Rating Augmentation with Generative Adversarial Networks towards Accurate Collaborative Filtering

Dong-Kyu Chae
Hanyang University
Seoul, Korea
kyu899@hanyang.ac.kr

Jin-Soo Kang
Hanyang University
Seoul, Korea
jensoo7023@hanyang.ac.kr

Sang-Wook Kim*
Hanyang University
Seoul, Korea
wook@hanyang.ac.kr

Jaeho Choi
NAVER Corporation
SeongNam, Korea
choi.jaeho@navercorp.com

ABSTRACT

Generative Adversarial Networks (GAN) have not only achieved a big success in various generation tasks such as images, but also boosted the accuracy of classification tasks by generating additional labeled data, which is called *data augmentation*. In this paper, we propose a Rating Augmentation framework with GAN, named RAGAN, aiming to alleviate the *data sparsity problem* in *collaborative filtering* (CF), eventually improving recommendation accuracy significantly. We identify a unique challenge that arises when applying GAN to CF for rating augmentation: naive RAGAN tends to generate values *biased towards high ratings*. Then, we propose a refined version of RAGAN, named RAGAN^{BT}, which addresses this challenge successfully. Via our extensive experiments, we validate that our RAGAN^{BT} is really effective to solve the data sparsity problem, thereby providing existing CF models with great improvement in accuracy under various situations such as basic top-N recommendation, long-tail item recommendation, and recommendation to cold-start users.

CCS CONCEPTS

• **Information systems → Recommender systems**.

KEYWORDS

Collaborative filtering, generative adversarial networks, data sparsity, data augmentation, top-N recommendation

ACM Reference Format:
Dong-Kyu Chae, Jin-Soo Kang, Sang-Wook Kim, and Jaeho Choi. 2019. Rating Augmentation with Generative Adversarial Networks towards Accurate Collaborative Filtering. In *Proceedings of the 2019 World Wide Web Conference (WWW '19), May 13–17, 2019, San Francisco, CA, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3308558.3313413

*Corresponding author

WWW '19, May 13–17, 2019, San Francisco, CA, USA

ACM ISBN 978-1-4503-6674-8/19/05.
https://doi.org/10.1145/3308558.3313413

1 INTRODUCTION

In recommender systems, *collaborative filtering* (CF) is one of the most widely used frameworks. The aim of CF is to generate a personalized item recommendation to a user based on her prior preference. Users' past preferences are typically presented in the form of a user-item *rating matrix*, where each cell value contain a rating given by a user for an item, mostly within a specific range (e.g., [1, 5]). However, the majority of entries in a rating matrix are missing: this *data sparsity problem* [13] is a universal obstacle to most CF methods, making them difficult to capture true relationships between users and items, thereby causing unsatisfactory accuracy in recommendation. To mitigate such a problem, most studies try to acquire additional information from social/trust networks among users [24] and auxiliary content of items [32]. However, their methods are applicable only when such additional data is available. The goal of this work is to investigate a generic solution to alleviate the sparsity issue in the context of pure CFs, i.e., a recommendation task given with only the user-item interaction data.

Recently, Goodfellow et al. [9] proposed the *Generative Adversarial Networks* (GAN), which learn to generate samples that mimic the distribution of the training data. Through an adversarial process between a *generative model* G and a *discriminator model* D, G learns to capture the data distribution in order to generate *synthetic but realistic* data. GAN has been very successful in image generation [25], and has gained popularity in performing other tasks, such as natural language generation [35] and music generation [7]. In addition, GAN's impressive ability to generate synthetic but realistic data has been used to improve the accuracy of a classification task, where the synthetic data is used as additional training data, which is called *data augmentation* [1, 2]. For example, [8] synthesized images labeled with Liver Lesion using GAN, resulting in 7% increase in terms of accuracy in Liver Lesion classification.

Motivated by such a success of data augmentation with GAN, this paper explores the *use of GAN for relieving the data sparsity problem in CF* under the context of data augmentation. We expect that using GAN to augment plausible ratings to a rating matrix would help CF models to provide more accurate recommendation by learning the augmented rating matrix, rather than the original one. Along this line, we propose a novel Rating Augmentation framework based on GAN, named RAGAN. RAGAN first trains GAN by using the observed ratings in a rating matrix as a training data, and

then makes its G generate plausible ratings for unobserved user-item interactions, which are to be augmented in the original rating matrix. Finally, conventional CF models can be trained by using the augmented rating matrix to produce final recommendations.

When applying this idea of data augmentation by GAN to the field of CF, we face a unique challenge that comes from the inherent *selection bias* of the rating data. We observed that only a small fraction (i.e., 10−17%) of ratings are low values (i.e., 1 or 2) and the rest are high values (i.e., 3, 4, or 5) from four real-life datasets used in our experiments, which has been also noticed in other studies [6, 12]. This might be because people have a tendency to select (and thus to rate) the items likely to give high satisfaction to them, and to disregard those items likely to result in low satisfaction (i.e., selection bias [6]). If GAN learns from the original rating matrix skewed to such high ratings, it would also generate *high ratings for most unobserved entries* in the matrix. However, this augmentation result seems not satisfactory since the majority of unobserved entries are likely to indicate users' negative preferences; in other words, users would not give a rating to those items that they are not interested in. If CF models are trained with such a *biased* augmented rating matrix, they would show poor recommendation accuracy since they fail to capture users' true preferences. We will demonstrate this phenomenon clearly through our experiments.

Towards this challenge, we propose RAGAN$^{\text{BT}}$, a refined version of RAGAN, which takes users' *negative items* (i.e., the items having users' negative preferences) additionally into account when training GAN, in order to avoid G from being biased towards high ratings in rating augmentation. Here, we formulate the problem of identifying such negative items as the well-known *one-class collaborative filtering* (OCCF) [23, 34] that infers the interestingness (or uninterestingness) of every user-item pair. By training GAN with the identified negative items as an additional source together with the original rating matrix, RAGAN$^{\text{BT}}$ successfully avoids the bias towards high ratings, and thus could generate the plausible ratings that better reflect the characteristics of unrated items. As a result, by augmenting the plausible ratings to the rating matrix, RAGAN$^{\text{BT}}$ solves the data sparsity problem, thereby contributing towards significant improvements of accuracy in CF.

In order to validate the effectiveness of our RAGAN$^{\text{BT}}$, we conduct extensive experiments using four real-world datasets. Notably, the results demonstrate that the key idea behind RAGAN$^{\text{BT}}$ (i.e., involving negative items in training GAN) is quite effective in generating more realistic ratings, leading to higher recommendation accuracy. The results also demonstrate that our RAGAN$^{\text{BT}}$ consistently and universally outperforms the state-of-the-art CF models not only under basic top-N recommendations, but also under more-difficult situations: (1) long-tail item recommendations and (2) recommendations to cold-start users. We believe our RAGAN$^{\text{BT}}$ sheds a light on the practical adoption of GAN for alleviating the data sparsity problem, which has been a universal obstacle to accuracy improvement in recommender systems.

2 PRELIMINARIES

Let two sets be denoted as $U = \{u_1, u_2, .., u_m\}$ and $I = \{i_1, i_2 ..., i_n\}$, where U consists of m users and I contains a set of n items. I_u is defined as a set of items that a user u has rated. A corresponding m-by-n rating matrix is $\mathbb{R}$, each entry r_{ui} is a rating on item i given by user u, and the ratings are within a specific range (e.g., [1, 5]). We denote $\mathbf{r_u}$ and $\mathbf{r_i}$ as rating vectors of user u and item i, respectively.

Given complicated real-world data, the *Generative Adversarial Network* (GAN) allows for a novel training method for generative models. GAN is made up of two models (neural networks in general), where one is a generative model, G, and the other is a discriminative model, D. The training process is a minimax game between G and D (i.e., $\min_G \max_D \mathcal{L}_{GAN}$): G tries to increase the error rate of D by generating seemingly authentic data, while D tries to accurately distinguish whether the data has come from the ground truth (i.e., real data) or from G. Formally, the objective function is:

$$\mathcal{L}_{GAN} = \mathbb{E}_{\mathbf{x}\sim p_{data}}[\log D(\mathbf{x})] - \mathbb{E}_{\hat{\mathbf{x}}\sim p_G}[\log(D(\hat{\mathbf{x}}))], \qquad (1)$$

where $\mathbf{x}$ is a ground-truth data from the data distribution p_{data} while $\hat{\mathbf{x}}$ is a synthetic data from the model distribution p_G, typically derived by $\hat{\mathbf{x}} = \text{G}(\mathbf{z})$ where $\mathbf{z}$ is a random noise vector. D$(\cdot)$ indicates the estimated probability of its input being a ground truth [4].

3 RAGAN

The main idea of this paper is to train GAN with the rating data, generate plausible ratings, and augment the generated values in the rating matrix. This section illustrates a basic solution to this objective, called RAGAN (rating augmentation framework based on GAN). As a first step, we need to train GAN by using the rating data. In this context, there exist several recently proposed GANs that could be proper candidates for our framework, such as IRGAN [33], GraphGAN [31], and CFGAN [3], all of which focus on recommendation tasks. Among them, we note that both IRGAN and GraphGAN can only learn from users' implicit feedback data, and expanding these methods to also learn from explicit ratings (i.e., the degree of satisfaction) and generate such data is non-trivial. In contrast, CFGAN can be easily extended to learn from and generate the rating data, even though it was also proposed to perform CF based on implicit feedback dataset. For this reason, our GAN for rating data is based on CFGAN.

Formally, $\mathcal{L}_{GAN}$ in our work is defined as:

$$\begin{aligned}\mathcal{L}_{GAN} &= \mathbb{E}_{\mathbf{x}\sim P_{data}}[lnD(\mathbf{x}|c)] - \mathbb{E}_{\hat{\mathbf{x}}\sim P_G}[lnD(\hat{\mathbf{x}}|c)] \\ &= \frac{1}{m}\Big(\Sigma_u lnD(\mathbf{r_u}|\mathbf{r_u}) - \Sigma_u lnD(\hat{\mathbf{r}}_\mathbf{u} \odot \mathbf{e_u}|\mathbf{r_u})\Big), \qquad (2)\end{aligned}$$

where $\hat{\mathbf{r}}_\mathbf{u}$ is an n-dimensional rating vector generated by G corresponding to a user u and $\mathbf{r_u}$ is her real rating vector (i.e., ground truth). $\hat{\mathbf{r}}_\mathbf{u}$ is derived by $\hat{\mathbf{r}}_\mathbf{u} = G(\{\mathbf{z}, \mathbf{r_u}\})$ where {} indicates the concatenation of two vectors inside. From the viewpoint of *conditional GAN*[22], $\mathbf{r_u}$ corresponds to a user u's specific condition; our G generates output conditioned by $\mathbf{r_u}$. Likewise, conditioned by $\mathbf{r_u}$, D outputs a single scalar value representing the probability that its input came from the real rating data, rather than from G. Both G and D are represented by neural networks. $\mathbf{e_u}$ is an n-dimensional indicator vector specifying whether user u has given a rating to item i (e_{u_i}=1) or not (e_{u_i}=0), and $\odot$ stands for element-wise multiplication. $\mathbf{e_u}$ is used to mask G's output $\hat{\mathbf{r}}_\mathbf{u}$ by multiplying it with $\mathbf{e_u}$, and thus the model parameters are learned based solely on the observed ratings and will be used later to generate potential (yet unobserved) user-item ratings.

We employ the *stochastic gradient descent* (*and ascent*) with mini-batch and back-propagation to train our G (and D). We update the model parameters of G and D alternately, keeping one fixed when

Table 1: Accuracy with/without RAGAN (P@5)

	ML100K	Ciao	Watcha	ML1M
AutoRec	.0696	.0191	.0272	.0727
RAGAN+AutoRec	.0543	.0150	.0210	.0805
SVD	.0976	.0128	.0384	.0419
RAGAN+SVD	.1016	.0125	.0321	.0444
ItemKNN	.0326	.0079	.0071	.0362
RAGAN+ItemKNN	.0339	.0079	.0049	.0297

the other is being trained. After training of GAN is completed, we feed $\mathbf{z}$[1] and $\mathbf{r_u}$ for G to generate a dense rating vector $\hat{\mathbf{r}}_\mathbf{u}$, which contains predicted ratings on all the items for user u. Among all the predicted ratings, we collect element j in $\hat{\mathbf{r}}_\mathbf{u}$ where $j \in I \setminus I_u$, i.e., the predicted ratings only for the unrated items, and augment the original rating matrix $\mathbb{R}$ with those ratings.

4 RAGAN^{BT}

4.1 Motivation

We first point out a unique challenge of treating biases in RAGAN. We note that the ratings in our datasets are skewed towards high ratings (i.e., 3, 4, or 5 that implies *positive preferences*), coinciding with the results in other studies [6, 12]. This is mainly because people tend to choose (thus, to give ratings to) only the items that they are likely to love, and never to give ratings to those items that do not attract their interest. We claim that this *selection bias* is the main reason to make GAN trained to generate the values for unobserved entries *biased towards high ratings*. As a result, the CF models learning the rating matrix augmented with such biased ratings would fail to capture users' true preferences.

In order to make the above challenge clearer, we conducted preliminary experiments with four real-world datasets. We compare two *precisions* of top-5 recommendations performed by CF methods: one obtained by learning from the original rating matrix and the other from the rating matrix augmented with the synthetic ratings generated by RAGAN. As CF methods, we employ ItemKNN [27], SVD [16] and AutoRec [28], which are very popular ones among memory-based methods, MF-based models, and DNN-based models, respectively.

As shown in Table 1, the effect of the rating augmentation does not seem that positive: RAGAN did not help improve the accuracy in all cases and even worsened it in some cases. We analyzed the results and found the reason as shown Figure 1. First, in the original rating matrix, we observe severe *imbalance* between low (i.e., 1 or 2) and high (i.e., 3, 4, or 5) ratings from all the datasets: only a small fraction (i.e., 10–17%) of ratings are 1 and 2 which indicate negative preferences, and the rest are 3, 4, and 5 which indicate positive preferences. Since RAGAN is trained with the data biased to those high ratings, it generates mostly high ratings as in Figure 1. We note, however, this result is quite far from the reality: the majority of unobserved entries would imply users' negative preferences since users would not give a rating to those items that they are not interested in [6, 13]. Consequently, the CF methods learning the augmented matrix having unrealistic ratings could not provide accuracy improvement as shown in Table 1.

[1]In practice, however, we did not use the random noise vector $\mathbf{z}$ as done in [3].

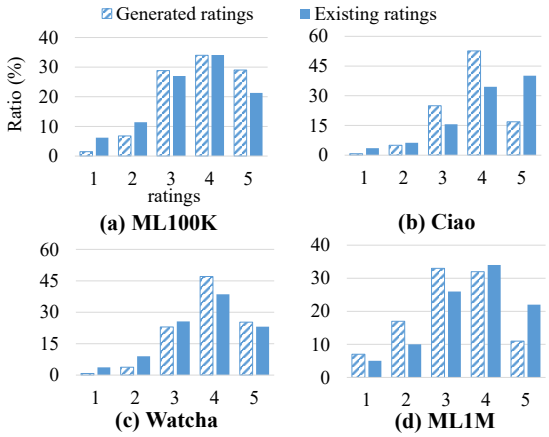

Figure 1: Rating distributions.

4.2 Overview of RAGAN^{BT}

A challenge then arises: how can we prevent GAN from being biased towards high ratings, thus generating realistic ratings for accurate CF? As a solution to this challenge, we now propose RAGAN^{BT}: a refined RAGAN with *bias treatment*. Figure 2 shows an overview of our RAGAN^{BT}, composed of the following three steps: first, we identify the *negative items* for users in a rating matrix (Figure 2(a)); we add low ratings to the negative items and use the newly added ratings along with the original ratings to train GAN (Figure 2(b)); finally, G in the trained GAN generates future ratings, which are to be augmented in the rating matrix (Figure 2(c)). We will elaborate the details of each step in the following sections.

4.3 Finding Negative Items

If user u's rating on item i is missing in a rating matrix, it will be due to one of the following reasons [11, 21, 29]:

- **lack of awareness**: u might prefer i, but was not aware of the existence of i. So, u could not rate i.
- **lack of interest**: u was aware of the existence of i, but was not interested in i because u has a low preference on i. So, u did not buy i, thus not rating i.

Obviously, a user's negative items would be her items unrated due to the latter reason. Along this line, our goal is to capture a user's interest on her unrated items and to select the items on which she has low interest. To this end, we employ the *one-class collaborative filtering* (OCCF) framework [23] following several recent work [13, 17, 19, 20]: The *implicit feedback* of a user does not always indicate her eventual preference on items represented by ratings, but naturally implies her interest on items *before using them* [10, 13, 23]; therefore, OCCF can be seen as a task of analyzing a user's interest on her unrated items and then finding top-N items that are likely to be of interest to her. Note that we have the goal *opposite* to that of OCCF: while the goal in OCCF is to find the items *most interesting* to a user, our goal is to find the items *least interesting* to her.

There have been numerous OCCF methods proposed; for example, BPR [26], FISM [14], and CDAE [34]. Among them, we employ Collaborative Denoising Auto-Encoder (CDAE), which is a DNN-based latent factor model and has shown a state-of-the-art accuracy

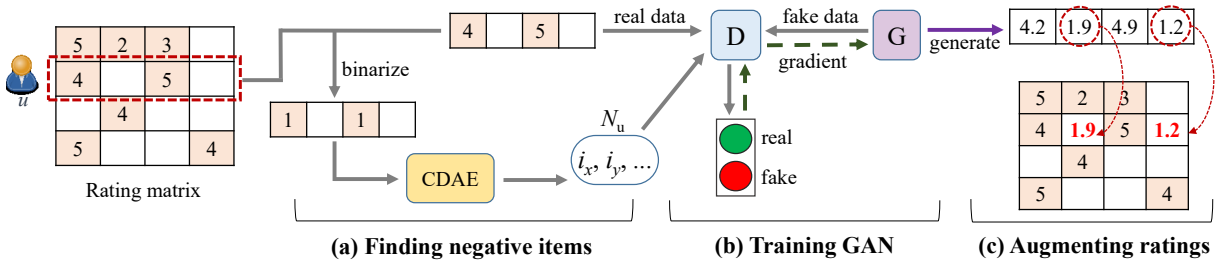

Figure 2: Overview of our proposed RAGAN^{BT}.

under OCCF setting. CDAE's input is a sparse implicit feedback vector of a user u, $\mathbf{r}_u^b$ (i.e., u's binarized ratings). This vector is first corrupted by masking noise: formally, the corrupted input vector, $\tilde{\mathbf{r}}_u^b$, is drawn by making some fraction (chosen at random for each user) of observed entries of $\mathbf{r}_u^b$ 0. Then, fed with $\tilde{\mathbf{r}}_u^b$, CDAE maps it to the hidden layer as $\mathbf{h} = \sigma(W_1^T \tilde{\mathbf{r}}_u^b + \mathbf{V_u} + \mathbf{b_1})$ where $\mathbf{V_u}$ is user u's latent feature vector. Finally, it tries to reconstruct the initial input (i.e., not corrupted one) by $\hat{\mathbf{r}}_u^b = \sigma(W_2\mathbf{h} + \mathbf{b_2})$. The model parameters of CDAE (i.e., W_1, b_1, W_2, b_2, and V) are learned by optimizing the objective function in the following:

$$\arg\min_{\psi} \frac{1}{m} \sum_{u=1}^{m} \mathbb{E}_{p(\tilde{\mathbf{r}}_u^b | \mathbf{r}_u^b)} [\ell(\mathbf{r}_u^b, \hat{\mathbf{r}}_u^b)] + \Omega(\psi), \tag{3}$$

where $\ell(\cdot)$ is a loss function and $\Omega(\cdot)$ is a term for so-called L2 regularization. While $\ell(\cdot)$ can be either square loss or logistic loss, we choose square loss that provides better empirical results in our work. We use ψ to denote a collection of all edge weights (W_1 and W_2) and biases ($\mathbf{b_1}$ and $\mathbf{b_2}$) in the network.

After ψ is learned, CDAE is fed with $\mathbf{r}_u^b$, and then reconstructs a *dense vector* that contains predicted scores of interestingness on all *unrated* items for user u. We now define a user u's *negative items*, N_u, as the bottom S percent of unrated items whose predicted scores of interestingness are the lowest [13].

4.4 Training GAN with Negative Items

We now present how we leverage those negative items to GAN's learning process. Our goal is to prevent G from being biased towards high ratings. To this end, we add a *low rating* for the negative items to $\mathbf{r_u}$ which is fed to D, as a new ground truth vector that G tries to mimic. By doing so, G would try to generate not only high ratings on the observed entries in $\mathbf{r_u}$ but also low ratings on the identified negative items in order to deceive D. Thus, we expect that G will no longer be biased towards high ratings, being able to learn users' positive and negative preferences *together in a balanced way*.

Let δ denote a low rating to be added. There are three choices for δ assigned to $\mathbf{r_u}$: 0, 1, and 2. Assigning ratings of 0 (i.e., $\delta = 0$) poses strong implication that a user does not like the negative items at all, while assigning ratings of 2 (i.e., $\delta = 2$) indicates a less severe uninterestingness towards the negative items. For each user u given, we add ratings of δ for N_u to u's real rating vector $\mathbf{r_u}$ (i.e., $r_{uj} \leftarrow \delta$, $\forall j \in N_u$), and then the imputed $\mathbf{r_u}$ is fed with D as a ground truth vector that G tries to mimic.

4.5 Recommendation

Once the training of GAN is complete, we feed G with $\mathbf{z}$ and $\mathbf{c_u}$ to generate plausible ratings. Then, we fill all missing entries in $\mathbb{R}$ with those generated ratings. We denote this augmented rating matrix as $\mathbb{R}'$. Finally, any CF models such as ItemKNN, SVD, and AutoRec can be trained to predict the preferences on the unobserved entries in $\mathbb{R}$ in their own way by using $\mathbb{R}'$ as training data. Finally, for each user, we pick the top-N items having the highest predicted preferences for recommendation.

5 EVALUATION

5.1 Experimental Settings

For evaluation, we used four real-world datasets of Watcha[2], Ciao [30], Movielens 100K, and Movielens 1M[3] (shortly, ML100K and ML1M). Table 2 summarizes their detailed statistics. For each dataset, we randomly select 80% of ratings for training data and the rest 20% for testing. In addition, we again set aside 20% of the training data for validation in order to use it for tuning hyper-parameters, such as the number of hidden layers and hidden nodes in G and D. Among the ratings in the test and validation sets, we only consider the ratings of 5 as our ground truth [5, 18]. Also, we employed four popular accuracy metrics: *precision* (P@N), *recall* (R@N), *normalized discounted cumulative gain* (G@N), and *mean reciprocal rank* (M@N). We set N as 5 and 20[4].

Table 2: Statistics of datasets

Datasets	# users	# items	# ratings	Sparsity
Ciao	996	1,927	18,648	98.72%
Watcha	1,391	1,927	101,073	96.98%
ML100K	943	1,682	100,000	93.69%
ML1M	6,039	3,883	1,000,209	95.72%

5.2 Experimental Results and Analyses

5.2.1 Q1: Effectiveness of our idea of bias treatment. Our key contribution is to address a unique challenge arises when data augmentation by GAN is applied to CF, and to suggest RAGAN^{BT}, a solution to this challenge. To show the effectiveness of this solution in rating augmentation, we visualize the accuracies of ItemKNN, SVD, and AutoRec that learn from two different rating matrices: (1) $\mathbb{R}'$ augmented by RAGAN and (2) $\mathbb{R}'$ augmented by RAGAN^{BT}.

Figure 3 reports the learning curves provided by each method *A+B* where *A* indicates a rating augmentation method and *B* does a CF model. We observe that RAGAN^{BT} makes great improvements in accuracy for all three CF models: for instance, RAGAN^{BT}+AutoRec

[2]http://watcha.net
[3]https://grouplens.org/datasets/movielens/
[4]Throughout the paper, however, we report the result of P@5 due to space limitations. We observed that the results of the other metrics exhibited very similar tendency.

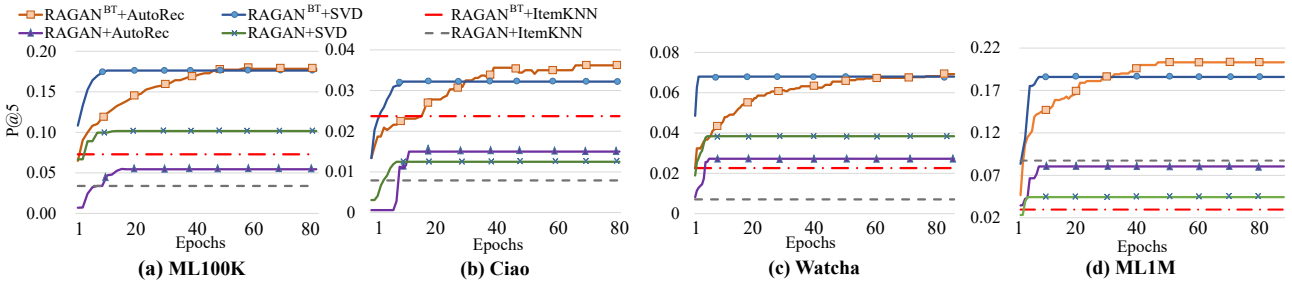

Figure 3: Learning curves (P@5).

provides up to 141.3% improvement, compared to RAGAN+AutoRec on Ciao and up to 228.5% on ML100K. This great improvement is mainly because RAGAN^{BT} generates plausible ratings by treating successfully the bias towards high ratings, thereby alleviating the data sparsity problem by augmenting those ratings.

Figure 4 reports the distributions of ratings generated by RAGAN^{BT}. Compared to the distribution of ratings generated by naive RAGAN, which is shown in Figure 1, most of generated ratings for unrated items are shown lower than or equal to 2. This result coincides with the observation that people tend to have low preferences on most of their unrated items [6, 13]. As a result, RAGAN^{BT} helps CF models to better understand users' true preferences, thereby improving the accuracy of top-N recommendation significantly.

5.2.2 Q2: Accuracy of RAGAN^{BT} according to varying hyper-parameters. This subsection investigates the impact of the following key hyper-parameters, which are unique in our RAGAN^{BT} framework: (1) the portion of negative items, S, for bias treatment and (2) which value to use as δ. Here, we employ SVD as our CF model. The results on Ciao and ML1M are shown only due to limited space.

Figure 5 reports the experimental results, where the three graphs in each figure represent the P@5 value over varying S when δ is given to negative items with 0, 1, and 2, respectively. We observe that the proposed framework works well when δ is 0 and 1. We also observe that when S reaches sufficiently large number (e.g., 25%), the accuracy of RAGAN^{BT} converges and does not change much after that point. These results show that setting S as a value moderately larger than 25% ensures CF models to understand users' positive and negative preferences in a balanced way, owing to our bias treatment. Thus, in the following experiments, we fix $\delta = 0$ and $S = 25$, and then compare it with the state-of-the-art methods.

5.2.3 Q3: Comparisons with state-of-the-arts. We now compare the accuracy of RAGAN^{BT} with those of the state-of-the-art top-N recommenders. Notably, we performed the comparisons not only under basic top-N recommendation, but also under the following two more-difficult settings: (1) long-tail items (i.e., items having a few ratings) recommendation and (2) recommendation to cold-start users (i.e., users giving a few ratings) [13]. To set up the first situation, we divided all items, by their popularity, into *top-head* items[5] that involve 30% of all ratings and *long-tail* items that are the rest. Then, we recommended only long-tail items to users. To create the second setting, we randomly selected 200 users as cold-start users. Then, we also randomly selected five ratings for each

[5] 140, 107, 101, and 202 items are included as the *top-head* items in Ciao, Watcha, ML100K, and ML1M, respectively.

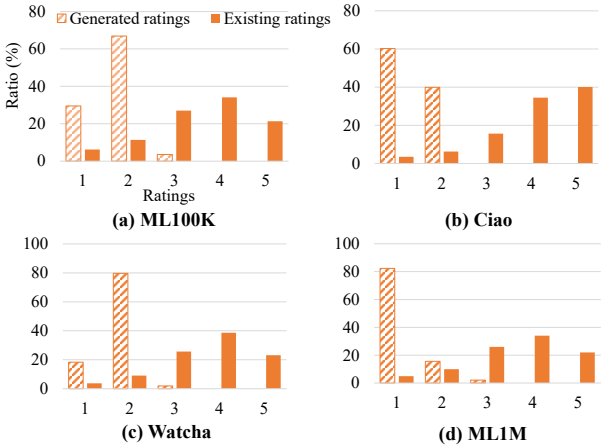

Figure 4: Rating distributions.

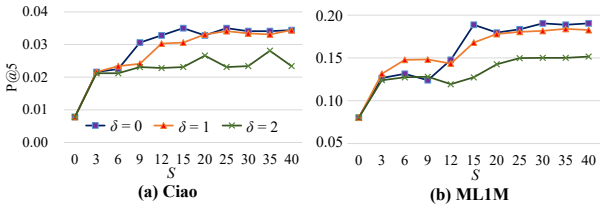

Figure 5: Impact of key hyper-parameters (P@5).

cold-start user and discarded the rest of her ratings. We finally recommended items to the cold-start users only.

Next, we introduce our competitors: (1) ItemPop, the simplest non-personalized method that ranks items based on the descending order of popularity; (2) ItemKNN, a popular neighborhood based method [27]; (3) SVD, a standard MF-based model along with user and item bias terms [16]; (4) SVD++, a combined idea of an MF and a neighborhood based method by approximating similarities among items via MF model [15]; (5) AutoRec, an autoencoder based, non-linear latent factor model [28]. Furthermore, we include two models that employ the *zero rating imputation*: PureSVD [5] and Zero-Injection [13]. Notably, Zero-Injection is the most similar to ours: both Zero-Injection and ours find a set of negative items, but after then Zero-Injection runs SVD to directly produce recommendations while our RAGAN^{BT} runs GAN to generate ratings to be imputed. PureSVD is also similar to ours: it regards *all unrated items* as negative ones and imputes zeros to them before running SVD.

Table 3: Accuracy comparisons with the state-of-the-art CF methods (P@5)

Datasets	Ciao			ML100K			Watcha			ML1M		
Situations	Basic	Long-tail	Cold-start	Basic	Long-tail	Cold-start	Basic	Long-tail	Cold-start	Basic	Long-tail	Cold-start
ItemPop	.0209	.0050	.0185	.0685	.0210	.0390	.0256	.0088	.0244	.1071	.0322	.0498
ItemKNN	.0079	.0075	.0064	.0326	.0262	.0159	.0087	.0071	.0099	.0362	.0199	.0284
SVD	.0128	.0072	.0164	.0976	.0312	.0431	.0384	.0204	.0221	.0419	.0155	.0390
SVD++	.0197	.0063	.0174	.0910	.0297	.0455	.0324	.0207	.0240	.0755	.0256	.0520
PureSVD	.0278	.0131	.0185	.1466	.0753	.0572	.0573	.0377	.0290	.1483	.0923	.0620
AutoRec	.0191	.0078	.0174	.0696	.0324	.0440	.0272	.0212	.0242	.0727	.0231	.0511
Zero-Injection	.0347	.0197	.0256	.1429	.0617	.0601	.0672	.0389	.0255	.1856	.0926	.0600
RAGAN^{BT}+ItemKNN	.0237	.0206	.0231	.0729	.0637	.0573	.0369	.0288	.0346	.1035	.0346	.0610
RAGAN^{BT}+SVD	**.0366**	.0197	.0256	.1762	.0880	.0660	.0680	.0423	**.0350**	.1860	**.0997**	**.0780**
RAGAN^{BT}+AutoRec	.0362	**.0267**	**.0277**	**.1784**	**.0884**	**.0670**	**.0701**	**.0426**	.0330	**.2032**	.0957	.0680

Table 3 reports the comparison results. Overall, we see that AutoRec and SVD along with our RAGAN^{BT} significantly outperform all the other competitors. In particular, we beat PureSVD and Zero-Injection, which have shown the best accuracy among all CF models, by a wide margin: the average of relative improvements of our RAGAN^{BT}+AutoRec over Zero-Injection is 15.8%. We note that PureSVD and Zero-Injection can be considered as simplified versions of our work: they assign 0 ratings to all the unrated items (PureSVD) or to carefully chosen negative items (Zero-Injection), and then run SVD to produce recommendation without GAN for further imputation. However, their important drawback is that they do not take users' *personalized preferences* on items into account, just adding the *same zero* ratings to all (PurSVD) and some (Zero-Injection) unrated items. In contrast, we exploit GAN so as to generate more realistic and personalized ratings carefully. Our experimental results indeed verify the effectiveness of our RAGAN^{BT} over the two competitors: it successfully relieves the data sparsity problem by carefully generating plausible ratings, rather than simply imputing zero ratings as PureSVD and Zero-Injection did.

Another notable point is that RAGAN^{BT} is more successful under the long-tail and cold-start settings, which are more-difficult situations than basic top-N recommendation. More specifically, RAGAN^{BT}+AutoRec performs well under the long-tail setting on Ciao (35.5% improved, compared to the best performer among our competitors) and ML100K (17.3% improved), and RAGAN^{BT}+SVD performs well under the cold-start setting on Watcha (20.7% improved) and ML1M (25.8% improved). It can be a very promising result since those difficult situations are very common in most real-world recommender systems. Therefore, our RAGAN^{BT} could be a good choice for those situations suffering from lots of cold-start users and long-tail items.

5.2.4 Q4: Execution time analysis. Lastly, we analyzed our proposed RAGAN^{BT} in terms of the execution time. Specifically, we measured each time consumed by (1) finding negative items with CDAE, (2) augmenting ratings with GAN, and (3) recommending items with AutoRec, which has achieved higher accuracy than SVD and ItemKNN. Here, we used a single machine equipped with an i9 7700K Intel CPU, 64GB RAM, and NVIDIA TITAN XP GPU. The results are summarized in Table 4. For the largest dataset (i.e., ML1M), total time required is about 6,891 seconds: each epoch of CDAE takes around 8 seconds, and needs 150 epochs to sample appropriate negative items for each user; each epoch of GAN takes approximately 12.5 seconds, and needs 450 epochs to generate plausible ratings; finally, AutoRec needs 100 epochs to get satisfactory accuracy where each epoch takes about 1 second. Note that the total execution time consumed by SVD++, which is one of our baselines, is around 5,078 seconds. In summary, we see that our framework augments plausible ratings and provides high-quality recommendations within a reasonable time. This is because, even though our RAGAN^{BT} involves three procedures of model training, all of them are based on neural networks whose training can be easily parallelized by fast GPU.

Table 4: Execution time (s)

	Total	Finding negative items (**CDAE**)	Augmenting ratings (**GAN**)	Recommending items (**AutoRec**)
Ciao	1,062.76	123.49	927.11	6.08
Watcha	2,060.70	222.69	1,827.21	10.80
ML100K	2,500.42	382.08	2,102.03	16.31
ML1M	6,891.66	1,164.88	5,623.58	103.20

6 CONCLUSIONS

In this paper, we have presented RAGAN, a novel rating augmentation framework based on GAN, to relieve the data sparsity problem in CF. Then, we have identified a challenge of treating the biases to high ratings that GAN faces when learning from the rating data. We have proposed RAGAN^{BT}, which successfully addresses the challenge by involving negative items when training GAN. From our comprehensive experiments, we have demonstrated that RAGAN^{BT} significantly improves the accuracies of existing CF methods, making them outperform the existing state-of-the-art CF methods, not only under basic top-N recommendations but also under more-difficult situations of long-tail item recommendations and recommendations to cold-start users.

ACKNOWLEDGMENTS

This work was supported by (1) the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT: Ministry of Science and ICT) (No. NRF-2017R1A2B3004581) and (2) Next-Generation Information Computing Development Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science and ICT (No. NRF-2017M3C4A7083678). Also, we thank the Naver Corporation for their support including computing environment and data, which helped us greatly in performing this research successfully.

REFERENCES

[1] Antreas Antoniou, Amos Storkey, and Harrison Edwards. 2017. Data augmentation generative adversarial networks. *arXiv preprint arXiv:1711.04340* (2017).

[2] Konstantinos Bousmalis, Nathan Silberman, David Dohan, Dumitru Erhan, and Dilip Krishnan. 2017. Unsupervised pixel-level domain adaptation with generative adversarial networks. In *Proceedings of the IEEE international conference on computer vision and pattern recognition*, Vol. 1. 7.

[3] Dong-Kyu Chae, Jin-Soo Kang, Sang-Wook Kim, and Jung-Tae Lee. 2018. CFGAN: A generic collaborative filtering framework based on generative adversarial networks. In *Proceedings of the 27th ACM international conference on information and knowledge management*. 137–146.

[4] Edward Choi, Siddharth Biswal, Bradley Malin, Jon Duke, Walter F Stewart, and Jimeng Sun. 2017. Generating multi-label discrete electronic health records using generative adversarial networks. *arXiv preprint arXiv:1703.06490* (2017).

[5] Paolo Cremonesi, Yehuda Koren, and Roberto Turrin. 2010. Performance of recommender algorithms on top-n recommendation tasks. In *Proceedings of the 4th ACM conference on recommender systems*. 39–46.

[6] Nilesh N Dalvi, Ravi Kumar, and Bo Pang. 2013. Para'normal'activity: on the distribution of average ratings.. In *Proceedings of the 7th international AAAI conference on weblogs and social media*. 110–119.

[7] Chris Donahue, Julian McAuley, and Miller Puckette. 2018. Synthesizing audio with generative adversarial networks. *arXiv preprint arXiv:1802.04208* (2018).

[8] Maayan Frid-Adar, Idit Diamant, Eyal Klang, Michal Amitai, Jacob Goldberger, and Hayit Greenspan. 2018. GAN-based synthetic medical image augmentation for increased CNN performance in Liver Lesion classification. *arXiv preprint arXiv:1803.01229* (2018).

[9] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*. 2672–2680.

[10] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings of the 26th international conference on World Wide Web*. 173–182.

[11] José Miguel Hernández-Lobato, Neil Houlsby, and Zoubin Ghahramani. 2014. Probabilistic matrix factorization with non-random missing data. In *Proceedings of the 31st international conference on machine learning*. 1512–1520.

[12] Nan Hu, Jie Zhang, and Paul A Pavlou. 2009. Overcoming the J-shaped distribution of product reviews. *Commun. ACM* 52, 10 (2009), 144–147.

[13] Won-Seok Hwang, Juan Parc, Sang-Wook Kim, Jongwuk Lee, and Dongwon Lee. 2016. "Told you i didn't like it": Exploiting uninteresting items for effective collaborative filtering. In *Proceedings of the IEEE 32nd international conference on data engineering*. 349–360.

[14] Santosh Kabbur, Xia Ning, and George Karypis. 2013. Fism: Factored item similarity models for top-n recommender systems. In *Proceedings of the 19th ACM SIGKDD international conference on knowledge discovery and data mining*. 659–667.

[15] Yehuda Koren. 2008. Factorization meets the neighborhood: A multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on knowledge discovery and data mining*. 426–434.

[16] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 42, 8 (2009).

[17] Jongwuk Lee, Won-Seok Hwang, Juan Parc, Youngnam Lee, Sang-Wook Kim, and Dongwon Lee. 2019. l-Injection: Toward effective collaborative filtering using uninteresting items. *IEEE transactions on knowledge and data engineering* 31, 1 (2019), 3–16.

[18] Sang-Chul Lee, Sang-Wook Kim, Sunju Park, and Dong-Kyu Chae. 2018. An approach to effective recommendation considering user preference and diversity simultaneously. *IEICE transactions on information and systems* 101, 1 (2018), 244–248.

[19] Youngnam Lee, Sang-Wook Kim, Sunju Park, and Xing Xie. 2018. How to impute missing ratings?: Claims, solution, and its application to collaborative filtering. In *Proceedings of the 27th international conference on World Wide Web*. 783–792.

[20] Yeon-Chang Lee, Sang-Wook Kim, and Dongwon Lee. 2018. gOCCF: Graph-theoretic one-class collaborative filtering based on uninteresting items. In *Proceedings of the 2018 AAAI international conference on artificial intelligence*.

[21] Dawen Liang, Laurent Charlin, James McInerney, and David M Blei. 2016. Modeling user exposure in recommendation. In *Proceedings of the 25th international conference on World Wide Web*. 951–961.

[22] Mehdi Mirza and Simon Osindero. 2014. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784* (2014).

[23] Rong Pan, Yunhong Zhou, Bin Cao, Nathan N Liu, Rajan Lukose, Martin Scholz, and Qiang Yang. 2008. One-class collaborative filtering. In *Proceedings of the 2008 IEEE 8th international conference on data mining*. 502–511.

[24] Chanyoung Park, Donghyun Kim, Jinoh Oh, and Hwanjo Yu. 2016. Improving top-k recommendation with truster and trustee relationship in user trust network. *Information Sciences* 374 (2016), 100–114.

[25] Alec Radford, Luke Metz, and Soumith Chintala. 2015. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434* (2015).

[26] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the 25th international conference on uncertainty in artificial intelligence*. 452–461.

[27] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*. 285–295.

[28] Suvash Sedhain, Aditya Krishna Menon, Scott Sanner, and Lexing Xie. 2015. Autorec: Autoencoders meet collaborative filtering. In *Proceedings of the 24th international conference on World Wide Web*. 111–112.

[29] Harald Steck. 2010. Training and testing of recommender systems on data missing not at random. In *Proceedings of the 16th international conference on knowledge discovery and data mining*. 713–722.

[30] Jiliang Tang, Huiji Gao, and Huan Liu. 2012. mTrust: Discerning multi-faceted trust in a connected world. In *Proceedings of the 5th ACM international conference on Web search and data mining*. 93–102.

[31] Hongwei Wang, Jia Wang, Jialin Wang, Miao Zhao, Weinan Zhang, Fuzheng Zhang, Xing Xie, and Minyi Guo. 2017. GraphGAN: Graph Representation Learning with Generative Adversarial Nets. *arXiv preprint arXiv:1711.08267* (2017).

[32] Hao Wang, Naiyan Wang, and Dit-Yan Yeung. 2015. Collaborative deep learning for recommender systems. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*. 1235–1244.

[33] Jun Wang, Lantao Yu, Weinan Zhang, Yu Gong, Yinghui Xu, Benyou Wang, Peng Zhang, and Dell Zhang. 2017. 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*. 515–524.

[34] Yao Wu, Christopher DuBois, Alice X Zheng, and Martin Ester. 2016. Collaborative denoising auto-encoders for top-n recommender systems. In *Proceedings of the 9th ACM international conference on Web search and data mining*. 153–162.

[35] Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2017. SeqGAN: Sequence generative adversarial nets with policy gradient.. In *Proceedings of the 2017 AAAI international conference on artificial intelligence*. 2852–2858.