vector
- Calculate Next FP User Vector
- Calculate Next FVP User Vector
- Calculate Next SI User Vector
- Update CP Feature Vector
- Update Time-Bias Vectors
- Estimate Next Time-FP User Vector
- Estimate Next Time-FVP User Vector
- Calculate Next FP User Vector
- Calculate Next FVP User Vector
- Calculate Next SI User Vector
- Estimate Next Time-FP User Vector
- Estimate Next Time-FVP User Vector
Iteration++
- Update FP User Vectors
- Update FVP User Vectors
- Update Time-FP User Vector
- Update Time-FVP User Vector
Extracted Preference Model
Finish
(b)

Fig. 2. (a) The high-level representation of Aspect-MF and (b) its flow chart.

the dependencies between the features and their values, and therefore, are applied to the matrices that account for the users' preferences over feature values. Social influence is applied to the aspects of preferences over features and preferences over feature values. However, applying social influence to the user and item biases showed no observable benefits and user or item biases do not seem to be influenced by social interactions. Therefore, we concluded that user and item biases are not much influenced by the social interactions (Zafari & Moser, 2017). Therefore, in the most abstract view of the model as depicted in the high-level representation in Fig. 2a, the model is comprised of four main modules. Initialising the model parameters (Model Initialiser), learning the intrinsic constituting aspects of preferences (i.e. preferences over features, preferences over feature values, conditional dependencies, and user and item bias values) and the drifting properties of preferences (Intrinsic Trainer), learning the social influence of the friends over the drifting intrinsic preference aspects (Social Trainer), and finally updating the model to reflect the new information extracted from the data about user ratings, time, and social connections (Model Updater). These modules will be discussed in more details later, when we introduce the algorithm in Section 3.2.4.

3.2.2. Aspect-MF model formulation

In this section, we provide the mathematical formulation of the preferences captured in Aspect-MF. Basically, in Aspect-MF, the user preferences are modelled as a *Bayesian Network* (Korb & Nicholson, 2010). Fig. 3 shows the topology or the structure of the Bayesian Network for user preferences that are modelled by Aspect-MF.

As mentioned earlier, Aspect-MF extends CondTrustFVSVD, by adding the time factor to the aspects of preferences as depicted in Fig 1. In CondTrustFVSVD, the user preferences were captured using the matrices P, Q, W, Z, Y, ω, y, with the hyper-parameters σ_P, σ_Q, σ_W, σ_Z, σ_ω, σ_y, σ_Y, σ_{bu} and σ_{bi}.

In Aspect-MF, the drifting social influence of friends in the user's social network are captured through Eq. (5) to (7).

$$\hat{T}^t_{uv} = \frac{1}{|I^t_u|} \sum^D_{\forall t_{uj} \in I^t_u} \sum^D_{f=1} P_{uf}(t_{uj})\omega_{vf} \tag{5}$$

$$\hat{S}^t_{uv} = \frac{1}{|I^t_u|} \sum^D_{\forall t_{uj} \in I^t_u} \sum^D_{f=1} (1 - W_{uf}(t_{uj}))\omega_{vf} \tag{6}$$

$$\hat{G}^t_{uv} = \frac{1}{|I^t_u|} \sum^D_{\forall t_{uj} \in I^t_u} \sum^D_{f=1} Z_{uf}(t_{uj})\omega_{vf} \tag{7}$$

where $\hat{T}^t_{uv}$, $\hat{S}^t_{uv}$, $\hat{G}^t_{uv}$ model the time-dependent influence of user v on the preferences of user u for the preferences over features

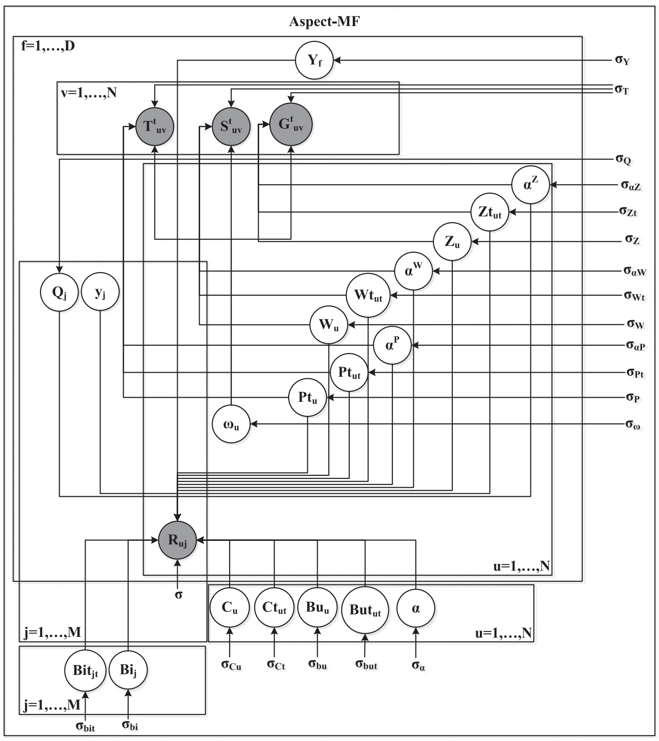


Fig. 3. Bayesian network of Aspect-MF.

(captured by $P_{uf}(t)$) and preferences over feature values (captured by $W_{uf}(t)$ and $Z_{uf}(t)$), and similar to CondTrustFVSVD, ω_{vf} captures *the implicit influence of user v on other users over factor f* and is obtained using the matrix factorisation process. As can be seen in Fig. 1, the user preferences over features and feature values in Aspect-MF are subject to social influence, and they also drift over time. In Eqs. (5) to (7), I^t_u is the set of timestamps for all the ratings given by user u. Therefore, using these equations, the influence of the user v on the preferences of user u is calculated for all the time points, and then it is averaged. Intuitively, these equations are telling us that the trust of user u in user v can be estimated by calculating the average of the weighted averages of user v's influence on user u's preferences for different features, in different times. Intuitively, if user u strongly trusts user v, his preferences would be more strongly influenced by user v. Furthermore, depending on the trust strength of user u in user v and the influence he gets from user v and its direction (positive or negative), the user's preference can be positively or negatively affected. Therefore in Aspect-MF, the user preferences are subject to social influence, and the social influence depends on the strength of their trust in the friends. According to these equations, if there is no relationship between user u and user v, user u's preferences will not be directly affected by the social influence of user v.

In Aspect-MF, the drifting preference value of the user u over an item j at time t is obtained according to Eq. (8).

$$\begin{aligned}\hat{R}_{uj}(t_{uj}) \\ = \ & \mu + bu_u(t_{uj}) + bi_j(t_{uj}) + \sum_{f=1}^{D}(P_{uf}(t_{uj}) + |I_u|^{-\frac{1}{2}} \sum_{\forall i \in I_u} y_{if} \\ & + |T_u|^{-\frac{1}{2}} \sum_{\forall v \in T_u} \omega_{vf})(W_{uf}(t_{uj})Q_{jf} + Z_{uf}(t_{uj})) \\ & + \sum_{f'=1}^{D}\left(\sum_{f=1}^{D}(W_{uf}(t_{uj})Q_{jf} + Z_{uf}(t_{uj}))Y_{ff'}\right)(W_{uf}(t_{uj})Q_{jf'} + Z_{uf}(t_{uj}))\end{aligned} \quad (8)$$

According to Eq. (8), in Aspect-MF, different aspects of preferences as well as user and item biases are subject to temporal drift. As can be seen in Eqs. (5)–(8), the user bias, item bias, preferences over features captured by the matrix P, and preferences over feature values captured by the matrices W and Z are subject to temporal drift. In order to model the drifting properties of these aspects, we use Eqs. (9)–(13).

$$bu_u(t_{uj}) = bu_u + \alpha_u dev_u(t_{uj}) + but_{ut_{uj}} \quad (9)$$

$$bi_j(t_{uj}) = (bi_j + bi_{jBin(t_{uj})})(C_u + Ct_{ut_{uj}}) \quad (10)$$

$$P_{uf}(t_{uj}) = P_{uf} + \alpha^P_u dev_u(t_{uj}) + Pt_{uft_{uj}} \quad (11)$$

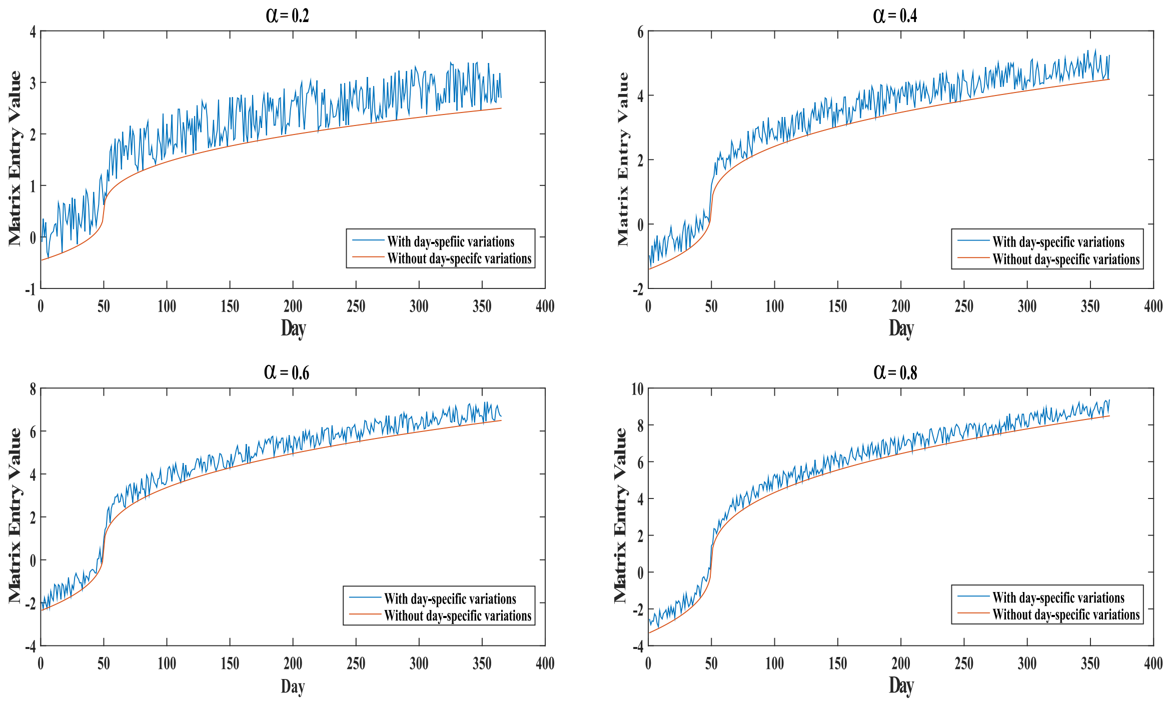
$$Z_{uf}(t_{uj}) = Z_{uf} + \alpha^Z_u dev_u(t_{uj}) + Zt_{uft_{uj}} \quad (12)$$

$$W_{uf}(t_{uj}) = W_{uf} + \alpha^W_u dev_u(t_{uj}) + Wt_{uft_{uj}} \quad (13)$$

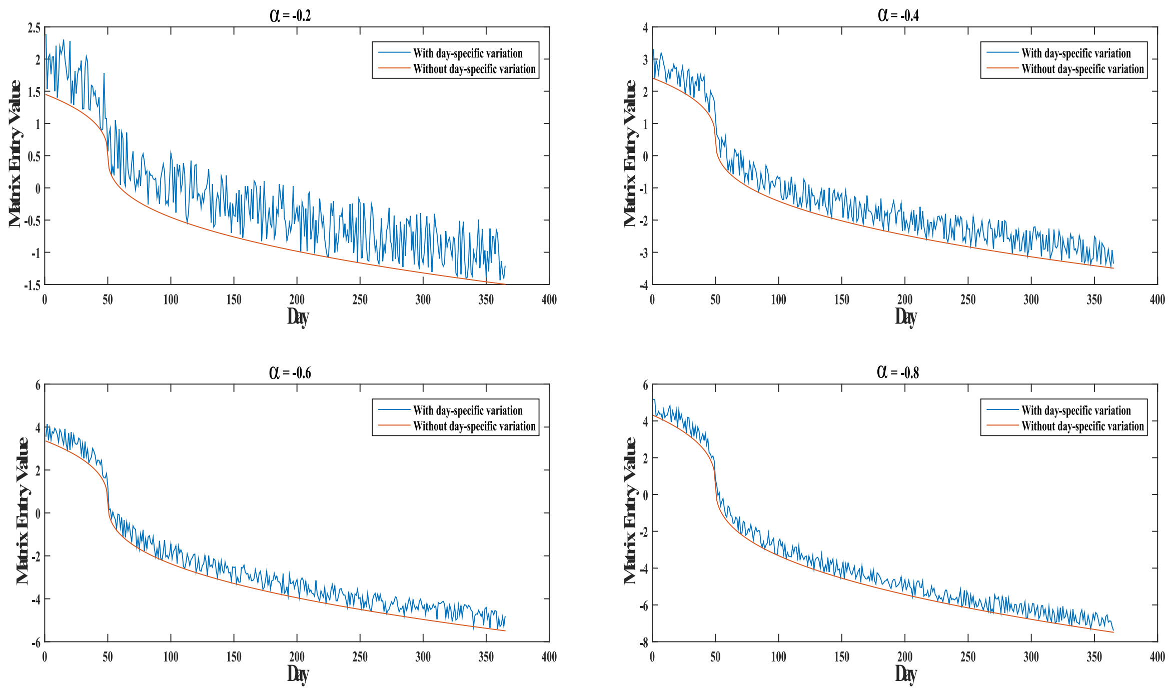
where P_{uf}, W_{uf}, and Z_{uf} capture the static preferences of the user u, while the variables $P_{uft_{uj}}$, $W_{uft_{uj}}$, $Z_{uft_{uj}}$ capture the day-specific variations in the user preferences (e.g. due to the mood of the users in a particular day), and α^P_u, α^W_u, and α^Z_u model the users' long term preference shifts, and $dev_u(t_{uj})$ is obtained according to Eq. (14) (Koren, 2010).

$$dev_u(t_{uj}) = sign(t_{u_{uj}} - t_u).|t_{uj} - t_u|^{\beta} \quad (14)$$

where t_u is the mean of the dates for the ratings given by the user u, and β is a constant value. In Eq. (10), all the dates are placed in a fixed number of bins, and the function $Bin(.)$ returns the bin number for a particular date. For example, if the maximum period of the ratings is 30 years and 30 bins are used, all the rates given in a particular year are placed in a bin, and the function $Bin(.)$ returns the year number for that particular year. The reason why this function is only used for items is that items are not expected to change on a daily basis, and as opposed to users' biases, longer time periods are expected to pass, before we see any changes in the items' popularity. In simple words, $dev_u(t_{uj})$ shows how much the time of the rating given by user u to the item j deviates from the average time of the ratings given by that user. Therefore, if a rating is given at the same time as the average time of the ratings, then the according to these equations, there will be no long-term preference shift for that aspect. However, for instance, if the average time of the rates given by user u is 11/04/2006, the rating of the same item by that user on 11/04/2016 would be different, and this shift is captured by the coefficients of the function $dev_u(t_{uj})$ in Eq. (9) and Eqs. (11)–(13). The drifting preferences captured using Eq. (9) and Eqs. (11)–(13) are depicted in Fig. 4. In these figures, the mean of the dates on which the user has given the ratings are assumed to be 50 (the fiftieth day in a year), and the variations of the user preferences over a period of one year are captured for different values of α in Eq. (9) and Eqs. (11)–13. The red lines in these figures represent the case in which the day-specific variations in the user preferences are not captured, while the blue lines also include the day-specific variations. Therefore, as can be seen, in these figures there are two types of preference shifts, long term drifts (captured by the values of α, α^P, α^W, and α^Z), and short-term or day-specific drifts (captured by the values of *but*, *Pt*, *Wt*, and *Zt*). Therefore, the preference drifts are comprised of small variations from one day to the other, mainly because of temporary factors such as the mood of the user, and the large variations which happen in the long term, as the user changes preferences because of the shift in the his/her tastes. The blue lines show the preference shift patterns that can be learnt by Aspect-MF. Furthermore, the first three terms in Eq. (18) model the social influence of the feature preferences and feature value preferences captured



(a)



(b)

Fig. 4. An example of drifting preferences in Eq. (9) and Eqs. (11)–(13) for (a) positive α values and (b) negative α values.

by P, α^P, Pt, W, α^W, Wt, Z, α^Z, Zt. Therefore, assuming that two users have established the social relationship from the very beginning (which is not essentially true, but usually social relationships do not contain time-stamps), using the Eqs. (5)–(7), the social influence is applied to the preferences of the user over the entire period for which the rating data is record. Therefore, the formulation of the estimated ratings in Aspect-MF (8) allows it to learn the drifting conditional feature value preferences, and the formulation of the optimisation in Aspect-MF (Eq. (18)) enables it to learn the influence of social friends on the drifting preferences of a user.

Eqs. (9)–(13) show how Aspect-MF can capture long-term and short-term drifts in each one of the preference aspects (user bias, item bias, feature preferences, and feature value preferences). The advantage of formulating the problem using Eq. (8) is that each one these aspects can be arbitrarily switched on/off. This results in a component-based approach, in which the model aspects interact with each other, with the purpose of extracting as much preference patterns from the raw data as possible.

3.2.3. Aspect-MF model training

According to the Bayesian network of Aspect-MF in Fig. 3, this model minimises the log-posterior probability of matrices that define the user preferences, given the model hyper-parameters and the training matrix. Formally,

$$argmin_{P,Pt,\alpha^P,Q,W,Wt,\alpha^W,Z,Zt,\alpha^Z,Y,\omega,y,bu,\alpha,but,C,Ct,bi,bit} \{lnp(P,Q,W,Z,\omega,y,bu,bi,\alpha_u,bu_t,bi_{Bin(t)},c,c_t,\alpha^P,\alpha^Z,\alpha^W,P_t,Z_t,W_t \\ |R,T^t,S^t,G^t,\sigma_N\} \tag{15}$$

$\sigma_N=\{\sigma,\sigma_T,\sigma_P,\sigma_{Pt},\sigma_{\alpha^P},\sigma_Q,\sigma_W,\sigma_{Wt},\sigma_{\alpha^W},\sigma_Z,\sigma_{Zt},\sigma_{\alpha^Z},\sigma_\omega,\sigma_y,\sigma_{bu},\sigma_\alpha,\sigma_{but},\sigma_C,\sigma_{Ct},\sigma_{bi},\sigma_{bit},\sigma_Y\}$ denotes the set of all the hyper-parameters. T^t, S^t, G^t respectively denote the real values for the estimated matrices $\hat{T}^t$, $\hat{S}^t$, and $\hat{G}^t$ in Eqs. (5)–(7). According to the Bayesian network in Fig. 3 and by decomposing the full joint distribution using chain rule of probability theory (Korb & Nicholson, 2010) according to the conditional dependencies between the variables defined in this figure, minimising the probability above is equal to minimising the value given in Eq. (16) (Korb & Nicholson, 2010).

$$argmin_{P,Pt,\alpha^P,Q,W,Wt,\alpha^W,Z,Zt,\alpha^Z,Y,\omega,y,bu,\alpha,but,C,Ct,bi,bit,Y} \\ \{lnp(R|P(t),Q,W(t),Z(t),bu(t),bi(t),Y,\sigma)+lnp(Q|\sigma_Q) \\ +lnp(P(t)|\sigma_P)+lnp(W(t)|\sigma_W)+lnp(Z(t)|\sigma_Z) \\ +lnp(bu(t)|\sigma_{bu})+lnp(bi(t)|\sigma_{bi})+lnp(y|,\sigma_y)+lnp(Y|,\sigma) \\ +lnp(T^t_{uv}|\omega,P(t),\sigma_T)+lnp(S^t_{uv}|\omega,W(t),\sigma_T)+lnp(G^t_{uv}|\omega,Z(t),\sigma_T) \\ +lnp(P(t)|\sigma_T)+lnp(W(t)|\sigma_T)+lnp(Z(t)|\sigma_T)+lnp(\omega|,\sigma_T)\} \tag{16}$$

Provided that all the probabilities above follow a normal distribution, it can be shown that minimising the function in Eq. (16) is equivalent to minimising the error value using Eqs. (17) to (19).

$$\begin{aligned} E_R =& \frac{1}{2}\sum_{u=1}^{N}\sum_{j=1}^{M}(R_{uj}-\hat{R}_{uj})^2+\frac{\lambda_Q}{2}\sum_{j=1}^{M}\|Q_j\|^2_{Frob} \\ &+\frac{\lambda_y}{2}\sum_{i=1}^{M}|U_i|^{-\frac{1}{2}}\|y_i\|^2_{Frob} \\ &+\sum_{u=1}^{N}\frac{\lambda_P}{2}|I_u|^{-\frac{1}{2}}(\|P_u\|^2_{Frob}+\|Pt_{ut}\|^2_{Frob}+\|\alpha^P\|^2_{Frob}) \\ &+\sum_{u=1}^{N}\frac{\lambda_W}{2}|I_u|^{-\frac{1}{2}}(\|W_u\|^2_{Frob}+\|Wt_{ut}\|^2_{Frob}+\|\alpha^W\|^2_{Frob}) \\ &+\sum_{u=1}^{N}\frac{\lambda_Z}{2}|I_u|^{-\frac{1}{2}}(\|Z_u\|^2_{Frob}+\|Zt_{ut}\|^2_{Frob}+\|\alpha^Z\|^2_{Frob}) \\ &+\sum_{u=1}^{N}\frac{\lambda_Z}{2}|I_u|^{-\frac{1}{2}}(\|Z_u\|^2_{Frob}+\|Zt_{ut}\|^2_{Frob}+\|\alpha^Z\|^2_{Frob}) \\ &+\frac{\lambda_{bu}}{2}\sum_{u=1}^{N}|I_u|^{-\frac{1}{2}}(bu_u^2+\alpha_u^2+C_u^2+\|bu_u\|^2_{Frob}+\|Ct_u\|^2_{Frob}) \\ &+\frac{\lambda_{bi}}{2}\sum_{j=1}^{M}|U_j|^{-\frac{1}{2}}bi_j^2+\frac{\lambda_{bi}}{2}\sum_{j=1}^{M}\sum_{\forall t\in I^t_j}|U_j|^{-\frac{1}{2}}bit^2_{j,Bin(t)} \\ &+\frac{\lambda_Y}{2}\sum_{f=1}^{D}\sum_{f'=1}^{D}Y^2_{ff'} \end{aligned} \tag{17}$$

$$\begin{aligned} E_T =& \frac{\lambda_t\eta_P}{2}\sum_{u=1}^{N}\sum_{\forall v\in T_u}(T_{uv}-\hat{T}_{uv})^2+\frac{\lambda_t\eta_W}{2}\sum_{u=1}^{N}\sum_{\forall v\in T_u}(T_{uv}-\hat{S}_{uv})^2 \\ &+\frac{\lambda_t\eta_Z}{2}\sum_{u=1}^{N}\sum_{\forall v\in T_u}(T_{uv}-\hat{G}_{uv})^2 \\ &+\sum_{u=1}^{N}\frac{\lambda_T}{2}|T_u|^{-\frac{1}{2}}(\|P_u\|^2_{Frob}+\|Pt_{ut}\|^2_{Frob}+\|\alpha^P\|^2_{Frob}) \\ &+\sum_{u=1}^{N}\frac{\lambda_T}{2}|T_u|^{-\frac{1}{2}}(\|W_u\|^2_{Frob}+\|Wt_{ut}\|^2_{Frob}+\|\alpha^W\|^2_{Frob}) \\ &+\sum_{u=1}^{N}\frac{\lambda_T}{2}|T_u|^{-\frac{1}{2}}(\|Z_u\|^2_{Frob}+\|Zt_{ut}\|^2_{Frob}+\|\alpha^Z\|^2_{Frob}) \\ &+\frac{\lambda_\omega}{2}\sum_{v=1}^{N}|T_v^+|^{-\frac{1}{2}}\|\omega_v\|^2_{Frob} \end{aligned} \tag{18}$$

$$argmin_{P,Pt,\alpha^P,Q,W,Wt,\alpha^W,Z,Zt,\alpha^Z,Y,\omega,y,bu,\alpha,but,C,Ct,bi,bit}[E=E_R+E_T] \tag{19}$$

where I^t_j is the set of timestamps, for all the ratings given to item j, and η_P, η_W, and η_Z are constants added to control the weights of the components related to the social aspect in this equation. The details of the model training can be found in Appendix A.

3.2.4. Aspect-MF algorithm

Algorithm 1 describes the details of the gradient descent method Aspect-MF uses to train the model parameters (P, Pt, α^P, Q, W, Wt, α^W, Z, Zt, α^Z, Y, ω, y, bu, α, but, C, Ct, bi, bit) as expressed in Eq. (19).

The algorithm receives the set of model hyper-parameters λ and the set of learning rates γ as input, and trains the model parameters according to the Bayesian approach described in Section 3.2.2. As we showed in the high-level representation of the algorithm in Fig. 2a, the model is comprised of four basic components. A model initialiser, which initialises the model parameters after the input data is loaded into memory, an intrinsic trainer, which trains the model parameters using the user-item ratings, a social trainer which trains the model parameters using the social relationship data, and finally, a model updater, which updates the model based on the trained parameters for a particular iteration.

Algorithm 1 Model training.

```
1: void ModelTrainer($\lambda$, $\gamma$, maxIter)[a]
2: $\lambda=\{\lambda_T,\lambda_P,\lambda_{Pt},\lambda_{\alpha^P},\lambda_Q,\lambda_W,\lambda_{Wt},\lambda_{\alpha^W},\lambda_Z,\lambda_{Zt},\lambda_{\alpha^Z},\lambda_\omega,\lambda_y,\lambda_{bu},\lambda_\alpha,\lambda_{but},\lambda_C,$
   $\lambda_{Ct},\lambda_{bi},\lambda_{bit},\lambda_Y\}$
3: $\gamma=\{\gamma_T,\gamma_P,\gamma_{Pt},\gamma_{\alpha^P},\gamma_Q,\gamma_W,\gamma_{Wt},\gamma_{\alpha^W},\gamma_Z,\gamma_{Zt},\gamma_{\alpha^Z},\gamma_\omega,\gamma_y,\gamma_{bu},\gamma_\alpha,\gamma_{but},\gamma_C,\gamma_{Ct},$
   $\gamma_{bi},\gamma_{bit},\gamma_Y\}$
4: {
5: //Creating matrices P, $\omega$, W, and Z and temporary matrices $P^S$, $\omega^S$, $W^S$, and $Z^S$:
6: Matrix P, $P^S$; Matrix $\omega^S$; Matrix $W^S$; Matrix $Z^S$;
7: //Creating vectors $\alpha^P$, $\alpha^W$, and $\alpha^W$, and temporary vectors $\beta^P$, $\beta^W$, and $\beta^W$:
8: Vector $\alpha^P$, $\beta^P$; Vector $\alpha^W$, $\beta^W$; Vector $\alpha^Z$, $\beta^Z$;
9: //Creating tables Pt, Wt, and Zt, and temporary tables $Pt^S$, $Wt^S$, and $Zt^S$:
10: Table Pt, $Pt^S$; Table Wt, $Wt^S$; Table Zt, $Zt^S$;
11: ModelInitialiser();
12: $l \leftarrow 1$;
13: for $l \preceq maxIter$ do
14:     IntrinsicTrainer();
15:     SocialTrainer();
16:     ModelUpdater();
17:     error ← error × 0.5;
18:     $l \leftarrow l+1$;
19: }
```

[a] λ is the set of the model hyper-parameters as specified in Eqs. (17) and (18) and Fig. 1. N, M, and D respectively denote number of users, number of items, and number of features. γ denotes the set of learning rates, $maxIter$ denotes the maximum number of learning iterations.

Algorithm 2 Model initialising.

```
1: void ModelInitialiser($\lambda$, $\gamma$)
2: {
3: $initMean \leftarrow 0$; $initStd \leftarrow 1$;
4: $P.init(initMean, initStd)$; $\alpha^P.initConst(0)$; $Pt.initConst(0)$;
5: $P^S.init(initMean, initStd)$; $\beta^P.initConst(0)$; $Pt^S.initConst(0)$; $W$
6: $W.initConst(0)$; $\alpha^W.initConst(0)$; $Wt.initConst(0)$;
7: $W^S.init(initMean, initStd)$; $\beta^W.initConst(0)$; $Wt^S.initConst(0)$;
8: $Z.initConst(0)$; $\alpha^Z.initConst(0)$; $Zt.initConst(0)$;
9: $Z^S.init(initMean, initStd)$; $\beta^Z.initConst(0)$; $Zt^S.initConst(0)$;
10: $\omega.init(initMean, initStd)$; $\omega^S.init(initMean, initStd)$;
11: $bu.init(initMean, initStd)$; $\alpha.init(0)$; $but.init(0)$; $C.init(0)$; $Ct.init(0)$;
12: $bi.init(initMean, initStd)$; $\beta.initConst(0)$; $bit.initConst(0)$;
13: $Q.init(initMean, initStd)$; $y.init(initMean, initStd)$;[b]
14: }
```

[b] *initMean* and *initStd* are the mean and standard deviation values that are used to initialise the model parameters. init(initMean, initStd) is a function that initialises a bias vector (e.g. *bu* and *bi*) and a matrix (e.g. *P*, and *Q*) using Gaussian distribution with mean value of *initMean* and standard deviation of *initStd*. initConst(initMean, initStd) initialises a matrix (e.g. *W* and *Z*) with a constant value.

As can be seen in line 11 in Algorithm 1, the training starts with initialising the model parameters. The matrices *P, Q, y*, and ω and user and item bias vectors (*bu* and *bi*) are randomly initialised using a Gaussian distribution with a mean of zero and the standard deviation of one. The new matrices *Pt, W, Wt, Z, Zt, Ct, but, bit*, and *Y* and the vectors α, α^P, α^W, α^Z, *C* are initialised with constant values. By using constant values to initialise the matrices and vectors, the algorithm starts the search process at the same starting point as CTFVSVD, and explores the modified search space to find more promising solutions, by considering the possible conditional dependencies between the features and the differences between users in preferring item feature values, as well as dynamic properties of the preferences, and the influence of social friends in the preferences of a user.

The main algorithm consists of a main loop, which implements the learning iterations of the model. Each iteration is comprised of one model intrinsic training operation (Algorithm 3), one model social training operation (Algorithm 4), and one model updating operation (Algorithm 5). In the model intrinsic trainer, the model parameters are updated using the gradient values in Eqs. (A.1)–(A.41), using a rating value that is read from the user-item ratings matrix. First in line 8, the estimated rating is calculated according to Eq. (8). Then the basic parameters of the model, *P, Q, W, Z, Y, bu*, and *bi*, and the temporal parameters *but, bit*, α, *C, Ct*, α^P, α^W, α^Z, *Pt, Wt*, and *Zt* are updated using the rating-related gradient values ($\frac{\partial E_R}{\partial (.)}$) in the Eqs. (A.1)–(A.41). Since this trainer only learns the intrinsic user preferences, only the error value in Eq. (17) will be used to update the model parameters. After learning the intrinsic preferences, the function in Algorithm 4 is invoked to train the social aspects of the preferences. Similar to IntrinsicTrainer, SocialTrainer is also comprised of a main loop, which iterates over the social relationship data in the social matrix. In each iteration, one entry from the social matrix is read, and the socially-influenced parameters of the model are updated though the gradient values that are obtained using the error in Eq. (18). Finally, the ModelUpdater in Algorithm 5 is invoked, and the calculated model updates are applied to the model parameters. This process is repeated for a fixed number of iterations, or until a specific condition is met. At the end of this process, the model parameters (*P, Pt*, α^P, *Q, W, Wt*, α^W, *Z, Zt*, α^Z, *Y*, ω, *y, bu*, α, *but, C, Ct, bi, bit*) are trained using the input data, and can be used to estimate the rating value given by a user *u* to an item *j* according to Eq. (8).

Algorithm 3 Intrinsic training.

```
1: void IntrinsicTrainer($\lambda$, $\gamma$)
2: {
3: $u \leftarrow 1$;
4: for $u \preceq N$ do
5:     $j \leftarrow 1$;
6:     for $j \preceq M$ do
7:         if $R_{uj} \neq 0$ then
8:             Calculate $\hat{R}_{uj}$ according to Eq. 8.
9:             Get the time $t$ that the rating $R_{uj}$ has been given.
10:            Update $bu_u$, $but_{ut}$, and $\alpha_u$ according to Eqs. A.1–A.3 using $\gamma_\alpha$, $\gamma_{bu}$, $\gamma_{but}$;
11:            Update $bi_j$ and $bit_{jt}$ according to Eqs. A.4–A.5 using $\gamma_{bi}$ and $\gamma_{bit}$;
12:            Update $C_u$ and $Ct_{ut}$ according to Eqs. A.6–A.7 using $\gamma_C$ and $\gamma_{Ct}$;
13:            $f \leftarrow 1$;
14:            for $f \preceq D$ do
15:                Update $P^S_{uf}$, $Pt^S_{uft}$, and $\beta^P_u$ according to Eqs. A.9, A.12, and A.15 using $\gamma_P$, $\gamma_{Pt}$, and $\gamma_{\alpha^P}$;
16:                Update $Q_{jf}$ according to Eq. A. 40 using $\gamma_Q$;
17:                Update $W^S_{uf}$, $Wt^S_{uft}$, and $\beta^W_u$ according to Eqs. A.18, A.21, and A.24 using $\gamma_W$, $\gamma_{Wt}$, and $\gamma_{\alpha^W}$;
18:                Update $Z^S_{uf}$, $Zt^S_{uft}$, and $\beta^Z_u$ according to Eqs. A.27, A.30, and A.33 using $\gamma_Z$, $\gamma_{Zt}$, and $\gamma_{\alpha^Z}$;
19:                $\forall v \in T_u$: Update $\omega^S_{vf}$ according to Eq. A.35 using $\gamma_\omega$;
20:                $\forall i \in I_u$: Update $y_{if}$ according to Eq. A.33 using $\gamma_y$;
21:                $f' \leftarrow f + 1$;
22:                for $f' \preceq D$ do
23:                    Update $Y_{ff'}$ and $Y_{f'f}$ according to Eq. A.39 using $\gamma_Y$;
24:                    $f' \leftarrow f' + 1$;
25:                $f \leftarrow f + 1$;
26:        $j \leftarrow j + 1$;
27:    $u \leftarrow u + 1$;
28: }
```

Algorithm 4 Social training.

```
1: void SocialTrainer($\lambda$, $\gamma$)
2: {
3: $u \leftarrow 1$;
4: for $u \preceq N$ do
5:     $v \leftarrow 1$;
6:     for $v \preceq N$ do
7:         if $v \in T_u$ then
8:             for $f \preceq D$ do
9:                 Update $P^S_{uf}$, $W^S_{uf}$, and $Z^S_{uf}$ according to Eqs. A.10, A.19, A.28 using $\gamma_P$, $\gamma_W$, and $\gamma_Z$;
10:                $\forall t \in I^t_u$ : Update $Pt^S_{uft}$, $Wt^S_{uft}$, and $Zt^S_{uft}$ according to Eqs. A.13, A.16, A.19 using $\gamma_{Pt}$, $\gamma_{Wt}$, and $\gamma_{Zt}$;
11:                Update $\beta^P_{uf}$, $\beta^W_{uf}$, and $\beta^Z_{uf}$ according to Eqs. A.16, A.19, A.22 using $\gamma_{\alpha^P}$, $\gamma_{\alpha^W}$, and $\gamma_{\alpha^Z}$;
12:                $\forall t \in I^t_u$ : Update $\omega^t_{vf}$ according to Eq. A.26 using $\gamma_\omega$;
13:                $f \leftarrow f + 1$;
14:        $v \leftarrow v + 1$;
15:    $u \leftarrow u + 1$;
16: }
```

Algorithm 5 Model updating.

```
1: void ModelUpdater($\lambda$, $\gamma$)
2: {
3: $\forall u, f : P_{uf} \leftarrow -\gamma_U \times P^S_{uf}$;
4: $\forall u : \alpha^P_u \leftarrow -\gamma_{\alpha^P} \times \beta^P_u$;
5: $\forall u, f : W_{uf} \leftarrow -\gamma_W \times W^S_{uf}$;
6: $\forall u : \alpha^W_u \leftarrow -\gamma_{\alpha^W} \times \beta^W_u$;
7: $\forall u, f : Z_{uf} \leftarrow -\gamma_Z \times Z^S_{uf}$;
8: $\forall u : \alpha^Z_u \leftarrow -\gamma_{\alpha^Z} \times \beta^Z_u$;
9: $\forall u, f : \omega_{uf} \leftarrow -\gamma_\omega \times \omega^S_{uf}$;
10: }
```

3.2.5. Computational complexity analysis

The model training in Algorithm 1 is comprised of one main loop that iterates for a fixed number of iterations (maxIter). Therefore, the computation time of the model trainer is expressed in Eq. (20).

$$C(ModelTrainer) = C(IntrinsicTrainer) + C(SocialTrainer) + C(ModelUpdater) \quad (20)$$

First, we examine the computational complexity of Intrinsic Training in Algorithm 3. On the highest level, this algorithm is comprised of two loops that iterate over the non-zero ratings in the rating matrix R. In the following, $|R|$ and $|T|$ denote the number of non-zero entries in the rating matrix R and adjacency matrix T respectively. In Intrinsic Trainer:

- The number of repetitions to calculate the estimated ratings ($\hat{R}$) in line 8 is $(D^2 \times |R|) + (D \times \sum_{u=1}^{N} |I_u|^2) + (D \times \sum_{u=1}^{N} |I_u| \times |T_u|)$.
- The number of repetitions to update parameters related to user and item biases in lines 10, 11, and 12 is $7 \times |R|$.
- The number of repetitions needed to update the parameters P, Q, W, and Z in lines 15, 16, 17, and 18 is $10 \times D \times |R|$.
- The number of repetitions needed to update the parameters ω in line 19 is $D \times \sum_{u=1}^{N}(|I_u| \times |T_u|)$.
- The number of repetitions needed to update the parameters y in line 20 is $D \times \sum_{u=1}^{N} |I_u|^2$.
- The number of repetitions needed to update the dependency matrix Y in line 23 is $D^2 \times |R|$.

Therefore, the overall number of repetitions for the Intrinsic Trainer is obtained according to Eq. (21).

$$\begin{aligned} N(IntrinsicTrainer) &= D^2 \times |R| + D \times \sum_{u=1}^{N} |I_u| \times |T_u| \\ &+ 7 \times |R| + 10 \times D \times |R| \\ &+ D \times \sum_{u=1}^{N}(|I_u| \times |T_u|) + D \times \sum_{u=1}^{N} |I_u|^2 \\ &+ D^2 \times |R| \end{aligned} \quad (21)$$

Assuming that on average, each user rates c items, and trusts k users, the computation time can be obtained as Eq. (22).

$$C(IntrinsicTrainer) = O(D^2 \times |R|) + O(D \times c \times |R|) + O(D \times k \times |T|) \quad (22)$$

Assuming that c, $k \lll N$, we can ignore the values of c and k. Therefore, the computational time of the Intrinsic Trainer would be obtained according to Eq. (23).

$$\begin{aligned} C(IntrinsicTrainer) &= O(D^2 \times |R|) + O(D \times |R|) + O(D \times |T|) \\ &= O(D^2 \times |R|) + O(D \times |T|) \end{aligned} \quad (23)$$

Consequently, the overall computation time is linear with respect to the number of observed ratings as well as observed trust statements. Social Trainer consists of two loops that iterate over the non-zero trust relations in the adjacency matrix T. The number of repetitions needed to update the parameters P, W, Z, and β^P, β^W, and β^Z is $6 \times D \times |T|$. The number of repetitions to update the values of Pt, Wt, Zt, and ω is equal to $4 \times (\sum_{u=1}^{N} |I_u| \times |T_u| \times D)$. Therefore, the computation time of Social Trainer is equal to:

$$C(IntrinsicTrainer) = O(D \times |R|) + O(D \times |T|) \quad (24)$$

In the Model Updater, the values of matrices P, W, Z, and vectors ω, α^P, α^W, and α^Z need to be updated. The computation time needed to update these parameters is $O(N \times D)$. Assuming that each user has rated at least one item, it is safe to say that $|R|$ is greater than the number of users N. Therefore, the computation time of Model Updater does not exceed the maximum computation time of Intrinsic Trainer and Social Trainer. Finally, the computation time of the Model trainer is obtained as Eq. (25).

$$C(ModelTrainer) = O(D^2 \times |R|) + O(D \times |T|) \quad (25)$$

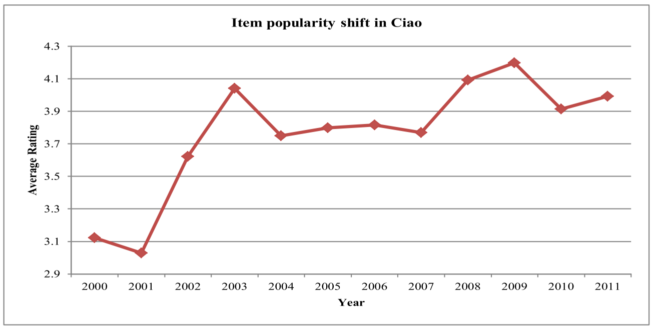
The number of latent factors D is fixed, hence the computation time is only a function of $|R|$ and $|T|$. Since both ratings matrix and social network matrix are sparse, the algorithm is scalable to the problems with millions of users and items.

4. Experiments

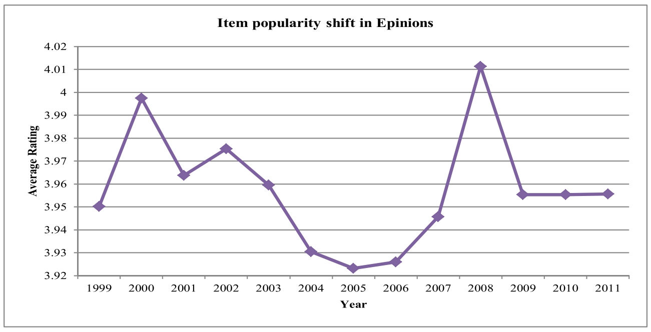
4.1. Datasets

We tested Aspect-MF on three popular datasets, Ciao, Epinions, and Flixster. Ciao is a dataset crawled from the ciao.co.uk website. This dataset includes 35,835 ratings given by 2248 users over 16,861 movies. Ciao also includes the trust relationships between users. The number of trust relationships in Ciao is 57,544. Therefore the dataset density of ratings and trust relationships are 0.09% and 1.14% respectively. The ratings are integer values between 1 and 6. The Epinions dataset consists of 664,824 ratings from 40,163 users on 139,738 items of different types (software, music, television show, hardware, office appliances, ...). Ratings are integer values between 1 and 5, and data density is 0.011%. Epinions also enables the users to issue explicit trust statements about other users. This dataset includes 487,183 trust ratings. The density of the trust network is 0.03%. Flixster is a social movie site which allows users to rate movies and share the ratings with each other, and become friends with others with similar movie taste. The Flixster dataset which is collected from the Flixster website includes 8,196,077 ratings issued by 147,612 users on 48,794 movies. The social network also includes 7,058,819 friendship links. The density of the ratings matrix and social network matrix are 0.11% and 0.001% respectively. The item popularity shift depicted for the Epinions, Ciao, and Flixster datasets in Fig. 5 shows that the ratings drift over time. In particular, it can be observed that an abrupt shift of items rating scale has happened at year 2005, and 2006 for Epinions and Flixster datasets respectively. We can also see that over time, generally items have grown in popularity in the Ciao dataset.

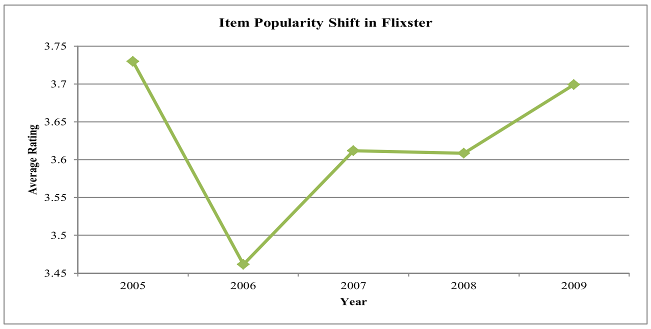
In all the experiments in Sections 4.3–4.5, 80% of the datasets are used for training and the remaining 20% are used for evaluation. In order to achieve statistical significance, each model training is repeated for 30 times and the average values are used. In Section 4.6, we analyse the behaviour of the models in other cases, where 60% and 40% of the ratings are used for training.



(a)



(b)



(c)

Fig. 5. The drift of average item ratings in the (a) Ciao, (b) Epinions, and (c) Flixster datasets.

4.2. Comparisons

In order to show the effectiveness of Aspect-MF, we compared the results against the recommendation quality of some of the most popular state of the art models that have reported the highest accuracies in the literature. The following models are compared across the experiments in this section:

- *TrustSVD* (Guo, Zhang, & Yorke-Smith, 2015), which builds on SVD++ (Koren & Bell, 2011). The missing ratings are calculated based on explicit and implicit feedback from user ratings and user's trust relations.
- *CondTrustFVSVD* (Zafari & Moser, 2017), this method extends TrustSVD by adding the conditional preferences over feature values to TrustSVD. Experimental results show that this method is significantly superior to TrustSVD in terms of accuracy. This model is denoted CTFVSVD in the experiments section.
- *Aspect-MF*, which is the model proposed in this paper. The component-based approach that we took in designing this model enabled us to arbitrarily switch on/off the dynamicity over different preference aspects. Therefore, in the experiments we try all the combinations of dynamic preference aspects. This results in 7 combinations denoted by *b*, *bf*, *bffv*, *bfv*, *f*, *ffv*, and *fv*[1]

Guo, Zhang, and Yorke-Smith (2016) carried out comprehensive experiments, and showed that their model, TrustSVD outperformed all the state of the art models. Recently, Zafari and Moser (2017) showed that their model CondTrustFVSVD significantly outperforms TrustSVD. Therefore, in this section, we limited our comparisons to these two models from the state of the art since they outperform a comprehensive set of state of the art recommendation models (Guo et al., 2016; Zafari & Moser, 2017).

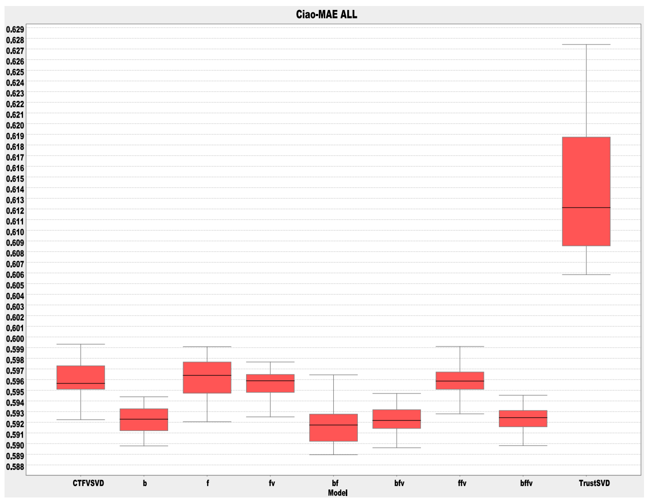
The optimal experimental settings for each method are determined either by our experiments or suggested by previous works (Guo et al., 2015; 2016; Zafari, Moser, & Rahmani, 2017). Since the model was designed using a component-based approach, we could switch off an aspect easily by setting the hyper-parameters and learning rates to zero. To find the appropriate values for each aspect, we performed grid search. We first set the values to zero and recorded the accuracy. Then we increased the values and monitored the accuracy. The accuracy kept improving before it dropped. After finding a set of sub-optimal values by this trial and error approach, we used the same values through our experiments for TCT-FVSVD.

Due to the over-fitting problem, the accuracy of iterative models improves for a number of iterations, after which it starts to degrade. Therefore, we recorded the best accuracy values achieved by each model during the iterations, and compared the models based on the recorded values. We believe that this approach results in a fairer comparison of the models than setting the number of iterations to a fixed value, because the models over-fit at different iterations, and using a fixed number of iterations actually prevents us from fairly comparing the models based on their real capacity in uncovering hidden patterns from data. Therefore, the reported results for iterative models here are the best results that they could achieve using the aforementioned parameters. MAE and RMSE measures are used to evaluate and compare the accuracy of the models. MAE and RMSE are two standard and popular measures that are used to measure and compare the performance of preference modelling methods in recommender systems. In the following sections, we consider the performances separately for All Users and Cold-start Users. Cold-start Users are the users who have rated less than 5 items, and All Users include all the users regardless of the number of items they have rated.

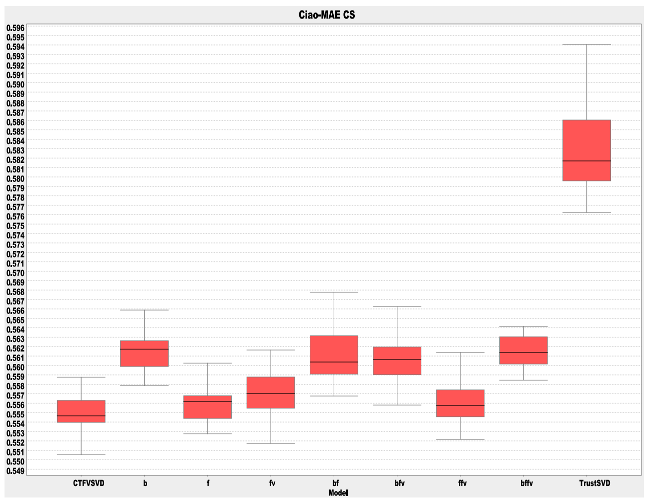
4.3. Discussion

All latent factor approaches have been evaluated with 5 factors, because no clear ideal value could be established. In Section 4.3.1, first we analyse the performance of the models from different perspectives. Since the results are subject to randomness, we also performed a *t* test to guarantee that the out-performances achieved do not happen by chance. The results are discussed in Section 4.4. As we mentioned in Section 1, one of the research questions we are interested in, in this paper is related to the interplay between the dynamicity of preference aspects and the preference domain. In Section 4.5, we consider the performance of combinations of Aspect-MF, in order to pinpoint the aspects that are more subject to temporal drift in each dataset. In Section 4.6, we also consider the effect of the amount of training data that is fed to the model as input, and analyse the robustness of the models to the shortage of training data.

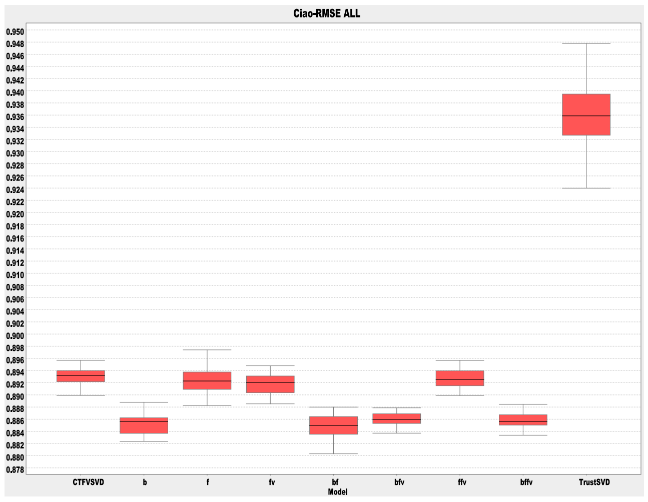
[1] *fv* denotes feature value preferences, *f* denotes feature preferences, and *b* denotes bias. Therefore, *bffv* denotes a model with all the three aspects.



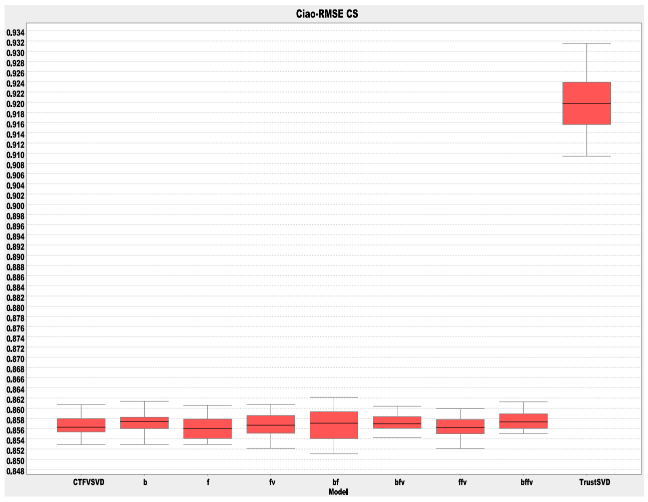
(a) MAE, all users



(b) MAE, cold-start users



(c) RMSE, all users



(d) RMSE, cold-start users

Fig. 6. Box plots of the Aspect-MF's combinations (b, bf, bffv, f, ffv, fv) and CTFVSVD versus TrustSVD in Ciao dataset in terms of MAE and RMSE measures for cold-start users (CS) and all users (ALL).

4.3.1. Model performances

We can consider the performance of the models from different perspectives. A preference model's performance can be considered with respect to the dataset on which it is trained, the accuracy measure that is used to evaluate the model's performance, and the performance of the model on cold-start users vs the performance on all users.

Datasets. The error values in Fig. 6 show that the Aspect-MF results in substantial improvements over TrustSVD in all three datasets for both measures and for all users and cold-start users. As we can see in this figure, the box plots of Aspect-MF's combinations do not have much overlap with the box plot of TrustSVD, which means that the differences are definitely statistically significant. In this figure, we can also see that the box plot widths for Aspect-MF's combinations are usually much smaller than that for TrustSVD. This suggests that Aspect-MF's combinations are more stable than TrustSVD, meaning that they find roughly the same solutions across different model executions. This is a favourable property of the model, since it makes the model performance less subject to randomness. Clearly, a model that performs well sometimes and worse at other times is less reliable. The model's superior performance is likely due to its taking multiple preference aspects into account, therefore, it has more clues as to where the optimal solutions might reside in the solution space.

In particular, we can see that the model is more stable in the case of the Ciao and Epinions datasets than the Flixster dataset. On the Epinions dataset, each typical user and cold-start user rates 41.61 items and 4.08 items on average. These numbers respectively are 15.94 and 2.94 for the Ciao dataset, and 11.12 and 1.94 for the Flixster dataset. This could explain why the variations are larger on Flixster dataset than Epinions and Ciao datasets. Since more ratings per user are available in the Ciao and Epinions dataset, different executions lead the model to more similar solutions than the solutions that are found on the Flixster dataset across different model executions. We can also see from Table 2, that on the Ciao and Flixster datasets, the improvements are more significant for RMSE, while more significant improvements are achieved for RMSE. We can also clearly observe that the model variations are smaller for all users in the Epinions dataset, and for cold-start users in the Flixster dataset.

Accuracy measures. As the statistical analysis of the models in Table 2 show, the differences are generally more significant when

Table 2
The t values and p values for Aspect-MF's combinations vs TrustSVD in Ciao, Epinions, and Flixster datasets for MAE and RMSE measures on all users (ALL) and cold-start users (CS).

Dynamic model	Measure	Ciao			Epinions			Flixster		
		t value	*p* value	Sig.	*t* value	*p* value	Sig.	*t* value	*p* value	Sig.
Aspect-MF(b)	MAE-ALL	−19.9867	9.44E−20	**yes**	−144.389	3.57E−49	**yes**	−18.3981	2.33E−21	**yes**
Aspect-MF(b)	RMSE-ALL	−48.7869	2.09E−33	**yes**	−138.903	6.05E−49	**yes**	−11.0414	8.14E−14	**yes**
Aspect-MF(b)	MAE-CS	−24.9813	9.60E−29	**yes**	−60.0446	2.75E−40	**yes**	−37.612	1.36E−34	**yes**
Aspect-MF(b)	RMSE-CS	−61.0847	1.94E−40	**yes**	−35.8673	7.73E−32	**yes**	−27.6887	9.62E−29	**yes**
Aspect-MF(bf)	MAE-ALL	−20.2987	1.75E−20	**yes**	−144.517	2.80E−49	**yes**	−17.6976	1.66E−21	**yes**
Aspect-MF(bf)	RMSE-ALL	−48.3137	8.60E−35	**yes**	−137.679	3.67E−50	**yes**	−11.3137	6.28E−14	**yes**
Aspect-MF(bf)	MAE-CS	−24.2062	5.78E−30	**yes**	−57.2661	2.06E−44	**yes**	−37.6646	2.67E−34	**yes**
Aspect-MF(bf)	RMSE-CS	−58.0125	2.90E−44	**yes**	−35.1151	7.69E−34	**yes**	−28.4271	2.65E−28	**yes**
Aspect-MF(bffv)	MAE-ALL	−20.1253	1.36E−19	**yes**	−144.792	1.23E−48	**yes**	−18.9756	1.21E−21	**yes**
Aspect-MF(bffv)	RMSE-ALL	−48.7184	2.40E−32	**yes**	−138.854	1.08E−48	**yes**	−11.9005	1.72E−14	**yes**
Aspect-MF(bffv)	MAE-CS	−26.7303	1.60E−26	**yes**	−57.8678	1.57E−42	**yes**	−37.5037	4.99E−35	**yes**
Aspect-MF(bffv)	RMSE-CS	−62.6826	2.88E−37	**yes**	−35.6445	1.44E−31	**yes**	−27.5707	1.45E−29	**yes**
Aspect-MF(bfv)	MAE-ALL	−20.0161	5.51E−20	**yes**	−140.994	5.74E−52	**yes**	−18.304	9.83E−23	**yes**
Aspect-MF(bfv)	RMSE-ALL	−48.7855	2.22E−32	**yes**	−135.65	1.02E−51	**yes**	−11.7557	1.50E−14	**yes**
Aspect-MF(bfv)	MAE-CS	−25.0275	4.47E−30	**yes**	−57.4134	9.13E−44	**yes**	−39.183	1.14E−31	**yes**
Aspect-MF(bfv)	RMSE-CS	−61.8785	4.54E−39	**yes**	−35.8765	9.44E−33	**yes**	−28.9199	9.04E−27	**yes**
Aspect-MF(f)	MAE-ALL	−16.021	1.28E−17	**yes**	−126.805	1.94E−50	**yes**	−15.2094	8.52E−20	**yes**
Aspect-MF(f)	RMSE-ALL	−40.3613	2.93E−33	**yes**	−120.674	2.03E−50	**yes**	−9.14701	2.32E−11	**yes**
Aspect-MF(f)	MAE-CS	−31.3473	9.40E−33	**yes**	−59.4225	5.52E−42	**yes**	−34.6759	2.87E−36	**yes**
Aspect-MF(f)	RMSE-CS	−62.008	1.76E−40	**yes**	−36.5189	3.76E−31	**yes**	−27.0282	2.79E−28	**yes**
Aspect-MF(ffv)	MAE-ALL	−16.4344	1.37E−17	**yes**	−131.061	1.17E−46	**yes**	−15.2416	1.30E−18	**yes**
Aspect-MF(ffv)	RMSE-ALL	−41.942	1.31E−30	**yes**	−124.216	4.87E−47	**yes**	−9.28969	4.00E−11	**yes**
Aspect-MF(ffv)	MAE-CS	−30.1691	9.47E−34	**yes**	−60.2686	2.61E−40	**yes**	−36.8921	3.11E−34	**yes**
Aspect-MF(ffv)	RMSE-CS	−60.9112	2.12E−41	**yes**	−34.5646	1.19E−32	**yes**	−27.7828	1.94E−28	**yes**
Aspect-MF(fv)	MAE-ALL	−16.8998	1.42E−17	**yes**	−127.613	1.55E−49	**yes**	−16.0515	2.37E−19	**yes**
Aspect-MF(fv)	RMSE-ALL	−42.4293	1.93E−31	**yes**	−121.779	2.10E−49	**yes**	−9.22016	4.64E−11	**yes**
Aspect-MF(fv)	MAE-CS	−30.0768	2.88E−32	**yes**	−57.1987	4.54E−43	**yes**	−38.164	6.52E−33	**yes**
Aspect-MF(fv)	RMSE-CS	−61.5278	2.68E−40	**yes**	−33.735	7.11E−33	**yes**	−27.2901	2.64E−28	**yes**

the accuracies are measured in terms of the RMSE. This can be explained by the formulation of these models as an optimisation problem. These models focus on maximising accuracy using RMSE and achieving better MAE values is a secondary goal that is only pursued through minimising RMSE.

Cold-start vs all users. By taking a close look at the statistical analysis results in Table 2 and also the box plots of CTFVSVD vs Aspect-MF's combinations in Fig. 6, we can see that in all three datasets, the improvements of the Aspect-MF are more significant over all users than cold-start users. This can be explained by the amount of dynamic information that the models receive for each one of these groups of users. For all users, the model is trained using all ratings and also all associated time stamps for those ratings. Therefore the model can more successfully discern the temporal patterns in the preferences, and the accuracy improvements are larger. However, for the cold-start users, the model does not have access to much temporal information about these users, since they do not have many ratings. As a result, the model cannot identify the shift in the preferences of these users, and the improvements are smaller. From this, we conclude that temporal models are more successful on all users, because for them, temporal information is available.

4.4. Statistical analysis

The statistical analysis of the performances provided in Table 2 shows that all Aspect-MF's combinations achieve significantly better results than TrustSVD, which does not include the temporal information. The values in Table 3 also show that Aspect-MF's combinations also result in improvements over CTFVSVD that are statistically significant, which means that in all three datasets, Aspect-MF has been successful in extracting the temporal patterns in the users' preferences. We can also see that the all the p values in Table 2 are 0.0000, which means that with almost 100% probability, the two model executions (Aspect-MF and TrustSVD) do not come from distributions with equal mean performances. Therefore, we are almost 100% sure that the observed differences in performance are due to the superiority of Aspect-MF over TrustSVD, and not the result of chance. Similarly, the p values in Table 3 are almost zero, which means that we are certain that Aspect-MF is better than CTFVSVD, in cases where the *t* test shows a statistically significant improvement.

4.5. Dynamic aspects

The close comparison of the error values achieved by Aspect-MF in Figs. 6 and 7 show that in terms of MAE for all users, Aspect-MF achieves the best performance on the Ciao and Epinions datasets, for the models including dynamic *b* and *f* aspects. However, on the Flixster dataset, the model combination with dynamic *b* and *fv* aspects performs best. Interestingly, for cold-start users, different models perform the best. In particular, on the Ciao dataset, the model including dynamic *f* performs best, whereas on the Epinions and Flixster datasets, the model including dynamic *b*, *f*, and *fv* aspects, and the model with drifting *f* aspect achieve the best results respectively.

As shown in Fig. 1, the social aspect does not directly help capture the temporal drifts, but interacts with the other aspects that are subject to social influence, such as feature preferences and feature value preferences. Figs. 6 through 8 show that addition of time aspect to CTFVSVD significantly improves the accuracy. This is because the feature preferences and feature value preferences are subject to change over time, and capturing the temporal properties of these aspects helps improve the recommendation quality. Modelling the social aspect is also critical, since it helps better model feature value preferences and feature preferences. In fact, the improvements achieved by CTFVSVD (Zafari & Moser, 2017) over TrustSVD (Guo et al., 2016) are the result of modelling feature value preferences and feature preferences, and their interplay with social aspect, and the improvements achieved over CTFVSVD by

Table 3
The *t* values and *p* values for Aspect-MF's combinations vs CTFVSVD in Ciao, Epinions, and Flixster datasets for MAE and RMSE measures on all users (ALL) and cold-start users (CS).

Dynamic model	Measure	Ciao			Epinions			Flixster		
		t value	*p* value	Sig.	*t* value	*p* value	Sig.	*t* value	*p* value	Sig.
Aspect-MF(b)	MAE-ALL	−7.9254	0.0000	**yes**	−40.0588	0.0000	**yes**	−3.51234	8.76E−04	**yes**
Aspect-MF(b)	RMSE-ALL	−17.5792	0.0000	**yes**	−34.3869	0.0000	**yes**	−3.76619	3.90E−04	**yes**
Aspect-MF(b)	MAE-CS	8.9344	0.0000	**yes**	0.8529	0.3973	no	−0.85677	0.39517	no
Aspect-MF(b)	RMSE-CS	0.5979	0.5522	no	1.4063	0.1650	no	−1.70069	0.094359	no
Aspect-MF(bf)	MAE-ALL	−8.6722	0.0000	**yes**	−40.4729	0.0000	**yes**	−2.88722	0.005453	**yes**
Aspect-MF(bf)	RMSE-ALL	−16.9178	0.0000	**yes**	−32.7924	0.0000	**yes**	−4.13771	1.16E−04	**yes**
Aspect-MF(bf)	MAE-CS	7.6174	0.0000	**yes**	0.0021	0.9983	no	−0.65626	0.514293	no
Aspect-MF(bf)	RMSE-CS	0.4274	0.6709	no	0.4595	0.6476	no	−2.23892	0.029057	**yes**
Aspect-MF(bffv)	MAE-ALL	−8.2488	0.0000	**yes**	−40.8591	0.0000	**yes**	−4.31919	6.39E−05	**yes**
Aspect-MF(bffv)	RMSE-ALL	−17.2079	0.0000	**yes**	−34.0360	0.0000	**yes**	−5.11012	3.90E−06	**yes**
Aspect-MF(bffv)	MAE-CS	10.1601	0.0000	**yes**	1.2975	0.1996	no	−1.11763	0.26848	no
Aspect-MF(bffv)	RMSE-CS	2.0086	0.0495	**yes**	1.9330	0.0582	no	−2.18501	0.032959	**yes**
Aspect-MF(bfv)	MAE-ALL	−8.0096	0.0000	**yes**	−33.8218	0.0000	**yes**	−4.19593	9.50E−05	**yes**
Aspect-MF(bfv)	RMSE-ALL	−17.3742	0.0000	**yes**	−30.0663	0.0000	**yes**	−4.92428	7.44E−06	**yes**
Aspect-MF(bfv)	MAE-CS	7.3472	0.0000	**yes**	0.5237	0.6025	no	−0.31262	0.755723	no
Aspect-MF(bfv)	RMSE-CS	1.0133	0.3151	no	0.5666	0.5732	no	−1.76305	0.083675	no
Aspect-MF(f)	MAE-ALL	0.6250	0.5345	no	0.6578	0.5139	no	0.169793	0.865773	no
Aspect-MF(f)	RMSE-ALL	−1.1529	0.2539	no	1.7569	0.0846	no	−0.53474	0.594882	no
Aspect-MF(f)	MAE-CS	1.0076	0.3179	no	−0.0942	0.9253	no	0.644178	0.522319	no
Aspect-MF(f)	RMSE-CS	−0.8901	0.3771	no	1.2122	0.2306	no	−0.62142	0.536762	no
Aspect-MF(ffv)	MAE-ALL	0.1020	0.9191	no	−0.0859	0.9318	no	1.302566	0.197918	no
Aspect-MF(ffv)	RMSE-ALL	−1.1275	0.2643	no	1.7583	0.0840	no	−0.32867	0.743682	no
Aspect-MF(ffv)	MAE-CS	1.0049	0.3191	no	0.5245	0.6019	no	0.350768	0.727059	no
Aspect-MF(ffv)	RMSE-CS	−0.2707	0.7876	no	2.1764	0.0336	**yes**	−1.57954	0.119666	no
Aspect-MF(fv)	MAE-ALL	−0.5989	0.5520	no	0.9751	0.3343	no	0.086061	0.931718	no
Aspect-MF(fv)	RMSE-ALL	−2.9847	0.0042	**yes**	1.6778	0.0990	no	−0.22597	0.822071	no
Aspect-MF(fv)	MAE-CS	2.3939	0.0200	**yes**	1.5299	0.1315	no	−0.03512	0.972102	no
Aspect-MF(fv)	RMSE-CS	0.0763	0.9394	no	2.8533	0.0060	**yes**	−0.92614	0.358215	no

TCTFVSVD are the result of modelling the temporal properties of feature value preferences and feature preferences that were made subject to social aspect in CTFVSVD.

As we reviewed in Section 2, Guo, Zhu, Qu, and Wang (2018) recently proposed a temporal-based latent factor model, and compared it with some of the state of the art temporal-based models. They used both 80%-20% ratio train-test splitting and 5-fold cross validation, and empirically found that both approaches produced similar performances. The results reported by them show that TCTFVSVD beats BPTF and PCCF by a large margin. The MAE and RMSE values achieved by BPTF on the Ciao dataset are 0.76 and 1.05, while TCTFVSVD achieves 0.59 and 0.88 on the same dataset. This result is even better than PCCF which achieved 0.69 and 0.92 respectively. This is expected, since TCTFVSVD also includes social aspect which potentially includes a large fraction of preference patterns in data. Therefore, TCTFVSVD easily beats other methods including BPTF which do not include the social aspect. TCTFVSVD shows how all the preference aspects can be captured in a model and that is one main contribution of the current work. To the best of our knowledge, TCTFVSVD is the first to model all these aspects together.

From Figs. 6 through 8, we can make several conclusions. The first conclusion is that the dynamic patterns are dataset-dependent. Therefore, users and the items in different dataset can have preferences with aspects with different levels of dynamicity. This finding supports our component-based approach in modelling the dynamic properties of the preference aspects.

The second conclusion is that the prediction of the ratings for the cold-start users is less dependent on the drifting bias than that of all users. As we see in this Figs. 9 and 10, for all users, the combinations that include dynamic *b* aspects are strictly better than the other combinations, whilst this is less consistent for cold-start users, where sometimes the models with only dynamic *f* aspects perform best. This suggests that the preferences of cold-start users are not much affected by the shifts in the popularity of the items, while other users' preferences are more influenced by such shifts. Therefore, the accurate modelling of such temporal effects is of greater importance in the case of all users than cold-start users. As previous studies have shown (Koenigstein et al., 2011), bias is a very important aspect in human preferences. Since the cold-start users do not have enough ratings, there is also not enough temporal data to train the preferences for these models. Therefore, the trained temporal aspects of these users are probably not very accurate, and therefore, the combinations that include bias perform poorly on these users, due to imprecise predictions.

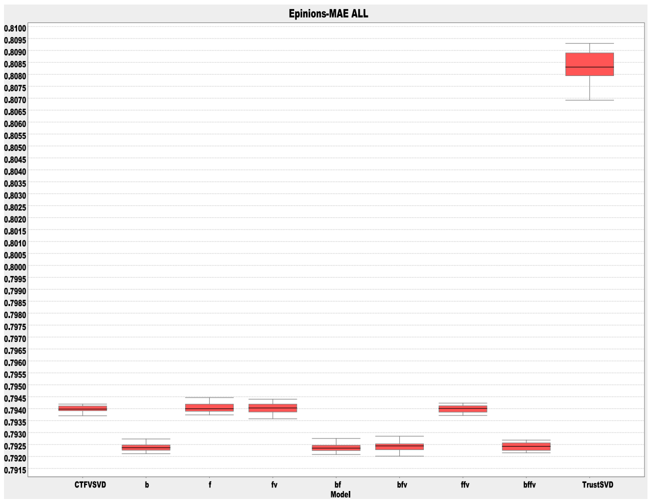
The third conclusion is that both measures reveal roughly the same preference patterns. This seems justifiable, since the shift in user preferences should naturally be independent of how the differences in estimated preferences and real preferences are measured.

To summarise, it is very advantageous to have a component-based model in which the temporal aspects of preferences can be arbitrarily captured in different conditions. This enables us to capture the patterns only when they are actually helpful, and consequently, build the most accurate preference models, tailored to different datasets and domains with disparate temporal patterns.

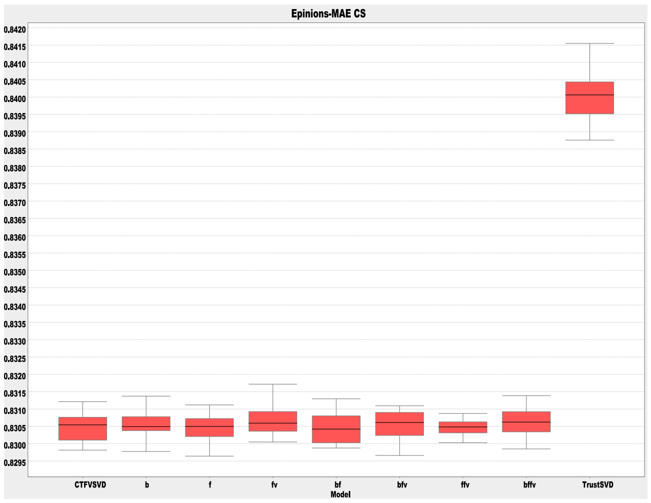
4.6. Effect of the size of the training dataset

The main purpose of this section is to evaluate the robustness of the models against shortage of training data. In the experiments in Sections 4.3 through 4.5, 80% of the ratings matrix was used for training the models and the remaining data was used for evaluation. The question that arises here is how the models would perform if less amount of data was fed to the models for training.

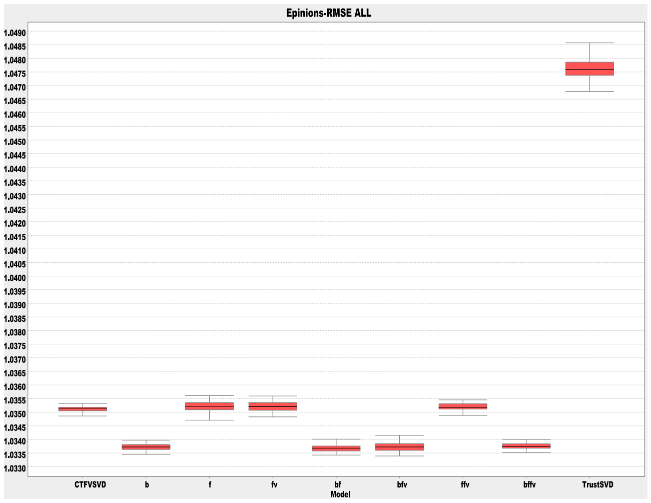
In order to analyse the behaviour of the models with respect to the amount of training data, we can reduce the amount of the training data, and consider how much the accuracy drops as the training data is decreased. Therefore, we also evaluate the models in two additional cases. The first case includes 60% of the data for



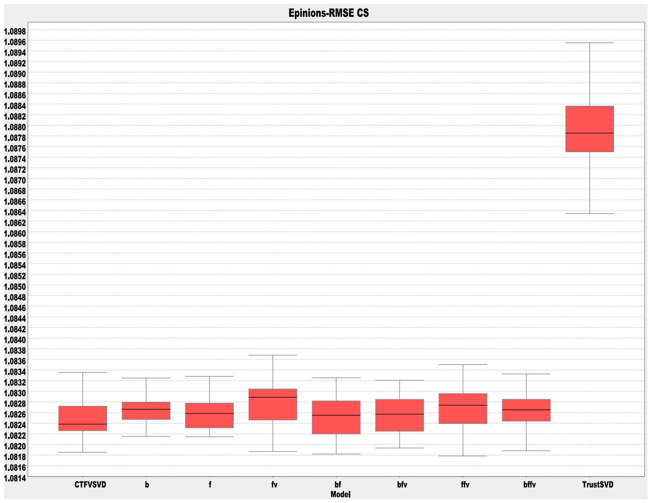
(a) MAE, all users



(b) MAE, cold-start users



(c) RMSE, all users



(d) RMSE, cold-start users

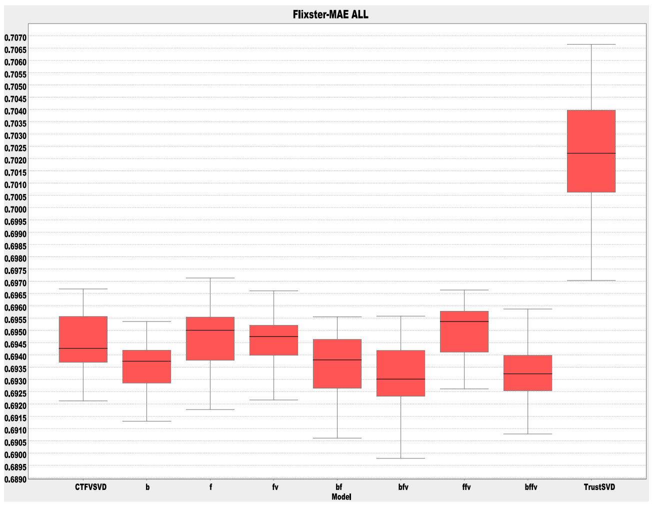
Fig. 7. Box plots of the Aspect-MF's combinations (b, bf, bffv, f, ffv, fv) and CTFVSVD versus TrustSVD in Epinions dataset in terms of MAE and RMSE measures for cold-start users (CS) and all users (ALL).

training, and the remaining 40% for testing, and the second case uses 40% of ratings data for training and the rest for evaluation. The results for the Flixster and Ciao datasets are demonstrated in Figs. 11 and 12 respectively. These figures show the percentage of error increase as the amount of training data is decreased.

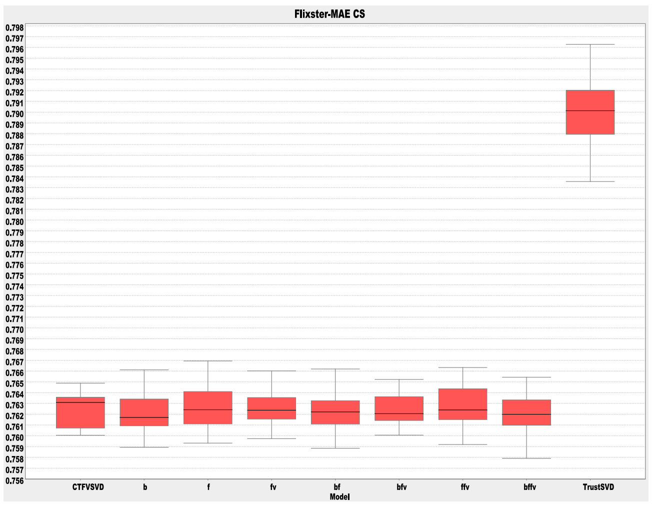
All users. As can be seen in Fig. 11, on the Flixster dataset, in the case of all users, all combinations of Aspect-MF result in a smaller increase in the error when the training data is decreased from 80% to 60% (denoted by 80-60 in these diagrams), and from 60% to 40% (denoted by 60–40 in these diagrams). Furthermore, we can observe that in terms of MAE, the combination that includes *f* and *fv* resulted the smallest error increase when the training data decreased from 80% to 60%, and the model that included *fv* resulted in the smallest error increase when the training data decreased from 60% to 40%. This suggests that the dynamic model is more robust to the shortage of training data, when the error is measured in terms of MAE for all users. In terms of RMSE, the least accuracy deterioration happened for the model combination with the *f* aspect, both when the training data amount drops to 60%, and when it drops to 40%.

Cold-start users. For cold-start users however, a different pattern is evident. Interestingly, we can see that for cold-start users, the error increases more when the training data is decreased from 80% to 60%, compared to when it is decreased from 60% to 40%. This means that the accuracy degrades more when the training data drops to 60%. Judging by the higher error increase for cold-start users in comparison with all users, cold-start users seem to be more sensitive to the decrease in the amount of training data. This seems understandable, since the cold-start users do not have many ratings. Therefore, when evaluating the model accuracy for cold-start users, less accurate predictions for each rating have a larger effect on the overall accuracy.

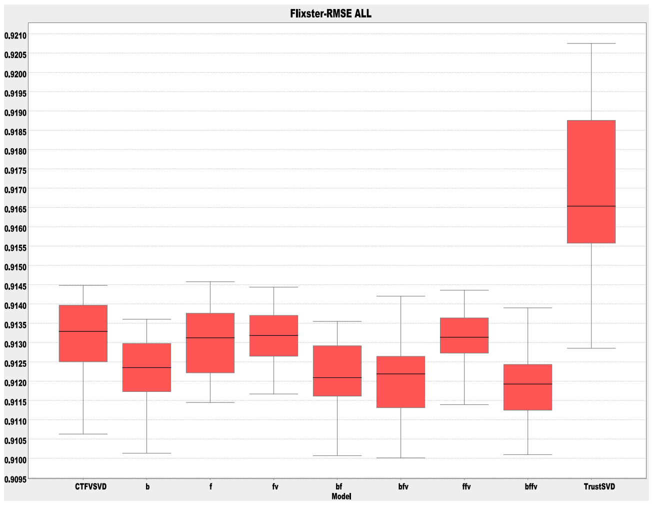
TrustSVD seems to be more robust to the shortage of training data for cold-start users, when the training data drops from 60% to 40%. This can be attributed to the fact that the dynamic model contains time information, and this information can be misleading if we substantially decrease the amount of training data, and evaluate the accuracy for cold-start users who do not have much ratings. A similar observation was made in Figs. 9 and 10, where the dynamic model including the *b* aspect performed poorly on the cold-start users.



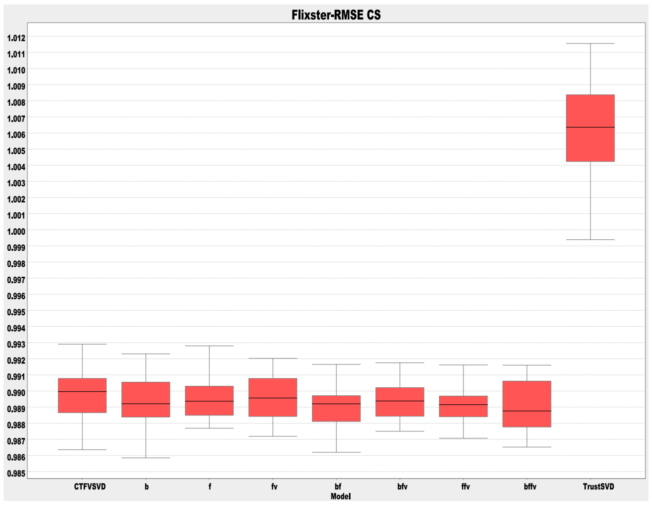
(a) MAE, all users



(b) MAE, cold-start users



(c) RMSE, all users



(d) RMSE, cold-start users

Fig. 8. Box plots of the Aspect-MF's combinations (b, bf, bffv, f, ffv, fv) and CTFVSVD versus TrustSVD in Flixster dataset in terms of MAE and RMSE measures for cold-start users (CS) and all users (ALL).

All users vs cold-start users. A similar trend to the one observed in Flixster dataset can also be seen in the Ciao dataset in Fig. 12. As this figure shows, the accuracy deterioration for cold-start users is much larger compared with that for all users. Again, we attribute this to the high sensitivity of cold-start users to inaccurate predictions. For the case where the training data amount drops from 80% to 60%, the model combination with all the dynamic aspects (*bffv*) results in the lowest increase in MAE for all users. For cold-start users, the model combination with *b* and *f* aspects achieve the smallest deterioration of accuracy.

However, in terms of RMSE for all users, TrustSVD incurs the lowest increase in the error, while for cold-start users, the model with the dynamic *fv* aspect is the most robust. In the second case where the training data amount is decreased from 60% to 40%, at least one of the model combinations performs best (incurs the lowest accuracy deterioration) for each measure, among the models tested. We can also see that when the training data amount is decreased from 80% to 60%, the error increase is much lower than when the training data amount drops from 60% to 40%. This means that the models are still quite robust with 60% of the ratings data as training data, but their accuracy considerably drops when the training data decreases to 40%.

Flixster vs Ciao. One of the key differences between the behaviour of the models on the Flixster and Ciao datasets, as can be seen in Figs. 11 and 12, is the threshold at which the accuracy sharply drops for cold-start users. For the Flixster dataset, the accuracy of cold-start users sharply worsens when the training data amount is decreased from 80% to 60%, while for the Ciao dataset, the sharp decrease in accuracy happens when the training data amount decreases from 60% to 40%. This can be easily justified by looking at the statistics of these two datasets for cold-start users. On the Flixster dataset as we mentioned before, each cold-start user rates 1.94 items on average, while this number is 2.94 in the Ciao dataset. Therefore, the accuracy of cold-start users on the Flixster dataset is more sensitive to inaccurate predictions than that on the Ciao dataset.

Considering all four measures on the two datasets, in general, we can observe that Aspect-MF's combinations are more robust to the decrease in the amount of training information than TrustSVD and CTFVSVD. The combinations in this paper are particularly more helpful in cases where enough time related data is fed into the model as input.

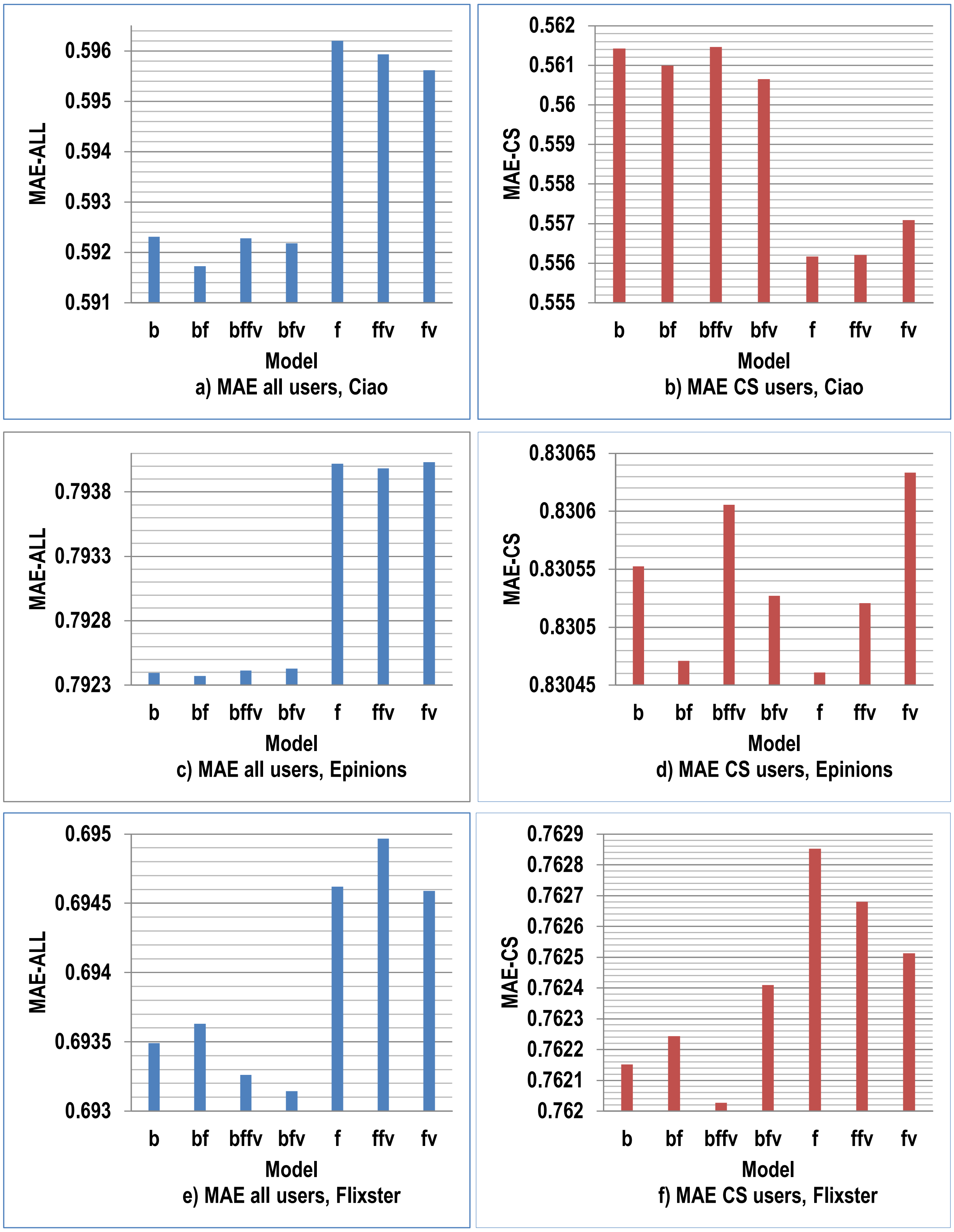


Fig. 9. Comparisons of the MAE values of Aspect-MF's combinations in (a,b) Ciao, (c,d) Epinions, and (e,f) Flixster datasets for all users (MAE-ALL) and cold-start users (MAE-CS).

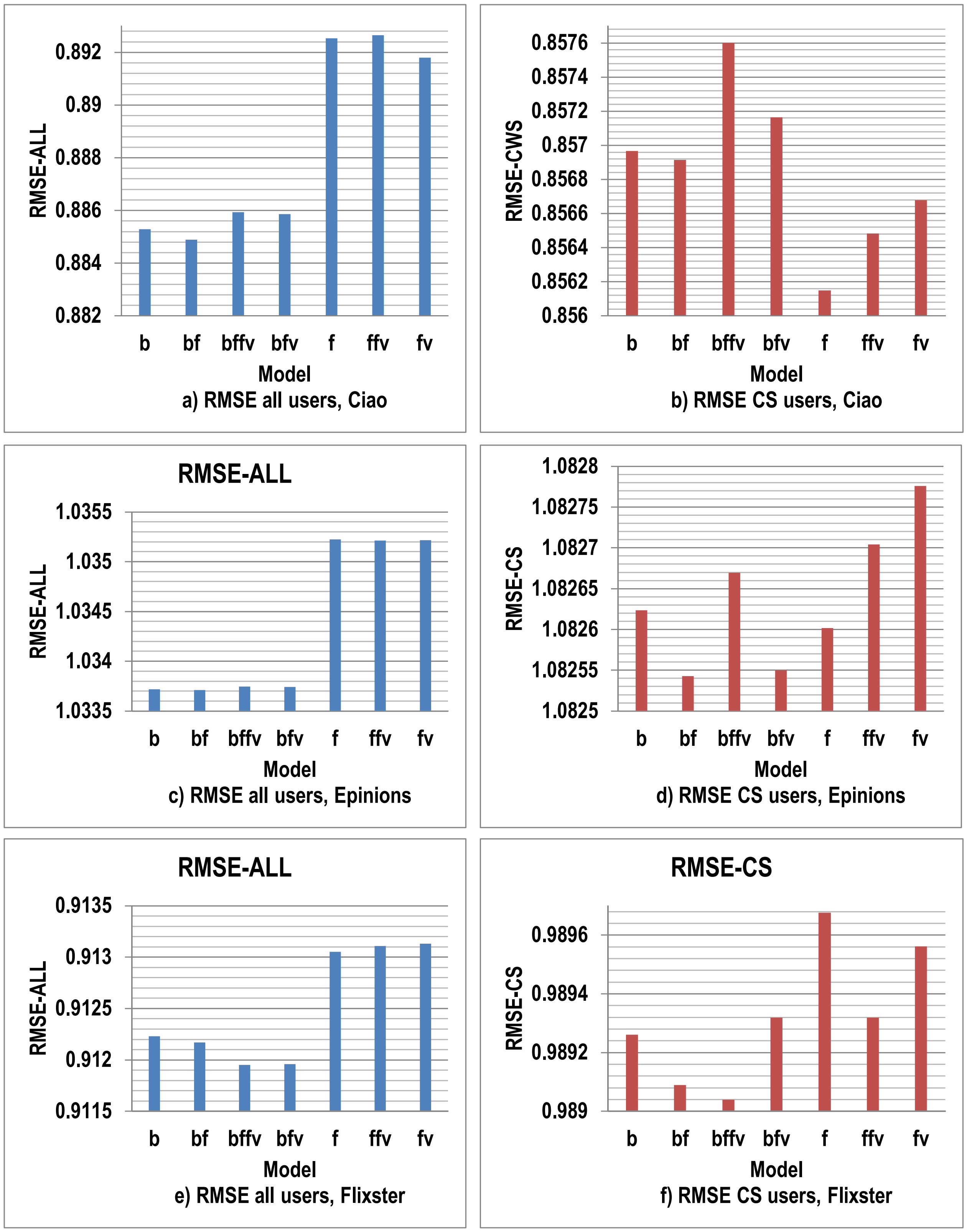


Fig. 10. Comparisons of the MAE values of Aspect-MF's combinations in (a,b) Ciao, (c,d) Epinions, and (e,f) Flixster datasets for all users (RMSE-ALL) and cold-start users (RMSE-CS).

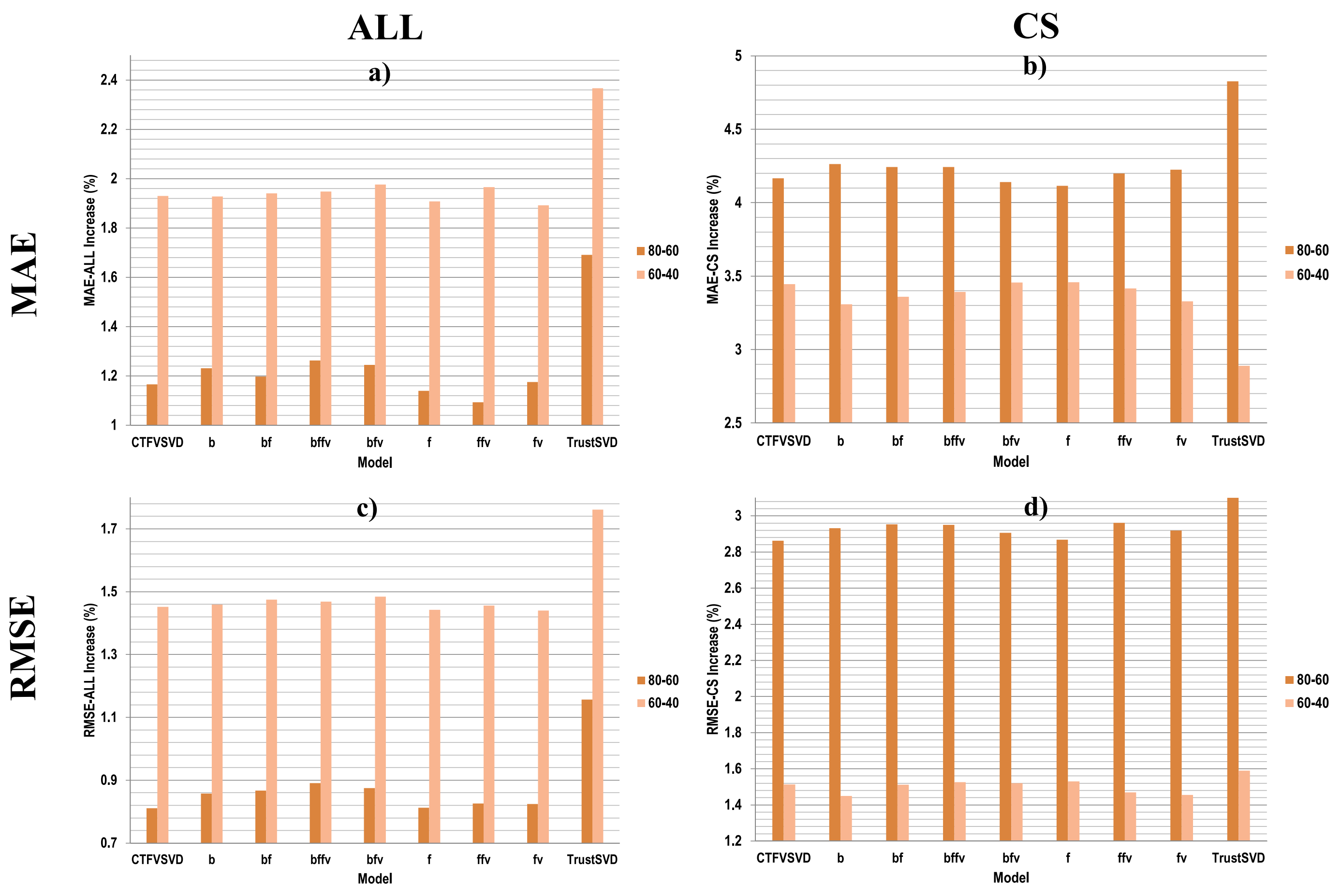


Fig. 11. Effect of the training amount on Flixster dataset for (a) MAE for all users, (b) MAE for cold-start users, (c) RMSE for all users, (d) RMSE for cold-start users.

Insights. From the observations for cold-start users, we can conclude that in order for the time information to be helpful, we need to provide the model with enough time-related data as input, so that the accuracy can be improved, and the importance of such data is more pronounced for the cold-start users, whose predictions are more sensitive to the inaccuracies. Otherwise, if the amount of training data is insufficient, the model can learn unrealistic temporal patterns that directly result from a shortage of training information.

We also saw that the degree of deterioration of the accuracy is somewhat dependent on the dataset. On the Flixster, the accuracy degrades somewhere between just under 1% to just under 5%. On Ciao, however, the accuracy deteriorates much more (roughly between 6.5% and 19.5%). Therefore, it is up to the system users to decide whether they would like to use smaller datasets and sacrifice the accuracy, or spend more time on training more accurate models using more information. We did not observe any tangible differences between the execution times of these cases (80%–60%–40%), and the computational complexity analysis of the model in Section 3.2.5 showed that the model time is of linear order. Therefore, it is probably advisable for the system owners to use as much data as available to achieve the highest accuracies, as long as their computational limitations allow.

5. Conclusion and future work

In this paper, we addressed the problem of modelling the temporal properties of human preferences in recommender systems. In order to tackle this problem, we proposed a novel latent factor model called Aspect-MF. Aspect-MF built on the basis of CTFVSVD, a model that we proposed earlier, in order to capture socially-influenced conditional preferences over feature values. In Aspect-MF, three major preference aspects were assumed to be subject to temporal drift. These aspects included user and item biases, preferences over features, and preferences over feature values. Moreover, we also analysed the temporal behaviour of each of these preference aspects and their combinations. We also considered the robustness of Aspect-MF's combinations with respect to the shortage of training data.

In order to evaluate the model, we carried out extensive experiments on three popular datasets in the area of recommender systems. We considered the model errors in terms of MAE and RMSE measures on all users and cold-start users. We also performed statistical analyses on the performances observed, to make sure that the differences in accuracies are significant, and hence do not happen by chance. The experiments revealed that in all three datasets, all combinations of Aspect-MF for both measures on all users and cold-start users significantly outperformed TrustSVD, which had proven to be the most accurate static social recommendation model before CTFVSVD. The experiments also proved that most of the Aspect-MF's combinations were significantly more accurate than CTFVSVD. In particular, we found that Aspect-MF with all dynamic aspects outperformed CTFVSVD in all three datasets on all users.

The analysis of the temporal behaviour of preference aspects and their combinations on the three datasets showed that different datasets included different temporal patterns, and therefore, required models with different dynamic aspects. This supported our component-based approach in modelling the basic preference

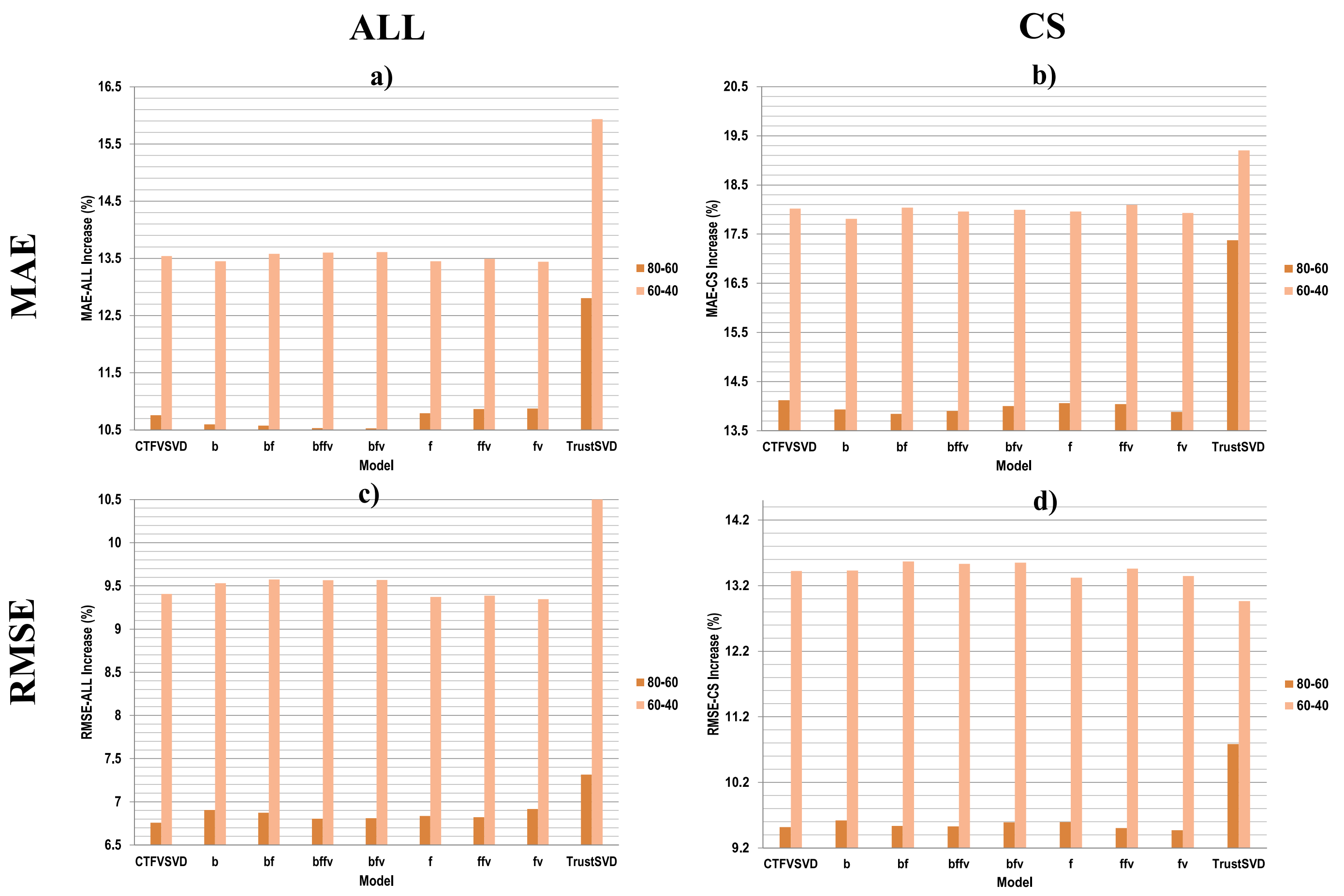


Fig. 12. Effect of the training amount on Ciao dataset for (a) MAE for all users, (b) MAE for cold-start users, (c) RMSE for all users, (d) RMSE for cold-start users.

aspects and their temporal properties. We also concluded that the dynamic models are more helpful in cases there is enough training data to discern the temporal properties. In particular, we concluded that the models proposed in this paper are more successful in modelling all users, because more time-related data is available for all users than cold-start users, and therefore the temporal characteristics were extracted more accurately. The analysis of the robustness of the models with respect to the shortage of training data also revealed that Aspect-MF was in general more robust than CTFVSVD and TrustSVD. The models were also more robust for all users than cold-start users, because cold-start users were more sensitive to the inaccurate predictions.

A direction that we would like to pursue in the future is related to explaining the resulting recommendations to the users. Explaining the recommendations to the users is believed to improve transparency and to instill trust in the users. So far we have pursued our main goal in improving the accuracy of the recommendations, and in this paper we showed how we could achieve significant improvements by taking the temporal aspects into consideration. As the next step, in particular we are interested in how we can explain the temporal properties of the trained models to the users. Furthermore, the component-based structure followed in designing Aspect-MF is generally beneficial in extracting explanations.

Acknowledgements

This work is part of the Veni research programme with project number 639.021.751, which is financed by the Netherlands Organisation for Scientific Research (NWO). We would like to acknowledge the SunCorp Group for partially funding this project, and also thank the National eResearch Collaboration Tools and Resources (Nectar) for providing us with the necessary computational resources to carry out the experiments.

Appendix A. Aspect-MF training equations

In Aspect-MF, we use gradient descent to optimise Eq. (19). The gradients for the model parameters are obtained using Eqs. (A.20) to A.41.

$$\frac{\partial E}{\partial bu_u} = \frac{\partial E_R}{\partial bu_u} = e_{uj} + \lambda_{bu}|I_u|^{-\frac{1}{2}} bu_u \tag{A.1}$$

$$\forall t_{uj} \in I_u^t : \frac{\partial E}{\partial but_{ut}} = \frac{\partial E_R}{\partial but_{ut}} = e_{uj} + \lambda_{bu}|I_u|^{-\frac{1}{2}} but_{ut} \tag{A.2}$$

$$\frac{\partial E}{\partial \alpha_u} = \frac{\partial E_R}{\partial \alpha_u} = e_{uj} dev_u(t_{uj}) + \lambda_{bu}|I_u|^{-\frac{1}{2}} \alpha_u \tag{A.3}$$

$$\frac{\partial E}{\partial bi_j} = \frac{\partial E_R}{\partial bi_j} = e_{uj}(C_u + C_{ut}) + \lambda_{bi}|J_j|^{-\frac{1}{2}} bi_j \tag{A.4}$$

$$\frac{\partial E}{\partial bit_{jBin(t_{uj})}} = \frac{\partial E_R}{\partial bit_{jBin(t_{uj})}} = e_{uj}(C_u + C_{ut}) + \lambda_{bi}|J_j|^{-\frac{1}{2}} bit_{jBin(t_{uj})} \tag{A.5}$$

$$\frac{\partial E}{\partial C_u} = \frac{\partial E_R}{\partial C_u} = e_{uj}(bi_j + bit_{jBin(t_{uj})}) + \lambda_{bi}|J_j|^{-\frac{1}{2}} C_u \tag{A.6}$$

$$\frac{\partial E}{\partial Ct_u} = \frac{\partial E_R}{\partial Ct_u} = e_{uj}(bi_j + bit_{jBin(t_{uj})}) + \lambda_{bi}|J_j|^{-\frac{1}{2}} Ct_u \tag{A.7}$$

$$\frac{\partial E}{\partial P_{uf}} = \frac{\partial E_R}{\partial P_{uf}} + \frac{\partial E_T}{\partial P_{uf}} \tag{A.8}$$

$$\frac{\partial E_R}{\partial P_{uf}} = e_{uj}(W_{uf}(t_uj)Q_{jf}(t_uj) + Z_{uf}(t_uj)) + \lambda_P|I_u|^{-\frac{1}{2}}P_{uf} \tag{A.9}$$

$$\frac{\partial E_T}{\partial P_{uf}} = \lambda_T|T_u|^{-\frac{1}{2}}P_{uf} + \lambda_t\eta_P\sum_{\forall v\in T_u} e^{(1)}_{uv}\omega_{vf} \tag{A.10}$$

$$\forall t_{uj}\in I^t_u:\ \frac{\partial E}{\partial Pt_{uft}} = \frac{\partial E_R}{\partial Pt_{uft}} + \frac{\partial E_T}{\partial Pt_{uft}} \tag{A.11}$$

$$\frac{\partial E_R}{\partial Pt_{uft}} = e_{uj}(W_{uf}(t_uj)Q_{jf}(t_{uj}) + Z_{uf}(t_uj)) + \lambda_{Pt}|I_u|^{-\frac{1}{2}}Pt_{uft} \tag{A.12}$$

$$\frac{\partial E_T}{\partial Pt_{uft}} = \lambda_T|T_u|^{-\frac{1}{2}}Pt_{uft} + \frac{\lambda_t\eta_P}{|I^t_u|}\sum_{\forall v\in T_u} e^{(1)}_{uv}\omega_{vf} \tag{A.13}$$

$$\frac{\partial E}{\partial \alpha^P_{uf}} = \frac{\partial E_R}{\partial \alpha^P_{uf}} + \frac{\partial E_T}{\partial \alpha^P_{uf}} \tag{A.14}$$

$$\frac{\partial E_R}{\partial \alpha^P_{uf}} = e_{uj}dev_u(t_{uj})(W_{uf}(t_uj)Q_{jf}(t_uj) + Z_{uf}(t_uj)) + \lambda_{\alpha^P}|I_u|^{-\frac{1}{2}}\alpha^P_{uf} \tag{A.15}$$

$$\frac{\partial E_T}{\partial \alpha^P_{uf}} = \lambda_T|T_u|^{-\frac{1}{2}}\alpha^P_{uf} + \frac{\lambda_t\eta_P}{|I^t_u|}\sum_{\forall v\in T_u}\sum_{\forall t_{uj}\in I^t_u} e^{(1)}_{uv}\omega_{vf}dev_u(t_{uj}) \tag{A.16}$$

$$\frac{\partial E}{\partial W_{uf}} = \frac{\partial E_R}{\partial W_{uf}} + \frac{\partial E_T}{\partial W_{uf}} \tag{A.17}$$

$$\begin{aligned}\frac{\partial E_R}{\partial W_{uf}} &= e_{uj}Q_{jf}(t_uj)(W_{uf}(t_uj)Q_{jf}(t_uj) + Z_{uf}(t_uj)) \\ &\quad + 2Q_{jf}(t_uj)\sum_{f'=1}^{D}(W_{uf}(t_uj)Q_{jf'}(t_uj) + Z_{uf'}(t_uj)) \\ &\quad + \lambda_W|I_u|^{-\frac{1}{2}}W_{uf}\end{aligned} \tag{A.18}$$

$$\frac{\partial E_T}{\partial W_{uf}} = \lambda_T|T_u|^{-\frac{1}{2}}W_{uf} + \lambda_t\eta_W\sum_{\forall v\in T_u} e^{(2)}_{uv}\omega_{vf} \tag{A.19}$$

$$\forall t_{uj}\in I^t_u:\ \frac{\partial E}{\partial Wt_{uft}} = \frac{\partial E_R}{\partial Wt_{uft}} + \frac{\partial E_T}{\partial Wt_{uft}} \tag{A.20}$$

$$\begin{aligned}\frac{\partial E_R}{\partial Wt_{uft}} &= e_{uj}Q_{jf}(t_uj)(W_{uf}(t_uj)Q_{jf}(t_{uj}) + Z_{uf}(t_uj)) \\ &\quad + 2Q_{jf}(t_uj)\sum_{f'=1}^{D}(W_{uf}(t_uj)Q_{jf'}(t_uj) + Z_{uf'}(t_uj)) \\ &\quad + \lambda_{Wt}|I_u|^{-\frac{1}{2}}W_{uf}\end{aligned} \tag{A.21}$$

$$\frac{\partial E_T}{\partial Wt_{uft}} = \lambda_T|T_u|^{-\frac{1}{2}}W_{uf} + \frac{\lambda_t\eta_W}{|I^t_u|}\sum_{\forall v\in T_u} e^{(2)}_{uv}\omega_{vf} \tag{A.22}$$

$$\frac{\partial E}{\partial \alpha^W_{uf}} = \frac{\partial E_R}{\partial \alpha^W_{uf}} + \frac{\partial E_T}{\partial \alpha^W_{uf}} \tag{A.23}$$

$$\begin{aligned}\frac{\partial E_R}{\partial \alpha^W_{uf}} &= e_{uj}dev_u(t_{uj})(W_{uf}(t_uj)Q_{jf}(t_uj) + Z_{uf}(t_uj)) \\ &\quad + 2Q_{jf}(t_uj)\sum_{f'=1}^{D}(W_{uf}(t_uj)Q_{jf'}(t_uj) + Z_{uf'}(t_uj)) \\ &\quad + \lambda_{\alpha^W}|I_u|^{-\frac{1}{2}}\alpha^W_{uf}\end{aligned} \tag{A.24}$$

$$\frac{\partial E_T}{\partial \alpha^W_{uf}} = \lambda_T|T_u|^{-\frac{1}{2}}\alpha^W_{uf} + \frac{\lambda_t\eta_W}{|I^t_u|}\sum_{\forall v\in T_u}\sum_{\forall t_{uj}\in I^t_u} e^{(2)}_{uv}\omega_{vf}dev_u(t_{uj}) \tag{A.25}$$

$$\frac{\partial E}{\partial Z_{uf}} = \frac{\partial E_R}{\partial Z_{uf}} + \frac{\partial E_T}{\partial Z_{uf}} \tag{A.26}$$

$$\begin{aligned}\frac{\partial E_R}{\partial Z_{uf}} &= e_{uj}(W_{uf}(t_uj)Q_{jf}(t_uj) + Z_{uf}(t_uj)) \\ &\quad + 2\sum_{f'=1}^{D}(W_{uf}(t_uj)Q_{jf'}(t_uj) + Z_{uf'}(t_uj)) \\ &\quad + \lambda_Z|I_u|^{-\frac{1}{2}}Z_{uf}\end{aligned} \tag{A.27}$$

$$\frac{\partial E_T}{\partial Z_{uf}} = \lambda_T|T_u|^{-\frac{1}{2}}Z_{uf} + \lambda_t\eta_Z\sum_{\forall v\in T_u} e^{(3)}_{uv}\omega_{vf} \tag{A.28}$$

$$\forall t_{uj}\in I^t_u:\ \frac{\partial E}{\partial Zt_{uft}} = \frac{\partial E_R}{\partial Zt_{uft}} + \frac{\partial E_T}{\partial Zt_{uft}} \tag{A.29}$$

$$\begin{aligned}\frac{\partial E_R}{\partial Zt_{uft}} &= e_{uj}(W_{uf}(t_uj)Q_{jf}(t_{uj}) + Z_{uf}(t_uj)) \\ &\quad + 2\sum_{f'=1}^{D}(W_{uf}(t_uj)Q_{jf'}(t_uj) + Z_{uf'}(t_uj)) \\ &\quad + \lambda_Z|I_u|^{-\frac{1}{2}}Zt_{uft}\end{aligned} \tag{A.30}$$

$$\frac{\partial E_T}{\partial Zt_{uft}} = \lambda_T|T_u|^{-\frac{1}{2}}Zt_{uft} + \frac{\lambda_t\eta_Z}{|I^t_u|}\sum_{\forall v\in T_u} e^{(3)}_{uv}\omega_{vf} \tag{A.31}$$

$$\frac{\partial E}{\partial \alpha^Z_{uf}} = \frac{\partial E_R}{\partial \alpha^Z_{uf}} + \frac{\partial E_T}{\partial \alpha^Z_{uf}} \tag{A.32}$$

$$\begin{aligned}\frac{\partial E_R}{\partial \alpha^Z_{uf}} &= e_{uj}dev_u(t_{uj})(W_{uf}(t_uj)Q_{jf}(t_uj) + Z_{uf}(t_uj)) \\ &\quad + 2\sum_{f'=1}^{D}(W_{uf}(t_uj)Q_{jf'}(t_uj) + Z_{uf'}(t_uj)) \\ &\quad + \lambda_{\alpha^Z}|I_u|^{-\frac{1}{2}}\alpha^Z_{uf}\end{aligned} \tag{A.33}$$

$$\frac{\partial E_T}{\partial \alpha^Z_{uf}} = \lambda_T|T_u|^{-\frac{1}{2}}\alpha^Z_{uf} + \frac{\lambda_t\eta_Z}{|I^t_u|}\sum_{\forall v\in T_u}\sum_{\forall t_{uj}\in I^t_u} e^{(3)}_{uv}\omega_{vf}dev_u(t_{uj}) \tag{A.34}$$

$$\forall i\in I_u:\ \frac{\partial E}{\partial y_{if}} = \frac{\partial E_R}{\partial y_{if}} = e_{uj}|I_u|^{-\frac{1}{2}}(W_{uf}V_{jf} + Z_{uf}) + (\lambda_y|J_j|^{-\frac{1}{2}}y_{if}) \tag{A.35}$$

$$\forall v\in T_u:\ \frac{\partial E}{\partial \omega_{vf}} = \frac{\partial E_R}{\partial \omega_{vf}} + \frac{\partial E_T}{\partial \omega_{vf}} \tag{A.36}$$

$$\frac{\partial E_R}{\partial \omega_{vf}} = e_{uj}|T_u|^{-\frac{1}{2}}(W(t)_{uf}Q_{jf} + Z(t)_{uf}) \tag{A.37}$$

$$\begin{aligned}\frac{\partial E_T}{\partial \omega_{vf}} &= (\lambda_T|T^+_v|^{-\frac{1}{2}})\omega_{vf} + \frac{\lambda_t\eta_P}{|I^t_u|}\sum_{\forall t_{uj}\in I^t_u} e^{(1)}_{uv}P(t_{uj})_{uf} \\ &\quad + \frac{\lambda_t\eta_W}{|I^t_u|}\sum_{\forall t_{uj}\in I^t_u} e^{(2)}_{uv}(1 - W(t_{uj})_{uf}) \\ &\quad + \frac{\lambda_t\eta_Z}{|I^t_u|}\sum_{\forall t_{uj}\in I^t_u} e^{(3)}_{uv}Z(t_{uj})_{uf}\end{aligned} \tag{A.38}$$

$$\frac{\partial E}{\partial Y_{ff'}} = \frac{\partial E_R}{\partial Y_{ff'}} = e_{uj}(W_{if}V_{jf} + Z_{if})(W_{if'}V_{jf'} + Z_{if'}) - \lambda_Y Y_{ff'} \tag{A.39}$$

$$\frac{\partial E}{\partial Q_{jf}} = \frac{\partial E_R}{\partial Q_{jf}} = e_{uj}[W_{uf}(P_{uf} + |I_u|^{-\frac{1}{2}}\sum_{\forall i\in I_u} y_i$$

$$+ |T_u|^{-\frac{1}{2}} \sum_{\forall v \in T_u} \omega_v) + 2W_{uf} \sum_{f'=1}^{D} (W_{if'} V_{jf'} + Z_{if'}) Y_{ff'}]$$
$$+ \lambda_Q |U_j|^{-\frac{1}{2}} Q_{jf} \quad (A.40)$$

where:

$$e_{uj} = R_{uj} - \hat{R_{uj}} \quad (A.41)$$

Therefore, the gradients in Eqs. (A.20)–(A.41) will be used to update the values matrices used to capture socially-influenced drifting conditional feature value preferences using an incremental gradient descent method.

References

Aldrich, S. E. (2011). Recommender systems in commercial use. *AI Magazine, 32*(3), 28–34.

Baltrunas, L., Ludwig, B., & Ricci, F. (2011). Matrix factorization techniques for context aware recommendation. In *Proceedings of the fifth ACM conference on recommender systems* (pp. 301–304). ACM.

Chatzis, S. (2014). Dynamic bayesian probabilistic matrix factorization.. In *AAAI* (pp. 1731–1737).

Chen, L., Chen, G., & Wang, F. (2015). Recommender systems based on user reviews: The state of the art. *User Modeling and User-Adapted Interaction, 25*(2), 99–154.

D'Addio, R. M., & Manzato, M. G. (2015). A sentiment-based item description approach for KNN collaborative filtering. In *Proceedings of the 30th annual ACM symposium on applied computing* (pp. 1060–1065). ACM.

Guo, G., Zhang, J., & Yorke-Smith, N. (2013). A novel bayesian similarity measure for recommender systems. In *Proceedings of the 23rd international joint conference on artificial intelligence (IJCAI)* (pp. 2619–2625).

Guo, G., Zhang, J., & Yorke-Smith, N. (2015). Trustsvd: Collaborative filtering with both the explicit and implicit influence of user trust and of item ratings.. In *AAAI* (pp. 123–129).

Guo, G., Zhang, J., & Yorke-Smith, N. (2016). A novel recommendation model regularized with user trust and item ratings. *IEEE Transactions on Knowledge and Data Engineering, 28*(7), 1607–1620.

Guo, G., Zhu, F., Qu, S., & Wang, X. (2018). Pccf: Periodic and continual temporal co-factorization for recommender systems. *Information Sciences, 436*, 56–73.

Jahrer, M., Töscher, A., & Legenstein, R. (2010). Combining predictions for accurate recommender systems. In *Proceedings of the 16th ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 693–702). ACM.

Jamali, M., & Ester, M. (2010). A matrix factorization technique with trust propagation for recommendation in social networks. In *Proceedings of the fourth ACM conference on recommender systems* (pp. 135–142). ACM.

Karatzoglou, A., Amatriain, X., Baltrunas, L., & Oliver, N. (2010). Multiverse recommendation: n-dimensional tensor factorization for context-aware collaborative filtering. In *Proceedings of the fourth ACM conference on recommender systems* (pp. 79–86). ACM.

Koenigstein, N., Dror, G., & Koren, Y. (2011). Yahoo! music recommendations: Modeling music ratings with temporal dynamics and item taxonomy. In *Proceedings of the fifth ACM conference on recommender systems* (pp. 165–172). ACM.

Korb, K. B., & Nicholson, A. E. (2010). *Bayesian artificial intelligence*. CRC Press.

Koren, Y. (2010). Collaborative filtering with temporal dynamics. *Communications of the ACM, 53*(4), 89–97.

Koren, Y., & Bell, R. (2011). Advances in collaborative filtering. In *Recommender systems handbook* (pp. 145–186). Springer.

Koren, Y., Bell, R., & Volinsky, C. (2009). Matrix factorization techniques for recommender systems. *Computer, 42*(8), 30–37.

Lee, T. Q., Park, Y., & Park, Y.-T. (2008). A time-based approach to effective recommender systems using implicit feedback. *Expert Systems with Applications, 34*(4), 3055–3062.

Lewis, K., Gonzalez, M., & Kaufman, J. (2012). Social selection and peer influence in an online social network. *Proceedings of the National Academy of Sciences, 109*(1), 68–72. doi:10.1073/pnas.1109739109.

Li, R., Li, B., Jin, C., Xue, X., & Zhu, X. (2011). Tracking user-preference varying speed in collaborative filtering. *AAAI*.

Liu, W., Wu, C., Feng, B., & Liu, J. (2015). Conditional preference in recommender systems. *Expert Systems with Applications, 42*(2), 774–788.

Liu, X., & Aberer, K. (2013). Soco: A social network aided context-aware recommender system. In *Proceedings of the 22nd international conference on world wide web* (pp. 781–802). ACM.

Luo, C., & Cai, X. (2016). Bayesian wishart matrix factorization. *Data Mining and Knowledge Discovery, 30*(5), 1166–1191.

Ma, H., Yang, H., Lyu, M. R., & King, I. (2008). Sorec: Social recommendation using probabilistic matrix factorization. In *Proceedings of the 17th ACM conference on information and knowledge management* (pp. 931–940). ACM.

Ma, H., Zhou, D., Liu, C., Lyu, M. R., & King, I. (2011). Recommender systems with social regularization. In *Proceedings of the fourth ACM international conference on web search and data mining* (pp. 287–296). ACM.

Pan, J., Ma, Z., Pang, Y., & Yuan, Y. (2013). Robust probabilistic tensor analysis for time-variant collaborative filtering. *Neurocomputing, 119*, 139–143.

Rafailidis, D. (2018). A multi-latent transition model for evolving preferences in recommender systems. *Expert Systems with Applications, 104*, 97–106.

Salakhutdinov, R., & Mnih, A. (2008). Bayesian probabilistic matrix factorization using markov chain monte carlo. In *Proceedings of the 25th international conference on machine learning* (pp. 880–887). ACM.

Salakhutdinov, R., & Mnih, A. (2011). Probabilistic matrix factorization. In *NIPS: 20* (pp. 1–8).

Widmer, G., & Kubat, M. (1996). Learning in the presence of concept drift and hidden contexts. *Machine Learning, 23*(1), 69–101.

Xiang, L., & Yang, Q. (2009). Time-dependent models in collaborative filtering based recommender system. In *Web intelligence and intelligent agent technologies, 2009. WI-IAT'09. IEEE/WIC/ACM international joint conferences on: 1* (pp. 450–457). IEEE.

Xiong, L., Chen, X., Huang, T.-K., Schneider, J., & Carbonell, J. G. (2010). Temporal collaborative filtering with bayesian probabilistic tensor factorization. In *Proceedings of the 2010 SIAM international conference on data mining* (pp. 211–222). SIAM.

Zafarani, R., Abbasi, M. A., & Liu, H. (2014). *Social media mining: An introduction*. Cambridge University Press.

Zafari, F., & Moser, I. (2016). Feature-aware factorised collaborative filtering. In *Australasian joint conference on artificial intelligence* (pp. 561–569). Springer.

Zafari, F., & Moser, I. (2017). Modelling socially-influenced conditional preferences over feature values in recommender systems based on factorised collaborative filtering. *Expert Systems with Applications, 87*, 98–117. https://doi.org/10.1016/j.eswa.2017.05.058.

Zafari, F., Moser, I., & Rahmani, R. (2017). Proposing a highly accurate hybrid component-based factorised preference model in recommender systems. In *Proceedings of the 26th international joint conference on artificial intelligence (IJCAI)*.

Zafari, F., & Nassiri-Mofakham, F. (2016). Popponent: Highly accurate, individually and socially efficient opponent preference model in bilateral multi issue negotiations. *Artificial Intelligence, 237*, 59–91.

Zafari, F., & Nassiri-Mofakham, F. (2017). Popponent: Highly accurate, individually and socially efficient opponent preference model in bilateral multi issue negotiations (extended abstract). In *Proceedings of the 26th international joint conference on artificial intelligence (IJCAI)*.

Zafari, F., Nassiri-Mofakham, F., & Hamadani, A. Z. (2015). Dopponent: A socially efficient preference model of opponent in bilateral multi issue negotiations. *Journal of Computing and Security, 1*(4), 283–292.

Zhang, Y., Lai, G., Zhang, M., Zhang, Y., Liu, Y., & Ma, S. (2014). Explicit factor models for explainable recommendation based on phrase-level sentiment analysis. In *Proceedings of the 37th international ACM SIGIR conference on research & development in information retrieval* (pp. 83–92). ACM.

Zhao, H., Wang, S., Chen, Q., & Cao, J. (2015). Probabilistic matrix factorization based on similarity propagation and trust propagation for recommendation. In *2015 IEEE conference on collaboration and internet computing (CIC)* (pp. 90–98). IEEE.