CnGAN: Generative Adversarial Networks for Cross-network user preference generation for non-overlapped users

Dilruk Perera
School of Computing
National University of Singapore
dilruk@comp.nus.edu.sg

Roger Zimmermann
School of Computing
National University of Singapore
rogerz@comp.nus.edu.sg

ABSTRACT

A major drawback of cross-network recommender solutions is that they can only be applied to users that are overlapped across networks. Thus, the non-overlapped users, which form the majority of users are ignored. As a solution, we propose CnGAN, a novel multi-task learning based, encoder-GAN-recommender architecture. The proposed model synthetically generates source network user preferences for non-overlapped users by learning the mapping from target to source network preference manifolds. The resultant user preferences are used in a Siamese network based neural recommender architecture. Furthermore, we propose a novel user-based pairwise loss function for recommendations using implicit interactions to better guide the generation process in the multi-task learning environment. We illustrate our solution by generating user preferences on the Twitter source network for recommendations on the YouTube target network. Extensive experiments show that the generated preferences can be used to improve recommendations for non-overlapped users. The resultant recommendations achieve superior performance compared to the state-of-the-art cross-network recommender solutions in terms of accuracy, novelty and diversity.

CCS CONCEPTS

• **Information systems → Recommender systems**; **Personalization**; **Multimedia and multimodal retrieval**.

KEYWORDS

Cross-network Recommendations; Generative Adversarial Networks; Collaborative Filtering; Deep learning; Implicit Feedback

ACM Reference Format:
Dilruk Perera and Roger Zimmermann. 2019. CnGAN: Generative Adversarial Networks for Cross-network user preference generation for non-overlapped users. 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/10.1145/3308558.3313733

1 INTRODUCTION

Cross-network recommender systems utilize auxiliary information from multiple source networks to create comprehensive user profiles for recommendations on a target network. Therefore, unlike traditional recommender solutions which are limited to information within a single network, cross-network solutions are more robust against cold-start and data sparsity issues [17]. For example, the additional information from Twitter and Flicker, increased recommender precision on Delicious by 10% [1]. Similarly, the transfer of information from various source networks to target networks such as Google+ to YouTube [10], Twitter to YouTube [20, 24] and Wikipedia to Twitter [18] have consistently improved recommender accuracy. Furthermore, cross-network solutions allow user preferences to be captured from diverse perspectives, which increases the overall recommender quality in terms of diversity and novelty [20, 21]. However, despite the growing success of cross-network recommender solutions, the majority of existing solutions can only be applied to users that exist in multiple networks (overlapped users). The remaining non-overlapped users, which form the majority are unable to enjoy the benefits of cross-network solutions.

Deep learning and generative modeling techniques have been successfully used in the recommender systems domain in the past few years. For example, restricted Boltzmann machines [26], Autoencoders [19], Hidden Markov Models [25] and Recurrent Neural Networks (RNN) [13] are some of the models used for rating prediction tasks. Recent advancements in Generative Adversarial Networks (GANs) have also drawn recommender systems researchers to apply the new minimax game framework to information retrieval. For example, IRGAN [28] and RecGAN [5] were designed to predict relevant documents to users. Existing solutions use GAN to generate simple rating values [5, 28, 30] or synthetic items (e.g., images [14]). In contrast, we use GAN to generate complex user data (i.e., user preferences on the source network).

We propose a novel GAN model (CnGAN) to generate cross-network user preferences for non-overlapped users. Unlike the standard GAN model, we view the learning process as a mapping from target to source network preference manifolds. The proposed model solves two main tasks. First, a generator task learns the corresponding mapping from target to source network user preferences and generates auxiliary preferences on the source network. Second, a recommender task uses the synthetically generated preferences to provide recommendations for users who only have interactions on the target network. Furthermore, we propose two novel loss functions to better guide the generator and recommender tasks. The proposed solution is used to provide recommendations for YouTube users by incorporating synthesized preferences on Twitter. However, the solution is generic and can be extended to incorporate multiple source networks.

In this paper, we first briefly provide preliminaries to the proposed model and present the proposed CnGAN solution detailing its two main components, the generator and recommender tasks. Then, we compare the proposed model against multiple baselines to demonstrate its effectiveness in terms of accuracy, novelty and diversity. We summarize our main contributions as follows:

- To the best of our knowledge, this is the first attempt to apply a GAN based model to generate missing source network preferences for non-overlapped users.
- We propose CnGAN, a novel GAN based model which includes a novel content loss function and user-based pairwise loss function for the generator and recommender tasks.
- We carry out extensive experiments to demonstrate the effectiveness of CnGAN to conduct recommendations for non-overlapped users and improve the overall quality of recommendations compared to state-of-the-art methods.

2 MODEL

2.1 Model Preliminaries

2.1.1 Bayesian Personalized Ranking (BPR):. Learning user preferences from implicit feedback is challenging since implicit feedback does not indicate preference levels toward items, it only indicates the presence or absence of interactions with items. BPR [23] was proposed to rank items based on implicit feedback using a generic optimization criterion. BPR is trained using pairwise training instances $S = \{(u, i, j) | u \in U \wedge i \in I_{u+} \wedge j \in I \setminus I_{u+}\}$, where I is the set of all items and I_{u+} is the set of items user u has interacted with. BPR uses a pairwise loss function to rank interacted items higher than non-interacted items, and maximizes their margin using a pairwise loss as follows:

$$L_{BPR}(S) = \sum_{(u,i,j) \in S} - \ln \sigma(\hat{r}_{uij}) + \lambda_{\Theta} \|\Theta\|^2 \tag{1}$$

where $\hat{r}_{uij} = \hat{r}_{ui} - \hat{r}_{uj}$, Θ are model parameters, $\sigma(x) = e^x / e^{x+1}$ is a sigmoid function and λ_{Θ} are regularization parameters.

2.1.2 Feature extraction on a continuous space: CnGAN is trained using target network interactions of each user at each time interval and their corresponding source network interactions. We used topic modeling to capture user interactions on a continuous topical space since CnGAN is a GAN based model and requires inputs in a continuous space. Let $u_o \in U = [Tn_u^t, Sn_u^t]$ denote an overlapped user at time t, where Tn_u^t and Sn_u^t are target and source network interactions spanned over $T = [1, \dots, t]$ time intervals. A non-overlapped user $u_{no} \in U = [Tn_u^t]$ is denoted only using target network interactions. We used YouTube as the target and Twitter as the source network to conduct video recommendations on YouTube. Therefore, Tn_u^t is the set of interacted videos (i.e., liked or added to playlists) and Sn_u^t is the set of tweets. We assumed that each interaction (video or tweet) is associated with multiple topics, and extracted the topics from the textual data associated with each interaction (video titles, descriptions and tweet contents). We used the Twitter-Latent Dirichlet Allocation (Twitter-LDA) [32] for topic modeling since it is most effective against short and noisy contents. Based on topic modeling, each user u on the target network is represented as a collection of topical distributions $tn_u = \{\boldsymbol{tn_u^1}; \dots ; \boldsymbol{tn_u^t}\} \in \mathbb{R}^{T \times K^t}$ over T time intervals, where K^t is the number of topics. Each vector $\boldsymbol{tn_u^t} = \{tn_u^{t,1}; \dots ; tn_u^{t,K^t}\} \in \mathbb{R}^{K^t}$ is the topical distribution at time t, where $tn_u^{t,k} = \hat{tn}_u^{t,k} / \sum_{c=1}^{K^t} \hat{tn}_u^{t,c}$ is the relative frequency of topic k, and $\hat{tn}_u^{t,k}$ is the absolute frequency of the corresponding topic. Similarly, on the source network, interactions are represented as $sn_u = \{\boldsymbol{sn_u^1}; \dots ; \boldsymbol{sn_u^t}\} \in \mathbb{R}^{T \times K^t}$. The resultant topical frequencies indicate user preference levels towards corresponding K^t topics. Therefore, $u = \{tn_u, sn_u\} \in \mathbb{R}^{T \times 2K^t}$ represents user preferences over T time intervals on a continuous topical space, and forms the input to the proposed model.

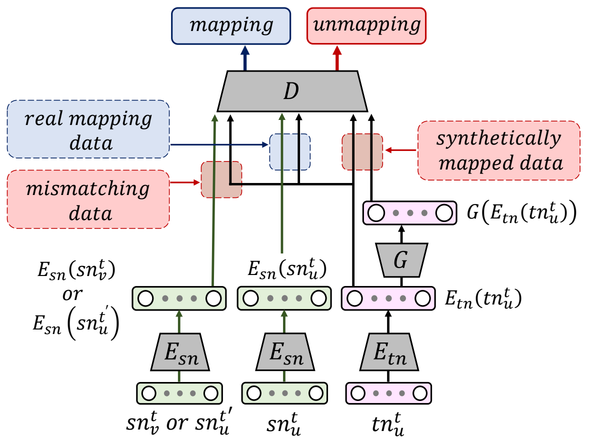

Figure 1: Adversarial learning process between E, D and G.

2.2 Generator Task

The generator task learns the function that maps the target network preferences manifold to the source network preferences manifold using Encoders (E), Discriminator (D) and Generator (G).

2.2.1 Encoders: User preferences captured as topical distribution vectors ($\boldsymbol{tn_u^t}$ and $\boldsymbol{sn_u^t}$) become sparse when $\boldsymbol{K^t}$ is set to a sufficiently large value to capture finer level user preferences, and/or the number of interactions at a time interval is low. One of the most effective methods to train machine learning models with highly sparse data is to model interactions between input features [7] (e.g., using neural networks [12]). Therefore, we used two neural encoders E_{tn} and E_{sn} for target and source networks to transform the input topical distributions to dense latent encodings. The resultant encodings are represented as $E_{tn}(tn_u) \in \mathbb{R}^{T \times En}$ and $E_{sn}(sn_u) \in \mathbb{R}^{T \times En}$, where En is the dimensionality of the latent space. These encodings form the input to the generator task.

2.2.2 Generator task formulation: For a non-overlapped user u, let $E_{tn}(\boldsymbol{tn_u^t})$ denote the target network encoding at time interval t. The generator task aims to synthetically generate the mapping source network encoding that closely reflects the missing source network encoding $E_{sn}(\boldsymbol{sn_u^t})$. Hence, G uses target encoding $E_{tn}(\boldsymbol{tn_u^t})$ and attempts to generate the mapping source encoding to fool D. Similarly, D tries to differentiate between real source encoding $E_{sn}(\boldsymbol{sn_u^t})$ and generated encoding $G(E_{tn}(\boldsymbol{tn_u^t}))$. Note that, we refer to the actual and generated target and source network encodings of non-overlapped users $\left(E_{tn}(\boldsymbol{tn_u^t}), G(E_{tn}(\boldsymbol{tn_u^t}))\right)$ as *synthetically mapped data* and the actual target and source network encodings of overlapped users $\left(E_{tn}(\boldsymbol{tn_u^t}), E_{sn}(\boldsymbol{sn_u^t})\right)$ as *real mapping data.*

2.2.3 Discriminator: Analogous to D in standard GANs, the real and synthetically mapped pairs could be used to learn D. However, D may only learn to differentiate between actual and generated source network encodings in a given pair of inputs, without learning to check if the given pair is a mapping pair. Therefore, during training, we input mismatching source and target network encodings to ensure that an effective mapping is learned. We modified the loss function for D to minimize output scores for mismatching pairs and maximize output scores for matching pairs. For a given target network encoding, we drew mismatching source encodings only from real data to avoid potential biases. Mismatching pairs were created by pairing real encodings from different users at the same time interval or by pairing encodings from the same user at different time intervals (see Figure 1). Accordingly, the loss function for D is formed as follows:

$$\begin{aligned}&\max_{E_{tn},E_{sn},D} V(E_{tn},E_{sn},D)\\&=\mathbb{E}_{\boldsymbol{tn_u^t},\boldsymbol{sn_u^t}\sim p_{data}(tn,sn)}L_{real}\Big(E_{tn}(\boldsymbol{tn_u^t}),E_{sn}(\boldsymbol{sn_u^t})\Big)\\&+\mathbb{E}_{\boldsymbol{tn_u^t}\sim p_{data}(tn)}L_{fake}\Big(E_{tn}(\boldsymbol{tn_u^t}),G\Big(E_{tn}(\boldsymbol{tn_u^t})\Big)\Big)\\&+\mathbb{E}_{\boldsymbol{tn_u^t},\overline{\boldsymbol{sn}}_u^t\sim \overline{p}_{data}(tn,\overline{sn})}L_{mismatch}\Big(E_{tn}(\boldsymbol{tn_u^t}),E_{sn}(\overline{\boldsymbol{sn}}_{\boldsymbol{u}}^{\boldsymbol{t}})\Big)\end{aligned} \tag{2}$$

where $p_{data}(tn,sn)$ and $\overline{p}_{data}(tn,\overline{sn})$ are matching and mismatching target and source network topical distributions, $p_{data}(tn)$ are the target network topical distributions and $G(x)$ is the generated matching source network encoding for the given target network encoding x. Furthermore, $L_{real}(x,y)=[\log(D(x,y))]$, $L_{fake}(x,y)=[1-\log(D(x,y))]$ and $L_{mismatch}(x,y)=[1-\log(D(x,y))]$, where $D(x,y)$ is the probability that x and y are matching target and source network encodings. Therefore, D and E work together to maximize the discriminator value function $V(E_{tn},E_{sn},D)$. The encoders (E_{tn} and E_{sn}) assist the process by encoding the topical distributions that are optimized for learning D (e.g., extract encodings that express the latent mapping between target and source networks).

2.2.4 Generator: Once D and E are trained, they adversarially guide G by effectively identifying real and synthetically generated data. Specifically, G takes in a target network encoding $E_{tn}(\boldsymbol{tn_u^t})$ and synthetically generates the mapping source network encoding that resemble $E_{sn}(\boldsymbol{sn_u^t})$ drawn from real mapping data.

GAN models are typically used to generate real-like images from random noise vectors. For a given noise vector, different GANs can learn multiple successful mappings from the input space to the output space. In contrast, CnGAN aims to learn a specific mapping from target to source network encodings. Hence, for given pairs of real mapping data $(\boldsymbol{tn_u^t},\boldsymbol{sn_u^t})$, G minimizes an additional content loss between the generated $G(E_{tn}(\boldsymbol{tn_u^t}))$ and real $E_{sn}(\boldsymbol{sn_u^t})$ source encodings to provide additional guidance for G. Therefore, we formulated the loss function for G as follows:

$$\begin{aligned}&\min_G V(G)=\mathbb{E}_{\boldsymbol{tn_u^t}\sim p_{data}(tn)}L_{fake}\Big(E_{tn}(\boldsymbol{tn_u^t}),G\Big(E_{tn}(\boldsymbol{tn_u^t})\Big)\Big)\\&+\mathbb{E}_{\boldsymbol{sn_u^t},\boldsymbol{tn_u^t}\sim p_{data}(tn,sn)}L_{content}\Big(E_{sn}(\boldsymbol{sn_u^t}),G\Big(E_{tn}(\boldsymbol{tn_u^t})\Big)\Big)\end{aligned} \tag{3}$$

where $L_{content}(x,y)=\|x,y\|_1$ is the content loss computed as the $\ell1$-norm loss between real and generated encodings.

2.3 Recommender Task

The recommender task uses the generated preferences to conduct recommendations for non-overlapped users.

2.3.1 Recommender task formulation: Let $E_{tn}(\boldsymbol{tn_u^t}),G(E_{tn}(\boldsymbol{tn_u^t}))$ denote a non-overlapped user u at time t based on synthetically mapped data, which reflects current preferences. To capture previous preferences, we used a previous interaction vector $R_u^{t-}\in\mathbb{R}^M$, where M is the number of items. Each element $r_{u,i}^{t-}\in\boldsymbol{R_u^{t-}}$ is set to 1 if the user had an interaction with item i before time t, and 0 if otherwise. Thus, we formulated the recommender task as a time aware Top-N ranking task where, current and previous user preferences are used to predict a set of N items that the user is most likely interact with, in the next time interval $t+1$.

2.3.2 Recommender loss: Typically, BPR optimization is performed using Stochastic Gradient Descent (SGD) where, for each training instance (u,i,j), parameter (Θ) updates are performed as follows:

$$\begin{aligned}\Theta &\leftarrow \Theta-\alpha\frac{\partial}{\partial\Theta}L_{BPR}(S)\\ \Theta &\leftarrow \Theta+\alpha\left(\frac{e^{-\hat{r}_{uij}}}{1+e^{-\hat{r}_{uij}}}\cdot\frac{\partial}{\partial\Theta}\hat{r}_{uij}+\lambda_\Theta\Theta\right)\end{aligned} \tag{4}$$

where α is the learning rate, $\hat{r}_{uij}=\hat{r}_{ui}-\hat{r}_{uj}$ is the pairwise loss for a given (u,i,j) triplet and $\hat{r}_{ui}$ is the predicted rating for user u and item i. Since BPR is a generic optimization criterion, $\hat{r}_{ui}$ can be obtained using any standard collaborative filtering approach (e.g., MF). Considering the rating prediction function in MF ($\hat{r}_{ui}=\sum_{f=1}^{K}\boldsymbol{w_{uf}}\cdot\boldsymbol{h_{if}}$, where $\boldsymbol{w_{uf}}\in\mathbb{R}^K$ and $\boldsymbol{h_{if}}\in\mathbb{R}^K$ are latent representations of user u and item i), $\hat{r}_{uij}$ for MF based BPR optimization can be defined as $\hat{r}_{uij}=\sum_{f=1}^{K}w_{uf}\cdot(h_{if}-h_{jf})$.

For each (u,i,j) instance, three updates are performed when $\Theta=w_{uf}$, $\Theta=h_{if}$ and $\Theta=h_{jf}$. For each update on the latent user representation (w_{uf}), two updates are performed on the latent item representations (h_{if} and h_{jf}). Hence, the training process is biased towards item representation learning, and we term the standard BPR optimization function as an *item-based* pairwise loss function. Since our goal is to learn effective user representations to conduct quality recommendations, we introduce a *user-based* pairwise loss function (R loss), which is biased towards user representation learning.

We used a fixed time-length sliding window approach in the training process where, at each time interval, the model is trained using the interactions within the time interval (see Section 3.2.1). Thus, we denote the training instances for the new user-based BPR at each time interval t as $S^t=\{(u,v,i)|u\in U_{i+}^t\wedge v\in U\setminus U_{i+}^t\wedge i\in I\}$ where U_{i+}^t is the set of users who have interactions with item i, at time t. Compared to (u,i,j) in the standard item-based BPR, user-based BPR contains (u,v,i) where, for item i at time t, user u has an interaction and user v does not. Thus, intuitively, the predictor should assign a higher score for (u,i) compared to (v,i). Accordingly, we defined the user-based loss function to be minimized as follows:

$$L_{UBPR}(S^t)=\sum_{(u,v,i)\in S^t}-\ln\sigma(\hat{r}_{uvi}^t)+\lambda_\Theta\|\Theta\|^2 \tag{5}$$

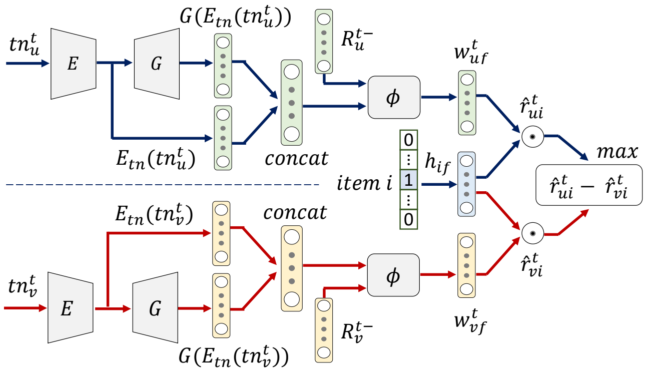

Figure 2: Recommender task architecture as a Siamese network.

where $\hat{r}_{uvi}^t = \hat{r}_{ui}^t - \hat{r}_{vi}^t = \sum_{f=1}^{K}(w_{uf}^t - w_{vf}^t)h_{if}$. During optimization, Θ updates are performed as follows:

$$\Theta \leftarrow \Theta + \alpha\left(\frac{e^{-\hat{r}_{uvj}^t}}{1+e^{-\hat{r}_{uvj}^t}} \cdot \frac{\partial}{\partial\Theta}\hat{r}_{uvj}^t + \lambda_\Theta\Theta\right) \tag{6}$$

For different Θ values, the following partial derivations are obtained:

$$\frac{\partial}{\partial\Theta}\hat{r}_{uvj}^t = \begin{cases} h_{if}, \text{if } \Theta = w_{uf}^t \\ -h_{if}, \text{if } \Theta = w_{vf}^t \\ (w_{uf}^t - w_{vf}^t), \text{if } \Theta = h_{if} \\ 0, \text{otherwise} \end{cases} \tag{7}$$

Since the user-based BPR performs two user representation updates for each training instance, it is able to efficiently guide G to learn user representations during training.

2.3.3 Recommender architecture: We used a Siamese network for the recommender architecture since it naturally supports pairwise learning (see Figure 2). Given a non-overlapped user u with current preferences $E_{tn}(\boldsymbol{tn_u^t})$, $G(E_{tn}(\boldsymbol{tn_u^t}))$ and previous preferences $\boldsymbol{R_u^{t-}}$ at time t, we learned a transfer function Φ, which maps the user to the latent user space $w_{u,f}^t$ for recommendations as follows:

$$w_{u,f}^t = \Phi(E_{tn}(\boldsymbol{tn_u^t}), G(E_{tn}(\boldsymbol{tn_i^t})), \boldsymbol{R_u^{t-}}) \tag{8}$$

The transfer function Φ is learned using a neural network since neural networks are more expressive and effective than simple linear models. Accordingly, for a non-overlapped user u at time t, the predicted rating $\hat{r}_{ui}$ for any given item i is obtained using the inner-product between the latent user and item representations as follows:

$$\hat{r}_{ui}^t = \sum_{f=1}^{K} \Phi(E_{tn}(tn_u^t), G(E_{tn}(tn_u^t)), \boldsymbol{R_u^{t-}}) \cdot \boldsymbol{h_{if}} \tag{9}$$

2.4 Multi-Task Learning (MTL)

Although the generator and recommender tasks are separate, they depend on each other to achieve a higher overall performance. The generator task increases the accuracy of the recommender by synthesizing source network encodings that effectively represent user preferences. The recommender task guides the generator to efficiently learn the target to source mapping by reducing the search space. Therefore, we trained both interrelated tasks in a MTL environment to benefit from the training signals of each other. Hence, we formulated the multi-task training objectives as follows:

$$\min_{G} \max_{E_{tn}, E_{sn}, D} V(E_{tn}, E_{sn}, G, D) \tag{10}$$

$$\min_{w_{uf}^t, w_{vf}^t, h_{if}} L_{UBPR}(S^t) \tag{11}$$

where $V(E_{tn}, E_{sn}, G, D)$ is the value function for E, G and D, $L_{UBPR}(S^t)$ is the R loss function, and w_{uf}^t, w_{vf}^t and h_{if} are model parameters. The R loss is back-propagated all the way to G since w_{uf}^t and w_{vf}^t are compositions of the functions G and Φ (see Equation 8) and the generator and recommender tasks are trained as an end-to-end process. Hence, equation 11 can be restated as follows:

$$\min_{\Phi, G, h_{if}} L_{UBPR}(S^t) \tag{12}$$

3 EXPERIMENTS

3.1 Dataset

Due to the lack of publicly available timestamped cross-network datasets, we extracted overlapped users on YouTube (target) and Twitter (source) networks from two publicly available datasets [16, 29]. We scraped timestamps of interactions and associated textual contents (video titles, descriptions and tweet contents) over a 2-year period from 1st March 2015 to 29th February 2017. In line with common practices, we filtered out users with less than 10 interactions on both networks for effective evaluations. The final dataset contained 2372 users and 12,782 YouTube videos.

3.2 Experimental Setup

The recommender systems literature often uses training and testing datasets with temporal overlaps [9], which provides undue advantages as it does not reflect a realistic recommender environment. To avoid such biases, at each time interval, the proposed model is trained using only the interactions during the current and previous time intervals. We randomly selected 50% of users as non-overlapped users and removed their source network interactions. The model was first trained offline using overlapped users (see Section 3.2.1). Then, testing and online training were conducted using both overlapped and non-overlapped users (see Section 3.2.2).

3.2.1 Offline Training: We used the sliding window approach and data from the first 16 months ($2T/3$) of overlapped users to learn the mapping from target to source network preferences for the generator task, the neural transfer function (Φ), and item latent representations (h_{if}) for the recommender task. At each training epoch, the generator task is learned first. Hence, E, D and G are trained using real mapping data and the trained G is used to generate synthetically mapped data. The generated preferences are used as inputs to the recommender task to predict interactions at the next time interval. The ground truth interactions at the same time interval are used to propagate the recommender error from the recommender to the generator task and learn parameters of both processes.

3.2.2 Testing and Online Training: We used data from the last 8 months ($2T/3$ onward) to test and retrain the model online in a simulated real-world environment. At each testing time interval t, first, mapping source network preferences are generated for non-overlapped users based on their target network preferences. Second, the synthetically mapped data for non-overlapped users are used as inputs to conduct Top-N recommendations for $t+1$ (see Equation 9). The entire model is then retrained online before moving to the next time interval, as follows: First, the recommender parameters (Φ and $\boldsymbol{h}^t_{if}$) are updated based on the recommendations for both overlapped and non-overlapped users. Second, the components of the generator task (E, D and G) are updated based on the real mapping data from overlapped users. Finally, to guide G from the training signals of the recommender, the model synthesizes mapping data for overlapped users, which are used as inputs for recommendations. The error is back-propagated from the recommender task to G, and consequently, the parameters G, Φ and $\boldsymbol{h}^t_{if}$ are updated. This process is repeated at subsequent time intervals as the model continues the online retraining process. In line with common practices in sliding window approaches, the model is retrained multiple times before moving to the next time interval.

3.3 Evaluation

We formulated video recommendation as a Top-N recommender task and predicted a ranked set of N videos that the user is most likely to interact with at each time interval. To evaluate model accuracy, we calculated the Hit Ratio (HR) and Normalized Discounted Cumulative Gain (NDCG) [11]. Both metrics were calculated for each participating user at each time interval and the results were averaged across all users and testing time intervals.

3.3.1 Baselines. We evaluated CnGAN against single network, cross-network, linear factorization and GAN based baselines.

- **TimePop**: Recommends the most popular N items in the previous time interval to all users in the current time interval.
- **TBKNN** [8]: Time-Biased KNN computes a set of K neighbors for each user at each time interval based on complete user interaction histories, and recommends their latest interactions to the target user at the next time interval. Similar to the original work, we used several K values from 4 to 50, and the results were averaged.
- **TDCN** [20]: Time Dependent Cross-Network is an offline, MF based cross-network recommender solution, which provides recommendations only for overlapped users. TDCN learns network level transfer matrices to transfer and integrate user preferences across networks and conduct MF based recommendations.
- **CRGAN**: Due to the absence of GAN based cross-network user preference synthesizing solutions, we created Cross-network Recommender GAN, as a variation of our solution. Essentially, CRGAN uses the standard D and G loss functions and does not contain the $L_{mismatch}$ and $L_{content}$ loss components.
- **NUBPR-O** and **NUBPR-NO**: Due to the absence of neural network based cross-network recommender solutions, we created two Neural User-based BPR solutions, as variations of our solution. Essentially, both models do not contain the generator task. NUBPR-O considers both source and target network interactions and NUBPR-NO only considers target network interactions to conduct recommendations.

3.4 Model Parameters

We used Adaptive Moment Estimation (Adam) [15] for optimizations since Adam adaptively updates the learning rate during training. We set the initial learning rate to 0.1, a fairly large value for faster initial learning before the rate is updated by Adam. We used only one hidden layer for all neural architectures, and given the size of the output layer H_L, the size of the hidden layer was set to $H_L \times 2$ to reduce the number of hyper-parameters. We used the dropout regularization technique and the dropout was set to 0.4 during training to prevent neural networks from overfitting.

4 DISCUSSION

4.0.1 Offline training loss: We compared the offline training loss of D, G and R (see Figure 3). All three processes converge around 50 epochs, and compared to D and R, the drop in G is low. Therefore, we further examined the proposed G loss against the vanilla G loss for the same training process (see Figure 4). Inline with standard GANs, the vanilla G loss component in the proposed G loss increases while the proposed G loss slowly decreases. Hence, despite the low decrease in the overall G loss, $L_{content}$ loss has notably decreased.

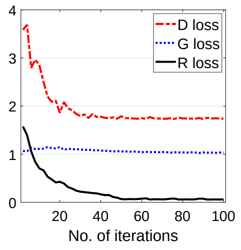

Figure 3: Offline training loss of D, G and R.

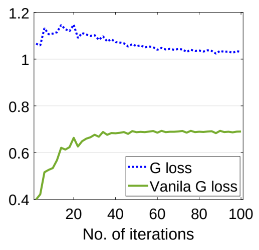

Figure 4: Offline training loss of G and vanilla G.

4.0.2 Online training loss: We observed R loss during the online training process (see Figure 5). Updates are most effective within the first 10-15 iterations and after around 20 iterations, the recommender accuracy reaches its peak. Additional iterations tend to overfit the model and degrade performance. Therefore, the proposed solution is feasible in real-world applications since online training requires only a few iterations and retraining is costly.

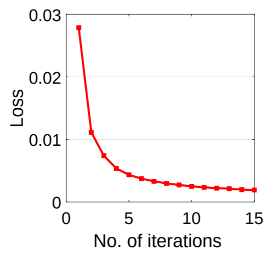

Figure 5: Online training loss of the recommender task.

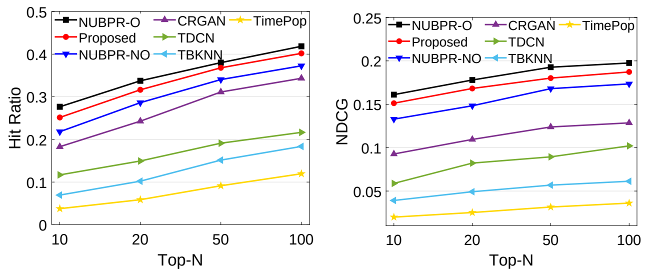

Figure 6: Hit Ratio and NDCG for different Top-N values.

4.0.3 Prediction accuracy: We compared recommender accuracy (HR and NDCG) against multiple Top-N values (see Figure 6). The neural network based solutions (Proposed, NUBPR-O, NUBPR-NO and CRGAN) show a higher accuracy since they capture complex relationships in user preferences. Among the non-neural network based solutions, TimePop has the lowest accuracy since it is based on a simple statistic - the popularity of videos. TDCN outperforms TBKNN and TimePop since it effectively utilizes both source and target network interactions to model user preferences.

As expected, NUBPR-O which does not have a data generation process and is only trained and tested with real mapping data from overlapped users has the best accuracy. We used NUBPR-O as the benchmark, since the goal is to synthesize data similar to the data used to train NUBPR-O. The proposed model shows the closest accuracy to NUBPR-O and consistently outperforms NUBPR-NO, which only used target network interactions. Therefore, the proposed model is able to generate user preferences that increases the recommender accuracy, even when the user overlaps are unknown. Furthermore, the improvements gained over CRGAN, which uses vanilla GAN loss functions, show the effectiveness of the proposed D and G loss functions.

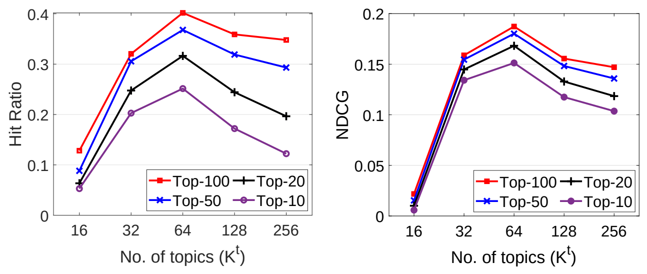

Figure 7: Hit Ratio and NDCG for different number of topics.

4.0.4 Prediction accuracy for different number of topics (K^t): We compared recomender accuracy against the dimensionality of the encoded topical space (K^t) (see Section 2.1.2 and Figure 7). A grid search algorithm was used to select an effective K^t value, and the highest accuracy is achieved when K^t is around 64 topics for all Top-N values. Smaller number of topics fail to effectively encode diverse user preferences, while a higher number of topics create highly sparse inputs and reduce recommender performance. Further experiments using the standard perplexity measure [6] showed that 64 topics lead to a smaller perplexity with faster convergence.

4.0.5 Diversity and novelty: A higher recommendation accuracy alone is insufficient to capture overall user satisfaction. For example, continuously recommending similar items (in terms of topics, genre, etc.) could lead to a decline in recommendation accuracy as users lose interest over time. Therefore, we compared the diversity [4] and novelty [31] of recommended items. The proposed model was able to recommend videos with better novelty and diversity, and was the closest to NUBPR-O model (only a 2.4% and 3.8% novelty and diversity drop). Compared to the single-network based solutions, cross-network solutions showed better results as they utilize rich user profiles with diverse user preferences. Against the closest CRGAN approach, the proposed model showed considerable improvements in novelty (by 8.9%) and diversity (by 12.3%).

4.0.6 Alternative GAN architectures: Designing and training GAN models can be challenging due to the unbalance between D and G, diminishing gradients and non-convergence [2, 27]. Various GAN architectures and training techniques were recently introduced to handle such issues. Therefore, despite the effectiveness of CnGAN, we designed two alternative solutions based on two widely popular GAN architectures, Deep convolutional GAN (DCGAN) [22] and Wasserstein GAN (WGAN) [3] to replace the G and D learning processes. We found that DCGAN becomes highly unstable in this environment, where D error is constantly increased. This can be because, DCGAN is based on Convolution Neural Networks, known to capture the local features within the input data. However, unlike typical image data, user preferences in the vectors may not provide any interesting local features. Hence, DCGAN was less effective. Further experiments using WGAN also did not improve the performances.

5 CONCLUSION AND FURTHER WORK

Typical cross-network recommender solutions are applied to users that are fully overlapped across multiple networks. Thus to the best of our knowledge, we propose the first Cross-network Generative Adversarial Network based model (CnGAN), which generates user preferences for non-overlapped users. The proposed model first uses a generator task to learn the mapping from target to source network user preferences and synthesize source network preferences for non-overlapped users. Second, the model uses a recommender task based on a Siamese network to incorporate synthesized source network preferences and conduct recommendations. The proposed model consistently outperformed multiple baselines in terms of accuracy, diversity and novelty of recommendations. As future work, we plan to investigate the recommender quality improvements when using both generated and real data for overlapped users. Furthermore, the model can be extended to use social information of the users. Overall, CnGAN alleviates a significant limitation in cross-network recommender solutions, and provides a foundation to make quality cross-network recommendations for all users.

Acknowledgments

This research has been supported by Singapore Ministry of Education Academic Research Fund Tier 2 under MOE's official grant number MOE2018-T2-1-103. We gratefully acknowledge the support of NVIDIA Corporation with the donation of a Titan Xp GPU used for this research.

REFERENCES

[1] Fabian Abel, Samur Araújo, Qi Gao, and Geert-Jan Houben. 2011. Analyzing cross-system user modeling on the social web. In *International Conference on Web Engineering*. Springer, 28–43.

[2] Martin Arjovsky and Léon Bottou. 2017. Towards principled methods for training generative adversarial networks. *arXiv preprint arXiv:1701.04862* (2017).

[3] Martin Arjovsky, Soumith Chintala, and Léon Bottou. 2017. Wasserstein gan. *arXiv preprint arXiv:1701.07875* (2017).

[4] Iman Avazpour, Teerat Pitakrat, Lars Grunske, and John Grundy. 2014. Dimensions and metrics for evaluating recommendation systems. In *Recommendation systems in software engineering*. Springer, 245–273.

[5] Homanga Bharadhwaj, Homin Park, and Brian Y Lim. 2018. RecGAN: Recurrent generative adversarial networks for recommendation systems. In *Proceedings of the 12th ACM Conference on Recommender Systems*. ACM, 372–376.

[6] David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of machine Learning research* 3, Jan (2003), 993–1022.

[7] Mathieu Blondel, Masakazu Ishihata, Akinori Fujino, and Naonori Ueda. 2016. Polynomial networks and factorization machines: New insights and efficient training algorithms. In *Proceedings of International Conference on Machine Learning*.

[8] Pedro G Campos, Alejandro Bellogín, Fernando Díez, and J Enrique Chavarriaga. 2010. Simple time-biased KNN-based recommendations. In *Proceedings of the Workshop on Context-Aware Movie Recommendation*. ACM, 20–23.

[9] Pedro G Campos, Fernando Dıez, and Iván Cantador. 2012. A performance comparison of time-aware recommendation models.

[10] Zhengyu Deng, Jitao Sang, Changsheng Xu, et al. 2013. Personalized video recommendation based on cross-platform user modeling. In *2013 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE, 1–6.

[11] Xiangnan He, Tao Chen, Min-Yen Kan, and Xiao Chen. 2015. Trirank: Review-aware explainable recommendation by modeling aspects. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. ACM, 1661–1670.

[12] 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*. International World Wide Web Conferences Steering Committee, 173–182.

[13] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2015. Session-based recommendations with recurrent neural networks. *International Conference on Learning Representations* (2015).

[14] Wang-Cheng Kang, Chen Fang, Zhaowen Wang, and Julian McAuley. 2017. Visually-aware fashion recommendation and design with generative image models. In *2017 IEEE International Conference on Data Mining (ICDM)*. IEEE, 207–216.

[15] Diederik P Kingma and Jimmy Lei Ba. 2015. Adam: Amethod for stochastic optimization. In *International Conference on Learning Representation*.

[16] Bang Hui Lim, Dongyuan Lu, Tao Chen, and Min-Yen Kan. 2015. # mytweet via instagram: Exploring user behaviour across multiple social networks. In *Advances in Social Networks Analysis and Mining (ASONAM), 2015 IEEE/ACM International Conference on*. IEEE, 113–120.

[17] Bhaskar Mehta, Claudia Niederee, Avare Stewart, Marco Degemmis, Pasquale Lops, and Giovanni Semeraro. 2005. Ontologically-enriched unified user modeling for cross-system personalization. In *International Conference on User Modeling*. Springer, 119–123.

[18] Miles Osborne, Saša Petrovic, Richard McCreadie, Craig Macdonald, and Iadh Ounis. 2012. Bieber no more: First story detection using Twitter and Wikipedia. In *SIGIR 2012 Workshop on Time-aware Information Access*.

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

[20] Dilruk Perera and Roger Zimmermann. 2017. Exploring the use of time-dependent cross-network information for personalized recommendations. In *Proceedings of the 2017 ACM on Multimedia Conference*. ACM, 1780–1788.

[21] Dilruk Perera and Roger Zimmermann. 2018. LSTM Networks for Online Cross-Network Recommendations.. In *IJCAI*. 3825–3833.

[22] Alec Radford, Luke Metz, and Soumith Chintala. 2015. Unsupervised representation learning with deep convolutional generative adversarial networks. *International Conference on Learning Representations (ICLR)*.

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

[24] Suman Deb Roy, Tao Mei, Wenjun Zeng, and Shipeng Li. 2012. Socialtransfer: Cross-domain transfer learning from social streams for media applications. In *Proceedings of the 20th ACM International Conference on Multimedia*. ACM, 649–658.

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

[26] Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey Hinton. 2007. Restricted Boltzmann machines for collaborative filtering. In *Proceedings of the 24th International Conference on Machine learning*. ACM, 791–798.

[27] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. 2016. Improved techniques for training gans. In *Advances in Neural Information Processing Systems*. 2234–2242.

[28] 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.

[29] Ming Yan, Jitao Sang, and Changsheng Xu. 2014. Mining cross-network association for youtube video promotion. In *Proceedings of the 22nd ACM International conference on Multimedia*. ACM, 557–566.

[30] Jaeyoon Yoo, Heonseok Ha, Jihun Yi, Jongha Ryu, Chanju Kim, Jung-Woo Ha, Young-Han Kim, and Sungroh Yoon. 2017. Energy-based sequence gans for recommendation and their connection to imitation learning. *Proceedings of ACM Conference*.

[31] Liang Zhang. 2013. The definition of novelty in recommendation system. *Journal of Engineering Science & Technology Review* 6, 3.

[32] Wayne Xin Zhao, Jing Jiang, Jianshu Weng, Jing He, Ee-Peng Lim, Hongfei Yan, and Xiaoming Li. 2011. Comparing twitter and traditional media using topic models. In *European Conference on Information Retrieval*. Springer, 338–349.