RecGAN: Recurrent Generative Adversarial Networks for Recommendation Systems

Homanga Bharadhwaj
Indian Institute of Technology Kanpur
homangab@cse.iitk.ac.in

Homin Park
National University of Singapore
bighp@nus.edu.sg

Brian Y. Lim
National University of Singapore
brianlim@comp.nus.edu.sg

ABSTRACT

Recent studies in recommendation systems emphasize the significance of modeling latent features behind temporal evolution of user preference and item state to make relevant suggestions. However, static and dynamic behaviors and trends of users and items, which highly influence the feasibility of recommendations, were not adequately addressed in previous works. In this work, we leverage the temporal and latent feature modelling capabilities of Recurrent Neural Network (RNN) and Generative Adversarial Network (GAN), respectively, to propose a Recurrent Generative Adversarial Network (RecGAN). We use customized Gated Recurrent Unit (GRU) cells to capture latent features of users and items observable from short-term and long-term temporal profiles. The modification also includes collaborative filtering mechanisms to improve the relevance of recommended items. We evaluate RecGAN using two datasets on food and movie recommendation. Results indicate that our model outperforms other baseline models irrespective of user behavior and density of training data.

CCS CONCEPTS

• **Information systems** → **Recommender systems**; • **Computing methodologies** → *Neural networks*;

KEYWORDS

Recommendation Systems, Generative Adversarial Networks, Recurrent Neural Networks

ACM Reference Format:
Homanga Bharadhwaj, Homin Park, and Brian Y. Lim. 2018. RecGAN: Recurrent Generative Adversarial Networks for Recommendation Systems. In *Twelfth ACM Conference on Recommender Systems (RecSys '18), October 2–7, 2018, Vancouver, BC, Canada.* ACM, New York, NY, USA, Article 4, 5 pages. https://doi.org/10.1145/3240323.3240383

1 INTRODUCTION

Recent publications have illustrated the effectiveness of Recurrent Neural Networks (RNN) [11, 19, 21], Generative Adversarial Networks (GAN) [8], and Collaborative Filtering (CF) [11, 22] in RS. Modeling the temporal evolution of user preferences and item states for effective recommendation systems (RS) is an active area of research. The necessity of learning static (long-term) and dynamic (short-term) behaviors and trends of users and items has also been well recognized but not adequately addressed. Moreover, traditionally, RS has focused only on discriminative retrieval and ranking of items, which aims to judge the relevancy of an user-item pair [4, 13, 17, 20]. We believe such a scope limits the effective learning of comprehensive latent representations of/between users and items.

In this work, inspired by Recurrent Recommender Networks (RRN) [21] and Information Retrieval GAN (IRGAN) [20], we propose Recurrent Generative Adversarial Networks for Recommendation Systems (RecGAN) to improve recommendation performance by learning temporal latent features of user and item under the GAN framework. We adopt the generative modelling framework to learn both the relevancy distribution of items for users (generator) and to exploit the unlabelled sequence of generated relevant items to achieve a better estimate of relevancy (discriminator). Furthermore, we model temporal aspects found in time-series data using RNN, which also naturally fits into the GAN framework as evidenced by recent works [7, 15]. We customize the Gated Recurrent Unit (GRU) [3], a form of RNN cell, to model the latent features observable from short-term and long-term user behaviors and item states effectively. Specifically, we introduced the use of ReLU activation function in the update gate of the GRU cell, and added collaborative techniques into the GRU model. Such a design allows RecGAN to attain more fine-grained user and item models from complex latent relationships.

We evaluate our work using two datasets, MyFitnessPal (MFP) [1] and Netflix [21], across two different application domains, namely food and movie, to test the versatility of RecGAN. Note that MFP (diet logs) and Netflix (movie ratings) consist of implicit and explicit feedback of user preference, respectively. Our results show that RecGAN effectively handles both scenarios leveraging adversarial training of the custom-GRU based RNNs and surpasses the current state-of-the-art models in various evaluation metrics. There is around 10% increase in precision values of RecGAN over Probabilistic Matrix Factorization (PMF) [14] on the MFP dataset. In addition, RecGAN has over 1.4% increase in RMSE over RRN and around 3% increase over PMF on the Netflix dataset.

2 RELATED WORK

2.1 Recurrent Neural Networks for RS

The temporal aspect of user preferences and item states were previously discussed in numerous works [9, 10, 18, 19]. Two of the most recent publications, which inspired our work, dealing with temporal dynamics for RS include Donkers et al. [6] and RRN [21].

RecSys '18, October 2–7, 2018, Vancouver, BC, Canada

ACM ISBN 978-1-4503-5901-6/18/10…$15.00
https://doi.org/10.1145/3240323.3240383

Donkers et al. proposed a GRU based RNN for sequential modelling of user preferences. A rectified linear integration of user states is done in their modified GRU cell to evolve each user state for personalized recommendation. On the other hand, RRN targeted movie recommendation tasks by exploiting explicit feedback from users. Customized LSTM cells were used for modelling the function through which user and item latent states evolve. Devooght et al. [5] leveraged sequence information to mitigate the scarcity of user-item interactions and identify items that will be eventually preferred and that those that will be immediately desired. However, these methods have not been able to precisely identify time varying latent representations for users and items that are relevant for temporal recommendations.

2.2 Generative Adversarial Networks for RS

Generative Adversarial Networks (GAN) were first introduced in 2014 and demonstrated their effectiveness in deep generative modelling. IRGAN [20] was the first paper to propose the use of this mini-max game framework to train Information Retrieval (IR) systems, including RS. Their evaluation results show significant gains over state-of-the-art baselines from different IR domains. However, a simple matrix factorization is used for the discriminator and generator without considering the temporal aspects of latent features. We extend and modify the GAN framework by incorporating a temporal modelling technique as described in following section.

3 THE RECGAN FRAMEWORK

Exploiting the temporal modelling capabilities of RNN and the latent feature modelling power of GAN we design a mini-max game framework for effective RS. Both the generator and the discriminator exploit GRU based RNN since it has lower computational overload than LSTM, and is more effective with small training data. This is desirable because data sparsity is one of the major bottlenecks for RS. The term *ratings* in the context of this paper is used both for explicit (e.g. movie ratings) and implicit (e.g. diet logs of repeated foods) feedback. For the implicit feedback, the rating represents the likelihood of a user consuming an item (e.g., food). It is important to note that for successful adversarial training, the generator and the discriminator must have equally sound modelling power and hence both of them are designed to have a similar recurrent architecture as described in Sections 3.1 and 3.2.

3.1 The Generator

For every user u, the ground-truth temporal preference distribution is denoted as $\mathbf{Y}^u = [\mathbf{y}_1^u, ..., \mathbf{y}_t^u, ..., \mathbf{y}_T^u]$ where $\mathbf{y}_t^u$ is a vector representing the ratings of user u over all items at time t. To handle both temporally static and dynamic preference of the users, we use Leaky ReLU activation in the update gate $\mathbf{x}_t^u$ of the GRU cell as shown in Eq. 1. This forces the model to place greater importance on past trends for users who show static preference trends and make routine choices. It also implies that, users who show dynamic preference trends and frequently make spontaneous choices (indicated by diverse preferences over time) would be automatically modelled favoring the collaborative aspect of the GRU cells described subsequently [11]. This is because the ReLU update gate will fire more strongly towards forgetting past trends. The GRU forward equations are as follows:

$$\mathbf{x}_t^u = RELU(\mathbf{W}_{xh}^u \mathbf{h}_{t-1}^u + \mathbf{W}_{xk}^u \mathbf{y}_t^u) \quad (1)$$

$$\mathbf{r}_t^u = \sigma(\mathbf{W}_{rh}^u \mathbf{h}_{t-1}^u + \mathbf{W}_{rk}^u \mathbf{y}_t^u) \quad (2)$$

$$\mathbf{m}_t^u = tanh(\mathbf{W}_h(\mathbf{r}_t^u \cdot \mathbf{h}_{t-1}^u) + \mathbf{W}_{input} \mathbf{y}_t^u) \quad (3)$$

$$\mathbf{h}_t^u = (1 - \mathbf{x}_t^u) \cdot \mathbf{h}_{t-1}^u + \mathbf{x}_t^u \cdot \mathbf{m}_t^u \quad (4)$$

where $\mathbf{r}_t^u$ is the reset gate of the GRU cell with sigmoid (σ) non-linearity, and $\mathbf{h}_t^u$ is the hidden state at time t. Furthermore, $\mathbf{W}_{xh}$, $\mathbf{W}_{xk}$, $\mathbf{W}_{rh}$ and $\mathbf{W}_{rk}$ denote weight matrices for the respective update x and reset r gates. Subscript h denotes the weight matrix for the corresponding hidden state while k denotes the weight matrix for the input $\mathbf{y}_t^u$.

Ultimately, the generator predicts a sequence of items that are highly likely to be consumed by the user u from $t = 0$ to T (one item per t). This is denoted as $\mathbf{Gen}^u = \mathbf{W}_{output}\mathbf{H}_g^u$, where $\mathbf{H}_g^u = [\mathbf{h}_1^u, ..., \mathbf{h}_t^u, ..., \mathbf{h}_T^u]$. During training, $\mathbf{Gen}^u$ is iteratively inferred and the model is trained by providing the ground-truth till a time index say $t_{train} < T$. So, the output $\mathbf{Gen}^u$ from $t = 0$ to $t = t_{train}$ is reconstruction and that from $t = t_{train}$ to $t =$ T is prediction. During testing, the recommendation can be made based on the $\mathbf{Gen}^u$ at the required time instances. Note that $\mathbf{W}_{input}$ and $\mathbf{W}_{output}$ matrices are invariant across users. This allows us to learn correlation between users' preferences over items and model dynamic temporal user preferences by modifying the GRU exploiting collaborative filtering and Leaky ReLU activation in the update gate.

3.2 The Discriminator

The Discriminator is another GRU based RNN that determines the probability of $\mathbf{Gen}^u$ being sampled from the true underlying distribution of users' preference over all items. The GRU forward equations for the discriminator are shown in Eq. 5 to 8. Note that variables and weight matrices are differentiated from the generator using the hat notation.

$$\hat{\mathbf{x}}_t^u = RELU(\hat{\mathbf{W}}_{xh}^u \hat{\mathbf{h}}_{t-1}^u + \hat{\mathbf{W}}_{xk}^u \mathbf{y}_t^u) \quad (5)$$

$$\hat{\mathbf{r}}_t^u = \sigma(\hat{\mathbf{W}}_{rh}^u \hat{\mathbf{h}}_{t-1}^u + \hat{\mathbf{W}}_{rk}^u \mathbf{y}_t^u) \quad (6)$$

$$\hat{\mathbf{m}}_t^u = tanh(\hat{\mathbf{W}}_h(\hat{\mathbf{r}}_t^u \cdot \hat{\mathbf{h}}_{t-1}^u) + \hat{\mathbf{W}}_{input} \mathbf{y}_t^u) \quad (7)$$

$$\hat{\mathbf{h}}_t^u = (1 - \hat{\mathbf{x}}_t^u) \cdot \hat{\mathbf{h}}_{t-1}^u + \hat{\mathbf{x}}_t^u \cdot \hat{\mathbf{m}}_t^u \quad (8)$$

The output of the discriminator, the probability of $\mathbf{Gen}^u$ being sampled from the true underlying distribution, is given by $\mathbf{Dis}^u = \sigma(\mathbf{V}_{output}\mathbf{H}_d^u)$, where $\mathbf{H}_d^u = [\hat{\mathbf{h}}_1^u, ..., \hat{\mathbf{h}}_T^u]$ is the vector of hidden states. The final sigmoid non-linearity in discriminator (not present in generator) is to constrain its output in the range $(0, 1)$ so that it is representative of a probability value. Over the course of training, the probability converges to 0.5, which means that the discriminator can no longer successfully discriminate between the generated samples of user-ratings over items and the real ratings.

3.3 The Mini-Max Game

As proposed in [8], we let the generator and the discriminator play a minimax game. The belief is that at each time index t, there exists a true underlying distribution of user-preferences over items, let us call it $D_{real|t}$. At this time index, the generator *generates* ratings for each user (i) and item (j) pair which we denote by $(r|i, j, t)$. The generator has in effect *learned* a distribution, which we call $D_{gen|t}$.

Algorithm 1 The Adversarial Training Procedure

Input: $\mathbf{Gen}_\Theta(r|i,j,t)$, $\mathbf{Dis}_\Phi(r|i,j,t)$, S_g, S_d
Data: Training Sequence of user ratings over items
Initialize: $\mathbf{Gen}_\Theta(r|i,j,t)$ and $\mathbf{Dis}_\Phi(r|i,j,t)$ with
random parameters Θ, Φ
while Convergence criteria not reached **do**
  for S_g number of steps **do**
    $\mathbf{Gen}_\Theta(r|i,j,t)$ generates rating values
    $\Theta \leftarrow$ UpdateParams(**Gen**)
  for S_d number of steps **do**
    $\mathbf{Dis}_\Phi(r|i,j,t)$ assigns likelihood value to the rating values being from the true distribution
    $\Phi \leftarrow$ UpdateParams(**Dis**)

In fact, what we observe and have access to is a sample from this distribution, $D_{gen|t}$ [2]. Now, the minimax game for our context can be formulated as follows

$$Q^* = \min_{\mathbf{Gen}} \max_{\mathbf{Dis}} \sum_{t=1}^{T}\sum_{i=1}^{N}\sum_{j=1}^{M}\Big(\mathbb{E}_{r\sim D(r|i,j)_{real|t}}[\log \mathbf{Dis}(r|i,j,t)] + \mathbb{E}_{r\sim D(r|i,j)_{gen|t}}[\log(1-\mathbf{Dis}(r|i,j,t))]\Big) \tag{9}$$

Here $Dis(r|i,j,t)$ refers to the discriminator's prediction of the likelihood of $(r|i,j,t)$ being sampled from the true distribution $D(r|i,j)_{real|t}$. The users are numbered from 1 to N, the items are numbered from 1 to M and time is indexed from 1 to T.

3.4 The Training Method

We use the conventional Backpropagation Through Time (BPTT) for training RecGAN [15]. During training, the parameters of the generator and the discriminator are alternately updated in each iteration. The training procedure is summarized in Algorithm 1. Here, S_g and S_d are the number of training iterations per epoch for the generator and the discriminator, respectively. Let Θ denote all the parameters of the generator and Φ denote all the parameters of the discriminator. The discriminator is optimized as shown.

$$\Phi^* = \arg\max_{\Phi} \sum_{t=1}^{T}\sum_{i=1}^{N}\sum_{j=1}^{M}\Big(\mathbb{E}_{r\sim D(r|i,j)_{real|t}}[\log \mathbf{Dis}(r|i,j,t)] + \mathbb{E}_{r\sim D(r|i,j)_{\Theta^*|t}}[\log(1-\mathbf{Dis}(r|i,j,t))]\Big) \tag{10}$$

where Φ is updated by BPTT after Θ has been updated by BPTT (Θ^* is the updated Θ). In each training iteration, while updating the parameters of the discriminator, the parameters of the generator are held constant and vice versa. The optimization problem for the generator is as follows.

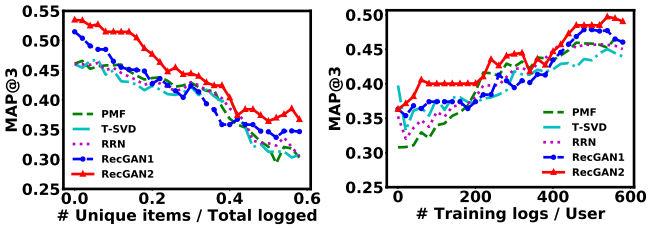
$$\Theta^* = \arg\min_{\Theta} \sum_{t=1}^{T}\sum_{i=1}^{N}\sum_{j=1}^{M}\Big(\mathbb{E}_{r\sim D(r|i,j)_{\Theta|t}}[\log(1-\mathbf{Dis}(r|i,j,t))]\Big) \tag{11}$$

Table 1: Hyperparameter values of the GRU for RecGAN2 for all the three datasets. Gradient cap is to trim large gradients that may cause a skip over optima.

	Batch Size	Hidden Units	Layers	Learning Rate	Gradient Cap
MFP	1500	1200	1	0.001	15.0
Netflix-6m	1000	1000	1	0.0005	12.0
Netflix-full	1200	1000	1	0.001	10.0

Table 2: Analysis of baseline models and RecGAN2 on the MFP dataset using MAP, NDCG and MRR. Higher is better.

	U-AR	I-AR	T-SVD	PMF	RRN	RecGAN2
MAP@3	0.36	0.36	0.38	0.37	0.37	**0.41**
MAP@5	0.34	0.33	0.35	0.35	0.36	**0.39**
NDCG@3	0.41	0.40	0.42	0.45	0.44	**0.50**
NDCG@5	0.40	0.38	0.41	0.43	0.43	**0.47**
MRR	0.34	0.34	0.35	0.38	0.40	**0.42**



(a) Impact of the # unique items logged over total # items logged **(b) Impact of the # training logs per user for the MFP dataset**

Figure 1: Performance of RecGAN 1 and 2 against various baseline models on MFP dataset using MAP@3 demonstrating the effectiveness of RecGAN.

4 SIMULATION RESULTS

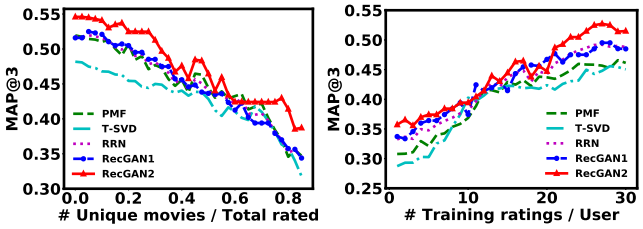
4.1 Setup

To evaluate the performance of RecGAN, we use two datasets from Netflix Challenge [21] and MyFitnessPal [1] of explicit and implicit feedback respectively. For each dataset, we compare RecGAN against various baselines dealing with temporal aspects of RS on several standard metrics. The baseline models are trained using open-source code [21], and suitable hyper-parameters for each model are identified using a grid search (Table 1). We split the dataset into training, validation, and test set by 85%, 7.5%, and 7.5%, respectively, while maintaining the chronological order. In the subsequent tables and figures, RecGAN1 denotes our model without the GRU modifications while RecGAN2 (used interchangeably with RecGAN) denotes the final model with GRU modifications. All the values reported in the tables are on the respective test sets.

4.2 MyFitnessPal Dataset

To illustrate the viability of RecGAN, we employ a diet log dataset scraped from MyFitnessPal, which was introduced in [1]. In total, 587,187 food diary records of 9,896 users were extracted. On average, the total number of food entries per user is 652.9. The baseline models in this evaluation include PMF [14], TimeSVD++ (T-SVD) [12], RRN [21] and AutoRec (I-AR and U-AR) [16].

Figure 1a shows a decreasing trend as the ratio of the number of unique items logged to the total number of items logged increases



(a) Impact of the # unique movies rated over total # ratings (b) Impact of the # training ratings per user

Figure 2: Performance of RecGAN 1 and 2 against baseline models on Netflix dataset using MAP@3 demonstrating the effectiveness of RecGAN.

due to a search-space increase owing to large combinations of potentially consumable items. When the ratio is small on a 0 to 1 scale, we interpret it as an indication of routine behavior due to static user preferences. On the contrary, users who tend to have dynamic preference trends will make diverse choices and have a larger ratio as they consume a diverse range of items. We postulate that RecGAN's robustness in handling routine and diverse choice, as evident from Figure 1a, is due to greater modelling flexibility by virtue of adversarial training having an implicit objective (unlike a maximum likelihhod objective). In addition, as mentioned in Section 3, we believe that the use of ReLU non-linearity in the update gate of RecGAN's GRU cells boosts performance by effectively weighing the two often competing effects of collaborative recommendation and leveraging of personal history.

We evaluate RecGAN using MAP, NDCG and MRR in Table 2, which shows that RecGAN has around 10% increase in MAP value and 9% increase in NDCG value over PMF and RRN. Figure 1b shows that RecGAN performs significantly better than PMF and T-SVD both for users who rate a lot and those that rate infrequently. This illustrates the effectiveness of RecGAN in addressing the cold-start problem (data sparsity) and also in leveraging dense data when the number of ratings per user is large.

4.3 Netflix Challenge Dataset

The Netflix Challenge Dataset [21] is one of the most popular benchmark datasets used to evaluate the performance of RS. It includes 100M ratings collected between 1999 and 2005 where each data point is a tuple of user ID, item ID, timestamp, and rating.

The RMSE results shown in Table 3 indicate that RecGAN outperforms state-of-the-art RRN by 1.4% on 6-months and by 0.5% on the full dataset. We selected Netflix-6months and Netflix-full to evaluate performance on both a moderately sparse dataset (6-months) and a relatively dense dataset (full). In addition, we evaluate RecGAN using MAP, NDCG and MRR in Tables 4, 5 and Figure 2, which show that RecGAN has a superior performance on all the evaluation metrics. We did not benchmark RecGAN against IRGAN, because IRGAN does not handle time-series data for recommendation. Figure 2a shows the relative performance of RecGAN surpasses the baseline models by greater than 10% in most cases. As discussed in Section 4.2, the non-linearity of ReLU in the update gate is postulated to play a significant role by being highly selective towards remembering and forgetting past trends of users who show differing behavior. When the past-trends are given less weight (forgotten) by

Table 3: RMSE of baseline models and RecGAN2 on Netflix challenge dataset. Lower is better.

	U-AR	I-AR	T-SVD	PMF	RRN	RecGAN2
Netflix-6m	0.983	0.977	0.958	0.958	0.943	**0.929**
Netflix-full	0.965	0.936	0.927	0.925	0.922	**0.918**

Table 4: Analysis of baseline models and RecGAN2 on MAP, NDCG and MRR for the Netflix-full dataset. Higher is better.

	U-AR	I-AR	T-SVD	PMF	RRN	RecGAN2
MAP@3	0.35	0.35	0.36	0.37	0.39	**0.41**
MAP@5	0.34	0.33	0.34	0.35	0.37	**0.38**
NDCG@3	0.51	0.49	0.52	0.52	0.54	**0.57**
NDCG@5	0.49	0.47	0.51	0.50	0.55	**0.55**
MRR	0.34	0.33	0.32	0.34	0.37	**0.39**

Table 5: Comparison of different versions of RecGAN on test data. Vanilla GRU denotes a plain GRU based RNN without the GAN framework. Higher is better.

		Vanilla GRU Without GAN	RecGAN1 GRU + GAN	RecGAN2 GRU + GAN
Netflix full	**MAP@3**	0.33	0.40	**0.41**
	MAP@5	0.30	0.37	**0.38**
	NDCG@3	0.46	0.55	**0.57**
	NDCG@5	0.46	0.53	**0.55**
	MRR	0.34	0.36	**0.39**
MFP	**MAP@3**	0.33	0.38	**0.41**
	MAP@5	0.30	0.37	**0.39**
	NDCG@3	0.37	0.47	**0.50**
	NDCG@5	0.37	0.44	**0.47**
	MRR	0.35	0.38	**0.42**

the model, naturally the collaborative aspect determines the type of recommendation.

Table 5 shows the improvement in recommender performance with the GRU modifications compared to Vanilla GRU. A similar performance trend shown in Figure 1b is evident on Netflix dataset, as illustrated in Figure 2b. Since GAN is an implicit density model [8], it successfully captures relevant information from both high-volume and low-volume data distributions, respectively corresponding to users who rate a lot and those who rate infrequently.

5 CONCLUSION AND FUTURE WORKS

In this work, we discussed the details of RecGAN, which was designed to improve the effectiveness of RS by accurately modelling temporal dynamics of user and item latent features. Our evaluation results demonstrate that RecGAN outperforms all baseline models in two different RS domains, namely movie and food recommendation. To further investigate and improve the performance of RecGAN, we are planning to conduct in-depth studies on the feasibility and impact of GRU modification, which was done to capture the temporal patterns of varying granularity. In particular, we intend to explore gate-variants of the GRU cell for use in the RecGAN framework. In addition, we also plan to expand the framework to handle auxiliary data like review texts, images and audio signals by modifying the generator and discriminator networks to include convolutional layers and bi-directional GRU-RNNs.

REFERENCES

[1] Palakorn Achananuparp and Ingmar Weber. 2016. Extracting food substitutes from food diary via distributional similarity. *arXiv preprint arXiv:1607.08807* (2016).

[2] Sanjeev Arora, Rong Ge, Yingyu Liang, Tengyu Ma, and Yi Zhang. 2017. Generalization and equilibrium in generative adversarial nets (gans). *arXiv preprint arXiv:1703.00573* (2017).

[3] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* (2014).

[4] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*. ACM, 191–198.

[5] Robin Devooght and Hugues Bersini. 2017. Long and Short-Term Recommendations with Recurrent Neural Networks. In *Proceedings of the 25th Conference on User Modeling, Adaptation and Personalization*. ACM, 13–21.

[6] Tim Donkers, Benedikt Loepp, and Jürgen Ziegler. 2017. Sequential User-based Recurrent Neural Network Recommendations. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*. ACM, 152–160.

[7] Cristóbal Esteban, Stephanie L Hyland, and Gunnar Rätsch. 2017. Real-valued (medical) time series generation with recurrent conditional GANs. *arXiv preprint arXiv:1706.02633* (2017).

[8] 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.

[9] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2015. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939* (2015).

[10] Balázs Hidasi, Massimo Quadrana, Alexandros Karatzoglou, and Domonkos Tikk. 2016. Parallel recurrent neural network architectures for feature-rich session-based recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*. ACM, 241–248.

[11] Young-Jun Ko, Lucas Maystre, and Matthias Grossglauser. 2016. Collaborative recurrent neural networks for dynamic recommender systems. In *Asian Conference on Machine Learning*. 366–381.

[12] Yehuda Koren. 2010. Collaborative filtering with temporal dynamics. *Commun. ACM* 53, 4 (2010), 89–97.

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

[14] Andriy Mnih and Ruslan R Salakhutdinov. 2008. Probabilistic matrix factorization. In *Advances in neural information processing systems*. 1257–1264.

[15] Olof Mogren. 2016. C-RNN-GAN: Continuous recurrent neural networks with adversarial training. *arXiv preprint arXiv:1611.09904* (2016).

[16] Yuanxin Ouyang, Wenqi Liu, Wenge Rong, and Zhang Xiong. 2014. Autoencoder-based collaborative filtering. In *International Conference on Neural Information Processing*. Springer, 284–291.

[17] Steffen Rendle. 2010. Factorization machines. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on*. IEEE, 995–1000.

[18] Nachiketa Sahoo, Param Vir Singh, and Tridas Mukhopadhyay. 2012. A hidden Markov model for collaborative filtering. *Mis Quarterly* (2012), 1329–1356.

[19] Hao Wang, SHI Xingjian, and Dit-Yan Yeung. 2016. Collaborative recurrent autoencoder: recommend while learning to fill in the blanks. In *Advances in Neural Information Processing Systems*. 415–423.

[20] 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*. ACM, 515–524.

[21] Chao-Yuan Wu, Amr Ahmed, Alex Beutel, Alexander J Smola, and How Jing. 2017. Recurrent recommender networks. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*. ACM, 495–503.

[22] Yao Wu, Christopher DuBois, Alice X Zheng, and Martin Ester. 2016. Collaborative denoising auto-encoders for top-n recommender systems. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*. ACM, 153–162.