Utilizing Generative Adversarial Networks for Recommendation based on Ratings and Reviews

Wang Chen
Tsinghua-Southampton Web Science Laboratory
Graduate School at Shenzhen, Tsinghua University
Shenzhen, China
chen-w16@mails.tsinghua.edu.cn

Hai-Tao Zheng*
Tsinghua-Southampton Web Science Laboratory
Graduate School at Shenzhen, Tsinghua University
Shenzhen, China
zheng.haitao@sz.tsinghua.edu.cn

Yang Wang
Tsinghua-Southampton Web Science Laboratory
Graduate School at Shenzhen, Tsinghua University
Shenzhen, China
yang-wan17@mails.tsinghua.edu.cn

Wei Wang
Tsinghua-Southampton Web Science Laboratory
Graduate School at Shenzhen, Tsinghua University
Shenzhen, China
w-w16@mails.tsinghua.edu.cn

Rui Zhang
School of Computing and Information Systems
University of Melbourne
Melbourne, Australia
rui.zhang@unimelb.edu.au

***Abstract*—Many existing rating-based recommendation algorithms have achieved relative success. However, the real-world datasets are extremely sparse and most rating-based algorithms are still suffering from the data sparsity problem. Along with integer-valued ratings, we consider that the user-generated review is also an important user feedback. Furthermore, compared with the traditional recommendation algorithms which have the limited ability to learn the distributions of ratings and reviews simultaneously, the generative adversarial networks can learn better representations for data. In this paper, we propose Rating and Review Generative Adversarial Networks (RRGAN), an innovative framework for recommendation, in which the generative model and discriminative model play a minimax game. Specifically, the generative model predicts the ratings of top-N list for users or items based on reviews, while the discriminative model aims to distinguish the predicted ratings from real ratings. With the competition between these two models, RRGAN improves the ability of understanding users and items based on ratings and reviews. We introduce the user profiles, item representations and ratings into a matrix factorization model to predict the top-N list for the users. In addition, we study three different architectures to learn reasonable user profiles and item representations based on ratings and reviews to achieve better recommendations. To evaluate the performance of our model, we conduct the extensive experiments on three real-world amazon datasets in three parts, which are top-N recommendation analysis, case study and long-tail users analysis. The experimental results show that our method significantly outperforms various state-of-the-art methods, including LFM, LambdaFM, HFT, DeepCoNN and IRGAN methods.**

I. Introduction

In recent years, generative adversarial networks [1] have been shown to produce state-of-the-art results on various tasks such as computer vision [2] and natural language processing [3], [4]. However, there are only rare researches exploiting the generative adversarial networks in recommendation task. Although many recommendation algorithms have been proposed [5]–[9], the real-world dataset is extremely sparse and degrades the performance of these rating-based methods significantly. Along with integer-valued ratings, we consider that the user-generated review is also an important user feedback. Ratings represent the users' preference degree of items. Reviews represent the reason why users like or dislike items. However, it is still difficult to ensure that the connection between ratings and reviews is correctly scaled because integer-valued ratings and plaintext reviews have different physical meanings. Besides that, the traditional recommendation algorithms have a limited ability to learn the distributions of ratings and reviews simultaneously. Comparing with the traditional recommendation algorithms, the generative adversarial networks can learn better representations for data. In this paper, we propose a novel framework named Rating and Review Generative Adversarial Networks (RRGAN) for high quality recommendation. There are three components in RRGAN, which are latent semantic model, generative model and discriminative model. Firstly, we jointly train them to generate user profiles and item representations based on ratings and reviews. In particular, the generative network tries to learn reasonable representations of users and items to predict the ratings, and the discriminative network tries to differentiate the predicted ratings from the real ratings. The learned representations are confined by a decoder of the latent semantic model. Secondly, we introduce the user profiles, item representations and ratings into a matrix factorization model and train the model for predicting ratings. Finally, we use the predicted ratings to rank the items for the users. Specifically, the main contributions of this paper are summarized as the following three aspects:

- To the best of our knowledge, we are the first to utilize Generative Adversarial Networks for recommendation based on ratings and reviews. We train our model in an adversarial process to fix the relationship between the ratings and the reviews.
- To achieve better recommendations, we study three different architectures to learn reasonable user profiles and item representations based on ratings and reviews.
- We conduct extensive experiments on real-world datasets.

*corresponding author.

The experimental results show that our method significantly outperforms various state-of-the-art methods.

The remainder of this paper is organized as follows. In Section 2, we review the related work on recommendation algorithms. In Section 3, we detail the framework of RRGAN. In Section 4, we evaluate RRGAN by comparing the state-of-the-art methods. In Section 5, we conclude our paper and discuss future work.

II. Related Work

Matrix factorization for recommendation. Matrix factorization techniques are widely used in many recommender systems. To achieve better recommendations, a variety of matrix factorization models have been proposed [5]–[8], [10]–[14]. They associate the users and items with latent factor. To alleviate the data sparsity problem, factorization machines model (FM) is proposed which is an extension to the linear regression and matrix factorization [15]. Neural Factorization Machine (NFM) combines the FM model and neural network for recommendation [16]. LambdaFM optimizes the rank for FM model by lambdaRank [17], [18]. We also adopt a matrix factorization model as one of the key components in our framework.

Rating and Review Model for recommendation. Reviews are a great supplement to the ratings because reviews provide rich semantic information about users and items. More recently, to improve the performance of recommender system, some researchers have attempted to exploit ratings and reviews based on the topic model [19]–[24]. With an exponential transformation function, HFT model links a Latent Dirichlet Allocation (LDA) [25] model in modeling the review text and a matrix factorization model in modeling the ratings [20]. RMR models the ratings with a mixture of Gaussian consistent with the topic distribution about the reviews [21]. Wu et al. propose RRN model combing ratings and reviews via interacting recurrent networks(LSTM) [26], [27]. DeepCoNN model exploit ratings and reviews based on two parallel neural networks [28]. These models are devoted to build a bridge between the ratings and the reviews. However, the reviews and the ratings have different physical meanings so it is difficult to ensure that these connections are correctly scaled. Besides that, the traditional recommendation algorithms have a limited ability to learn the distributions of ratings and reviews simultaneously. Comparing with the traditional recommendation algorithms, the generative adversarial networks can learn better representations for data. The generative model and the discriminative model play a minimax game to improve their performance till reaching a Nash equilibrium. Therefore, we utilize the generative adversarial networks to fix the relationship between the ratings and the reviews.

Generative adversarial network for recommendation. For information retrieval task, IRGAN is proposed which consists of the generative retrieval model and the discriminative retrieval model [29]. IRGAN achieved impressive performances in three tasks including web search, item recommendation, and question answering. IRGAN aims to unify two schools of thinking in information retrieval modelling. Different from IRGAN, RRGAN unifies the ratings and the reviews for recommendation. In the framework of RRGAN, we generate the user profiles and item representations based on ratings and reviews. We introduce the user profiles, the item representations, and the ratings into a matrix factorization model to improve the recommendation performance.

III. Rating and Review Generative Adversarial Networks

A. Framework

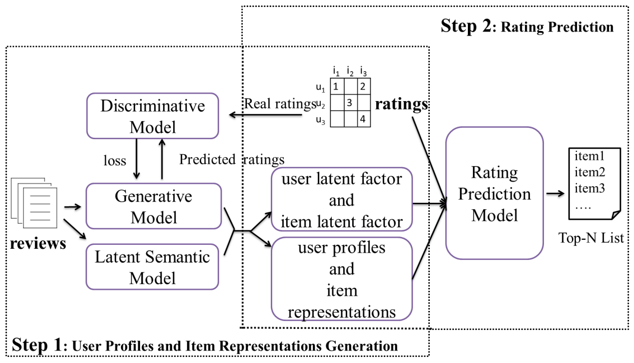


Fig. 1. Recommendation Framework

Fig. 1 mainly shows the recommendation framework. There are two main steps in the framework. Fig. 1 contains the models using in these two steps. Fig. 2 demonstrates the relationship between these models in detail.

Step 1: User Profiles and Item Representations Generation. We utilize the RRGAN to generate user profiles and item representations for recommendation. There are three key components in RRGAN, which are: (1) The latent semantic model generates the user profiles and the item representations from plaintext reviews. (2) The generative model predicts the ratings of top-N list for the user or the item based on reviews. (3) The discriminative model aims to distinguish the predicted ratings from real ratings based on the user profiles and the item representations.

Step 2: Rating Prediction. We introduce the user profiles, item representations and ratings into a matrix factorization model. We train the model for predicting ratings. Finally we produce a top-N list for users.

B. Generating User Profiles and Item Representations

To make the relationship between the models more clear, Fig. 2 provides a deeper technical detail. As shown in Fig. 2, we propose a GAN framework to generate user profiles and item representations. There are three components in RRGAN, which are latent semantic model, generative model and discriminative model. The latent semantic model generates user profiles and item representations based on reviews. The generative model and the discriminative model play a minimax

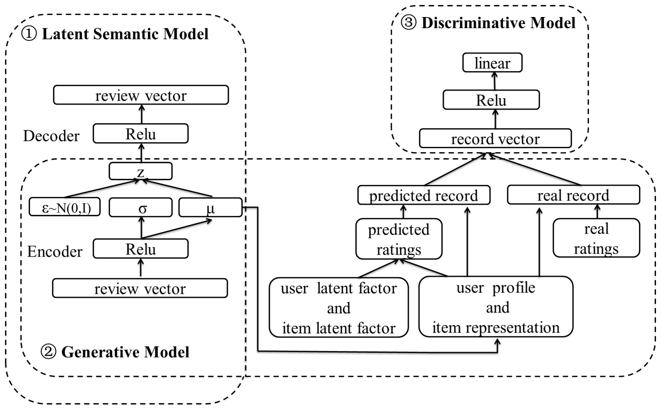


Fig. 2. Rating and Review Generative Adversarial Networks

game to improve the accuracy of user profiles and item representations. The generated user profile and item representation will share the same latent semantic space and the operation of them will be more reasonable. $\boldsymbol{\mu}$ is the result output we need and is regarded as user profiles and item representations. The parameters of latent semantic model encoder are c and the parameters of decoder are θ_1. The user latent matrix and the item latent matrix are θ_2. The parameters of generative model θ contains θ_1 and θ_2. The parameters of discriminative model are ϕ.

Latent Semantic Model. The latent semantic model generates the user profiles and the item representations based on reviews. We adopt variational autoencoder model (VAE) [30] as the latent semantic model. VAE model is a generative model which uses variational approach for latent representation learning. A VAE consists of two networks: the recognition network (encoder) and the generative network (decoder). The recognition network is written $q(z|x)$ which approximates the latent representation z on a data sample x. The generator network is written $p(x|z)$ which generates x given on the latent representation z. The objective function of the VAE model takes the following form:

$$\begin{aligned}\mathcal{L}^{VAE} = \min \sum_{x} \Big(&- \mathbb{E}_{q(z|x)}[logp(x|z)] \\ &+KL(q(z|x)||p(z))\Big),\end{aligned} \tag{1}$$

where $KL(\cdot||\cdot)$ is the KL Divergences.

As shown in Fig. 2, multi-layer neural network is utilized in the encoder and decoder of the latent semantic model. We map the reviews of users or received reviews for items into a vector $\boldsymbol{x} = \{w_1, w_2...w_n\}$ by the bag-of-words model. w_n means whether the nth word is appeals in the review. $\boldsymbol{\mu}$ is the mean and $\boldsymbol{\sigma}$ is the standard deviation while they are the output of the encoder network. $\boldsymbol{\epsilon}$ is the sample from $\mathcal{N}(0, I)$. $\boldsymbol{z}$ can be approximated by $\boldsymbol{\mu}$, $\boldsymbol{\sigma}$, and $\boldsymbol{\epsilon}$. They can be computed as follows:

$$\boldsymbol{\mu}, \boldsymbol{\sigma} = Encoder(\boldsymbol{x}), \tag{2}$$

$$\boldsymbol{z} = \boldsymbol{\mu} + \boldsymbol{\sigma} \otimes \boldsymbol{\epsilon}, \tag{3}$$

$$\tilde{\boldsymbol{x}} = Decoder(\boldsymbol{z}) \ , \tag{4}$$

where $\boldsymbol{z}$, $\boldsymbol{\mu}$, $\boldsymbol{\sigma}$, and $\boldsymbol{\epsilon}$ are all l-dimensional vectors. l is the number of latent factors dimension.

For the users, the input of VAE are all reviews posted by user, defined as $\boldsymbol{x}_u$. For the items, the input of VAE are all received reviews for item, defined as $\boldsymbol{x}_i$. We only use one VAE model to generate $\boldsymbol{\mu}_u$ or $\boldsymbol{\mu}_i$. The model generates user profile $\boldsymbol{\mu}_u$ when we feed $\boldsymbol{x}_u$, while generates item representation $\boldsymbol{\mu}_i$ when we feed $\boldsymbol{x}_i$. We use the same VAE model to ensure that $\boldsymbol{\mu}_u$ and $\boldsymbol{\mu}_i$ share a consistent latent semantic space. $\boldsymbol{\mu}$ rather than $\boldsymbol{z}$ is regarded as user profiles and item representations. $\boldsymbol{z}$ is stochastic because it is computed by $\boldsymbol{\epsilon}$ while $\boldsymbol{\epsilon}$ is the sample from $\mathcal{N}(0, I)$.

Generative Model. Before introducing our model, we provide a brief description of LFM here. Matrix factorization techniques are widely used in many recommender systems. LFM [8] is one of the most popular algorithms in Matrix factorization techniques. In the LFM, we can predict the rating as:

$$\tilde{r}_{ui} = (\boldsymbol{v}_u)^T \boldsymbol{v}_i + r_{avg} + b_u + b_i \ , \tag{5}$$

where $\tilde{r}_{ui}$ is the predicted rating that user u for item i. $\boldsymbol{v}_u$ is the latent factor of the user u while $\boldsymbol{v}_i$ is the latent factor of the item i. r_{avg} is the global average rating. b_u is the bias of user u while b_i is the bias of item i.

Based on reviews, the generative model predicts the ratings of top-N items for the special user or the ratings of top-N users for the special item. The real-world dataset is extremely sparse and most users have few behaviors. We focus on only top-N ratings rather than all ratings for each user and each item. Firstly, we generate the user profiles $\boldsymbol{\mu}_u$ and the item representations $\boldsymbol{\mu}_i$ by the encoder based on reviews. Secondly, inspired by the LFM, the generative model predicts the rating as:

$$\tilde{r}_{ui} = (\boldsymbol{v}_u)^T \boldsymbol{v}_i + \alpha * \frac{1}{distance(\boldsymbol{\mu}_u, \boldsymbol{\mu}_i) + 1}, \tag{6}$$

where $\boldsymbol{v}_u$ and $\boldsymbol{v}_i$ are the latent factors as in Equation 5. $\boldsymbol{v}_u$ and $\boldsymbol{v}_i$ are both k-dimensional vectors. k is the number of latent factor's dimension. $\boldsymbol{\mu}_u$ is the user profile of user u and $\boldsymbol{\mu}_i$ is the item representation of item i. $\boldsymbol{\mu}_u$ and $\boldsymbol{\mu}_i$ are both l-dimensional vectors. $distance(\boldsymbol{\mu}_u, \boldsymbol{\mu}_i)$ denotes the distance between the $\boldsymbol{\mu}_u$ and $\boldsymbol{\mu}_i$. Because $\boldsymbol{\mu}_u$ and $\boldsymbol{\mu}_i$ share a consistent latent semantic space, we directly measure the distance between them. Euclidean distance is used for $distance(\boldsymbol{\mu}_u, \boldsymbol{\mu}_i)$. Other kinds of distance are also acceptable, such as the Manhattan distance. We convert the Euclidean distance to range 0 and 1 by $\frac{1}{distance(\boldsymbol{\mu}_u, \boldsymbol{\mu}_i)+1}$. α is the weight factor which decides how much the rating function depends on the reviews. Here we omit the biases of user b_u, the biases of items b_i and the global average rating r_{avg} because this part is considered by the discriminative model. In Equation 6, when the data is very sparse, the performance of matrix factorization $(\boldsymbol{v}_u)^T \boldsymbol{v}_i$ is excessively poor. Nevertheless, the other part $\frac{1}{distance(\boldsymbol{\mu}_u, \boldsymbol{\mu}_i)+1}$ is not much affected by the problem of data sparsity because user reviews and item representations can be extracted from only one review.

Finally, the objective function of the generative model can be formulated as follows:

$$\mathcal{L}^{G} = \min_{\theta} \sum_{u,i \in Train} \Big(- \mathbb{E}_{\tilde{r} \sim g_{\theta}(\tilde{r}|\boldsymbol{\mu}_u, \boldsymbol{\mu}_i)} D(\tilde{r}|\boldsymbol{\mu}_u, \boldsymbol{\mu}_i) \Big), \quad (7)$$

where the generative model G is written $g_{\theta}(\tilde{r}|\boldsymbol{\mu}_u, \boldsymbol{\mu}_i)$. $g_{\theta}(\tilde{r}|\boldsymbol{\mu}_u, \boldsymbol{\mu}_i)$ represents that the generative model generates the predicted ratings $\tilde{r}$ given the user profiles $\boldsymbol{\mu}_u$ and item representations $\boldsymbol{\mu}_i$. The details are shown in Equation 6. $D(\cdot|\cdot)$ represents the discriminative model. The discriminative model D distinguishes the predicted ratings $\tilde{r}$ from real ratings r given the user profiles $\boldsymbol{\mu}_u$ and item representations $\boldsymbol{\mu}_i$.

Discriminative Model. The discriminative model is a multi-layer neural network. Given the user profiles and the item representations, the discriminative model D attempts to distinguish the predicted ratings $\tilde{r}$ from real ratings r about top-N recommendation list. As shown in Fig. 2, the "record vector" is the input of the discriminative model. The "record vector" here is predicted record or real record. The predicted record consists of three parts which are predicted ratings, user profile and item representation. The real record also consists of three parts which are real ratings, user profile and item representation. The objective function of the discriminative model can be formulated as follows:

$$\mathcal{L}^{D} = \min_{\phi} \sum_{u,i \in Train} \Big(\mathbb{E}_{\tilde{r} \sim g_{\theta}(\tilde{r}|\boldsymbol{\mu}_u, \boldsymbol{\mu}_i)} D(\tilde{r}|\boldsymbol{\mu}_u, \boldsymbol{\mu}_i) - \mathbb{E}_{r \sim g_{true}(r|\boldsymbol{\mu}_u, \boldsymbol{\mu}_i)} D(r|\boldsymbol{\mu}_u, \boldsymbol{\mu}_i) \Big), \quad (8)$$

where r and $\tilde{r}$ are both n-dim dimensional vectors. For the user sample, the number of the user profile $\boldsymbol{\mu}_u$ is one and the number of the item representations $\boldsymbol{\mu}_i$ is n. All the item representations and user profile are concatenated as input. For the item sample, the number of the user profile $\boldsymbol{\mu}_u$ is n and the number of the item-representations $\boldsymbol{\mu}_i$ is one. For a user sample, if the user with m items and m is less than n, we concatenate $n - m$ user profiles with m item representations as item representations and pad $n - m$ ratings which are $(\boldsymbol{v}_u)^T \boldsymbol{v}_u + \alpha$ to the predicted ratings and real ratings. For a item sample, if the item with less than n users, the process is similar.

On the one hand, the discriminative model D can learn the relationship between the ratings, the user profiles and the item representations. On the other hand, the discriminative model learns the biases of the user and the item. For example, some users are used to giving high ratings while the other are used to giving low ratings. The bias of the item is similar.

To briefly conclude, the generative model and discriminative model play a minimax game for optimizing the user profiles and the item representations. Simultaneously, the user profiles and the item representations are confined by a decoder of the latent semantic model. Consequently, the objective function of this game can be formulated as follows:

$$\mathcal{L}^{G,D} = \min_{\theta} \max_{\phi} \sum_{u,i \in Train} \Big(\mathbb{E}_{r \sim g_{true}(r|\boldsymbol{\mu}_u, \boldsymbol{\mu}_i)} D(r|\boldsymbol{\mu}_u, \boldsymbol{\mu}_i) - \mathbb{E}_{\tilde{r} \sim g_{\theta}(\tilde{r}|\boldsymbol{\mu}_u, \boldsymbol{\mu}_i)} D(\tilde{r}|\boldsymbol{\mu}_u, \boldsymbol{\mu}_i) \Big). \quad (9)$$

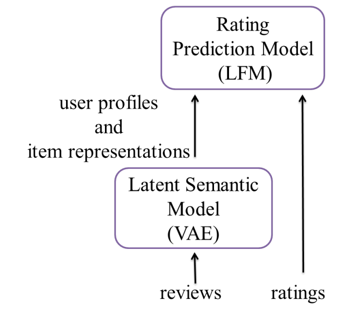
Algorithm 1: Rating and Review Generative Adversarial Networks.

```
1  Input: latent semantic model $\text{VAE}_{c,\theta_1}$, generative model $G_{\theta_1,\theta_2}$, discriminative model $D_\phi$, training data S. ( $c$ are the parameters of $\text{VAE}_{c,\theta_1}$ encoder, $\theta_1$ are the parameters of $\text{VAE}_{c,\theta_1}$ decoder, $\theta_2$ are the user latent matrix and item latent matrix, $\phi$ are the parameters of D.)
2  There are 2 types of samples in $S$, which are user sample and item sample. The user sample contains one user and top-N items for the user while the item sample contains one item and top-N users for the item.
3  Initialize models $\text{VAE}_{c,\theta_1}$, $G_{\theta_1,\theta_2}$ and $D_\phi$ with random weights.
4  repeat
5  for vae-steps do
6      if sample is user sample then
7          Generate one user profile via Eq. 2.
8      else if sample is item sample then
9          Generate one item representation via Eq. 2.
10     Update latent semantic model $\text{VAE}_{c,\theta_1}$ via Eq. 1.
11 for g-steps do
12     if sample is user sample then
13         Generate $n$ item representations via Eq. 2.
14         Predict the ratings of top-N items based on $n$ item representations and user profile generated in vae-steps via Eq. 6.
15     else if sample is item sample then
16         Generate $n$ user profiles via Eq. 2.
17         Predict the ratings of top-N users based on $n$ user profiles and item representation generated in vae-steps via Eq. 6.
18     Update the generative model $G_{\theta_1,\theta_2}$ via Eq. 7.
19 for d-steps do
20     if sample is user sample then
21         Distinguish the predicted ratings from real ratings based on one user profile and $n$ item representations.
22     else if sample is item sample then
23         Distinguish the predicted ratings from real ratings based on one item representation and $n$ user profiles.
24     Update the discriminative model $D_\phi$ via Eq. 8.
25 until convergence
```

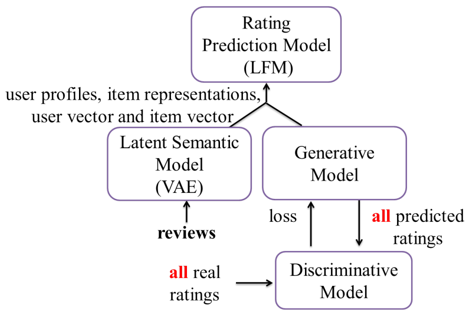
The overall procedure is summarized in Algorithm 1. We jointly train the latent semantic model, the generative model and the discriminative model to generate user profiles and item representations based on ratings and reviews.

C. Rating Prediction

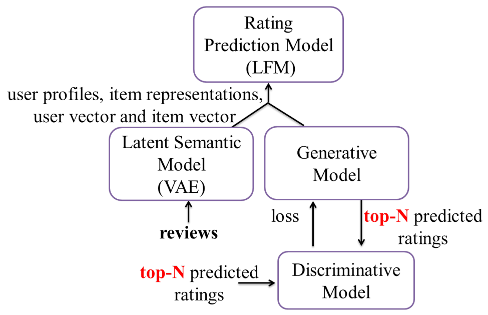
The top-N task needs to give recommendation list according to model's predicted ratings. As shown in Equation 6, the



(a) RRGAN-V1: RRGAN without GAN loss



(b) RRGAN-V2: RRGAN focus on all ratings



(c) RRGAN-V3: RRGAN focus on top-N ratings

Fig. 3. Three architectures to learn reasonable user profiles and item representations based on ratings and reviews for recommendation.

generative model predicts the ratings, and the discriminative model ties to differentiate the predicted ratings from the real ratings. In the minimax game, the models learn the user profiles $\boldsymbol{\mu}_u$, the item representations $\boldsymbol{\mu}_i$, the latent factors of user $\boldsymbol{v}_u$ and the latent factors of item $\boldsymbol{v}_i$. We feed $\boldsymbol{\mu}_u$, $\boldsymbol{\mu}_i$, $\boldsymbol{v}_u$ and $\boldsymbol{v}_i$ into a final rating prediction model to predict the top-N list. The prediction function is the same as Equation 6. In training process, we fix the parameters $\boldsymbol{\mu}_u$ and $\boldsymbol{\mu}_i$, and use the the parameters $\boldsymbol{v}_u$ and $\boldsymbol{v}_i$ pre-trained by the generative model to initialize the same parameters in final rating prediction model. In other words, the model only learns the parameters $\boldsymbol{v}_u$ and $\boldsymbol{v}_i$. The objective function of our final rating prediction model can be formulated as follows,

$$\mathcal{L} = \min_{\boldsymbol{v}_u, \boldsymbol{v}_i} \sum_{u,i \in Train} (r_{ui} - \tilde{r}_{ui})^2 \ , \tag{10}$$

where r_{ui} are the real ratings and $\tilde{r}_{ui}$ are the predicted ratings. Finally, we use the predicted ratings to rank the items for the users.

As shown in Fig.3, to achieve better recommendations, we study three different architectures to learn reasonable user profiles and item representations based on ratings and reviews. The details are described below.

RRGAN-V1. As shown in Fig. 3(a), this is a simple version without GAN loss and this version only contains a VAE model and a LFM model. In this version, the VAE model learns reasonable representations of users and items based on reviews. The learned representations are fed into the LFM model to predict the top-N list. Specially, we randomly initialize the parameters $\boldsymbol{v}_u$ and $\boldsymbol{v}_i$ for LFM because these two parameters are not pre-trained by the VAE model. We consider this simple version as baseline to evaluate the benefits of introducing GAN loss.

RRGAN-V2. As shown in Fig. 3(b), the architecture of this version is consistent with the architecture in Fig. 1. Specially, the generative model predicts the ratings of all items for the special user or the ratings of all users for the special item while the discriminative model attempts to distinguish all predicted ratings from all real ratings.

RRGAN-V3. The real-world dataset is extremely sparse and there are a large number of long-tail users. As shown in Fig. 3(c), the difference between RRGAN-V3 and RRGAN-V2 is that the generative model and the discriminative model focus on only top-N ratings for each user and each item in RRGAN-V3 rather than all ratings.

IV. Experiments

A. Datasets

Dataset	Watches	Patio	Kindle Store
#users	62041	166832	116191
#items	10318	19531	4372
#ratings (#reviews)	68356	206250	160793
#ratings / #users	1.10	1.23	1.38
#ratings / #items	6.62	10.56	36.77
% of long-tail users	99.45%	98.40%	97.42%
training set	12/11-11/12	12/11-11/12	12/11-11/12
validation set	12/12-1/13	12/12-1/13	12/12-1/13
test set	2/13-3/13	2/13-3/13	2/13-3/13
#training samples	10135	44017	10887
#validation samples	5959	16435	6917
#test samples	1749	5111	2740

TABLE I
STATISTICS OF THE DATASETS.

To evaluate the performance of our model, we conduct the extensive experiments on three real-world amazon datasets[1]. The datasets span a period of 18 years [20]. The three datasets contain three categories of products, namely Watches, Patio, Kindle Store respectively. Each sample in the dataset contains a user id, an item id, a rating (ranging from 1 to 5 stars), and a plaintext review. Table I shows statistics of these three datasets.

From the Table I, we can find that these real-world datasets are extremely sparse. Each user posts 1.26 reviews on average. Specially, more than 98.26% of users are long-tail users who have no more than three ratings.

B. Experimental Setup

Preprocess. As the Table I shown, we split each dataset into training/validation/test dataset based on time. We regard the 5-start ratings as positive feedback and all other as unknown (negative) feedback as in [29]. We map star-ratings to positive/negative feedback because we only predict whether the user like or dislike the item. This transformation decreases the

[1]https://snap.stanford.edu/data/web-Amazon.html

	Recall@3	Recall@5	NDCG@3	NDCG@5
LFM	0.0035	0.0071	0.0033	0.0046
HFT	0.0009	0.0018	0.0009	0.0013
LambdaFM	0.0107	0.0179	0.0083	0.0113
DeepCoNN	0.0009	0.0018	0.0006	0.0023
IRGAN	0.0331	0.0547	0.0209	0.0300
RRGAN-V1	0.0286	0.0511	0.0182	0.0277
RRGAN-V2	0.0332	0.0538	0.0216	0.0303
RRGAN-V3	**0.0349**	**0.0556**	**0.0248**	**0.0334**
Improvement	5.4%	1.6%	18.6%	11.3%

TABLE II
RESULTS ON WACTHES.

	Recall@3	Recall@5	NDCG@3	NDCG@5
LFM	0.0012	0.0019	0.0011	0.0013
HFT	0.0021	0.0027	0.0017	0.0020
LambdaFM	0.0006	0.0006	0.0005	0.0005
DeepCoNN	0.0009	0.0071	0.0006	0.0032
IRGAN	0.0287	0.0362	0.0228	0.0263
RRGAN-V1	0.0302	0.0399	0.0235	0.0273
RRGAN-V2	0.0303	0.0393	0.0257	0.0298
RRGAN-V3	**0.0306**	**0.0420**	**0.0260**	**0.0309**
Improvement	6.6%	16.0%	14.0%	17.4%

TABLE III
RESULTS ON PATIO.

	Recall@3	Recall@5	NDCG@3	NDCG@5
LFM	0.0024	0.0047	0.0028	0.0039
HFT	0.0029	0.0070	0.0031	0.0051
LambdaFM	0.0218	0.0312	0.0194	0.0235
DeepCoNN	0.0041	0.0053	0.0031	0.0037
IRGAN	0.1115	0.1328	**0.1141**	0.1231
RRGAN-V1	0.0944	0.1275	0.0922	0.1060
RRGAN-V2	0.1133	0.1481	0.1113	0.1231
RRGAN-V3	**0.1227**	**0.1582**	0.1123	**0.1251**
Improvement	10.0%	19.1%	-1.6%	1.6%

TABLE IV
RESULTS ON KINDLE STORE.

complexity of training model in some degree under the sparse data. Basing on the reviews, we remove the stop words and extract top 5000 words based on term frequencies to build the word list for each dataset. Then we map the reviews of users and received reviews for items into a 5000-dimensional vector based on the word list. These word vectors are the inputs of latent semantic model and generative model.

Model Setting and Training. In the latent semantic model, the encoder and the decoder contain one hidden layer. The hidden size of each hidden layer is 200. The latent size of $\boldsymbol{\mu}$, $\boldsymbol{\sigma}$ and $\boldsymbol{z}$ is 25. The number of the predicted ratings N is 3 in the generative model and the discriminative model. The latent factor number k in the generative model is 5. The discriminative model contains one hidden layer with 25-bits. We set the number of batch and the number of epoch 100 and 5 respectively. We jointly train these three models and the practical embedding model by SGD algorithm with learning rate 1×10^{-4}. We clam the weights of the discriminative model to the range of [-0.1, 0.1] after each gradient update. We use the SGD algorithm to train the final rating prediction model with a learning rate 1×10^{-3}. We also set the latent factor number $k = 5$ consistent with the baseline methods and our generative model. For the optimization objective of the matrix factorization model, we let $\alpha = 0.9$. In RRGAN-V1, we randomly initialize the parameters $\boldsymbol{v}_u$ and $\boldsymbol{v}_i$ for final rating prediction model. In RRGAN-V2 and RRGAN-V3, we use the parameters $\boldsymbol{v}_u$ and $\boldsymbol{v}_i$ pre-trained by the generative model to initialize the same parameters in final rating prediction model. We use grid search to determine the optimal hyperparameters for all the baselines.

Baseline Methods. We implemented our framework RRGAN by using TensorFlow[2]. We compare RRGAN with five start-of-the-art methods, including LFM [8], HFT [20], LambdaFM [17], DeepCoNN [28] and IRGAN [29].

LFM model is a popular matrix factorization algorithm and it predicts the unknown ratings only based on the ratings. We run the LFM model with the source code implemented by LibRec[3]. LambdaFM is a very strong baseline method which optimizes the rank for FM model. We run the LambdaFM model with the source code[4]. HFT model exploit ratings and reviews based on the topic model. This is the offcial implementation[5] of HFT released by McAuley. DeepCoNN model exploit ratings and reviews based on two parallel neural networks. We run the DeepCoNN model with the source code[6]. IRGAN model unifies the generative model and discriminative model via adversarial training. This is the offcial implementation[7] of IRGAN released by Wang et al..

In particular, to achieve better recommendations, we study three different architectures to learn reasonable user profiles and item representations based on ratings and reviews. RRGAN-V1 is a simple version without GAN loss and only contains a VAE model and a LFM model. RRAGN-V2 is a version which focuses on all ratings in the generative model and the discriminative model. RRGAN-V3 focuses on only top-N ratings for each user and each item rather than all ratings. We conduct the experiment of self-comparisons.

Evaluation Metric. To evaluate the performance of our model, we adopt the commonly used metrics $Recall@N$ and $Precision@N$.

$$Recall@N = \frac{Hit@N}{M \times |U_{test}|} \tag{11}$$

$$Precision@N = \frac{Hit@N}{N \times |U_{test}|} \tag{12}$$

where $Hit@N$ denotes the total number of hits, $|U_{test}|$ is the number of user in the test set, N is the size of the recommendation size and M is the average number of relevant item for each user. Because $Recall@N/Precision@N = N/M$, we only show the $Recall@N$. Moreover, we also adopt Normalised Discounted Cumulative Gain (NDCG@N) as evaluation metric.

C. Top-N Recommendation Analysis

In the top-N recommendation task, the system attempts to recommend the user N items that the user is most interested in.

[2]https://www.tensorflow.org
[3]https://www.librec.net
[4]https://github.com/fajieyuan/LambdaFM
[5]http://cseweb.ucsd.edu/ jmcauley/
[6]https://github.com/chenchongthu/DeepCoNN
[7]https://github.com/geek-ai/irgan

	Ground truth	HFT	LambdaFM	DeepCoNN	IRGAN	RRGAN-V3
Dataset:Watches Userid:A3M6L0KYHLJT87	B000GAYQKY	[1] B000INA75C [2]B0007N546A [3]B000HP1EXU [4]B000EJPDOK [5]B0007N53E8	[1]B0000C9ZBX [2]B000F1OGWW [3]B0000C9ZBQ [4]B000ETYF30 [5]B000CK4FGI	[1]B000BDH8II [2]B000P0WURQ [3]B000K2L62Y [4]B000GAWSGI [5]B000GB1RB4	***[1]B000GAYQKY** [2]B000EQS1JW [3]B000H6AQ0Q [4]B000CK4FGI [5]B000GAWSDG	***[1]B000GAYQKY** [2]B000CK4FGI [3]B000GAYQL8 [4]B000GAYQLI [5]B000EQS1JW
Dataset:Patio Userid:A1VGDM55HCLAC2	B000HCNEWM B0002ZINDY	[1]B000HCNEVI [2]B000NCWP44 [3]B0000775FZ [4]B000ND7DTK [5]B000JKONAO	[1]B00002N67I [2]B000MR7B26 [3]B00004RB23 [4]B0001WV010 [5]B000668Z96	[1]B0006HDILK [2]B000KISUVI [3]B00004TR7S [4]B000FBMFDO [5]B0002ZINDY	***[1]B000HCNEWM** [2]B00004SD7B [3]B000071NUS [4]B000Q5S7RM [5]B000BWFESU	***[1]B000HCNEWM** [2]B000Q5S7RM [3]B0007LQ3RQ [4]B00004SD7B ***[5]B0002ZINDY**
Dataset:Kindle Store Userid:A74XNUC3NJI27	B000FC1HA0 B000FC1FUW	[1]B000FC27NG [2]B000OI0IL4 [3]B000OVLJS2 [4]B000JMKR34 [5]B000JMKSZ6	[1]B000OZ0NXA [2]B000PC0S0K [3]B000PC0SEQ [4]B000FC1N2W [5]B000FC1CBO	[1]B000PC6ATS [2]B000OCXJP2 [3]B000GCFX68 [4]B000JMKOAA [5]B000P2A3ZU	[1]B000FC1N2C [2]B000NY133G [3]B000OI0G1Q [4]B000PDYVU2 ***[5]B000FC1HA0**	[1]B000FC1N2C ***[2]B000FC1HA0** ***[3]B000FC1FUW** [4]B000OIZT9U [5]B000PC0S5K

TABLE V
THE TOP-5 ITEM RECOMMENDATION LIST FOR EACH USER.

Table II, Table III and Table IV compare the performance of top-N recommendation task with all the baseline methods and RRGAN in three datasets respectively. In particular, we have explored several versions of our model with different architectures. Firstly, we can see that RRGAN-V3 obtains the best result in $11/12$ evaluation metrics, and on the rest one, RRGAN-V3 is still competitive to the best one. For example, RRGAN-V3 improves 5.4% on $Recall@3$, 1.6% on $Recall@5$, 18.6% on $NDCG@3$ and 11.3% on $NDCG@5$ comparing with all the baseline methods on Watches dataset. Secondly, RRGAN-V3 outperforms the RRAGN-V1 and RRAGN-V2 across all evaluation metrics.

The reasons why RRGAN-V3 performs best are as follows: (1) Reviews are a great supplement of ratings. We introduce the user-generated reviews into our model. The latent semantic information of reviews helps our model alleviate the data sparsity problem in a certain extent. The model without textual signal only learns the users' preference degree of items, but ignores the reason why users like or dislike items. Moreover, the real-world dataset is extremely sparse and degrades the performance of the model without textual signal. (2) For a recommendation task, reviews represent the reason why users like or dislike items. Ratings represent the users' preference degree of items. We unify the ratings and the reviews via an adversarial process rather than the designed connection rule adopted by HFT. With the adversarial process, our model can better fix the relationship between ratings and reviews. (3) RRGAN-V1 only learns representations of users and items based on reviews and build a bridge between the ratings and the reviews by the final rating predict model. The learned representations are only based on reviews which degrades the performance of RRGAN-V1. Moreover, with the benefits of introducing GAN loss, RRGAN-V3 outperforms the RRAGN-V1. (4) Considering that the real-world dataset is extremely sparse and most users have few behaviors, RRGAN-V3 focus on the top-N ratings rather than all ratings and it helps RRGAN-V3 outperform the RRAGN-V2.

D. Case Study

In this section, we show the advantages of our model through some examples. Table V shows the top-5 recommendation lists for three users who are randomly selected from the three datasets. RRGAN-V3 can recommend all the latent items for users in these three cases and provide a better order of these items. On Watches dataset, RRGAN-V3 and IRGAN both recommend the item 'B000GAYQKY' in 1st position. On Patio dataset, RRGAN-V3 selects the item 'B0002ZINDY' which is ignored by the baselines. On Kindle Store dataset, RRGAN-V3 recommends the item 'B000FC1HA0' in 2nd position which is higher than the 5th position in IRGAN. The results show that RRGAN improves the ability of understanding users and items based on ratings and reviews.

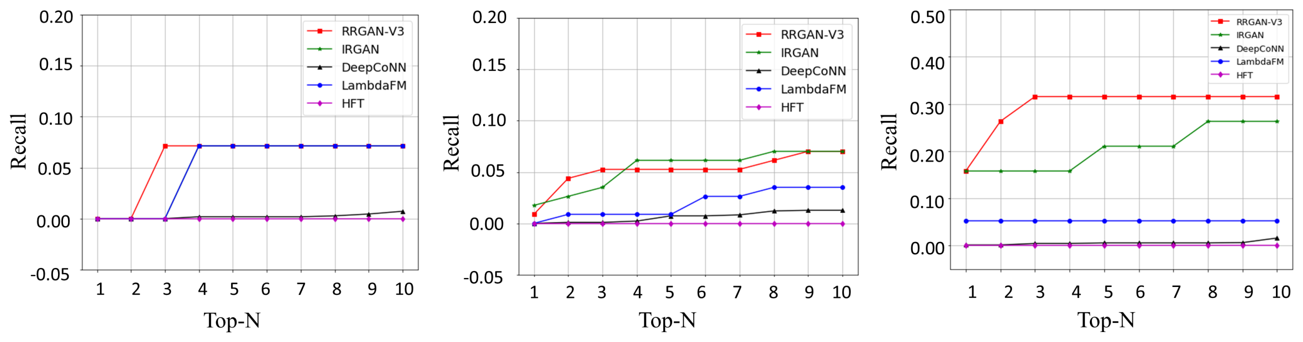
E. Long-tail Users Analysis

To evaluate the performance of RRGAN for long-tail users who have no more than three ratings, we conduct experiments on the three datasets. As the Fig. 4 shown, we can see that RRGAN-V3 obtains better Recall than other methods in most cases. Such as on Kindle Store dataset, the recall improvement of RRGAN-V3 is 0%, 66.7%, 100%, 100%, 50%, 50%, 50%, 20%, 20%, 20% respectively when the size of top-N ranges from 1 to 10. The experimental results demonstrate that RRGAN can generate user profiles and item representations effectively based on ratings and reviews for long-tail users. The user profiles and item representations are both the significant factors to achieve better recommendations.

V. CONCLUSIONS AND FUTURE WORK

In this paper, we propose an innovative framework named RRAGN to generate the user profiles and item representations by unifying ratings and reviews. We introduce the user profiles, item representations and ratings into rating prediction model and train the model for recommendation. Moreover, we study three different architectures to learn reasonable user profiles and item representations based on ratings and reviews. Experiments show that RRGAN outperforms various start-of-the-art methods.

In the future, we will investigate the way to unify explicit feedback and implicit feedback via an adversarial process. Moreover, we plan to exploit how to link the content-based method based on reviews and collaborative filtering based on



(a) Watches (b) Patio (c) Kindle Store

Fig. 4. Recall results of long-tail users

ratings via generative adversarial networks to achieve better recommendations.

Acknowledgements. This research is supported by National Natural Science Foundation of China (Grant No. 61773229) , Basic Scientific Research Program of Shenzhen City (Grant No. JCYJ20160331184440545), and Overseas Cooperation Research Fund of Graduate School at Shenzhen, Tsinghua University (Grant No. HW2018002).

References

[1] 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.

[2] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.

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

[4] Y. Zhang, Z. Gan, and L. Carin, "Generating text via adversarial training," in *NIPS workshop on Adversarial Training*, vol. 21, 2016.

[5] P. O. Hoyer, "Non-negative matrix factorization with sparseness constraints," *Journal of machine learning research*, vol. 5, no. Nov, pp. 1457–1469, 2004.

[6] 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.

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

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

[9] N. Liang, H.-T. Zheng, J.-Y. Chen, A. K. Sangaiah, and C.-Z. Zhao, "Trsdl: Tag-aware recommender system based on deep learning–intelligent computing systems," *Applied Sciences*, vol. 8, no. 5, p. 799, 2018.

[10] 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.

[11] R. Salakhutdinov and A. Mnih, "Bayesian probabilistic matrix factorization using markov chain monte carlo," in *Proceedings of the 25th international conference on Machine learning*. ACM, 2008, pp. 880–887.

[12] K. Yu, J. Lafferty, S. Zhu, and Y. Gong, "Large-scale collaborative prediction using a nonparametric random effects model," in *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM, 2009, pp. 1185–1192.

[13] 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.

[14] 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.

[15] S. Rendle, "Factorization machines," in *Data Mining (ICDM), 2010 IEEE 10th International Conference on*. IEEE, 2010, pp. 995–1000.

[16] X. He and T.-S. Chua, "Neural factorization machines for sparse predictive analytics," 2017.

[17] 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.

[18] C. J. Burges, R. Ragno, and Q. V. Le, "Learning to rank with nonsmooth cost functions," in *Advances in neural information processing systems*, 2007, pp. 193–200.

[19] C. Wang and D. M. Blei, "Collaborative topic modeling for recommending scientific articles," in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2011, pp. 448–456.

[20] J. McAuley and J. Leskovec, "Hidden factors and hidden topics: understanding rating dimensions with review text," in *Proceedings of the 7th ACM conference on Recommender systems*. ACM, 2013, pp. 165–172.

[21] G. Ling, M. R. Lyu, and I. King, "Ratings meet reviews, a combined approach to recommend," in *Proceedings of the 8th ACM Conference on Recommender systems*. ACM, 2014, pp. 105–112.

[22] Q. Diao, M. Qiu, C.-Y. Wu, A. J. Smola, J. Jiang, and C. Wang, "Jointly modeling aspects, ratings and sentiments for movie recommendation (jmars)," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2014, pp. 193–202.

[23] Y. Tan, M. Zhang, Y. Liu, and S. Ma, "Rating-boosted latent topics: Understanding users and items with ratings and reviews." in *IJCAI*, 2016, pp. 2640–2646.

[24] W. Chen, H.-T. Zheng, and X.-X. Mao, "Extracting deep semantic information for intelligent recommendation," in *International Conference on Neural Information Processing*. Springer, 2017, pp. 134–144.

[25] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *Journal of machine Learning research*, vol. 3, no. Jan, pp. 993–1022, 2003.

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

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

[28] L. Zheng, V. Noroozi, and P. S. Yu, "Joint deep modeling of users and items using reviews for recommendation," *web search and data mining*, pp. 425–434, 2017.

[29] 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," *arXiv preprint arXiv:1705.10513*, 2017.

[30] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.