Data Mining and Knowledge Discovery (2021) 35:837–862
https://doi.org/10.1007/s10618-021-00738-8

FuseRec: fusing user and item homophily modeling with temporal recommender systems

Kanika Narang[1] · Yitong Song[1] · Alexander Schwing[1] · Hari Sundaram[1]

Received: 14 January 2020 / Accepted: 15 January 2021 / Published online: 15 February 2021

Abstract
Recommender systems can benefit from a plethora of signals influencing user behavior such as her past interactions, her social connections, as well as the similarity between different items. However, existing methods are challenged when taking all this data into account and often do not exploit all available information. This is primarily due to the fact that it is non-trivial to combine the various information as they mutually influence each other. To address this shortcoming, here, we propose a 'Fusion Recommender' (FuseRec), which models each of these factors separately and later combines them in an interpretable manner. We find this general framework to yield compelling results on all three investigated datasets, Epinions, Ciao, and CiaoDVD, outperforming the state-of-the-art by more than 14% for Ciao and Epinions. In addition, we provide a detailed ablation study, showing that our combined model achieves accurate results, often better than any of its components individually. Our model also provides insights on the importance of each of the factors in different datasets.

Keywords Attention-based graph networks · Temporal recommender systems · Social recommendation · Item similarity modeling

1 Introduction

Recommender systems are ubiquitous and model user preferences on commercial and social websites as well as in apps. These systems predict with reasonable accuracy, products that we may be interested in, people which we may know, or songs and movies that we may appreciate. This success builds upon a long history of research. However, to this day, a large active community continues to improve recommender systems

Responsible editor: Ira Assent, Carlotta Domeniconi, Aristides Gionis, Eyke Hüllermeier.

✉ Kanika Narang
knarang2@illinois.edu

[1] University of Illinois at Urbana-Chamaign, Urbana, IL, USA

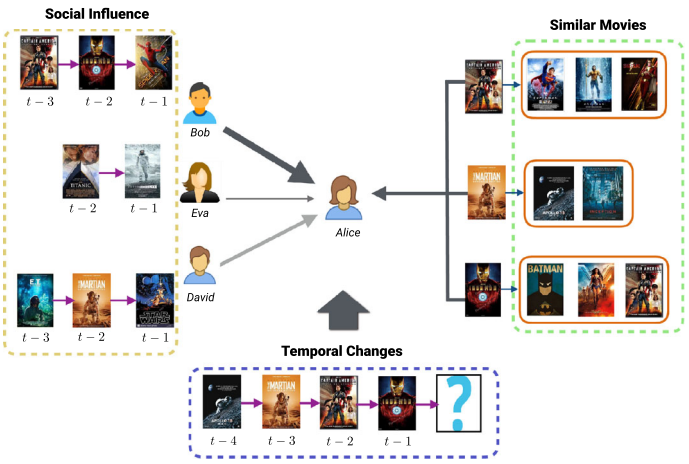

Fig. 1 Illustrative diagram of factors affecting Alice's decision on which movie to watch next in an online social movie viewing platform. Her current interests are towards superhero movies (temporal); she could either decide to watch recent superhero movies seen by her friends (social) or other superhero movies not watched by her friends (similar items)

as many questions remain open, e.g., How to effectively model and merge multiple factors influencing user preferences like (1) temporal context, (2) social influence, and (3) similarity between items? We explore this question in detail.

Classic collaborative filtering is one of the most successful approaches to model user preferences. It learns a low dimensional and often linear latent factor model for both users and items via matrix factorization of the user-item interaction matrix (Rendle et al. 2009). With deep learning taking a more prominent role, more complex models have been applied to learn increasingly non-linear relationships (He et al. 2017; Wu et al. 2016). However, those classical methods ignore all three of the factors above. Hence, many techniques have been developed, which augment classical recommender systems with one of those factors.

First, considering temporal context removes the assumption of a static interaction matrix, which generally doesn't hold as user preferences evolve with time. Thus, history from a distant past is not necessarily relevant to current preferences. To this end, Markov chains (Rendle et al. 2010) and recently, Convolution Neural Network (CNN) (Kang and McAuley 2018) and Recurrent Neural Network (RNN) (Hidasi et al. 2016) based methods have been proposed to model this temporal dependence in recommender systems. Those methods remove the static interaction matrix of classical collaborative filtering and learn a user's and an item's hidden representation based on their recent interactions.

Second, considering social influence removes the restriction that users operate in isolation. This idea is popularized by the social influence theory (Tang et al. 2009), which argues that a user's preference is influenced by their friends' behavior, leading to user homophily (similar user preferences). It is noteworthy that these influences are inherently dynamic as friends' preferences are evolving too (socio-temporal influence), a fact mostly ignored by current systems. For instance, recent works model static social

effect (Zhao et al. 2014; Wu et al. 2018; Wang et al. 2018; Fan et al. 2019). These methods look at the entire history of the user's friends instead of emphasizing the most recent actions. Moreover, these approaches assume uniform importance of all friends. While this is not suitable in general, it is an important first step to understand social influence for the recommendation.

Third, exploiting similarities between items (based on co-occurrence or similar features) alleviates the data sparsity issue (many items with few ratings). Similar items hold similar attractiveness to users, leading to item homophily. Deep net-based recommender models are prone to skew prediction results towards popular items with ample feedback in the training data (overfit to popular items) (Krishnan et al. 2018). This is counterproductive to user experience as it leads to similar recommendations across users. Also, compared to highly frequented items, long-tail items (items with fewer ratings) result in higher profit margins for the platforms (Yin et al. 2012). Item-Item collaborative filtering based methods (Sarwar et al. 2001) integrate these similarities but ignore the user's history; thus providing generic recommendations.

To make these three points concrete, let us consider the example shown in Fig. 1. Alice is using an online social movie viewing platform. She is currently hooked onto superhero movies (temporal). While deciding which movie to watch next, she will be influenced by recent superhero movies watched by her friends on the platform (socio-temporal influence). She could also decide to watch other superhero movies on the platform not seen by her social circle yet (item-to-item similarity).

As illustrated earlier, a user's behavior is affected by at least the aforementioned three factors (others could include time-of-day, mood, etc.). However, the relative importance of these factors remains unclear. It is indeed challenging to model these factors effectively and efficiently in a unified model as these cues mutually influence each other. This is emphasized by the fact that existing work often studies only a subset of those three cues.

In this work, we first propose three different modules to model each factor: (1) a user-temporal, (2) a user-social, and (3) an item-similarity module. To model the temporal behavior of a user's item viewing history, we use widely adopted recurrent neural nets. These networks are shown to capture complex and non-linear relationships of time-varying sequences. To capture the effect of a user's friends' behavior, we develop a novel attention-based social aggregation model. Different from existing works, it aggregates a user's friends' recent item history in a weighted manner. We learn attention weights separately for each pair of a user and her friend. Note that as user preferences evolve, our dynamic social recommender uses only recent interactions from friend's history relative to the current timestamp. For item-item similarity aggregation, we construct a 'social graph of items' based on similarity in item features and co-occurrence in the dataset. We develop a novel attention-based aggregation model for the item similarity graph too. In contrast to existing work, we learn an attention weight for each similar item and later aggregate information of neighboring items in a weighted manner.

We further develop a 'Fusion Recommender' (FuseRec) model to jointly and efficiently combine all these factors in a unified model. It takes into account a user's temporal changes along with homophily in the user and item space. It treats each of the modules equally and combines them in an interpretable manner. Specifically,

we compare the output from the user-temporal and user-social module with the output from item-similarity module. We then linearly combine these scores via learned weights to provide an understanding of the importance of the three factors. The magnitude of the learned weights permits a glimpse at the importance of the individual modules.

We evaluate our model on three representative benchmark datasets: Ciao, CiaoDVD, and Epinions. We compare to an array of collaborative filtering, temporal, and social methods and achieve a significant improvement of more than 14% for AUC on the Ciao and the Epinions dataset and perform similarly for the CiaoDVD dataset. In addition, we provide a study on the importance of the three factors. Across all datasets, we find the temporal module to be the most significant factor for modeling user preferences.

In summary, our main contributions are as follows:

- We propose a novel attention based aggregation model to capture homophily in both the user and item space. Our proposed user-social module also uses dynamic features as it operates on only the recent history of friends. The item-similarity module operates on explicit item-item similarity graphs based on co-occurrence and feature similarity to capture homophily in the item space.
- We further propose 'FuseRec,' a Fusion Recommender model to effectively combine temporal, social, and item similarity modules in an interpretable manner.
- We evaluate our method on three benchmark datasets and compare it to a variety of recent temporal, social, and socio-temporal models. We also provide a detailed study regarding the importance of different factors used in our model. Our experiments show that different factors play a role in providing an effective recommendation. However, user history contributes the most in predicting user preferences.

2 Related work

Since FuseRec leverages temporal, social and item similarity factors, in the following, we provide a brief review on temporal recommendation, social recommendation, and item similarity. Before doing so, we discuss classical collaborative filtering. We also provide background on Graph Convolution Nets, which we use in our model.

Collaborative filtering: Collaborative Filtering (CF) is one of the most popular techniques for explicit feedback (predicting item ratings by a given user) in recommender systems. Specifically, the methods employ Matrix Factorization (MF) to decompose a user-item rating matrix into user and item-specific latent factors. Classical and seminal work for MF-based recommender systems (Rendle et al. 2009) uses a Bayesian pairwise loss (BPR). Collaborative filtering is also performed in item space (Sarwar et al. 2001), where similar items are computed offline based on their rating similarity or co-occurrence in the dataset. Consequently, it recommends items similar to the ones used in the past by the user. We also use similar item-based collaborative filtering for our item-similarity module. Neural net approaches have been proposed recently to improve MF models. They learn more complex non-linearities in the user-item interaction data (He et al. 2017; Wu et al. 2016).

However, most MF approaches assume a static user-item interaction matrix. Often, this assumption is not accurate, particularly for online communities where user preferences evolve — sometimes quickly — necessitating temporal recommendation, which we use in FuseRec.

Temporal Recommendation: There has been significant work in the area of temporal recommender systems that model a user's past interactions to inform a user's current preference. These temporal models generally assume a linear relationship between the events and model it using a Markov chain (Cheng et al. 2013; Rendle et al. 2010). However, these are often 'shallow' (i.e., linear) methods that are inept at modeling the more complex dynamics of temporal changes. Recent works (Sun et al. 2018; Cai et al. 2017; Kang and McAuley 2018) use deep net approaches involving convolution layers, attention networks, and recurrent neural nets to model complex relations. For example, Tang and Wang (2018) apply convolutional filters on the embedding matrix computed from a few recent items of a user. This model captures a higher-order Markov chain, but it still has a limited scope as it does not consider the entire history of a user.

In contrast, to model long term dependencies, Hidasi et al. (2016) propose to model a user's sequential behavior within a session using recurrent neural nets. Wu et al. (2017) apply a recurrent architecture to both user and item sequences and hence model dynamic influences in popularity of movies on users' viewing preference. Kang and McAuley (2018) instead employ a self-attention module for next item recommendation that adaptively learns the importance of all past items in a user's history. However, these models are limited as they do not leverage the social connections of a user as we do in FuseRec.

Social Recommendation: Social recommenders integrate information from a user's social connections to mitigate data sparsity for cold-start users, i.e., users with no or minimal history. They exploit the principle of social influence theory (Tang et al. 2009), which states that socially connected users exert influence on each other's behavior, leading to a homophily effect: similar preferences towards items. Jamali and Ester (2010), Ma et al. (2011) use social regularization in matrix factorization models to constrain socially connected users to have similar preferences. The recently proposed SERec (Wang et al. 2018) embeds items seen by the user's social neighbors as a prior in a matrix factorization model. The SBPR model (Zhao et al. 2014) extends the pairwise BPR model to incorporate social signals so that users assign higher ratings to items preferred by their friends. However, these models assume equal influence among all social neighbors. TBPR (Wang et al. 2016) distinguishes between strong and weak ties only when computing social influence strength.

In contrast, we specifically model the influence between each user, friend pair via attention weights. Also, all of the aforementioned approaches model dependence on the entire history of friends. In contrast, in FuseRec, we model dynamic social influence, which emphasizes recent 'consumptions' of a user and their friends.

Socio-Temporal Recommendation: Few of the recent approaches have started to look at merging temporal dependence with social influence. Cai et al. (2017) extend Markov chain based temporal recommenders (Rendle et al. 2010) by incorporating information about the last interacted item of a user's friends. However, this method only assumes

linear dependence. FuseRec extends this work to capture more complexities in the data.

In the context of session-based recommendation, Sun et al. (2018) propose a socially aware recurrent neural network that uses a dynamic attention network to capture social influence. On the other hand, Song et al. (2019) use graph attention nets to model social influence on a user's behavior in the session. Both these models learn a unified user representation based on social influence with a user's temporal history.

In contrast, we model temporal and dynamic social influence factors separately. We subsequently combine them in an interpretable manner to obtain insights about the importance of different factors. We also use an additional module to model item similarity that is not used by any of these methods.

Graph Convolution Networks: More recently, Graph Convolution Networks (GCNs) have been proposed to learn embeddings for graph-structured data (Kipf and Welling 2017). In recommender systems, they have been used to model the user-item interaction graph. GCMC (van den Berg et al. 2017) extends GCN by training an auto-encoder framework on a bipartite user-item interaction graph that performs differentiable message passing, aggregating data from a user's and an item's 'neighbors.' PinSage (Ying et al. 2018) proposed a random walk based sampling of neighbors to scale GCNs to web-scale graphs. Fan et al. (2019) further extend these methods to incorporate information from a user's social connections. Similarly, Wu et al. (2019a) use graph neural networks to model diffusion of social influence in recommender systems.

However, different from FuseRec, these methods either do not take a user's social neighbors into account or operate on static features. All these models also assign a uniform weight to all their neighbors, which does not represent online social communities well. Typically in these communities, some friends are only superficially known while others are known personally for years. Thus, they exert a different degree of influence on a user's behavior.

Graph Attention Networks (Veličković et al. 2018) have been recently proposed to learn attention weights between each pair of nodes in a static graph. To account for a different degree of social influence, in FuseRec, we also use an attention mechanism to learn weights between a user and her friends in the user's social graph. In contrast to graph attention nets, we deal with graphs with dynamic features (recent items). However, we argue that a friend's influence on the user does not change with her recent interactions. Thus, attention weights are computed separately based on users' static preferences and then applied to the dynamic features.

In summary, in contrast to all the previous works, we develop a novel model that learns temporal dependence of a user's history along with socio-temporal influence and item similarity. We also learn how to linearly combine these factors. This combination helps to provide insights regarding the importance of each of the cues.

3 Proposed method

We first provide an overview of the proposed FuseRec approach to model user and item homophily via attention based nets while also modeling temporal relations. We subsequently discuss the details of each employed module.

3.1 Overview

Our goal is to create a ranked list of items, indicating the preference of a user u for interacting with item $i \in \mathcal{I}$ at time t. Here, $\mathcal{I}$ represents the set of all items available on the considered platform. We compute this ranked list by sorting probability scores $\hat{r}^t_{u,i}$ for a user u and item i at time t. Formally, given user u and time t, we obtain $\forall i$, the probability scores after scaling the output of a linear layer, i.e.,

$$\hat{r}^t_{u,i} = \sigma\left(\lambda_1 S_1 + \lambda_2 S_2 + \lambda_3 S_3 + \lambda_4 S_4 + b_c\right). \tag{1}$$

Hereby σ denotes the sigmoid function, b_c is a learnable bias, and $\lambda_k \in \mathbb{R}$, $k \in \{1, \ldots, 4\}$, are four learnable weights for the scores S_k. Importantly, because we learn a linear combination of scores, we can study their magnitude which provides evidence regarding the importance of the different factors in the proposed FuseRec model. See Sect. 4.5 for our experimental results.

As illustrated in Fig. 2, we obtain the scores S_k by combining information from the following three modules: (1) the user-temporal module which leverages temporal information about a user; (2) the user-social module which captures information about

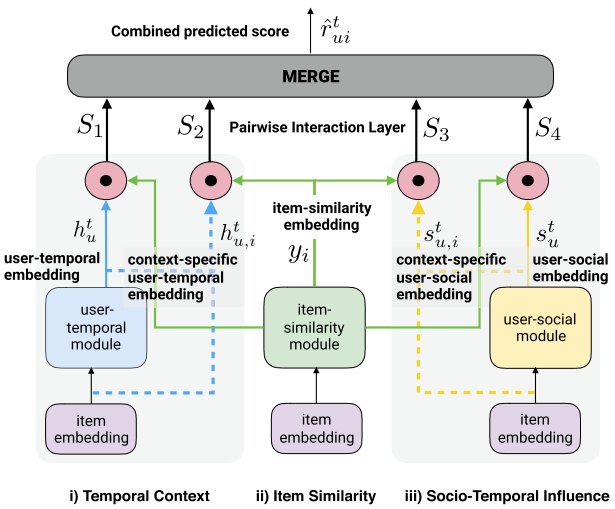

Fig. 2 Overview of the proposed FuseRec model. Our model computes pairwise interaction scores which compares item embeddings from the item-similarity module with user embeddings from both the user-temporal module and the user-social module. These scores are then merged using learnable weights to compute the final predicted score

the recent interactions of a user's friends; and (3) the item-similarity module, which captures information about item homophily.

Specifically, Eq. (1) combines four pairwise interactions: (1) $S_1 = h^t_u \odot y_i$, (2) $S_2 = h^t_{u,i} \odot y_i$, (3) $S_3 = s^t_u \odot y_i$, and (4) $S_4 = s^t_{u,i} \odot y_i$. Hereby, $\odot$ indicates an inner product between two embeddings. Note, all pairwise interaction scores S_k assess the similarity between a D-dimensional representation obtained from the item-similarity module ($y_i \in \mathbb{R}^D$) and information obtained from either the user-temporal module ($h^t_u, h^t_{u,i} \in \mathbb{R}^D$) or the user-social module ($s^t_u, s^t_{u,i} \in \mathbb{R}^D$).

The user-temporal module encapsulates information from a history of user interactions. This module computes a time-dependent embedding h^t_u for user $u \in \mathcal{U}$ at time t. This embedding denotes a user's current preferences in general. We also compute a context-specific user-temporal embedding, $h^t_{u,i}$ which encodes similarity of a user's current preferences (h^t_u) in the context of the candidate item $i \in \mathcal{I}$. Note that both embeddings capture different aspects (general and item-specific) and are time-dependent.

The user-social module captures a user's social preferences based on the recent history of the user's social graph. Specifically, for user u at time t, we encode this information in an embedding referred to as user-social embedding, s^t_u. Similar to the user-temporal module, we also compute a context-specific user-social embedding for a user, $s^t_{u,i}$, encoding similarity of a user's social preferences (s^t_u) with respect to the candidate item i.

The item-similarity module employs item-item collaborative filtering, building an implicit similarity network between items based on their features and co-occurrence in the dataset. This module computes an item-similarity embedding y_i for item $i \in \mathcal{I}$, which is identical across time. We think this is a reasonable assumption as properties of items do not change over time.[1]

We will next provide details about computation of the user-temporal embedding h^t_u, the context-specific user-temporal embedding $h^t_{u,i}$, a user-social embedding s^t_u, the context-specific user-social embedding $s^t_{u,i}$, and the item-similarity embedding y_i.

3.2 User-temporal module

Users constantly interact with items offered on online platforms, e.g., users rate or watch movies. Importantly, a user's preference does not remain constant and changes over time. To model temporal dynamics, classical methods have explored Markov chains, attention networks, and convolution networks (Cheng et al. 2013; Kang and McAuley 2018; Tang and Wang 2018). However, these methods assume dependence on only recent history and thus do not capture long term dependencies within user-item preferences. To address this concern, we use recurrent neural nets (RNNs) based on long-short-term-memory (LSTM) components. Those are widely used in natural language processing to capture sequential dynamics (Mikolov et al. 2010).

[1] Experiments with time-sensitive item embeddings decreased accuracy of the reported results.

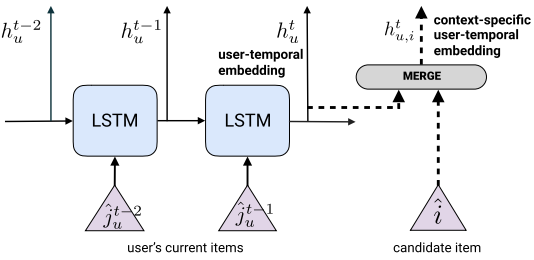

Fig. 3 The user-temporal module uses an LSTM to compute an embedding h_u^t based on a user's history h_u^{t-1} and the current item $\hat{j}_u^{t-1}$. We also compute context-specific user-temporal embedding $h_{u,i}^t$ with respect to candidate item i

In general, RNNs are based on the following recurrence relation, where h^t represents the hidden vector at time t, x^t is the input at time t and w refers to learnable weights:

$$h^t = f(h^{t-1}, x^{t-1}, w).$$

To specialize to our case, consider again a platform where users watch movies. Formally, for each user u at time $t-1$, let $j_u^{t-1} \in \mathcal{I}$ be the item which user u interacted with at time $t-1$. To compute its item embedding $\hat{j}_u^{t-1} \in \mathbb{R}^D$ (throughout we use '$\hat{\cdot}$' to indicate embeddings), we concatenate the item j_u^{t-1} with its one-hot category information $c(j_u^{t-1}) \in \{0,1\}^{|C|}$ and apply the linear transformation

$$\hat{j}_u^{t-1} = W_p[j_u^{t-1} \,||\, c(j_u^{t-1})] + b_p. \tag{2}$$

With slight abuse of notation j_u^{t-1} also denotes the one-hot representation. Here, W_p and b_p are trainable parameters and represent weight and bias of a linear layer, '||' represents the concatenation operation and C is the total number of item categories. This item embedding $\hat{j}_u^{t-1}$ is used as an input for an LSTM module based RNN which computes the user-temporal embedding h_u^t via

$$h_u^t = f'(h_u^{t-1}, \hat{j}_u^{t-1}, w). \tag{3}$$

Here, f' represents the LSTM recurrence relation. This final user-temporal representation encodes a user's past behavior.

While h_u^t captures a user's current preferences in general, we also separately capture the relevance of candidate item i with respect to the user's current preferences. For instance, if the user is currently watching action movies, we should capture if the candidate action movie i matches her preferences. Thus, context-specific user-temporal embedding $h_{u,i}^t$ encodes similarity between user-temporal embedding h_u^t and the candidate item embedding $\hat{i}$. Specifically, to capture similarity between the user-temporal embedding and the item embedding, we compute $h_{u,i}^t$ as

$$h_{u,i}^t = W_q[h_u^t \,||\, \hat{i} \,||\, h_u^t \otimes \hat{i}] + b_q, \tag{4}$$

where $\hat{i}$ is the embedding of candidate item i[2] and $\otimes$ represents an element-wise product of two vectors. W_q and b_q are learnable parameters. Figure 3 depicts the architecture of this user-temporal module.

3.3 User-social module

Beyond temporal changes, users are influenced by recent behavior or ratings of trusted friends. Also, influences are not equal among all friends. To model this heterogeneous social influence, we use an attention based aggregation of a user's friends recent past behavior. Figure 4a shows our user-social module. Formally, for user u at time t, the user-social embedding $s_u^t \in \mathbb{R}^D$ is computed as

$$s_u^t = \sum_{v \in F(u)} a_{uv} p_v^t, \tag{5}$$

where $F(u)$ represents social connections of user u, $a_{uv} \in \mathbb{R}$ is the attention weight for friend v and $p_v^t \in \mathbb{R}^D$ is a feature vector representing the recent history of friend v.

Most of the social recommender systems (Zhao et al. 2014; Wang et al. 2018; Pan and Chen 2013) employ a uniform weighting for all social friends when computing influence. However, we argue that this is sub-optimal, particularly for social media, where a user does not trust all friends equally. Indeed, we think modeling of trust is particularly important for online platforms due to large social circles with superficial acquaintance. Thus, we obtain the attention weights a_{uv} from an influence score $e(u, v)$ for each user u and friend v:

$$e(u, v) = \text{LeakyReLU}(W_q[\hat{u} \mid\mid \hat{v}] + b_q), \tag{6}$$

where $\hat{u}, \hat{v} \in \mathbb{R}^D$ are user embeddings for user u and v respectively,[3] while W_q and b_q are learnable parameters. The attention weight a_{uv} is obtained by normalizing the influence score via a soft-max:

$$a_{uv} = \frac{\exp(e(u, v))}{\sum_{v \in F(u)} \exp(e(u, v))}. \tag{7}$$

Each friend v is represented by a dynamic feature vector p_v^t which captures recent past behavior and is computed via

$$\hat{j}_v^{t'} = W_r[j_v^{t'} \mid\mid c(j_v^{t'})] + b_r, \tag{8}$$

$$p_v^t = \text{AGG}(\{\hat{j}_v^{t'} \mid t' < t\}). \tag{9}$$

Here $\hat{j}_v^{t'} \in \mathbb{R}^D$ is the item embedding for item $j_v^{t'}$ clicked by user v at time t'. Each past item rated by the friend before the current timestamp t can be used to compute

[2] $\hat{i} = W[i]$ where i represents the item index and $W \in R^{|\mathcal{I}| X D}$

[3] $\hat{u} = W[u]$ where i represents the item index and $W \in R^{|\mathcal{U}| X D}$

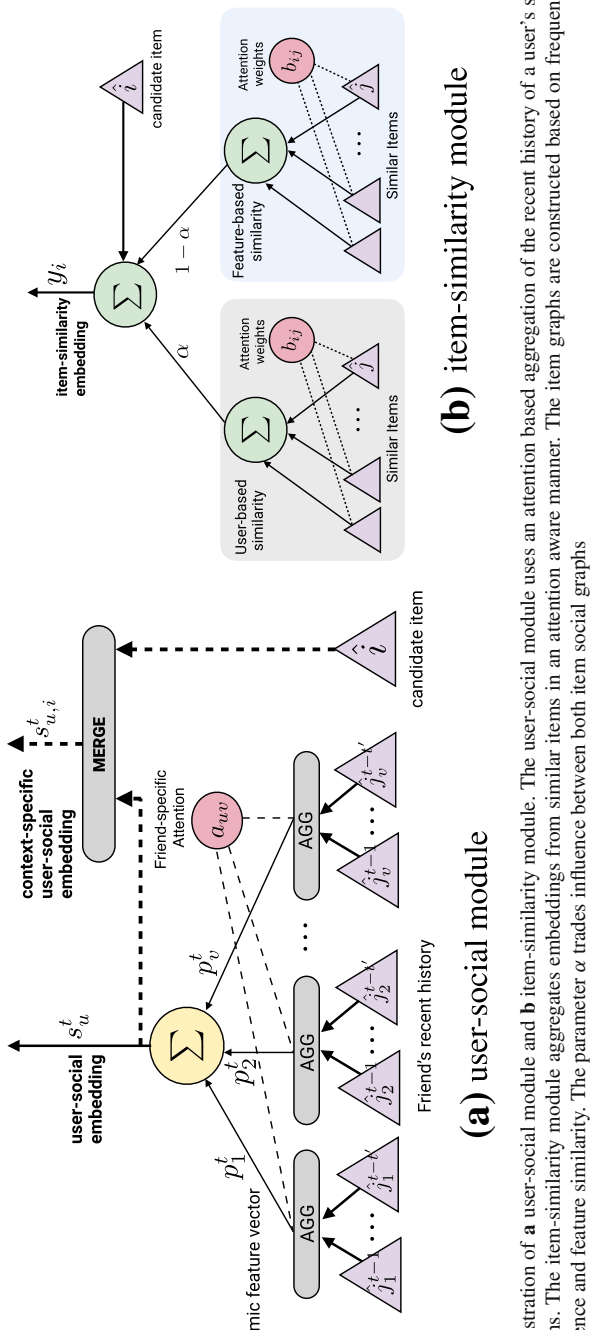

Fig. 4 Illustration of **a** user-social module and **b** item-similarity module. The user-social module uses an attention based aggregation of the recent history of a user's social connections. The item-similarity module aggregates embeddings from similar items in an attention aware manner. The item graphs are constructed based on frequency of co-occurrence and feature similarity. The parameter α trades influence between both item social graphs

the historical profile of friend v. In practice, we found that using the recent past gives similar performance compared to using all previous items. Therefore, we consider only the items from the last t' timesteps of each friend. We aggregate these historical item embeddings using the mean aggregation operation AGG. Note that it is possible to aggregate this information in multiple ways but mean aggregation performed well in our experiments. It is also worthwhile to note that attention weights remain static across time while a user's feature vectors are dynamic.

Similar to the user-temporal module, we also compute a context-specific user-social embedding $s^t_{u,i}$. This embedding captures similarity of the user's social preferences with respect to the candidate item i. Formally,

$$s^t_{u,i} = W_s[s^t_u \,||\, \hat{i} \,||\, s^t_u \otimes \hat{i}] + b_s, \tag{10}$$

where $\hat{i}$ is the item embedding of candidate item i and $\otimes$ is the element-wise product.

3.4 Item-similarity module

Online platforms operate with a large set of items, many of which are rated infrequently. It is consequently hard to construct meaningful representations of items, particularly if the available information is scarce. Also, users are similarly attracted to related items but it is non-trivial to implicitly learn item-item similarity. To address this concern, we propose to construct a similarity aware item embedding based on information available for users who have interacted, e.g., clicked this item, and item features like category information. Figure 4b illustrates our item-similarity module.

In particular, we represent each item $i \in \mathcal{I}$ via an n-hot vector $g_i \in \{0, 1\}^{|\mathcal{U}|}$, where n is the number of users who have interacted with item i while $|\mathcal{U}|$ is the total number of users on the platform. We then compute k-nearest neighbors for each item using cosine similarity between these n-hot vectors. This results in an implicit social network for item i, denoted by $F(i)$. All these items are similar as the same users interacted with them. However, this approach of computing the similarity between items is biased towards popular items with a high user degree. Thus, we construct another item similarity network based on item features. In particular, we compute the network $F'(i)$ based on items that belong to the same category. We randomly connect k' items of the same category in $F'(i)$. In our experiments, we let $k = k'$, i.e., we use the same neighborhood size for both item networks.

We then aggregate both of these item graphs to learn item-similarity embeddings via

$$y_i = \alpha \sum_{j \in F(i)} b_{ij}\hat{j} + (1 - \alpha) \sum_{j \in F'(i)} b_{ij}\hat{j} + \hat{i}, \tag{11}$$

where $\alpha \in [0, 1]$ is a learnable parameter which controls the effect of co-occurring similarity versus category relationship. To compute the attention weights b_{ij}, we follow our earlier approach: similar to the user model, $e(i, j)$ is the influence score for each

item i and its similar item j:

$$e(i, j) = \text{LeakyReLU}\left(W_n[\hat{i} \,||\, \hat{j}] + b_n\right), \tag{12}$$

where $\hat{i}$, $\hat{j}$ are item embeddings for item i and j respectively, and W_n and b_n are learnable parameters. The final attention weight is obtained by normalizing the influence score via a soft-max function:

$$b_{ij} = \frac{\exp(e(i, j))}{\sum_{j \in F(i)} \exp(e(i, j))}. \tag{13}$$

Note that we use the same embeddings to compute attention weights for both graphs.

The final estimated 'item-similarity' module models similarity between items with similar features and frequently co-occurring items. This helps to address the data sparsity in the item space by exploiting item homophily.

Each module learns separates factors that influence user choices in recommender systems. We fuse these factors via a linear operation as detailed in Eq. (1).

3.5 Training

We train all the three modules simultaneously end-to-end in a unified framework using the binary cross-entropy loss:

$$L_\theta = -\sum_{(u,i,t) \in \mathcal{B}} r^t_{ui} \log\left(\hat{r}^t_{ui}\right) + \left(1 - r^t_{ui}\right) \log\left(1 - \hat{r}^t_{ui}\right),$$

where θ represents all the learnable parameters in the model and $\mathcal{B}$ is the currently sampled mini-batch. Specifically, for each sample $(u, i, t) \in \mathcal{B}$, we also obtain user u's friends, $F(u)$ along with the corresponding item graph of the item i, i.e., $F(i)$ and $F'(i)$.

As we are dealing with implicit feedback, we don't have any negative samples. Thus, in each iteration of the training process, for every observed user-item interaction at time t, $(u, i, t) \in \mathcal{B}$, we sample m unobserved items for user u. Similar to Song et al. (2019), we assign a weight of $1/m$ for each negative instance to provide a weak negative signal. This is done as each unobserved item does not necessarily mean that the user will not interact with this item in the future. For our experiments, we set $m = 5$.

Parameter Settings. For all three modules, we use an embedding size $D = 32$ for all user and item embeddings. We also initialize the embedding matrix using a Gaussian distribution with a mean of 0 and a standard deviation of 0.01. For the user-social and item-similarity modules, in each iteration, we subsample a set of friends F at each timestamp for a user (item) instead of considering all friends. This sampling has two benefits: (1) it avoids overfitting by introducing random noise in the social module; and (2) it is computationally more tractable. We use a sample size of 10 for both user

$F(u)$ and item $F(i)$, $F'(i)$ social graphs. If a user has less than 10 friends, we pad the remainder with zeros. We set a friend's history length, $t' = 3$ in the user-social module. We will study the effect of this and other hyper-parameters subsequently.

For the user-temporal and user-social module, we set the length of the historical sequence of user interactions to 30 for all users. For users with a history length of less than 30, we utilize all the available interactions. We also study the effect of this sequence length on our results in the next section. We use the Adam optimizer for training our model and perform a grid search over {0.1, 0.01, 0.001} for a suitable learning rate on the validation set. The validation set is used for tuning the model hyper-parameters, e.g., learning rate, embedding dimension, sample size of a user's friends, etc. We report results on the test set for the best performance model on the validation set. We also use dropout along with gradient clipping with a value of 0.25 and L2 regularization on the user and item embeddings to avoid overfitting.

4 Experiments

In this section, we first describe the datasets we use to evaluate the proposed approach, followed by the evaluation setup and competing state-of-the-art baselines. We then analyze the performance of our model, followed by ablation studies on different modules and hyper-parameters. Finally, we analyze the importance of each module using the learned weight parameters.

4.1 Datasets

We use benchmark datasets available from three online social review platforms: Ciao, Epinions, and CiaoDVD.

Ciao is a product review website where users provide reviews along with ratings for products ranging across a variety of categories. This dataset was crawled by Tang et al. (2012) and contains rating information along with the creation timestamp given up to May 2011. Users also establish directed trust relations with other users on the platform and add them to their 'Circle of Trust.' Epinions is another popular online consumer review website like Ciao (Tang et al. 2012). However, this dataset is longer spanning a decade from Aug. 1999 to Nov. 2013. This dataset also contains trust relations between users.[4] CiaoDVD is a movie review dataset of DVDs crawled from dvd.ciao.co.uk in December 2013. This dataset contains user reviews of movies accompanied by their overall rating. It also contains directed trust relations between users.[5]

As we are dealing with implicit feedback (user-item interaction data), we convert all the observed interactions, i.e., ratings, into positive instances. For both datasets, we only keep users with at least five rated items. The final data statistics are summarized in Table 1. Epinions is by far the largest dataset.

[4] Both Ciao and Epinions datasets are available at www.cse.msu.edu/~tangjili/trust.html.

[5] Dataset available from www.librec.net/datasets.html.

Table 1 Dataset statistics

Dataset	#Users	#Items	#Interactions	#Trusts	#Cat	μ(len)	Social density	Data density
Ciao	1653	16,862	26,190	32,955	6	15.84	1.21%	0.09%
Epinions	22,143	296,278	464,249	83,363	28	20.97	0.02%	0.01%
CiaoDVD	2609	16,122	32,054	8926	17	12.29	0.10%	0.08%

#Cat refers to the number of categories while μ(len) denotes the average history length of users. Ciao has the most dense social network while Epinions has the most sparse user-item interaction data

4.2 Evaluation protocol

We split each user sequence into training, validation, and test set. For each user, the most recent item is held out for testing. The second most recent item is held out for validation while we train the model on the remainder of the sequence.

We evaluate model performance on the test set via two widely used evaluation metrics: HitRate@10 (HR@10) and Area under Curve (AUC). HitRate@10 is computed as follows:

$$HR@10 = \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} \mathbb{1}(L_{u,i_u^t} < 10), \tag{14}$$

where i_u^t is the ground truth item that user u clicked at test time t, L_{u,i_u^t} is the rank of the ground truth item in the predicted ranked list, and $\mathbb{1}(.)$ is the indicator function. The HR@10 metric checks if the ground truth item is present in the top-10 ranking. In contrast, AUC measures the rank of the test item in the predicted ranked list. This is a harder metric as it takes into account the exact position of the ground truth item. Formally,

$$AUC = \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} \frac{1}{|\mathcal{I}|} \sum_{j \in \mathcal{I}} \mathbb{1}(L_{u,i_u^t} > L_{u,j}). \tag{15}$$

where $L_{u,j}$ is the rank of item j in the predicted ranked list for user u. (Cai et al. 2017). Due to the sparsity of user-item interaction data (number of items is huge) and computational feasibility, following He et al. (2016), HitRate@10 is reported on a sample subset of negative items instead of the whole set. For each user-item pair in the test set, we sample 99 negative items that the user has not rated before. Similarly, for AUC computation, we sample a set of 5,000 unobserved items for each user for both our model and the baselines.

4.3 Baselines

We compare to a large variety of state-of-the-art collaborative filtering (CF), temporal, social, and socio-temporal recommenders:

- BPR-MF (Rendle et al. 2009) is a classic matrix factorization model that uses a pairwise ranking loss.
- NeuMF (He et al. 2017) is a recently proposed CF-based model with neural architecture. It merges matrix factorization and multi-layer perceptron modules to predict item ranking.
- NAIS (He et al. 2018) is an item-to-item CF-based model which employs user-specific attention between items to identify similar items.[6]
- TBPR (Wang et al. 2016) extends the BPR model to capture strong and weak ties separately in a user's social network in order to distinguish the degree of influence of a user's connections on her behavior.
- SERec (Wang et al. 2018) is a state-of-the-art CF-based social recommender model that augments collaborative filtering with social exposure through regularization and boosting.
- DiffNet (Wu et al. 2019a) is a state-of-the-art graph convolution-based social recommender that propagates user influence through their social network. We use the item category information for item feature vectors in the model .[7]
- GraphRec (Fan et al. 2019) is another recently proposed social recommender that uses graph convolution networks to incorporate information from both a user-item and a user's social graph. The original model was proposed for rating prediction, and we adopt it for our item ranking task. For that purpose, we change the loss function to a log loss and augment the training data with randomly sampled negative items following other ranking prediction models (He et al. 2017).
- SASRec (Kang and McAuley 2018) is a state-of-the-art model for the temporal recommendation that uses a self-attentive module to capture the long-term interests of a user.[8]
- SR-GNN (Wu et al. 2019b) is a graph neural network-based temporal recommender that aggregates information from all previous timesteps of the user.[9]
- SPMC (Cai et al. 2017) is an extension of a Markov chain based Rendle et al. (2010) model which captures both temporal and social dynamics for recommendation.[10]
- ARSE (Sun et al. 2018) is a state-of-the-art social session recommendation model. It employs a session LSTM module to model change in user preferences across sessions while incorporating social influence as an input to the LSTM module.[11] Similar to their paper, we divide the training dataset into monthly intervals and predicted for the test item in the next session.[12]
- DGRec (Song et al. 2019): Another social session recommender method that uses graph attention networks to merge preferences of social neighbors from the

[6] github.com/AaronHeee/Neural-Attentive-Item-Similarity-Model.

[7] github.com/PeiJieSun/diffnet.

[8] github.com/kang205/SASRec.

[9] github.com/CRIPAC-DIG/SR-GNN.

[10] github.com/cwcai633/SPMC.

[11] github.com/DeepGraphLearning/RecommenderSystems/.

[12] We also experimented with constraining each training session to comprise of just a single item, but that resulted in slightly worse performance.

previous session with a user's preference in the current session. We use similar month-wise intervals to denote each training session.[13]

BPR-MF, NeuMF, and NAIS are Collaborative Filtering models, while TBPR and SERec are social recommenders. Diffnet and GraphRec are graph convolution-based social recommenders. SASRec and SR-GNN are temporal recommender models. We omit a comparison with Fossil and GRU (Hidasi et al. 2016) as they are RNN based temporal recommendation methods. The results for our user-temporal module are equivalent to their models. SPMC is most similar to our model as it also models temporal behavior and socio-temporal influence. However, it is a shallow model as it extends a Markov chain to model linear dependence. ARSE and DGRec are also socio-temporal recommenders proposed for session recommendations.

We use the Adam optimizer for our models with a batch size of 256 for Ciao, CiaoDVD, and 1024 for Epinions. We used the implementation provided by the RecQ python library for BPR-MF,[14] NeuMF, TBPR, and SERec models. We used the authors' implementation for all the other models. Specifically, for ARSE and GraphRec, we wish to thank the authors as they generously shared their implementation. We keep the embedding dimension $D = 32$, for all the baselines to be comparable with our model. We report the best result of the baselines using either hyper-parameters based on the authors' specifications or values, which performed better on our validation set.

4.4 Performance analysis

Table 2 details a comparison of HR@10 and AUC for our model and the state-of-the-art baselines on all three datasets. We report mean results over five runs. Our model substantially outperforms all the baselines on the AUC metric by at least around 14% for Epinions, and 18% for Ciao while performing similarly for CiaoDVD. On the Ciao and Epinions dataset, our FuseRec model also improves the HR@10 metric by at least 4%. Note that each of the baselines models one or a subset of three factors (temporal, socio-temporal, and item similarity), while our FuseRec model considers all factors jointly. Our user-social module outperforms its respective baselines for HR@10 metric while the item-similarity module outperforms classical CF models on AUC metric for CiaoDVD and Epinions dataset. The user-temporal module performs comparably to the other temporal recommenders. Further, the combined FuseRec model considerably outperforms the individual components. This improvement indicates that our model is effective at combining the individual modules, each capturing a distinct factor affecting a user's preference.

Amongst our proposed modules, for the HR@10 metric, the user-social module performs best for Ciao and CiaoDVD, while the user-temporal module performs best for Epinions. This difference can be attributed to the fact that Ciao and CiaoDVD have much denser (100 and 10 times respectively) social networks than the Epinions dataset. Also, Epinions has the longest mean user history length among the three

[13] We also evaluated other intervals, but they all performed similarly.

[14] github.com/Coder-Yu/RecQ.

Table 2 Comparison of results of our model to state-of-the-art baselines and variants of our own model on three datasets.

Models	Ciao		Epinions		CiaoDVD	
	HR@10	AUC	HR@10	AUC	HR@10	AUC
BPR-MF (2009)	0.297±9.7e−3	0.630±5.4e−3	0.509±1.2e−3	0.731±0.7e−3	0.517±5.4e−3	0.759±4.3e−3
NeuMF (2017)	0.342±3.1e−3	0.570±2.4e−3	0.470±38.2e−3	0.709±15.6e−3	**0.551**±2.8e−3	0.760±4.2e−3
NAIS (2018)	0.241±12.9e−3	0.626±12.8e−3	0.510±4.4e−3	0.723±2.0e−3	0.487±1.5e−3	0.742±0.6e−3
SERec (2018)	0.295±7.0e−3	0.550±17.2e−3	0.421±4.9e−3	0.613±4.2e−3	0.385±7.0e−3	0.647±7.0e−3
TBPR (2016)	0.322±5.2e−3	0.601±3.9e−3	0.47±23.1e−3	0.717±10.4e−3	0.518±6.5e−3	0.745±5.4e−3
DiffNet (2019a)	0.343±4.2e−3	0.652±3.4e−3	–	–	0.549±25.9e−3	**0.779**±1.8e−3
GraphRec (2019)	0.234±12.9e−3	0.534±5.1e−3	0.452±27.6e−3	0.702±13.4e−3	0.33±82.2e−3	0.708±10.6e−3
SASRec (2018)	0.324±5.7e−3	0.575±0.6e−3	0.508±18.3e−3	0.724±2.4e−3	0.546±5.0e−3	0.761±2.3e−3
SR-GNN (2019b)	0.320±4.4e−3	0.590±3.7e−3	0.509±2.2e−3	0.732±1.6e−3	0.546±1.8e−3	0.759±1.2e−3
SPMC (2017)	0.223±84.8e−3	0.599±26.8e−3	0.483±22.6e−3	0.717±0.7e−3	0.53±40.1e−3	0.758±7.5**e**−3
ARSE (2018)	0.328±16.6e−3	0.583±10.3e−3	0.522±1.8e−3	0.726±1.0e−3	0.527±12.2e−3	0.754±3.5e−3
DGRec (2019)	0.329±9.8e−3	0.610±25.2e−3	0.479±26.9e−3	0.726±7.0e−3	0.521±8.9e−3	0.753±2.7e−3
User-Temporal	0.308±15.9e−2	0.666±16.6e−2	**0.559**±7.6e−3	0.726±9.1e−3	0.535±9.7e−2	0.751± 10.4e−2
User-Social	0.344±15.4e−3	0.637±14.2e−2	0.503±6.5e−3	0.736±17.9e−3	0.551±3.3e−3	0.741±10.0e−2
Item-Similarity	0.214±4.2e−3	0.606±11.0e−3	0.430±3.8e−3	0.758±1.2e−3	0.474±10.4e−3	0.767±2.7e−3
FuseRec	**0.355**±16.5e−3	**0.745**±7.9e−3	0.549± 14.3e−3	**0.834**±20.0e−3	0.538±7.8e−3	0.774±4.9e−3

Higher values are better for both metrics. Our model substantially outperforms the baselines that model a subset of three factors. '−' indicates an out of memory error

datasets. This difference further underlines that it is essential to model a variety of factors for an effective recommendation.

On the other hand, for the AUC metric, the item-similarity module outperforms the other modules for Epinions and CiaoDVD. This difference between the metrics could be because the user-social module improves the ranking of items currently popular among a user's social connections. In contrast, the item-similarity module emphasizes items that frequently co-occur in the entire dataset, irrespective of the user preferences. This increases the bias of the item ranking only slightly, as co-occurrence is a weak signal for a specific user (it is averaged across all users). Note that we use randomly initialized user embeddings (instead of user embeddings from the other two modules) to compute the individual performance of the item-similarity module.

Collaborative: CF-based approach BPR-MF performs competitively with the other sophisticated baselines when considering the AUC metric while NeuMF performs competitively for HR@10. This superior performance highlights that classical matrix factorization approaches are still very competitive for the recommendation. The user-specific attention-based NAIS model did not outperform the non-attention based CF models.

Social: Our proposed user-social module outperforms all baseline social recommenders for both metrics (except for the AUC metric on CiaoDVD). Among the baselines, the TBPR model that models the user's social graph with different weights for strong and weak ties performs better than the SERec that considers each friend's contribution equally. This reaffirms our assumption that each friend exerts a different influence on the user. In general, we found the recently proposed SERec to underperform on the three datasets used here. It proposes that social connections have a limited influence on a user's preference. Social influence is weakly modeled through exposure prior on the items. However, we think the low performance indicates the contrary, i.e., social influence plays a significant part in shaping a user's preference.

Our GCN based user-social module is inherently different from the other GCN based baselines, DiffNet, and GraphRec. Our user-social module is time-dependent with dynamic user features while both baselines operate on static features (entire item history). These dynamic features result in superior performance in the HR@10 metric for product review websites, Ciao, and Epinions. However, performance dip for movie review platform, CiaoDVD, shows that the user's movie preferences do not evolve rapidly and longer friend history is needed to accurately estimate social influence.

Further, our architecture also differs: we employ attention between a user and her social connections, whereas GraphRec computes attention based on user history rather than the user itself. Our superior performance supports the difference. The difference is particularly pronounced for Ciao that has a denser friendship network than the other two datasets (see Table 1).

Although the DiffNet model performs the best among the baseline social recommenders, it is unable to scale to our largest dataset, Epinions. Note that we used the authors' implementation, and our largest dataset is bigger than the one reported in their paper. In contrast, the other GCN based model GraphRec performs poorly on all the datasets. Note that for results obtained in their paper, they only consider users with

non-zero social connections and items previously seen in the training data, while we do not make any such assumptions.

Temporal: Our user-temporal module uses information from only the previous timestep for prediction at the current timestep while the baseline temporal recommenders aggregate information from all previous timesteps. However, we argue that information from a distant past is not useful. The comparable performance of our user-temporal module with the temporal baselines confirms our argument. In general, both the temporal recommenders, SASRec and SR-GNN, perform better than social recommenders and are comparable to the best performing BPR baseline on the Epinions and CiaoDVD dataset when using the AUC metric. While they perform competitively with other baselines on HR@10.

Social + Temporal: SPMC, which combines temporal and social influence, performs worse than baselines, which model either of the two factors. This worse performance is expected as it is a shallow model with linear dependence. The other socio-temporal models ARSE and DGRec perform similarly, with ARSE performing slightly better for the HR@10 metric. Note that despite modeling socio-temporal influence, in general, these methods do not outperform best performing social and temporal only baselines. This could be since these models were originally proposed for the session-based recommendation. Sessions are typically defined as a sequence of activities performed by a user in a single visit to the website or platform.

Thus, for the session prediction settings, models work on recommending the next item to consume by the user in the current session. They typically assume either none or limited past session information of the users and assume static user preferences per session. These methods also aggregate item information among each past session, thus, losing the sequential information. In contrast, for our temporal recommendation settings, we model the evolution per item for each user (user-temporal module) and expect a significant change in a user's preference over time.

Specifically, DGRec only models socio-temporal influence based on the friend's last session and ignores the evolution of the user's preference across sessions. In contrast, ARSE models the user's evolution across sessions but aggregates information per session resulting in limited flexibility. Also, it is worthy to note that none of these models exploit similarity in the item space that is used in our proposed model. Thus, the worse performance of these session-based socio-temporal methods indicates that they are not well suited for temporal recommendation owing to information aggregation within sessions.

4.5 Module analysis

We now evaluate the importance of each module (user-temporal, user-social, and item-similarity) in our combined model using learned weights, λ_k. Figure 5 shows the learned magnitude of the weights λ_k for $k \in \{1, \ldots, 4\}$ for each of the S_k (Eq. (1)) scores for all the datasets. Weight λ_1 corresponds to a user-temporal embedding (h^t_u) and contributes most to the final score. This high magnitude is expected as a user's history of interactions play a crucial role in modeling user preferences accurately. λ_2 corresponds to the context-specific user-temporal embedding ($h^t_{u,i}$) and is the second

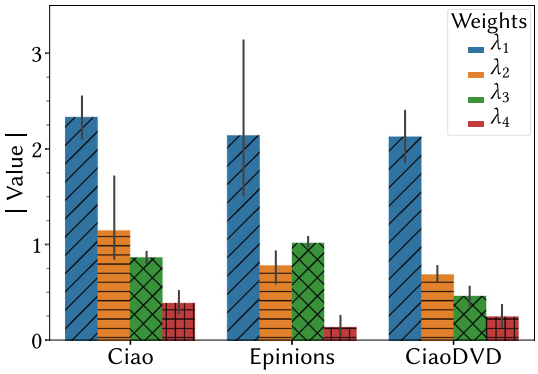

Fig. 5 Learned value of the weights λ_k for $k \in \{1, \ldots, 4\}$ for different scores in our model. Weight λ_1 corresponding to user-temporal embedding contributes most to the final score

Table 3 Performance of our FuseRec model for all three datasets, when removing one module at a time

Model variants	Ciao		Epinions		CiaoDVD	
	HR@10	AUC	HR@10	AUC	HR@10	AUC
{social + item}	0.329	0.574	0.478	0.713	0.526	0.738
{temporal + item}	0.335	0.631	0.557	0.798	0.531	0.743
{temporal + social}	0.327	0.679	0.536	0.736	0.540	0.745
FuseRec	0.355	0.745	0.549	0.834	0.538	0.774

All of these variants perform worse than the combined model

most important factor for Ciao and CiaoDVD datasets. In contrast, λ_3, which corresponds to a user-social embedding (s^t_u) is the second-highest factor for Epinions. This difference indicates that social influence plays an important factor for users in the Epinions dataset while it is slightly less important for the Ciao and the CiaoDVD datasets. λ_4, which corresponds to the context-specific user-social embedding ($s^t_{u,i}$) contributes the least across all datasets. Note that all scores use item-similarity embeddings.

Next, we perform ablation experiments by removing each module at a time from the final model, as shown in Table 3. All variants perform worse than the final FuseRec model for the AUC metric, emphasizing the need for each of the modules. For all the datasets, removing the user-temporal module results in the largest drop in performance. This is expected as a user's history encapsulates a great deal of information for the recommendation. Thus, personalized recommendations fare better than generic ones.

Apart from the user-temporal module, for CiaoDVD, the model without the item-similarity module performs best, while for Epinions and Ciao, the model without the user-social module performs the best. Thus, for the movie reviewing platform CiaoDVD, the preferences of a user's social connections play a significant role in predicting her movie preferences. In contrast, for Epinions and Ciao, both of which contain broad categories of products, items frequently bought together by users are better predictors of purchasing behavior. This seems intuitive as movie preferences are subjective in general, while users tend to believe strangers on online reviewing platforms for products like electronics, furniture, etc.

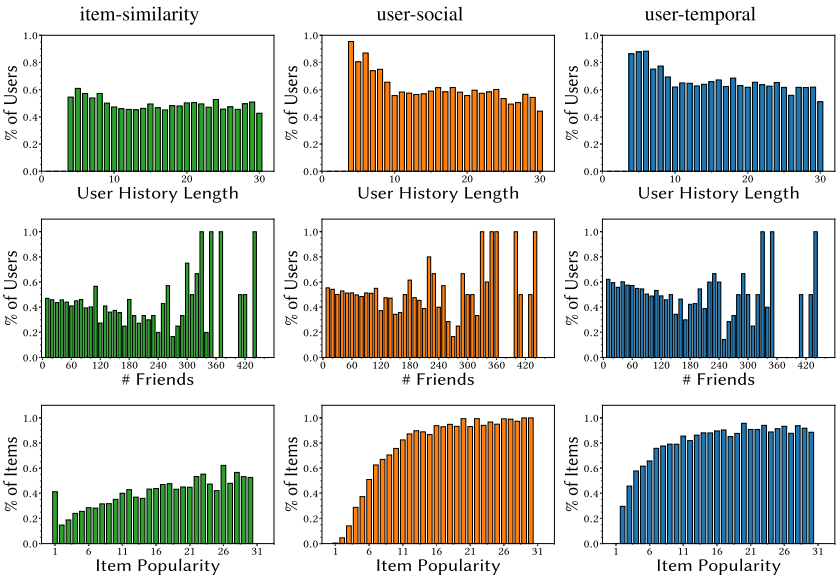

Fig. 6 Fraction of user with correct item predictions on test data by individual modules with different user sequence length and varying size of a user's social connections and fraction of correct item predictions with varying item popularity. The item-similarity module performs better for rare items while the user-social module better models cold-start users

Further, we evaluated individual modules with varying item popularity in the dataset, different sequence length for users, and varying size of a user's social graph for the Epinions dataset as shown in Fig. 6. The user-social module can predict cold-start users better than the user-temporal module as it takes information from a user's social connections into account (top row). Note, the item-similarity module performs uniformly for different user sequence length as it does not contain any user information. Information about the past behavior of social connections (user-social module) improves performance compared to not using social connections (item-similarity module). Note that this effect increases with more friends and does not create noise as our attention model can distinguish between strong and weak connections (middle row). The item-similarity module exploits item homophily in the user space and the feature space. Thus, it can accurately predict rarely reviewed items (in the left section of the figure, close to zero) better than the other two modules (bottom row) (also see item coverage analysis in the Appendix). Thus, our model provides an interpretable way of determining the importance of each of the factors used in our model through multiple experiments.

4.6 Cold-start analysis

Combining different factors in each of the three modules helps to alleviate data sparsity and to predict for cold-start users effectively. To verify this claim, we evaluate our model in cold-start settings using a decreasing number of past available interactions

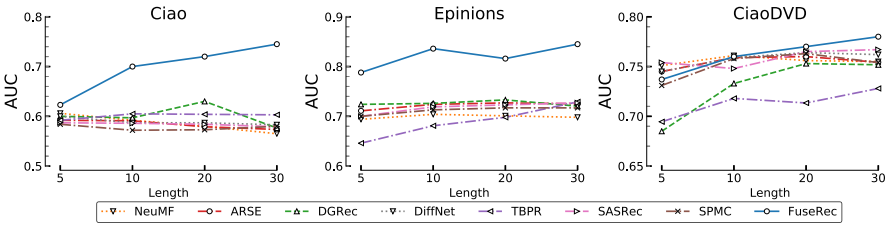

Fig. 7 AUC over different number of past interactions to mimic cold-start settings. Our model consistently outperforms all the social and temporal based baselines

for a user. Figure 7 provides our results with varying user sequence length compared to the baselines. Our model substantially outperforms the baselines modeling a subset of the factors for cold-start settings on all three datasets reaffirming our claim. Also, we observe that our model performs comparably for maximum user history lengths 10 and above, while the performance drops for a very short sequence length of up to 5. This further underlines that a user's temporal history is an essential cue for the recommendation.

4.7 Parameter sensitivity

Table 4 details performance results after removing attention from our user-social and item-similarity modules, respectively. The variant without attention on both the modules performs the worst. Comparing the user-social and item-similarity module: only using attention in the item-similarity module performs better than using attention in the user-social module exclusively. This is expected as the ties between similar items are weaker than explicit social connections between users. Thus, attention results in lower weights for noisy connections in the item-similarity module.

Figure 8 illustrates the performance of our model with different hyper-parameters on the Epinions dataset. For a user's friend's history length t' (Eq. (8)), we observe: using fewer past interactions performs better. However, no change in performance is observed if t' increases to more than the last five interacted items. Increasing the size of the hidden dimension D improves performance initially due to larger model capacity. However, results decline beyond a certain point due to overfitting. For the user and item

Table 4 Performance of our model without attention in the user-social and the item-similarity module

	Ciao		Epinions		CiaoDVD	
	HR@10	AUC	HR@10	AUC	HR@10	AUC
No attn	0.313	0.707	0.536	0.697	0.515	0.742
Userside attn	0.356	0.719	0.531	0.729	0.532	0.747
Itemside attn	0.356	0.737	0.541	0.740	0.536	0.752
FuseRec	0.355	0.745	0.549	0.834	0.538	0.774

Attention in item-similarity module only performs better than only user-social module attention

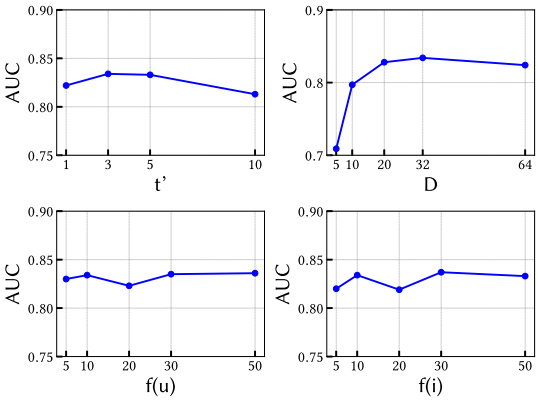

Fig. 8 Ablation results of different model hyper-parameters on Epinions datatset

friend sample size, $F(u)$ (Eq. (5)) and $F(i)$, $F'(i)$ (Eq. (11)) respectively, performance slightly increases with an increased size but plateaus for sample sizes larger than 10.

5 Conclusion

We presented a model that captures temporal changes and socio-temporal influence while taking into account item similarity and later combines them in an interpretable manner. We used an RNN model to capture the evolution of user preferences. We proposed an attention-based user social module that aggregates historical features for all of the user's neighbors. To capture item-based homophily, we proposed an attention-based item module to learn item embeddings using similar items frequently co-occurring in the platform. We compared our approach to a large number of temporal, social, and socio-temporal recommenders on three benchmark datasets. We report an improvement of more than 14% over state-of-the-art baselines on the Ciao and Epinions datasets for AUC metric. Our ablation study shows that each module is essential to capture different factors affecting user behavior in recommender systems. Further, the user-temporal module is the most important factor across all datasets.

We would also like to mention some of the limitations of our approach: 1) there are potentially many other factors affecting user preferences apart from the ones explored in this work; 2) the temporal module is sub-optimal for datasets where the preferences do not evolve rapidly (CiaoDVD in our work), and 3) the item network construction can be computationally demanding. We need to use scalable approaches to compute a KNN graph between items efficiently.

References

Cai C, He R, McAuley J (2017) SPMC: socially-aware personalized markov chains for sparse sequential recommendation. In: Proceedings of the 26th international joint conference on artificial intelligence, IJCAI'17, pp 1476–1482

Cheng C, Yang H, Lyu MR, King I (2013) Where you like to go next: successive point-of-interest recommendation. In: Proceedings of the twenty-third international joint conference on artificial intelligence, IJCAI '13, pp 2605–2611

Fan W, Ma Y, Li Q, He Y, Zhao E, Tang J, Yin D (2019) Graph neural networks for social recommendation. In: The world wide web conference, WWW '19, pp 417–426
He X, He Z, Song J, Liu Z, Jiang YG, Chua TS (2018) Nais: neural attentive item similarity model for recommendation. IEEE Trans Knowl Data Eng 30(12):2354–2366
He R, Fang C, Wang Z, McAuley J (2016) Vista: a visually, socially, and temporally-aware model for artistic recommendation. In: Proceedings of the 10th ACM conference on recommender systems, pp 309–316
He X, Liao L, Zhang H, Nie L, Hu X, Chua TS (2017) Neural collaborative filtering. In: Proceedings of the 26th international conference on world wide web, pp 173–182
Hidasi B, Karatzoglou A, Baltrunas L, Tikk D (2016) Session-based recommendations with recurrent neural networks. In: Proceedings of the international conference on learning representation, ICLR '16
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, RecSys '10, pp 135–142
Kang W, McAuley J (2018) Self-attentive sequential recommendation. In: Proceedings of the 2018 IEEE international conference on data mining (ICDM), IEEE, pp 197–206
Kipf TN, Welling M (2017) Semi-supervised classification with graph convolutional networks. In: Proceedings of the international conference on learning representations, ICLR'17
Krishnan A, Sharma A, Sankar A, Sundaram H (2018) An adversarial approach to improve long-tail performance in neural collaborative filtering. In: Proceedings of the 27th ACM international conference on information and knowledge management, CIKM '18, pp 1491–1494
Ma H, Zhou D, Liu C, Lyu MR, King I (2011) Recommender systems with social regularization. In: Proceedings of the fourth ACM international conference on web search and data mining, WSDM '11, pp 287–296
Mikolov T, Karafiát M, Burget L, Černockỳ J, Khudanpur S (2010) Recurrent neural network based language model. In: INTERSPEECH'10, ISCA, pp 1045–1048
Pan W, Chen L (2013) Gbpr: group preference based bayesian personalized ranking for one-class collaborative filtering. In: International joint conference on artificial intelligence
Rendle S, Freudenthaler C, Gantner Z, Schmidt-Thieme L (2009) Bpr: Bayesian personalized ranking from implicit feedback. In: Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence, UAI '09, pp 452–461
Rendle S, Freudenthaler C, Schmidt-Thieme L (2010) Factorizing personalized markov chains for next-basket recommendation. In: Proceedings of the 19th international conference on World Wide Web, WWW '10, pp 811–820
Sarwar B, Karypis G, Konstan J, Riedl J (2001) Item-based collaborative filtering recommendation algorithms. In: Proceedings of the 10th international conference on World Wide Web, WWW '01, pp 285–295
Song W, Xiao Z, Wang Y, Charlin L, Zhang M, Tang J (2019) Session-based social recommendation via dynamic graph attention networks. In: Proceedings of the twelfth ACM international conference on web search and data mining, pp 555–563
Sun P, Wu L, Wang M (2018) Attentive recurrent social recommendation. In: The 41st international ACM SIGIR conference on research and development in information retrieval, SIGIR '18, pp 185–194
Tang J, Gao H, Liu H, Sarma AD (2012) eTrust: Understanding trust evolution in an online world. In: Proceedings of the 18th ACM SIGKDD international conference on knowledge discovery and data mining, KDD '12, pp 253–261
Tang J, Sun J, Wang C, Yang Z (2009) Social influence analysis in large-scale networks. In: Proceedings of the 15th ACM SIGKDD international conference on knowledge discovery and data mining, KDD '09, pp 807–816
Tang J, Wang K (2018) Personalized top-n sequential recommendation via convolutional sequence embedding. In: Proceedings of the eleventh ACM international conference on web search and data mining, WSDM '18, pp 565–573
van den Berg R, Kipf TN, Welling M (2017) Graph convolutional matrix completion. arXiv preprint arXiv:1706.02263
Veličković P, Cucurull G, Casanova A, Romero A, Liò P, Bengio Y (2018) Graph attention networks. In: International conference on learning representations
Wang X, Lu W, Ester M, Wang C, Chen C (2016) Social recommendation with strong and weak ties. In: Proceedings of the 25th ACM international on conference on information and knowledge management, CIKM '16, pp 5–14

Wang M, Zheng X, Yang Y, Zhang K (2018) Collaborative filtering with social exposure: a modular approach to social recommendation. In: Thirty-second AAAI conference on artificial intelligence

Wu CY, Ahmed A, Beutel A, Smola AJ, Jing H (2017) Recurrent recommender networks. In: Proceedings of the tenth ACM international conference on web search and data mining, WSDM '17, pp 495–503

Wu Y, DuBois C, Zheng AX, Ester M (2016) Collaborative denoising auto-encoders for top-n recommender systems. In: Proceedings of the Ninth ACM international conference on web search and data mining, WSDM '16, pp 153–162

Wu L, Sun P, Fu Y, Hong R, Wang X, Wang M (2019a) A neural influence diffusion model for social recommendation. In: Proceedings of the 42nd international ACM SIGIR conference on research and development in information retrieval, pp 235–244

Wu L, Sun P, Hong R, Ge Y, Wang M (2018) Collaborative neural social recommendation. IEEE transactions on systems, man, and cybernetics: systems pp 1–13

Wu S, Tang Y, Zhu Y, Wang L, Xie X, Tan T (2019b) Session-based recommendation with graph neural networks. In: Proceedings of the AAAI conference on artificial intelligence, vol 33, pp 346–353

Yin H, Cui B, Li J, Yao J, Chen C (2012) Challenging the long tail recommendation. Proc VLDB Endow 896–907

Ying R, He R, Chen K, Eksombatchai P, Hamilton WL, Leskovec J (2018) Graph convolutional neural networks for web-scale recommender systems. In: Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery and data mining, pp 974–983

Zhao T, McAuley J, King I (2014) Leveraging social connections to improve personalized ranking for collaborative filtering. In: Proceedings of the 23rd ACM international conference on information and knowledge management, CIKM '14, pp 261–270