Leveraging Long and Short-term Information in Content-aware Movie Recommendation via Adversarial Training

Wei Zhao, Benyou Wang, Min Yang, Jianbo Ye, Zhou Zhao, Xiaojun Chen, Ying Shen

***Abstract*—Movie recommendation systems provide users with ranked lists of movies based on individual's preferences and constraints. Two types of models are commonly used to generate ranking results: long-term models and session-based models. The long-term based models represent the interactions between users and movies that are supposed to change slowly across time, while the session-based models encode the information of users' interests and changing dynamics of movies' attributes in short terms. In this paper, we propose an *LSIC* model, leveraging Long and Short-term Information for Content-aware movie recommendation using adversarial training. In the adversarial process, we train a generator as an agent of reinforcement learning which recommends the next movie to a user sequentially. We also train a discriminator which attempts to distinguish the generated list of movies from the real records. The poster information of movies is integrated to further improve the performance of movie recommendation, which is specifically essential when few ratings are available. The experiments demonstrate that the proposed model has robust superiority over competitors and achieves the state-of-the-art results.**

***Index Terms*—Top-n movie recommendation, adversarial learning, content-aware recommendation**

I. Introduction

With the sheer volume of online information, much attention has been given to data-driven recommender systems. Those systems automatically guide users to discover products or services respecting their personal interests from a large pool of possible options. Numerous recommendation techniques have been developed. Three main categories of them are: collaborative filtering methods, content-based methods and hybrid methods [1], [2]. In this paper, we aim to develop a method producing a ranked list of n movies to a user at a given moment (top-N movie recommendation) by exploiting both historical user-movie interactions and the content information of movies.

∗ Min Yang is corresponding author.

† First two authors contribute to this work equally.

W. Zhao and M. Yang are with Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen, China. E-mail:{min.yang, wei.zhao}@siat.ac.cn.

B. Wang is with Department of Information Engineering, University of Padova, Padova, Italy. E-mail: wang@dei.unipd.it.

J. Ye is with Department of computer science, Pennsylvania State University, PA, USA. E-mail: jianboye@gmail.com.

Z. Zhao is with School of Computer Science, Zhejiang University, Hangzhou, China. E-mail: zhaozhou@zju.edu.cn.

X. Chen is with College of Computer Science and Software, Shenzhen University, Shenzhen, China (e-mail: xjchen@szu.edu.cn).

Y. Shen is with School of Electronics and Computer Engineering, Peking University Shenzhen Graduate School, Shenzhen, China (e-mail: shenying@pkusz.edu.cn).

Matrix factorization (MF) [3] is one of the most successful techniques in the practice of recommendation due to its simplicity, attractive accuracy and scalability. It has been used in a broad range of applications such as recommending movies, books, web pages, relevant research and services. The matrix factorization technique is usually effective because it discovers the latent features underpinning the multiplicative interactions between users and movies. Specifically, it models the user preference matrix approximately as a product of two lower-rank latent feature matrices representing user profiles and movie profiles respectively.

Despite the appeal of matrix factorization, this technique does not explicitly consider the temporal variability of data [4]. Firstly, the popularity of a movie may change over time. For example, movie popularity booms or fades, which can be triggered by external events such as the appearance of an actor in a new movie. Secondly, users may change their interests and baseline ratings over time. This is well established by previous work [5], [4]. A user might like a particular actor, might discover the intricacies of a specific genre, or her interest in a particular show might wane, due to maturity or a change in lifestyle. For instance, a user who tended to rate an average movie as "4 stars", may now rate such a movie as "3 stars". Recently, recurrent neural network (RNN) [6] has gained significant attention by considering such temporal dynamics for both users and movies and achieved high recommendation quality [7], [4]. The basic idea of these RNN-based methods is to formulate the recommendation as a sequence prediction problem. They take the latest observations as input, update the internal states and make predictions based on the newly updated states. As shown in [8], such prediction based on short-term dependencies is likely to improve the recommendation diversity.

More recent work [4] reveals that combing matrix factorization based and RNN based recommendation approaches can achieve good performance for the reasons that are complementary to each other. Specifically, the matrix factorization based approaches make movie predictions based on users' long-term interests that change very slowly with respect to time. On the contrary, the RNN based recommendation approaches predict which movie will the user consume next, respecting the dynamics of users' behaviors and movies' attributes in the short term. It therefore motivates us to devise a joint approach that takes advantage of both matrix factorization and RNN, exploiting both long-term and short-term associations among users and movies.

Furthermore, most existing recommender systems take into account only the users' past behaviors when making recommendation. Compared with tens of thousands of movies in the corpus, the historical rating set is too sparse to learn a well-performed model. It is desirable to exploit the auxiliary information of movies (e.g., posters, movie descriptions, user reviews) for movie recommendation. For example, movie posters reveal a great amount of information to understand movies and users, as demonstrated in [9]. Such a poster is usually the first contact that a user has with a movie, and plays an essential role in the user's decision to watch it or not. When a user is watching the movie presented in cold, blue and mysterious visual effects, he/she may be interested in receiving recommendations for movies with similar styles, rather than others that are with the same actors or subject [9]. These visual features of movies are usually captured by the corresponding posters.

In this paper, we propose a novel LSIC model, which leverages **L**ong and **S**hort-term **I**nformation in **C**ontent-aware movie recommendation using adversarial training. The LSIC model employs an adversarial framework to combine the MF and RNN based models for the top-N movie recommendation, taking the best of each to improve the final recommendation performance. In the adversarial process, we simultaneously train two models: a generative model G and a discriminative model D. In particular, the generator G takes the user u_i and time t as input, and predicts the recommendation list for user i at time t based on the historical user-movie interactions. We implement the discriminator D with a siamese network that incorporates long-term and session-based ranking model in a pair-wise scenario. The two point-wise networks of siamese network share the same set of parameters. The generator G and the discriminator D are optimized with a minimax two-player game. The discriminator D tries to distinguish the real high-rated movies in the training data from the recommendation list generated by the generator G, while the training procedure of generator G is to maximize the probability of D making a mistake. Thus, this adversarial process can eventually adjust G to generate plausible and high-quality recommendation list. In addition, we integrate poster information of movies to further improve the performance of movie recommendation, which is specifically essential when few ratings are available.

We summarize our main contributions as follows:

- To the best of our knowledge, we are the first to use GAN framework to leverage the MF and RNN approaches for top-N recommendation. This joint model adaptively adjusts how the contributions of the long-term and short-term information of users and movies are mixed together.
- We propose hard and soft mixture mechanisms to integrate MF and RNN. We use the hard mechanism to calculate the mixing score straightforwardly and explore several soft mechanisms to learn the temporal dynamics with the help of the long-term profiles.
- Our model uses reinforcement learning to optimize the generator G for generating highly rewarded recommendation list. Thus, it effectively bypasses the non-differentiable task metric issue by directly performing policy gradient update.
- We explore the context-aware information (movie posters) to alleviate the cold-start problem and further improve the performance of movie recommendation. The release of the collected posters would push forward the research of integrating context-aware information in movie recommender systems.
- To verify the effectiveness of our model, we conduct extensive experiments on two widely used real-life datasets: Netflix Prize Contest data and MovieLens data. The experimental results demonstrate that our model consistently outperforms the state-of-the-art methods.

The rest of the paper is organized as follows. In Section II, we review the related work on recommender systems. Section III presents the proposed adversarial learning framework for movie recommendation in detail. In Section IV, we describe the experimental data, implementation details, evaluation metrics and baseline methods. The experimental results and analysis are provided in Section V. Section VI concludes this paper.

II. Related Work

Recommender system is an active research field [10], [11]. The authors of [1], [2] describe most of the existing techniques for recommender systems. In this section, we briefly review the following major approaches for recommender systems that are related to our work.

a) Matrix factorization for recommendation: Modeling the long-term interests of users, the matrix factorization method and its variants have grown to become dominant in the literature [12], [13], [3], [14], [15]. In the standard matrix factorization, the recommendation task can be formulated as inferring missing values of a partially observed user-item matrix [3]. The Matrix Factorization techniques are effective because they are designed to discover the latent features underlying the interactions between users and items. [16] suggested the Maximum Margin Matrix Factorization (MMMF), which used low-norm instead of low-rank factorizations. [17] presented the Probabilistic Matrix Factorization (PMF) model that characterized the user preference matrix as a product of two lower-rank user and item matrices. The PMF model was especially effective at making better predictions for users with few ratings. [15] proposed a new MF method which considers the implicit feedback for on-line recommendation. In [15], the weights of the missing data were assigned based on the popularity of items. To exploit the content of items and solve the data sparsity issue in recommender systems, [9] presented a movie recommendation model which used additional visual features (e.g.. posters and still frames) to further improve the performance of movie recommendation.

b) Recurrent neural network for recommendation: These traditional MF methods for recommendation systems are based on the assumption that the user interests and movie attributes are near static, which is however not consistent with reality. [18] discussed the effect of temporal dynamics in recommender systems and proposed a temporal extension of the SVD++ (called TimeSVD++) to explicitly model the temporal bias in data. However, the features used in TimeSVD++

were hand-crafted and computationally expensive to obtain. Recently, there have been increasing interests in employing recurrent neural network to model the temporal dynamics in recommendation systems. For example, [19] applied recurrent neural network (i.e. GRU) to session-based recommender systems. This work treats the first item a user clicked as the initial input of GRU. Each follow-up click of the user would then trigger a recommendation depending on all of the previous clicks. [20] proposed a recurrent neural network to perform the time heterogeneous feedback recommendation. [21] presented a visual and textural recurrent neural network (VT-RNN), which simultaneously learned the sequential latent vectors of user's interest and captured the content-based representations that contributed to address the cold-start problem. [22] characterized the short- and long-term profile of many collaborative filtering methods, and showed how RNNs can be steered towards better short or long-term predictions. [4] proposed a dynamic model, which incorporated the global properties learned by MF into the recurrent neural network. Different from their work, we use GAN framework to leverage the MF and RNN approaches for top-N recommendation, aiming to generate plausible and high-quality recommendation lists.

c) Generative adversarial network for recommendation: In parallel, previous work has demonstrated the effectiveness of generative adversarial network (GAN) [23] in various tasks such as image generation [24], [25], image captioning [26], and sequence generation[27]. The most related work to ours is [28], which proposed a novel IRGAN mechanism to iteratively optimize a generative retrieval component and a discriminative retrieval component. IRGAN reported impressive results on the tasks of web search, item recommendation, and question answering. Our approach differs from theirs in several aspects. First, we combine the MF approach and the RNN approach with GAN, exploiting the performance contributions of both approaches. Second, IRGAN does not attempt to estimate the future behavior since the experimental data is split randomly in their setting. In fact, they use future trajectories to infer the historical records, which seems not useful in real-life applications. Third, we incorporate poster information of movies to deal with the cold-start issue and boost the recommendation performance. To further improve the performance, we design a siamese network to independently learn the representation for each user and candidate item, and then maximize the distance between the two estimated representations via a margin constraint in pair-wise scenario.

III. Our Model

Suppose there is a sparse user-movie rating matrix R that consists of U users and M movies. Each entry $r_{ij,t}$ denotes the rating of user i on movie j at time step t. The rating is represented by numerical values from 1 to 5, where the higher value indicates the stronger preference. Instead of predicting the rating of a specific user-movie pair as is done in [29], [30], the proposed LSIC model aims to provide users with ranked lists of movies (top-N recommendation) [31].

In this section, we elaborate each component of LSIC model for content-aware movie recommendation. The main notations of this work are summarized in Table I for clarity. The LSIC model employs an adversarial framework to combine the MF and RNN based models for the top-N movie recommendation. The overview of our proposed architecture and its data-flow are illustrated in Figure 1. In the adversarial process, we simultaneously train two models: a generative model G and a discriminative model D.

TABLE I
NOTATION LIST. WE USE SUPERSCRIPT u TO ANNOTATE PARAMETERS RELATED TO A USER, AND SUPERSCRIPT m TO ANNOTATE PARAMETERS RELATED TO A MOVIE.

R	the user-movie rating matrix
U, M	the number of users and movies
r_{ij}	rating score of user i on movie j
$r_{ij,t}$	rating score of user i on movie j at time t
$\mathbf{e}_i^u$	MF user factors for user i
$\mathbf{e}_j^m$	MF movie factors for movie j
b_i^u	bias of user i in MF and RNN hybrid calculation
b_j^m	bias of movie j in MF and RNN hybrid calculation
$\mathbf{h}_{i,t}^u$	LSTM hidden-vector at time t for user i
$\mathbf{h}_{j,t}^m$	LSTM hidden-vector at time t for movie j
$\mathbf{z}_{i,t}^u$	the rating vector of user i at time t (LSTM input)
$\mathbf{z}_{j,t}^m$	the rating vector of movie j at time t (LSTM input)
α_t^i	attention weight of user i at time t
β_t^j	attention weight of movie j at time t
m_+	index of a positive (high-rating) movie drawn from the entire positive movie set $\mathcal{M}_+$
m_-	index of a negative (low-rating) movie randomly chosen from the entire negative movie set $\mathcal{M}_-$
$m_{g,t}$	index of an item chosen by generator G at time t

A. Matrix Factorization (MF)

The MF framework [17] models the long-term states (global information) for both users ($\mathbf{e}^u$) and movies ($\mathbf{e}^m$). In its standard setting, the recommendation task can be formulated as inferring missing values of a partially observed user-movie rating matrix R. The formulation of MF is given by:

$$\underset{\mathbf{e}^u,\mathbf{e}^m}{\operatorname{argmin}} \sum_{i,j} I_{ij} \left(r_{ij} - \rho\left((\mathbf{e}_i^u)^T \mathbf{e}_j^m\right)\right)^2 + \lambda^u \|\mathbf{e}^u\|_F^2 + \lambda^m \|\mathbf{e}^m\|_F^2 \tag{1}$$

where $\mathbf{e}_i^u$ and $\mathbf{e}_j^m$ represent the user and movie latent factors in the shared d-dimension space respectively. r_{ij} denotes the user i's rating on movie j. I_{ij} is an indicator function and equals 1 if $r_{ij} > 0$, and 0 otherwise. λ^u and λ^m are regularization coefficients. The $\rho(\cdot)$ is a logistic scoring function that bounds the range of outputs.

In most recommender systems, matrix factorization techniques [3] recommend movies based on estimated ratings. Even though the predicted ratings can be used to rank the movies, it is known that it does not provide the best prediction for the top-N recommendation since minimizing the objective function – the squared errors – does not perfectly align with the goal of optimizing the ranking order. A close prediction on the rating score of one item cannot guarantee to recover the relative preference relationship between two pairwised items. [31] also shows that the models well designed for rating prediction may lead to unsatisfactory performance on the task of item ranking.

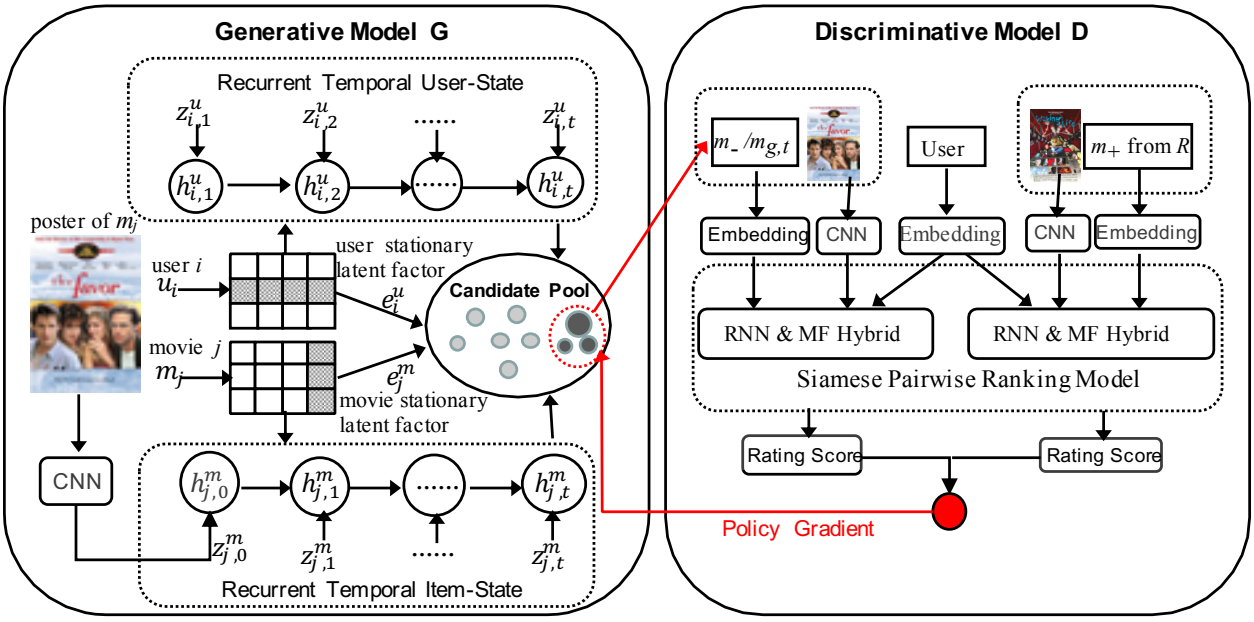


Fig. 1. The overall architecture of the proposed LSIC model, which leverages both RNN and MF models via an generative adversarial framework. LSIC consists of a generative model G and a discriminative model D. In the adversarial process, we train the generative model G to predict the recommendation list via the RNN and MF Hybrid model. We also train the discriminative model D to distinguish the generated recommendation list from the real ones in the training data. The generative model and the discriminative model are optimized with a minimax two-player game. Here, m_+ is a positive (high-rating) movie chosen from the training data, m_- is a negative movie randomly chosen from the entire negative (low-rating) movie space, $m_{g,t}$ is the generated movie by G given time t. $\mathbf{z}_{i,t}^u$ and $\mathbf{z}_{j,t}^m$ represent the rating vector of user i and movie j given time t. $\mathbf{h}_{i,t}^u$ and $\mathbf{h}_{j,t}^m$ denote the hidden states of LSTM for user i and movie j at time step t

In this paper, we have chosen a basic matrix factorisation model for ranking prediction (top-N recommendation) directly, and it would be straightforward to replace it with more sophisticated models such as Probabilistic Matrix Factorization (PMF) [17], whenever needed. For example, we may explore more advanced MF based methods to alleviate the overfitting problem when dealing with the severe sparse training data.

B. Recurrent Neural Network (RNN)

The RNN based recommender system focuses on modeling session-based trajectories instead of global (long-term) information [4]. It predicts future behaviors and provides users with a ranking list given the users' past history. The main purpose of using RNN is to capture time-varying state for both users and movies. Particularly, we use LSTM cell as the basic RNN unit. Each LSTM unit at time t consists of a memory cell c_t, an input gate i_t, a forget gate f_t, and an output gate o_t. These gates are computed from previous hidden state $\mathbf{h}_{t-1}$ and the current input $\mathbf{x}_t$:

$$[f_t, i_t, o_t] = \text{sigmoid}(W[\mathbf{h}_{t-1}, \mathbf{x}_t]) \tag{2}$$

The memory cell c_t is updated by partially forgetting the existing memory and adding a new memory content $\mathbf{l}_t$:

$$\mathbf{l}_t = \tanh(V[\mathbf{h}_{t-1}, \mathbf{x}_t]) \tag{3}$$

$$\mathbf{c}_t = f_t \odot \mathbf{c}_{t-1} + i_t \odot \mathbf{l}_t \tag{4}$$

Once the memory content of the LSTM unit is updated, the hidden state at time step t is given by:

$$\mathbf{h}_t = o_t \odot \tanh(\mathbf{c}_t) \tag{5}$$

For simplicity of notation, the update of the hidden states of LSTM at time step t is denoted as $\mathbf{h}_t = \text{LSTM}(\mathbf{h}_{t-1}, \mathbf{x}_t)$.

Here, we use $\mathbf{z}_{i,t}^u \in \mathbb{R}^U$ and $\mathbf{z}_{j,t}^m \in \mathbb{R}^M$ to represent the rating vector of user i and movie j given time t respectively. Both $\mathbf{z}_{i,t}^u$ and $\mathbf{z}_{j,t}^m$ serve as the input to the LSTM layer at time t to infer the new states of the user and the movie:

$$\mathbf{h}_{i,t}^u = \text{LSTM}(\mathbf{h}_{i,t-1}^u, \mathbf{z}_{i,t}^u) \tag{6}$$

$$\mathbf{h}_{j,t}^m = \text{LSTM}(\mathbf{h}_{j,t-1}^m, \mathbf{z}_{j,t}^m) \tag{7}$$

where $\mathbf{h}_{i,t}^u$ and $\mathbf{h}_{j,t}^m$ denote the hidden states of LSTM for user i and movie j at time step t respectively.

CNN Encoder for Poster Information In this work, we explore the potential of integrating posters of movies to boost the performance of movie recommendation. Inspired by the recent advances of deep convolutional neural networks in computer vision [32], [33], the poster is mapped to the same space of the movie by using a deep residual network [34]. Deep Residual Network (ResNet) makes it possible to train up to hundreds or even thousands of layers and achieves the state-of-the-art performance in many tasks of computer vision community. ResNet uses an end-to-end strategy to naturally integrate the multi-layer features. These feature annotations are greatly enriched with the stacked layers. Rather than expecting every few stacked layers fit an underlying mapping directly, the residual network makes these staked layers fit a residual mapping. More concretely, we encode each image into a FC-2k feature vector with Resnet-101 (101 layers) [34], resulting in a 2048-dimensional vector representation. The poster P_j of movie j is only inputted once, at $t = 0$, to inform the movie LSTM about the poster content:

$$\mathbf{z}_{j,0}^{m} = CNN(P_j). \tag{8}$$

C. RNN and MF Hybrid

Although the user and movie states are time-varying, there are also some stationary components that capture the fixed properties, e.g., the profile of a user and the genre of a movie [4]. To leverage both the long- and short-term information of users and movies, we supplement the dynamic user and movie representations $h_{i,t}^{u}$ and $h_{j,t}^{m}$ with the stationary ones e_i^u and e_j^m, respectively. Similar to [4], the rating prediction function is defined as:

$$r_{ij,t} = g(\mathbf{e}_i^u, \mathbf{e}_j^m, \mathbf{h}_{i,t}^u, \mathbf{h}_{j,t}^m) \tag{9}$$

where $g(.)$ is a score function, $\mathbf{e}_i^u$ and $\mathbf{e}_j^m$ denote the global latent factors of user i and movie j learned by Eq. (1); $\mathbf{h}_{i,t}^u$ and $\mathbf{h}_{j,t}^m$ denote the hidden states at time step t of two RNNs learned by Eq. (6) and Eq. (7) respectively. In this work, we study four strategies to calculate the score function g, integrating MF and RNN. The details are described below.

a) LSIC-V1: The first strategy simply combines the results from MF and RNN linearly, inspired by [4]. However, the scores from MF and RNN are calculated independently and there is no interaction between their calculations. Mathematically, the combination of MF and RNN is defined as follows:

$$r_{ij,t} = g(\mathbf{e}_i^u, \mathbf{e}_j^m, \mathbf{h}_{i,t}^u, \mathbf{h}_{j,t}^m) = \frac{1}{1+\exp(-\mathbf{s})} \tag{10}$$

$$\mathbf{s} = \mathbf{e}_i^u \cdot \mathbf{e}_j^m + \mathbf{h}_{i,t}^u \cdot \mathbf{h}_{j,t}^m + b_i^u + b_j^m \tag{11}$$

where b_i^u and b_j^m are the biases of user i and movie j; $\mathbf{h}_{i,t}^u$ and $\mathbf{h}_{j,t}^m$ are computed by Eq. (6) and Eq. (7).

In fact, LSIC-V1 does not exploit the global factors in learning the temporal dynamics. In this paper, we also design three soft mixture mechanisms and provide three strategies to account for the global factors $\mathbf{e}_i^u$ and $\mathbf{e}_j^m$ in learning $\mathbf{h}_{i,t}^u$ and $\mathbf{h}_{j,t}^m$, as described below (i.e., LSIC-V2, LSIC-V3 and LSIC-V4).

b) LSIC-V2: LSIC-V1 is not stable when the hidden state of RNN is initialized randomly [35]. Motivated by this observation, we propose LSIC-V2 which uses user and item representations learned by MF to initialize the hidden neurons of RNN. In particular, we use the latent factors of user i ($\mathbf{e}_i^u$) and movie j ($\mathbf{e}_j^m$) pre-trained by MF model to initialize the hidden states of the LSTM cells $\mathbf{h}_{i,0}^u$ and $\mathbf{h}_{j,0}^m$ respectively, as depicted in Figure 3(b).

c) LSIC-V3: Since the RNN has position bias to the input and ignores the global (long-term) information, we use the user and item representations learned by MF as the extra input of the RNN. As shown in Figure 3(c), we extend LSIC-V2 by treating $\mathbf{e}_i^u$ (for user i) and $\mathbf{e}_j^m$ (for movie j) as the static context vectors, and feed them as an extra input into the computation of the temporal hidden states of users and movies by LSTM. At each time step, the context information assists the inference of the hidden states of LSTM model.

d) LSIC-V4: The last strategy is inspired by the recent success of attention mechanism in natural language processing and computer vision [36], [37]. A user/item profile not only depends on itself, but also is affected by its neighbors. Thus, we design an attention mechanism to make use of the dynamic item and user representations learned by RNN to learn a weight for each user representation and item representation of MF. The mixing score function at time t can be reformulated by:

$$r_{ij,t} = g(\mathbf{e}_i^u, \mathbf{e}_j^m, \mathbf{h}_{i,t-1}^u, \mathbf{h}_{j,t-1}^m, \mathbf{c}_{i,t}^u, \mathbf{c}_{j,t}^m) = \frac{1}{1+\exp(-\mathbf{s})} \tag{12}$$

$$\mathbf{s} = \mathbf{e}_i^u \cdot \mathbf{e}_j^m + \mathbf{h}_{i,t}^u \cdot \mathbf{h}_{j,t}^m + b_i + b_j \tag{13}$$

where $\mathbf{c}_{i,t}^u$ and $\mathbf{c}_{j,t}^m$ are the context vectors at time step t for user i and movie j; b_i and b_j are bias terms for user i and movie j, respectively; $\mathbf{h}_{i,t}^u$ and $\mathbf{h}_{j,t}^m$ are the hidden states of LSTMs at time step t, computed by

$$\mathbf{h}_{i,t}^u = \mathrm{LSTM}(\mathbf{h}_{i,t-1}^u, \mathbf{z}_{i,t}^u, \mathbf{c}_{i,t}^u) \tag{14}$$

$$\mathbf{h}_{j,t}^m = \mathrm{LSTM}(\mathbf{h}_{j,t-1}^m, \mathbf{z}_{j,t}^m, \mathbf{c}_{j,t}^m) \tag{15}$$

The context vectors $\mathbf{c}_{i,t}^u$ and $\mathbf{c}_{j,t}^m$ act as extra input in the computation of the hidden states in LSTMs to make sure that every time step of the LSTMs can get full information of the context (long-term information). The context vectors $\mathbf{c}_{i,t}^u$ and $\mathbf{c}_{j,t}^m$ are the dynamic representations of the relevant long-term information for user i and movie j at time t, calculated by

$$\mathbf{c}_{i,t}^u = \sum_{k=1}^{U} \alpha_{k,t}^i \mathbf{e}_k^u; \quad \mathbf{c}_{j,t}^m = \sum_{p=1}^{M} \beta_{p,t}^j \mathbf{e}_p^m \tag{16}$$

where U and M are the number of users and movies. The attention weights $\alpha_{k,t}^i$ and $\beta_{p,t}^j$ for user i and movie j at time step t are computed by

$$\alpha_{k,t}^i = \frac{\exp(\sigma(\mathbf{h}_{i,t-1}^u, \mathbf{e}_k^u))}{\sum_{k'=1}^{U} \exp(\sigma(\mathbf{h}_{i,t-1}^u, \mathbf{e}_{k'}^u))} \tag{17}$$

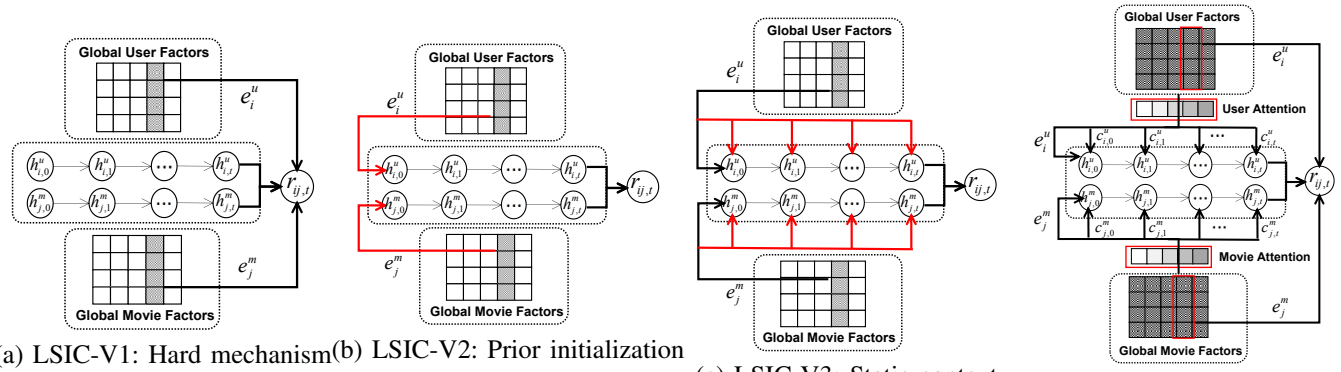
$$\beta_{p,t}^j = \frac{\exp(\sigma(\mathbf{h}_{j,t-1}^m, \mathbf{e}_p^m))}{\sum_{p'=1}^{M} \exp(\sigma(\mathbf{h}_{j,t-1}^m, \mathbf{e}_{p'}^m))} \tag{18}$$

where σ is a feed-forward neural network to produce a real-valued score. The attention weights α_t^i and β_t^j together determine which user and movie factors should be selected to generate $r_{ij,t}$.

D. Generative Adversarial Network for Recommendation

Generative adversarial network (GAN) [23] consists of a generator G and a discriminator D that compete in a minimax game with two players: the discriminator tries to distinguish real high-rated movies on training data from ranking or recommendation list predicted by G, and the generator tries to fool the discriminator and generate (predict) well-ranked recommendation list. Concretely, D and G play the following game on V(D,G):

$$\min_G \max_D v(D,G) = \mathbb{E}_{x\sim P_{true}(x)}[\log D(x)] + \mathbb{E}_{z_{noise}\sim P(z_{noise})}[\log(1 - D(G(z_{noise})))] \tag{19}$$



(a) LSIC-V1: Hard mechanism (b) LSIC-V2: Prior initialization (c) LSIC-V3: Static context (d) LSIC-V4: Attention model

Fig. 2. Four strategies to calculate the score function g, integrating MF and RNN.

Here, x is the input data from training set, z_{noise} is the noise variable sampled from normal distribution.

We propose an adversarial framework to iteratively optimize two models: the generative model G predicting recommendation list given historical user-movie interactions and the discriminative model D predicting the relevance of the generated list. Like the standard generative adversarial networks (GANs) [23], LSIC also optimizes the two models with a minimax two-player game. D tries to distinguish the real high-rated movies in the training data from the recommendation list generated by G, while G maximizes the probability of D making a mistake. Hopefully, this adversarial process can eventually adjust G to generate plausible and high-quality recommendation list. We further elaborate the generator and discriminator below.

1) Discriminative Model: As depicted in Figure 1 (right side), we implement the discriminator D via a Siamese Network that incorporates long and session-based ranking models in a pair-wise scenario. The discriminator D has two symmetrical point-wise networks that share parameters and are updated by minimizing a pair-wise loss.

The objective of discriminator D is to maximize the probability of correctly distinguishing the ground truth movies from generated recommendation movies. For G fixed, we can obtain the optimal parameters for the discriminator D with the following formulation.

$$\theta^* = \operatorname*{argmax}_{\theta} \sum_{i \in \mathcal{U}} \Big(\mathbb{E}_{m_+, m_- \sim p_{true}} \left[\log D_\theta(u_i, m_-, m_+|t) \right] + \mathbb{E}_{m_+ \sim p_{true}, m_{g,t} \sim G_\phi(m_{g,t}|u_i,t)} \left[\log(1 - D_\theta(u_i, m_{g,t}, m_+|t)) \right] \Big) \tag{20}$$

where $\mathcal{U}$ denotes the user set, u_i denotes user i, m_+ is a positive (high-rating) movie, m_- is a negative movie randomly chosen from the entire negative (low-rating) movie space, θ and ϕ are parameters of D and G, and $m_{g,t}$ is the generated movie by G given time t. Here, we adopt hinge loss as our training objective since it performs better than other training objectives. Hinge loss is widely adopted in various learning to rank scenario, which aims to penalize the examples that violate the margin constraint:

$$D(u_i, m_-, m_+|t) = \max \Big\{ 0, \epsilon - g(\mathbf{e}^u_i, \mathbf{e}^m_{m_+}, \mathbf{h}^u_{i,t}, \mathbf{h}^m_{m_+,t}) + g(\mathbf{e}^u_i, \mathbf{e}^m_{m_-}, \mathbf{h}^u_{i,t}, \mathbf{h}^m_{m_-,t}) \Big\} \tag{21}$$

where ϵ is the hyper-parameter determining the margin of hinge loss, and we compress the outputs to the range of $(0, 1)$.

2) Generative Model: Similar to conditional GANs proposed in [38], our generator G takes in the auxiliary information (user u_i and time t) as input, and generates the ranking list for user i. Specifically, when D is optimized and fixed after computing Eq. 20, the generator G can be optimized by minimizing the following formulation:

$$\phi^* = \operatorname*{argmin}_{\phi} \sum_{m \in \mathcal{M}} \big(\mathbb{E}_{m_{g,t} \sim G_\phi(m_{g,t}|u_i,t)} [\log(1 - D(u_i, m_{g,t}, m_+|t))] \big) \tag{22}$$

Here, $\mathcal{M}$ denotes the movie set. As in [23], instead of minimizing $\log(1 - D(u_i, m_{g,t}, m_+|t))$, we train G to maximize $\log(D(u_i, m_{g,t}, m_+|t))$.

3) Policy Gradient: Since the sampling of recommendation list by generator G is discrete, it cannot be directly optimized by gradient descent as in the standard GAN formulation. Therefore, we use policy gradient based reinforcement learning algorithm [39] to optimize the generator G so as to generate highly rewarded recommendation list. Concretely, we have the following derivations:

$$\begin{aligned}
\nabla_\phi J^G(u_i) &= \nabla_\phi \mathbb{E}_{m_{g,t} \sim G_\phi(m_{g,t}|u_i,t)} [\log D(u_i, m_{g,t}, m_+|t)] \\
&= \sum_{m \in \mathcal{M}} \nabla_\phi G_\phi(m|u_i,t) \log D(u_i, m, m_+|t) \\
&= \sum_{m \in \mathcal{M}} G_\phi(m|u_i,t) \nabla_\phi \log G_\phi(m|u_i,t) \cdot \\
&\qquad \log D(u_i, m, m_+|t) \\
&= \mathbb{E}_{m_{g,t} \sim G_\phi(m|u_i,t)} [\nabla_\phi \log G_\phi(m_{g,t}|u_i,t) \cdot \\
&\qquad \log D(u_i, m_{g,t}, m_+|t)] \\
&\approx \frac{1}{K} \sum_{k=1}^{K} \nabla_\phi \log G_\phi(m_k|u_i,t) \log D(u_i, m_k, m_+|t)
\end{aligned} \tag{23}$$

where K is number of movies sampled by the current version of generator and m_k is the k-th sampled item.

With reinforcement learning terminology, we treat the term $\log D(u_i, m_k, m_+|t)$ as the reward at time step t, and take an action m_k at each time step. To accelerate the convergence, the rewards within a batch are normalized with a Gaussian distribution to make the differences significant.

Algorithm 1: Long and Session-based Ranking Model with Adversarial Network

Input: generator G_ϕ, discriminator D_θ, training data S.
Initialize models G_ϕ and D_θ with random weights, and pre-train them on training data S.
repeat
  for *g-steps* **do**
    Generate recommendation list for user i at time t using the generator G_ϕ.
    Sample K candidates from recommendation list.
    for $k \in \{1, ..., K\}$ **do**
      Sample a positive movie m_+ from S.
      Compute the reward $\log D(u_i, m_k, m_+|t)$ with Eq.(21)
    Update generator G_ϕ via policy gradient Eq.(23).
  for *d-steps* **do**
    Use current G_ϕ to generate a negative movie and combined with a positive movie sampled from S.
    Update discriminator D_θ with Eq.(20).
until convergence

The overall procedure is summarized in Algorithm 1. During the training stage, the discriminator and the generator are trained alternatively in a adversarial manner via Eq.(20) and Eq.(23), respectively.

IV. Experimental Setup

A. Datasets

TABLE II
Characteristics of the datasets.

Dataset	MovieLens-100K	Netflix-3M	Netflix-Full
Users	943	326,668	480,189
movies	1,6831	17,751	17,770
Ratings	100,000	16,080,980	100,480,507
Rating Range	1-5	1-5	1-5
Train Data	09/97-03/98	9/05-11/05	12/99-11/05
Test Data	03/98-04/98	12/05	12/05
Train Ratings	77,714	13,675,402	98,074,901
Test Ratings	21,875	2,405,578	2,405,578
Density	0.493	0.406	0.093
Sparsity	0.063	0.003	0.012

In order to evaluate the effectiveness of our model, we conduct experiments on two widely-used real-life datasets: MovieLens100K and Netflix (called "Netflix-Full"). To evaluate the robustness of our model, we also conduct experiments on a 3-month Netflix (called "Netflix-3M") dataset, which is a small version of Netflix-Full and has different training and testing period. Each movie in both MovieLens and Netflix datasets has a rating from the customers, and the rating is on a five-star scale from 1 to 5. In addition, the movie files that include the years of release and the titles of the movies are also provided. Each user in MovieLens dataset has the demographic information, such as age, gender, occupation, etc.

For each dataset, we split the whole data into several training and testing intervals based on time, as is done in [4], to simulate the actual situation of predicting future behaviors of users given the data that occurred strictly before current time step. Then, each testing interval is randomly divided into a validation set and a testing set. We removed the users and movies that do not appear in training set from the validation and test sets. The detailed statistics are presented in Table II[1]. Following [28], we treat "5-star" in Netflix, "4-star" and "5-star" for MovieLens100K as positive feedback and all others as unknown (negative) feedback.

B. Implementation Details

There are several critical hyperparameters needed to be set for our proposed mdoel.

a) Matrix Factorization: 5-dimensional and 16 dimensional stationary latent factors are used for MovieLens and Netflix, respectively, as suggested in [28]. The item and user latent feature matrix is randomly initialized by a uniform distribution ranged in [-0.05, 0.05]. We take gradient-clipping to suppress the gradient to the range of [-0.2,0.2]. L_2 regularization (with $\lambda^u = \lambda^m = 0.05$) is used to the weights and biases of user and movie factors. The function ρ is a logistic scoring function that bounds the range of the outputs.

b) Convolutional Neural Network: We follow the same parameter settings as in [34] for the implementation of ResNet. We adopt batch normalization (BN) right after each convolution and before activation. We initialize the weights and train all plain/residual nets from scratch. We do not use dropout for ResNet, following the practice in [34].

c) Recurrent Neural Network: We use a single-layer LSTM with 10 hidden neurons in our experiments. The size of the input embeddings is set to 15. We adopt 4-dimensional dynamic states where each state contains 7-days users/movies behavioral trajectories. That is, we take one month as the length of a session. The weight parameters are randomly sampled from the uniform distribution $U(-0.01, 0.01)$, and the bias parameters are set to zero. L_2 regularization (with a weight decay value of 0.001) and dropout (with a dropout rate of 0.2) are used to avoid overfitting. We conduct mini-batch training (batch size = 128) using SGD optimization method to train the model which follows the suggested parameter setup in [28].

d) Generative Adversarial Nets: We pre-train G and D on the training data with a pair-wise scenario, and use SGD algorithm with learning rate 1×10^{-4} to optimize its parameters. This step aims at improving the recommendation performance of the generator G, based on the pretrained

[1]"Density" shows the average number of 5-ratings for the user per day. "Sparsity" shows the filling-rate of user-movie rating matrix as used in [4]

discriminator D. The number of sampled movies is set to 64 (i.e., $K = 64$). In addition, we use matrix factorization model to generate 100 candidate movies, and then re-rank these movies with LSTM. In all experiments, we conduct mini-batch training with batch size 128.

C. Evaluation Metrics

To quantitatively evaluate our method, we adopt the standard rank-based evaluation metrics to measure the performance of top-N recommendation [31], [40] , including Precision@N, Normalised Discounted Cumulative Gain (NDCG@N), Mean Average Precision (MAP) and Mean Reciprocal Ranking (MRR). The larger the values of these evaluation metrics are, the better the performance of the recommender system is.

The precision of recommendation describes the proportion of items that users prefer, which is defined as:

$$Precision@n = \frac{N_{rs}@n}{n} \tag{24}$$

where N_{rs} is the number of the recommended items that user prefer, and n is the first n recommended items.

The normalized discounted cumulative gain (nDCG) has been widely used to evaluate the ranked lists of the top-N recommendation systems. The premise of nDCG is that the highly relevant movies appearing lower in a list should be penalized as the graded relevance value is reduced logarithmically proportional to the position of the result. Formally, nDCG is defined as:

$$nDCG@N = \frac{DCG@N}{IDCG@N} \tag{25}$$

$$DCG@N = rel_1 + \sum_{i=2}^{N} \frac{rel(i)}{\log_2 i} \tag{26}$$

where N denotes the position up to which relevance is accumulated (the size of each recommendation list), $rel(i)$ is an indicator function equaling 1 if the item at rank i is a relevant movie, zero otherwise. $IDCG$ represents a perfect (ideal) ranking of discounted cumulative gain.

The mean average precision (MAP) is the average of different recommendation precisions. The formula of MAP is defined as:

$$MAP@N = \frac{\sum_{q=1}^{Q} AP@N(q)}{Q} \tag{27}$$

$$AP@N = \frac{\sum_{k=1}^{N} Precision@k \times rel(k)}{\textit{\# of relevant movies}} \tag{28}$$

where Q is the number of recommendation, $rel(k)$ is an indicator function equaling 1 if the item at rank k is a relevant movie, zero otherwise.

The mean reciprocal rank (MRR) can measure whether the recommender system places the user's favorite items in the front. The MRR is defined as follows:

$$MRR = \frac{\sum_{q=1}^{Q} 1/rank_q}{Q} \tag{29}$$

where Q is the number of recommendation, $rank_q$ represents the rank position of the first relevant document for the q-th recommendation.

D. Comparison to Baselines

In the experiments, we evaluate and compare our models with several state-of-the-art methods.

a) Bayesian Personalised Ranking (BPR): Given a positive movie, BPR uniformly samples negative movies to resolve the imbalance issue and provides a basic baseline for top-N recommendation [43].

b) Pairwise Ranking Factorization Machine (PRFM): This is one of the state-of-the-art movie recommendation algorithms, which applies Factorization Machine model to microblog ranking on basis of pairwise classification [44]. We use the same settings as in [44] in our experiments.

c) LambdaFM: It is a strong baseline for recommendation, which directly optimizes the rank biased metrics [42]. We run the LambdaFM model with the publicly available code[2], and use default settings for all hyperparameters.

d) Recurrent Recommender Networks (RRN): This model supplements matrix factorization with recurrent neural network model via a hard mixture mechanism [4]. We use the same setting as in [4].

e) IRGAN: This model trains the generator and discriminator alternatively with MF in an adversarial process [28]. We run the IRGAN model with the publicly available code[3], and use default settings for all hyperparameters.

f) CTR: Collaborative Topic Regression is a state-of-the-art recommendation model, which combines collaborative filtering (PMF) and topic modeling (LDA) to use both ratings and documents [45].

g) CDL: Collaborative Deep Learning is another state-of-the-art recommendation model, which enhances rating prediction accuracy by analyzing documents using Stacked Denoising AutoEncoder (SDAE) [46].

h) ConvMF: Convolutional MF (ConvMF) is a strong baseline that integrates convolutional neural network into probabilistic matrix factorization [41].

Overall, BPR, PRFM, LambdaFM, RRN, and IRGAN are reommender systems that leverage merely the rating information, while CTR, CDL and SDAE incorporate both rating and context information for item recommendation. Some important hyperparameters of each models are listed in Table III.

V. Experimental Results

In this section, we compare our model with baseline methods quantitatively and qualitatively.

A. Quantitative Evaluation

We first evaluate the performance of top-N recommendation. The experimental results are summarized in Tables IV ,V and VI. Our model substantially and consistently outperforms the baseline methods by a noticeable margin on all the experimental datasets. In particular, we have explored several versions of our model with different mixture mechanisms. As one anticipates, LSIC-V4 achieves the best results across all

[2]https://github.com/fajieyuan/LambdaFM
[3]https://github.com/geek-ai/irgan

TABLE III
THE KEY FEATURES AND HYPERPARAMETERS FOR THE STATE-OF-THE-ART BASELINE METHODS.

Baselines	Key features	Key hyperparameters
BPR	Rating matrix	The number of features k is set to 10. We randomly initialize to a small value the item and user feature matrix. The regularization terms for user and item are set to 0.025.
PRFM	Rating matrix	We use LIBPMF to implement this model. Grid search of regularization parameter over $\lambda \in \{10^0, 10^{-1}, \ldots, 10^{-5}\}$ and factor size over $k \in \{20, 40, 80, 160\}$ is performed to choose the best parameters.
LambdaFM	Rating matrix	We run LambdaFM with $\gamma_\pi, \gamma_\varepsilon \in \{0.5, 0.1, 0.05, 0.01, 0.005\}$ to find the best regularization parameters. We tune the value of $\rho \in \{0.01, 0.1, 0.3, 0.5, 0.8, 1.0\}$. The value of parameter ε is fixed at 1.
RRN	Rating matrix	The number of hidden states of LSTM is 15 for Netflix (including Netflix-3M and Netflix-full) and 10 for Movielens. 15-dimensional input embeddings and 4-dimensional dynamic states are used. Each dynamic state contains 7-days users/movies behavioral trajectories. For MF, we use 5-dimensional and 20-dimensional stationary latent factors for MovieLens and Netflix, respectively. ADAM optimizer is adopted and the learning rate is 1×10^{-4}.
IRGAN	Rating matrix	The factor numbers for matrix factorization are 5 and 16 for MovieLens and Netflix respectively. The number of hidden neurons of LSTM is 10. 15-dimensional input embeddings and 4 dimensional dynamic states. L_2 regularization (with $\lambda = 0.05$) is used to weights and biases of the LSTM layer to avoid overfitting. SGD with learning rate 1×10^{-4} is used to optimize the parameters of the model.
CTR	Rating matrix and Item text description	We use grid search to find that $K = 50, \lambda_u = 0.01, \lambda_v = 10, a = 1, b = 0.01$ achieves good performance on the held out recommendations.
CDL	Rating matrix and Item text description	We directly set $a = 1, b = 0.01, K = 50$ and perform grid search on the hyperparameters $\lambda_u, \lambda_v, \lambda_n, \lambda_w \in \{0.01, 0.1, 1.0, 10, 100\}$. We use a masking noise with a noise level of 0.3 to get the corrupted input and choose a dropout rate from {2, 33, 11} of 0.1 to achieve adaptive regularization. The 2-layer CDL model is adopted.
ConvMF	Rating matrix and Item text description	The maximum length of documents is set to 300. As in [41], we initialize the word latent vectors randomly with dimension size of 200, which are tuned through the training phase. The dropout is used to prevent overfitting and we set the dropout rate to 0.2. Mini-batch based RMSprop (with batch size 128) is used to optimize the model.

TABLE IV
MOIVE RECOMMENDATION RESULTS (MOVIELENS).

	Precision@3	Precision@5	Precision@10	NDCG@3	NDCG@5	NDCG@10	MRR	MAP
BPR	0.2795	0.2664	0.2301	0.2910	0.2761	0.2550	0.4324	0.3549
PRFM	0.2884	0.2699	0.2481	0.2937	0.2894	0.2676	0.4484	0.3885
LambdaFM	0.3108	0.2953	0.2612	0.3302	0.3117	0.2795	0.4611	0.4014
RRN	0.2893	0.2740	0.2480	0.2951	0.2814	0.2513	0.4320	0.3631
IRGAN	0.3022	0.2885	0.2582	0.3285	0.3032	0.2678	0.4515	0.3744
CTR	0.2824	0.2694	0.2493	0.2855	0.2836	0.2569	0.4505	0.3725
CDL	0.2875	0.2731	0.2504	0.2946	0.2865	0.2673	0.4491	0.3863
ConvMF	0.2901	0.2856	0.2545	0.2978	0.2914	0.2733	0.4592	0.3996
LSIC-V1	0.2946	0.2713	0.2531	0.2905	0.2801	0.2644	0.4595	0.4066
LSIC-V2	0.3004	0.2843	0.2567	0.3122	0.2951	0.2814	0.4624	0.4101
LSIC-V3	0.3105	0.3023	0.2610	0.3217	0.3086	0.2912	0.4732	0.4163
LSIC-V4	**0.3327**	**0.3173**	**0.2847**	**0.3512**	**0.3331**	**0.2939**	**0.4832**	**0.4321**

evaluation metrics and all datasets. For example, on MovieLens dataset, LSIC-V4 improves 7.45% on percision@5 and 6.87% on NDCG@5 over the baseline methods. The main strength of our model comes from its capability of prioritizing both long-term and short-term information in content-aware movie recommendation. In addition, our mixture mechanisms (hard and soft) also seem quite effective to integrate MF and RNN.

To better understand the adversarial training process, we visualize the learning curves of LSIC-V4 as shown in Figure 3. Due to the limited space, we only report the Precision@5 and NDCG@5 scores as in [28]. The other metrics exhibit a similar trend. As shown in Figure 3, after about 50 epoches, both Precision@5 and NDCG@5 converge and the winner is the generator which is used to generate recommendation list for our final top-N movie recommendation. The performance of generator G becomes better with the effective feedback (reward) from discriminator D. On the other hand, once we have a set of high-quality recommendation movies, the performance of D deteriorates gradually in the training procedure and makes mistakes for predictions. In our experiments, we use the generator G with best performance to predict test data.

B. Ablation Study

In order to analyze the effectiveness of different components of our model for top-N movie recommendation, in this section, we report the ablation test of LSIC-V4 by discarding poster information (w/o poster) and replacing the reinforcement learning with Gumbel-Softmax [47] (w/o RL), respectively.

TABLE V
MOVIE RECOMMENDATION RESULTS (NETFLIX-3M).

	Precision@3	**Precision@5**	**Precision@10**	**NDCG@3**	**NDCG@5**	**NDCG@10**	**MRR**	**MAP**
BPR	0.2670	0.2548	0.2403	0.2653	0.2576	0.2469	0.3829	0.3484
PRFM	0.2562	0.2645	0.2661	0.2499	0.2575	0.2614	0.4022	0.3712
LambdaFM	0.3082	0.2984	0.2812	0.3011	0.2993	0.2849	0.4316	0.4043
RRN	0.2759	0.2741	0.2693	0.2685	0.2692	0.2676	0.3960	0.3831
IRGAN	0.2856	0.2836	0.2715	0.2824	0.2813	0.2695	0.4060	0.3718
CTR	0.2564	0.2578	0.2605	0.2543	0.2622	0.2535	0.3244	0.3521
CDL	0.2703	0.2704	0.2646	0.2741	0.2733	0.2597	0.3765	0.3843
ConvMF	0.2841	0.2793	0.2694	0.2775	0.2790	0.2633	0.3968	0.3805
LSIC-V1	0.2815	0.2801	0.2680	0.2833	0.2742	0.2696	0.4416	0.4025
LSIC-V2	0.2901	0.2883	0.2701	0.2903	0.2831	0.2759	0.4406	0.4102
LSIC-V3	0.3152	0.3013	0.2722	0.2927	0.2901	0.2821	0.4482	0.4185
LSIC-V4	**0.3221**	**0.3193**	**0.2921**	**0.3157**	**0.3114**	**0.2975**	**0.4501**	**0.4247**

TABLE VI
MOVIE RECOMMENDATION RESULTS (NETFLIX-FULL).

	Precision@3	**Precision@5**	**Precision@10**	**NDCG@3**	**NDCG@5**	**NDCG@10**	**MRR**	**MAP**
BPR	0.3011	0.2817	0.2587	0.2998	0.2870	0.2693	0.3840	0.3660
PRFM	0.2959	0.2837	0.2624	0.2831	0.2887	0.2789	0.4060	0.3916
LambdaFM	0.3446	0.3301	0.3226	0.3450	0.3398	0.3255	0.4356	0.4067
RRN	0.3135	0.2954	0.2699	0.3123	0.3004	0.2810	0.3953	0.3768
IRGAN	0.3320	0.3229	0.3056	0.3319	0.3260	0.3131	0.4248	0.4052
CTR	0.2857	0.2939	0.2765	0.3001	0.3156	0.2886	0. 3736	0.3925
CDL	0.3045	0.3076	0.2844	0.3042	0.3244	0.2933	0.4195	0.4041
ConvMF	0.3257	0.3293	0.2946	0.3155	0.3396	0.2974	0.4302	0.4166
LSIC-V1	0.3127	0.3012	0.2818	0.3247	0.3098	0.2957	0.4470	0.4098
LSIC-V2	0.3393	0.3271	0.3172	0.3482	0.3401	0.3293	0.4448	0.4213
LSIC-V3	0.3501	0.3480	0.3291	0.3498	0.3451	0.3321	0.4503	0.4257
LSIC-V4	**0.3621**	**0.3530**	**0.3341**	**0.3608**	**0.3511**	**0.3412**	**0.4587**	**0.4327**

TABLE VII
ABLATION RESULTS FOR NETFLIX-3M DATASET.

	Precision@3	**Precision@5**	**Precision@10**	**NDCG@3**	**NDCG@5**	**NDCG@10**	**MRR**	**MAP**
LSIC-V4	**0.3221**	**0.3193**	**0.2921**	**0.3157**	**0.3114**	**0.2975**	**0.4501**	**0.4247**
w/o RL	0.3012	0.2970	0.2782	0.2988	0.2927	0.2728	0.4431	0.4112
w/o poster	0.3110	0.3012	0.2894	0.3015	0.3085	0.2817	0.4373	0.4005

Gumbel-Softmax is an alternative method to address the non-differentiation problem so that G can be trained straightforwardly.

Due to the limited space, we only illustrate the experimental results for Netflix-3M dataset that is widely used in movie recommendation (see Table VII). Generally, both factors contribute, and reinforcement learning contributes most. This is within our expectation since discarding reinforcement learning will lead the adversarial learning inefficient. With Gumbel-Softmax, G does not benefit from the reward of D, so that we do not know which movies sampled by G are good and should be reproduced. Not surprisingly, poster information also contributes to movie recommendation.

C. Computational Cost

We investigate the computational cost of baseline methods and the proposed LSIC model. All these methods are run on a single NVIDIA GeForce GTX 1080 Ti. The GAN framework is indeed computationally expensive, thus LSIC and IRGAN have longer training time than other methods. Specifically, the training time of each epoch on Netflix-3M dataset is about 2 hours for LSIC and half a hour for IRGAN. On the other hand, the RNN component is also computationally expensive, which is caused by the complex operations in each RNN unit along the input sequence. Therefore, the MF based methods (e.g., BPR, PRFM, LambdaFM) are faster than RNN [4] because they do not need recurrent calculators of input sequence length. In particular, RNN method takes about 50 minutes per epoch, while the MF based models take only about 10 minutes per

TABLE VIII
THE RECALLED MOVIES FROM TOP-10 CANDIDATES IN NEFLIX-3M DATASET

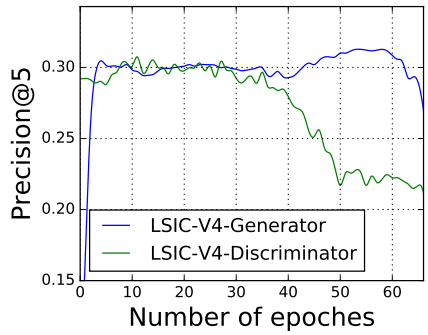
	Groundtruth	**IRGAN** [28]	**RRN** [4]	**LambdaFM** [42]	**LSIC-V4**
Userid: 1382	9 Souls The Princess Bride Stuart Saves His Family The Last Valley Wax Mask After Hours Session 9 Valentin	**[1]** The Beatles: Love Me Do **[2]** Wax Mask ✓ **[3]** Stuart Saves His Family ✓ **[4]** After Hours ✓ **[5]** Top Secret! **[6]** Damn Yankees **[7]** Dragon Tales: It's Cool to Be Me! **[8]** Play Misty for Me **[9]** The Last Round: Chuvalo vs. Ali' **[10]** La Vie de Chateau	**[1]** Falling Down **[2]** 9 Souls ✓ **[3]** Wax Mask ✓ **[4]** After Hours ✓ **[5]** Stuart Saves His Family ✓ **[6]** Crocodile Dundee 2 **[7]** The Princess Bride ✓ **[8]** Dragon Tales: It's Cool to Be Me! **[9]** They Were Expendable **[10]** Damn Yankees	**[1]** The Avengers '63 **[2]** Wax Mask ✓ **[3]** The Boondock Saints **[4]** Valentin ✓ **[5]** 9 Souls ✓ **[6]** The Princess Bride ✓ **[7]** After Hours ✓ **[8]** Tekken **[9]** Stuart Saves His Family ✓ **[10]** Runn Ronnie Run	**[1]**9 Souls ✓ **[2]**The Princess Bride ✓ **[3]**Stuart Saves His Family ✓ **[4]** The Last Valley ✓ **[5]** Wax Mask ✓ **[6]** Session 9 ✓ **[7]** Dragon Tales: It's Cool to Be Me! **[8]** Damn Yankees **[9]** After Hours ✓ **[10]** Valentin ✓
Userid: 8003	9 Souls Princess Bride	**[1]** Cheech Chong's Up in Smoke **[2]** Wax Mask **[3]** Damn Yankees **[4]** Dragon Tales: It's Cool to Be Me! **[5]** Top Secret! **[6]** Agent Cody Banks 2: Destination London **[7]** After Hours **[8]** Stuart Saves His Family **[9]** 9 Souls ✓ **[10]** The Beatles: Love Me Do	**[1]** Crocodile Dundee 2 **[2]** Session 9 **[3]** Falling Down **[4]** Wax Mask **[5]** After Hours **[6]** Stuart Saves His Family **[7]** 9 Souls ✓ **[8]** The Princess Bride ✓ **[9]** Dragon Tales: It's Cool to Be Me! **[10]** Scream 2	**[1]** The Insider **[2]**A Nightmare on Elm Street 3 **[3]** Dennis the Menace Strikes Again **[4]** Civil Brand **[5]** 9 Souls ✓ **[6]** Falling Down **[7]** The Princess Bride ✓ **[8]** Radiohead: Meeting People **[9]** Crocodile Dundee 2 **[10]** Christmas in Connecticut	**[1]** 9 Souls ✓ **[2]**The Princess Bride ✓ **[3]**The Last Valley **[4]**Stuart Saves His Family **[5]**Wax Mask **[6]**Dragon Tales: It's Cool to Be Me! **[7]**Session 9 **[8]**Crocodile Dundee 2 **[9]**Damn Yankees **[10]**Cheech Chong's Up in Smoke

epoch on an average for the Netflix-3M dataset. All models typically converge within less than 10 epochs using the early stopping criterion.

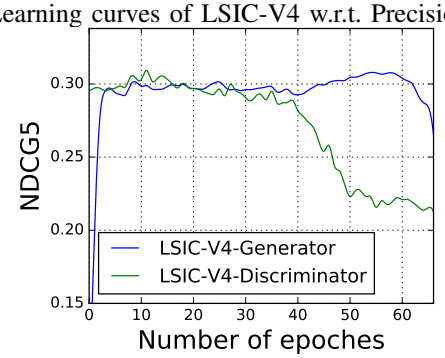
D. Case Study

In this section, we will further show the advantages of our models through some quintessential examples.

In Table VIII, we provide the recommendation lists generated by three state-of-the-art baseline methods (i.e., IRGAN, RNN, LambdaFM) as well as the proposed LSIC-V4 model for two users who are randomly selected from the Netflix-3M dataset. Our model can rank the positive movies in higher positions than other methods. For example, the ranking of the movie "*9 souls*" for user "8003" has increased from 5-th position (by LambdaFM) to 1st position (by LSIC-V4). Meanwhile some emerging movies such as "*Session 9*" and "*The Last Valley*" that are truly attractive to the user "1382" have been recommended by our models, whereas they are ignored by baseline methods. In fact, we can include all positive movies in the top-10 list for user "1382" and in top-3 list for user "8003". Our model benefits from the fact that both dynamic and static knowledge are incorporated into the model with adversarial training.



(a) Learning curves of LSIC-V4 w.r.t. Precision@5



(b) Learning curves of LSIC-V4 w.r.t. NDCG@5

Fig. 3. Learning curves of LSIC-V4 (the discriminative model D and generative model G) on Netflix-3M.

E. Re-rank Effect

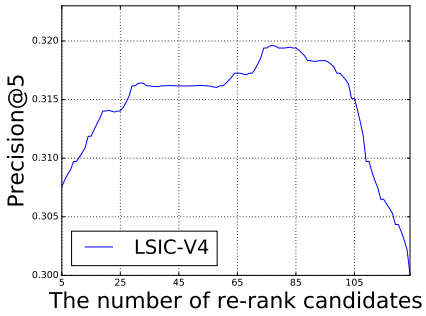
From our experiments, we observe that it could be time-consuming for RNN to infer all movies. In addition, users may be interested in a few movies that are subject to a long-tailed distribution. Motivated by these observations, we provide a re-ranking strategy as used in [48]. Specifically, we first generate N candidate movies by MF, and then re-rank these candidate movies with our model. In this way, the inference time can been greatly reduced. Figure 4 illustrates the performance curves over the number of candidate movies (i.e. N) generated by MF. We only report the Precision@5 and NDCG@5 results due to the limited space, the other metrics exhibit a similar trend. As shown in Figure 4, when the number of candidate movies is small, i.e., $N \leq 100$ for Netflix-3M dataset, the Percision@5 increases gradually with the increase in number of candidate movies. Nevertheless, the performance decreases rapidly with the increase in number of candidate movies when $N \geq 100$. It suggests that the generated candidates in the long-tail side is inaccurate, and these candidate movies deteriorate the overall performance of the re-rank strategy.

F. Cold Start Issue

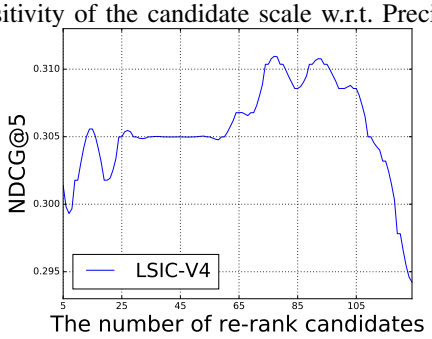
To investigate the effectiveness of our model in dealing with the cold-start problem, we also conduct experiments on the Movielens dataset for the *cold-start users*. In this paper, the users that have less than 10 ratings are categorized as *cold-start users* and there are 73 *cold-start users* in Movielens

TABLE IX
PERFORMANCE ON THE COLD-START USERS IN MOVIELENS.

	Precision@3	Precision@5	Precision@10	NDCG@3	NDCG@5	NDCG@10	MRR	MAP
LSIC-V4	**0.0356**	**0.0388**	**0.0447**	**0.0355**	**0.0373**	**0.0425**	**0.0952**	**0.0870**
w/o Poster	0.0144	0.0217	0.0195	0.0153	0.0199	0.0190	0.0500	0.0435
w/o RL	0.0337	0.0345	0.0406	0.0319	0.0320	0.0367	0.0905	0.0819
IRGAN	0.0246	0.0320	0.0345	0.0225	0.0281	0.0316	0.0685	0.0653

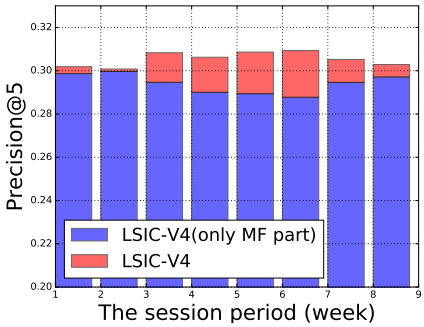


(a) Sensitivity of the candidate scale w.r.t. Precision@5

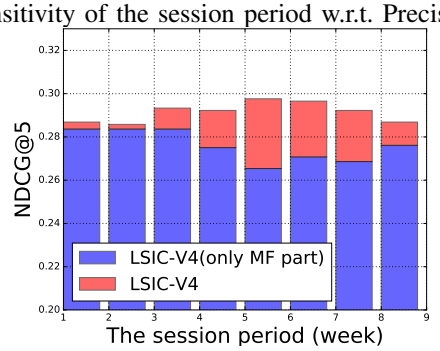


(b) Sensitivity of the candidate scale w.r.t. NDCG@5

Fig. 4. Sensitivity of the candidate scale on Netflix-3M



(a) Sensitivity of the session period w.r.t. Precision@5



(b) Sensitivity of the session period w.r.t. NDCG@5

Fig. 5. Sensitivity of the session period on Netflix-3M

dataset. The experimental results are reported in Table IX. From the results we can observe that the cold-start users benefit significantly from the movie posters. In particular, our model achieves much better performance than other models that do not consider the context information (e.g., IRGAN and w/o poster) on cold-start users. This verifies the effectiveness of our model, which incorporates the auxiliary information (posters) for alleviating the cold-start problem.

On the other hand, based on our empirical observation from Tables IV-VI, the previous models without considering context features (e.g., BPR, PRFM, LambdaFM, RRN and IRGAN) tend to recommend only the movies with a lot of ratings (historical interactions). For example, the average ratings of the top 3 recommended movies by LSIC-V4 is 185.3, which is much less than that by the RNN model (i.e., 321.7). That is, the new items (or cold-start items) have more chance to appear in users' recommendation lists by using our model.

G. Session Period Sensitivity

The above experimental results have shown that the session-based (short-term) information indeed improves the performance of top-N recommendation. We conduct an experiment on Netflix-3M to investigate how the session period influences the final recommendation performance. As shown in Figure 5, the bars in purple color (below) describe the basic performance of MF component while the red ones (above) represent the extra improvement by the LSIC-V4. The RNN plays an insignificant role in the very early sessions since it lacks enough historical interaction data. For later period of sessions, LSIC-V4 model achieves a clear improvement over the model with only MF component till an optimal value (when the session period is 7 weeks), after which the evaluation results decrease slightly. This may be because that the movie recommendation mainly relies on the short-term preferences of users and evolutional attributes of movies. With the increase of the session period, the performance of movie recommendation may be hindered by the long-term information.

VI. CONCLUSION

In this paper, we proposed a novel LSIC model, which leveraged both long- and short-term information for context-aware movie recommendation using adversarial training. The adversarial process simultaneously optimized a generative model G and a discriminative model D via a minimax two-player game. In particular, the discriminator D is implemented by a

siamese network to incorporate long-term based and session-based ranking model in a pair-wise scenario. We explored both hard and soft mixture mechanisms to integrate MF and RNN methods and learn the temporal dynamics with the help of the long-term profiles. To further enhance the recommendation performance, we also integrated the posters of movies that were effective when few ratings were available. We conducted extensive experiments to evaluate the effectiveness of LSIC on two real-life datasets: Netflix Prize Contest data and MovieLens data. Experimental results revealed that the proposed LSIC model substantially and consistently outperformed the baseline methods by a noticeable margin, obtaining the state-of-the-art performances.

In the future, we would like to extend our method by exploring a more advanced MF component to alleviate the data sparsity problem. In addition, a mutual attention will be designed to interactively learn attentions in the MF and RNN components. With this design, our model can well represent the long- and short-term information when performing movie recommendation in current time-step. What's more, we also plan to further improve the performance of movie recommendation by exploiting more auxiliary data such as movie descriptions, movie reviews and user profiles by modifying the generator and discriminator networks.

REFERENCES

[1] J. Bobadilla, F. Ortega, A. Hernando, and A. Gutierrez, "Recommender systems survey," *Knowledge-based systems*, vol. 46, pp. 109–132, 2013.

[2] J. Lu, D. Wu, M. Mao, W. Wang, and G. Zhang, "Recommender system application developments: a survey," *Decision Support Systems*, vol. 74, pp. 12–32, 2015.

[3] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, 2009.

[4] C.-Y. Wu, A. Ahmed, A. Beutel, A. J. Smola, and H. Jing, "Recurrent recommender networks," in *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*. ACM, 2017, pp. 495–503.

[5] C. Danescu-Niculescu-Mizil, R. West, D. Jurafsky, J. Leskovec, and C. Potts, "No country for old members: User lifecycle and linguistic change in online communities," in *Proceedings of the 22nd international conference on World Wide Web*. ACM, 2013, pp. 307–318.

[6] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[7] C.-Y. Wu, A. Ahmed, A. Beutel, and A. J. Smola, "Joint training of ratings and reviews with recurrent recommender networks," *ICLR*, 2016.

[8] R. Devooght and H. Bersini, "Collaborative filtering with recurrent neural networks," *arXiv preprint arXiv:1608.07400*, 2016.

[9] L. Zhao, Z. Lu, S. J. Pan, and Q. Yang, "Matrix factorization+ for movie recommendation." in *IJCAI*, 2016, pp. 3945–3951.

[10] P. Hao, G. Zhang, L. Martinez, and J. Lu, "Regularizing knowledge transfer in recommendation with tag-inferred correlation," *IEEE Transactions on Cybernetics*, no. 99, pp. 1–14, 2017.

[11] M. Mao, J. Lu, G. Zhang, and J. Zhang, "Multirelational social recommendations via multigraph ranking," *IEEE transactions on cybernetics*, vol. 47, no. 12, pp. 4049–4061, 2017.

[12] J. D. Rennie and N. Srebro, "Fast maximum margin matrix factorization for collaborative prediction," in *Proceedings of the 22nd international conference on Machine learning*. ACM, 2005, pp. 713–719.

[13] Y. Koren, "Factorization meets the neighborhood: a multifaceted collaborative filtering model," in *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2008, pp. 426–434.

[14] A. Hernando, J. Bobadilla, and F. Ortega, "A non negative matrix factorization for collaborative filtering recommender systems based on a bayesian probabilistic model," *Knowledge-Based Systems*, vol. 97, pp. 188–202, 2016.

[15] X. He, H. Zhang, M.-Y. Kan, and T.-S. Chua, "Fast matrix factorization for online recommendation with implicit feedback," in *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*. ACM, 2016, pp. 549–558.

[16] N. Srebro, J. Rennie, and T. S. Jaakkola, "Maximum-margin matrix factorization," in *Advances in neural information processing systems*, 2005, pp. 1329–1336.

[17] A. Mnih and R. R. Salakhutdinov, "Probabilistic matrix factorization," in *Advances in neural information processing systems*, 2008, pp. 1257–1264.

[18] Y. Koren, "Collaborative filtering with temporal dynamics," *Communications of the ACM*, vol. 53, no. 4, pp. 89–97, 2010.

[19] B. Hidasi, A. Karatzoglou, L. Baltrunas, and D. Tikk, "Session-based recommendations with recurrent neural networks," in *ICLR*, 2015.

[20] C. Wu, J. Wang, J. Liu, and W. Liu, "Recurrent neural network based recommendation for time heterogeneous feedback," *Knowledge-Based Systems*, vol. 109, pp. 90–103, 2016.

[21] Q. Cui, S. Wu, Q. Liu, and L. Wang, "A visual and textual recurrent neural network for sequential prediction," *arXiv preprint arXiv:1611.06668*, 2016.

[22] R. Devooght and H. Bersini, "Long and short-term recommendations with recurrent neural networks," in *Proceedings of the 25th Conference on User Modeling, Adaptation and Personalization*. ACM, 2017, pp. 13–21.

[23] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[24] S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee, "Generative adversarial text to image synthesis," in *Proceedings of the 33rd International Conference on International Conference on Machine Learning*. JMLR.org, 2016, pp. 1060–1069.

[25] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *International Conference on Machine Learning*, 2017, pp. 214–223.

[26] T.-H. Chen, Y.-H. Liao, C.-Y. Chuang, W.-T. Hsu, J. Fu, and M. Sun, "Show, adapt and tell: Adversarial training of cross-domain image captioner," *arXiv preprint arXiv:1705.00930*, 2017.

[27] L. Yu, W. Zhang, J. Wang, and Y. Yu, "Seqgan: Sequence generative adversarial nets with policy gradient." in *AAAI*, 2017, pp. 2852–2858.

[28] J. Wang, L. Yu, W. Zhang, Y. Gong, Y. Xu, B. Wang, P. Zhang, and D. Zhang, "Irgan: A minimax game for unifying generative and discriminative information retrieval models," in *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2017, pp. 515–524.

[29] G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions," *IEEE transactions on knowledge and data engineering*, vol. 17, no. 6, pp. 734–749, 2005.

[30] S. M. McNee, J. Riedl, and J. A. Konstan, "Being accurate is not enough: how accuracy metrics have hurt recommender systems," in *CHI'06 extended abstracts on Human factors in computing systems*. ACM, 2006, pp. 1097–1101.

[31] N. N. Liu and Q. Yang, "Eigenrank: a ranking-oriented approach to collaborative filtering," in *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2008, pp. 83–90.

[32] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[33] W.-T. Chu and H.-J. Guo, "Movie genre classification based on poster images with deep neural networks," in *Proceedings of the Workshop on Multimodal Understanding of Social, Affective and Subjective Attributes*. ACM, 2017, pp. 39–45.

[34] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[35] N. Mohajerin and S. L. Waslander, "State initialization for recurrent neural network modeling of time-series data," in *2017 International Joint Conference on Neural Networks*. IEEE, 2017, pp. 2330–2337.

[36] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[37] Y. Ding, Y. Liu, H. Luan, and M. Sun, "Visualizing and understanding neural machine translation," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, vol. 1, 2017, pp. 1150–1159.

[38] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *arXiv preprint arXiv:1411.1784*, 2014.
[39] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Advances in neural information processing systems*, 2000, pp. 1057–1063.
[40] P. Cremonesi, Y. Koren, and R. Turrin, "Performance of recommender algorithms on top-n recommendation tasks," in *Proceedings of the fourth ACM conference on Recommender systems*. ACM, 2010, pp. 39–46.
[41] D. Kim, C. Park, J. Oh, S. Lee, and H. Yu, "Convolutional matrix factorization for document context-aware recommendation," in *RecSys*. ACM, 2016, pp. 233–240.
[42] F. Yuan, G. Guo, J. M. Jose, L. Chen, H. Yu, and W. Zhang, "Lambdafm: learning optimal ranking with factorization machines using lambda surrogates," in *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. ACM, 2016, pp. 227–236.
[43] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "Bpr: Bayesian personalized ranking from implicit feedback," in *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence*. AUAI Press, 2009, pp. 452–461.
[44] R. Qiang, F. Liang, and J. Yang, "Exploiting ranking factorization machines for microblog retrieval," in *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*. ACM, 2013, pp. 1783–1788.
[45] C. Wang and D. M. Blei, "Collaborative topic modeling for recommending scientific articles," in *SIGKDD*. ACM, 2011, pp. 448–456.
[46] H. Wang, N. Wang, and D.-Y. Yeung, "Collaborative deep learning for recommender systems," in *SIGKDD*. ACM, 2015, pp. 1235–1244.
[47] M. J. Kusner and J. M. Hernández-Lobato, "Gans for sequences of discrete elements with the gumbel-softmax distribution," *arXiv preprint arXiv:1611.04051*, 2016.
[48] P. Covington, J. Adams, and E. Sargin, "Deep neural networks for youtube recommendations," in *Proceedings of the 10th ACM Conference on Recommender Systems*. ACM, 2016, pp. 191–198.

Wei Zhao is currently a PhD candidate at the department of Computer Science in Technische Universität Darmstadt, Germany. He received his M.Eng degree from University of Chinese Academy of Sciences in 2018. He was a visiting student in Nanyang Technological University, Singapore in 2018. Before that, he received B.A degree from Fudan University. His research interests are machine learning, information retrieval and natural language processing.

Benyou Wang is currently a Marie Curie Researcher, as well as a first-year Phd student at University of Padova. He was a NLP research engineer in Tencent. He received his M.S. degree in Feb. 2017 from School of Computer Science and technology, Tianjin University. Prior to that, Benyou received his B.S. degree from the HUAT in 2014. His research interests are natural language processing, information retrieval, quantum language model, and reinforcement learning.

Min Yang is currently an assistant professor at Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences. She received her Ph.D. degree from the University of Hong Kong in 2017. Her current research interests include machine learning and natural language processing.

Jianbo Ye received his B. S. degree in Mathematics from the University of Science and Technology of China in 2011. He worked as a research postgraduate at The University of Hong Kong, from 2011 to 2012, and a research intern at Intel Labs, China in 2013. He is currently a PhD candidate at the College of Information Sciences and Technology, The Pennsylvania State University. His research interests include statistical modeling and learning, numerical optimization and method, and affective image modeling.

Zhou Zhao received the BS and the Ph.D. degrees in computer science from the Hong Kong University of Science and Technology (HKUST), in 2010 and 2015, respectively. He is currently an associate professor with the College of Computer Science, Zhejiang University. His research interests include machine learning, data mining and information retrieval.

Xiaojun Chen received his Ph.D. degree from Harbin Institute of Technology in 2011. He is now an assistant professor at College of Computer Science and Software, Shenzhen University. His research interests include machine learning, clustering, feature selection and massive data mining.

Ying Shen is now an assistant professor in School of Electronics and Computer Engineering (SECE) at Peking University. She received her Ph.D. degree from the University of Paris Ouest Nanterre La Défense (France), specialized in Medical and Biomedical Information Science. Her research interest is mainly focused in the area of Medical Informatics, Natural Language Processing and Machine Learning.