It carefully finds so-called *uninteresting items* that each user has not rated yet but is unlikely to prefer even if recommended to her. Then, it injects *zero ratings* to the identified uninteresting items as her negative preferences. Very recently, several imputation methods employ the generator in GAN as an imputation model for missing data. Chae et al. [7] proposed *Rating Augmentation GAN* (RAGAN) that aims at exploiting GAN to generate plausible ratings to be imputed while resolving the inherent *selection bias* of the rating data. *Generative Adversarial Imputation Nets* (GAIN) [44] and MisGAN [24] are also well-designed data imputation methods based on GAN, even though their domain is not recommender systems. They assume that the data is *missing completely at random* (MCAR) [15, 35] and build their models on the basis of this philosophy.

Among the contents-based or the hybrid recommender systems, AugCF [41] and RSGAN [45] are relevant to our work. In the AugCF framework, $\mathcal{G}$ outputs the most plausible item for a given user and a given class (e.g., *like*/*dislike*) with the help of *side information* associated with the users and items, and $\mathcal{D}$ distinguishes whether the generated item is fake or real. RSGAN aims at generating a target user's social friends who would be reliable and able to bring better recommendation performance. In our future work, we plan to extend our AR-CF to deal with various types of side information and compare it with hybrid recommenders.

5 EVALUATION

This section reports and analyzes the results of our extensive experiments, which are carefully designed to answer the following key questions:

- **Q1**: Are the generated neighbors beneficial for resolving the cold-start problems?
- **Q2**: How does AR-CF perform compared with the state-of-the-arts for all users and items?
- **Q3**: How does AR-CF perform according to different values of key hyper-parameters?
- **Q4**: How much do the generated users and items influence the scalability of AR-CF?

5.1 Experimental Settings

5.1.1 Datasets. We used four real-world datasets: Movielens 100K, Movielens 1M[3], Watcha[4], and CiaoDVD [38] datasets, whose detailed statistics are summarized in Table 2. For each dataset, following [14, 39], we chronologically split the ratings into two subsets: the first 80% for training and the remaining 20% for testing [5]. Among the ratings in the test set, we considered the ratings of 4 and 5 as our ground truth.

5.1.2 Implementation details. As mentioned, one of the most important advantages of AR-CF is that it can be performed on top of any existing data imputation methods. Once an arbitrary imputation method produces the rating matrix with a specific amount of missing cells imputed, which we denote by $\check{\mathbb{R}}$, then we can simply run AR-CF on $\check{\mathbb{R}}$ as if we run it on the original rating matrix $\mathbb{R}$. Since the data imputation methods have been very successful in improving the accuracy of CF models, our AR-CF can enjoy such performance gains without requiring any modification. We are aware of several successful imputation methods summarized in Section 4. Among them, we chose Zero-Injection (ZI in short, hereafter) and applied it to $\mathbb{R}$ before running AR-CF, since ZI is simple but fast and has achieved state-of-the-art accuracy in the literature. It is also less sensitive to its hyper-parameters [17, 22] and its code is publicly available. However, we note any other imputation methods can be applied here, and we remain AR-CF's collaborating with various imputation methods for our future work.

[3]https://grouplens.org/datasets/movielens/
[4]http://watcha.net
[5]However, since Watcha does not have the timestamp information, we split only its ratings in a random manner.

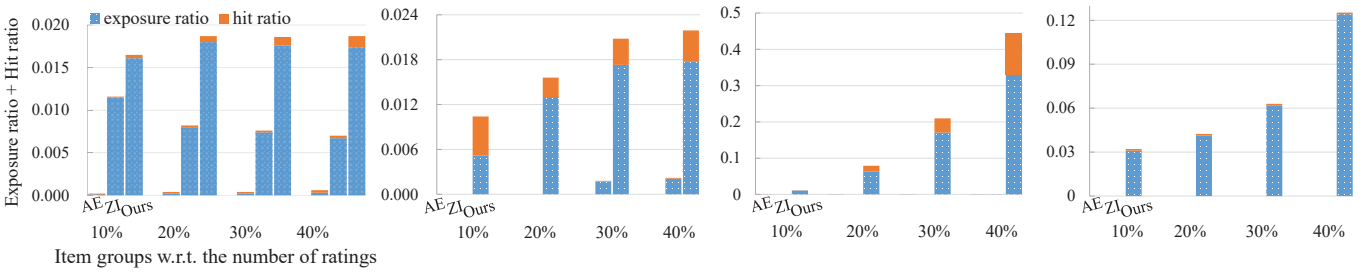
Table 2: Dataset statistics

Datasets	# users	# items	# ratings	Sparsity
Movielens 100K	943	1,682	100,000	93.69%
Watcha	1,391	1,927	101,073	96.98%
Movielens 1M	6,039	3,883	1,000,209	95.72%
CiaoDVD	7,628	15,536	62,358	99.94%

We performed the grid search to find the optimal values of hyper-parameters in our AR-CF. For the generators and discriminators in our four CGANs, we tested a different number of hidden layers with {1, 2, 3, 4, 5} and a different number of hidden nodes per hidden layer with {300, 400, 500, 600, 700}. We fixed the dimensionality of the random noise vector $\mathbf{z}$ as 128. For training the CGANs, the size of the minibatch was varied with {64, 128, 192, 256} and a learning rate with {0.005, 0.001, 0.0005, 0.0001}. We chose the numbers of virtual users and items, $\delta_{\mathcal{U}}$ and $\delta_{\mathcal{I}}$, by using the ratio to the entire number of real users (say, m) and real items (say, n) in a dataset, and varied them with {0.25, 0.5, 0.75, 1.0, 1.5, 2.0}; if $\delta_{\mathcal{U}} = 0.5$, for example, (int) $m \times 0.5$ users will be generated. We also varied the importance parameters, $\alpha_{\mathcal{U}}$ and $\alpha_{\mathcal{I}}$, with {0, 0.2, 0.4, 0.6, 0.8, 1.0}. We employed SVD [18] and AutoRec [32] as our CF models; we tested AutoRec's number of hidden layers with {1, 2, 3, 4} and its number of hidden nodes per hidden layer with {200, 300, 400, 500}; we tested SVD's number of latent factors with {20, 40, 60, 80, 100}. For training the CF models, we used the size of the minibatch with {64, 128, 256, 512} and the learning rate with {0.001, 0.0005, 0.0001}. For all the CGANs and CF models, we fixed their L2 regularization coefficients as 0.001. Following [7], we use a user's (or, an item's) rating vector as user-specific (or, item-specific) condition.

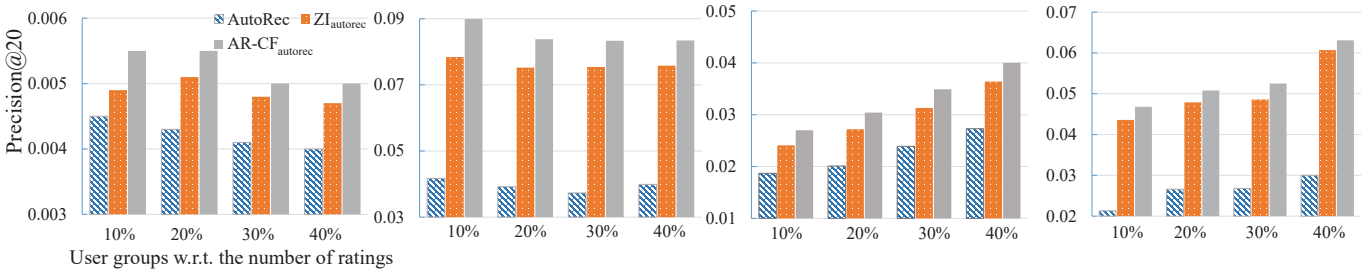
5.1.3 Competitors. We employed (1) two widely-used baseline CF methods of SVD [18] and AutoRec [32], (2) four CF methods based on state-of-the-art data imputation algorithms of Zero-Injection (ZI) [17], RAGAN [7], GAIN [44], and MisGAN [24], and (3) four state-of-the-art top-N recommenders of PureSVD [10], CDAE [43], CFGAN [5], and NGCF [42]. SVD is a standard MF-based model along with user and item bias terms [18]. AutoRec is an Autoencoder-based, non-linear latent factor model for CF. We already explained ZI, RAGAN, GAIN, and MisGAN in Section 4. For these imputation methods, we also employed SVD and AutoRec to provide the final recommendations after they completed their own imputation algorithms on $\mathbb{R}$. PureSVD, CDAE (*Collaborative De-noising Autoencoder*), CFGAN, and NGCF (*Neural Graph CF*) are the state-of-the-art recommender models based on SVD, Autoencoder, GAN, and GCN (*Graph Convolutional Networks*) [1], respectively.

For each competing method, we tested various values for its hyper-parameters (e.g., the numbers of hidden nodes and hidden



(a): CiaoDVD (b): Watcha (c): Movielens 1M (d): Movielens 100K

Figure 4: Exposure/hit ratio of cold-start items.



(a): CiaoDVD (b): Watcha (c): Movielens 1M (d): Movielens 100K

Figure 5: Accuracy of recommendation to cold-start users.

layers of $\mathcal{G}$ and $\mathcal{D}$ in the GAN-based models and the number of latent factors in the MF-based models) and selected a set of values that provide the highest accuracy. Since some of our competitors such as CFGAN and NGCF are tailored to perform the *one-class collaborative filtering* task [27, 33], which can be regarded as the special case of this paper's multi-class setting, we binarized $\mathbb{R}$ only when running them.

5.1.4 Evaluation metrics. We employed *precision*, *recall*, *normalized discounted cumulative gain* (nDCG), and *mean reciprocal rank* (MRR) for evaluating our model's recommendation accuracy [5, 7, 21]. *Precision* and *recall* focus on how many correct items are included in the recommendation list while *nDCG* and *MRR* account for the ranked positions of correct items in the recommendation list.

5.2 Experimental Results

5.2.1 Q1: Effectiveness on the cold-start problems. The problem in the cold-start items is that they are very difficult to be recommended, and thus rarely selected by users as well, as illustrated in Figure 1(b). We quantitatively measured such problems by defining the notions of "*exposure ratio*" and "*hit ratio*": when a CF model recommends top-N items to users, the exposure ratio reveals how many cold-start items are exposed (i.e., included in the top-N recommendation list) to users, and the hit ratio does how many those items are actually chosen by users. Formally, the exposure ratio is computed by B/A where B is the number of cold-start items which are exposed to at least one user, and A is the number of the entire cold-start items. The hit ratio is computed by C/A where C is the number of cold-start items which are recommended and *actually chosen* by at least one user (i.e., included in at least one user's ground truth).

Figure 4 reports each method's results of exposure ratio and hit ratio on each dataset. In each graph, $X\%$ in the x-axis indicates that the bottom-X percent of items according to the number of ratings were defined as cold-start items. AutoRec (AE in short, here) and $\text{ZI}_{\text{autorec}}$ (ZI in short, here) were compared with ours in this experiment. Since the results on this task provided by other methods are similar with or not on par with those of $\text{ZI}_{\text{autorec}}$, we omit their results in Figure 4 due to space limitations. We observed that AutoRec has difficulty in understanding the cold-start items, thereby recommending none or very few of them to users on all the four datasets. Even after the data imputation with ZI, $\text{ZI}_{\text{autorec}}$ still failed to exhibit at least one cold-start item to at least one user on many cases (e.g., dealing with bottom-10% and 20% cold-start items on Watcha, and all cases on Movielens 100K). However, after AR-CF generated their neighbors, we can observe that AR-$\text{CF}_{\text{autorec}}$ pushed a meaningful number of cold-start items to the top-N recommendation lists to users. Moreover, we can see that some of the recommended items are correct recommendation, i.e., they are actually chosen and highly rated by users. These results demonstrate the effectiveness of our AR-CF in resolving the cold-start item problems: by generating their realistic neighbors, AR-CF can "promote" the cold-start items and even result in purchases *without requiring any cost or human powers*.

Next, Figure 5 highlights the recommendation accuracy of our AR-CF to the cold-start users in terms of *precision@20*; the other metrics exhibited very similar tendency, hence they are omitted due to space limitations. We defined the cold-start users as the bottom-X percent of users according to their number of ratings, and then tried different X values; AutoRec, $\text{ZI}_{\text{autorec}}$, and AR-$\text{CF}_{\text{autorec}}$ are compared as well. The experimental results demonstrate that our AR-CF is really effective in solving the cold-start user problem. For example, AR-$\text{CF}_{\text{autorec}}$ performs well to the bottom-10% cold-start users on Watcha compared to $\text{ZI}_{\text{autorec}}$ (12.8% improved) and on Ciao (11% improved). It is also worth mentioning that AR-CF

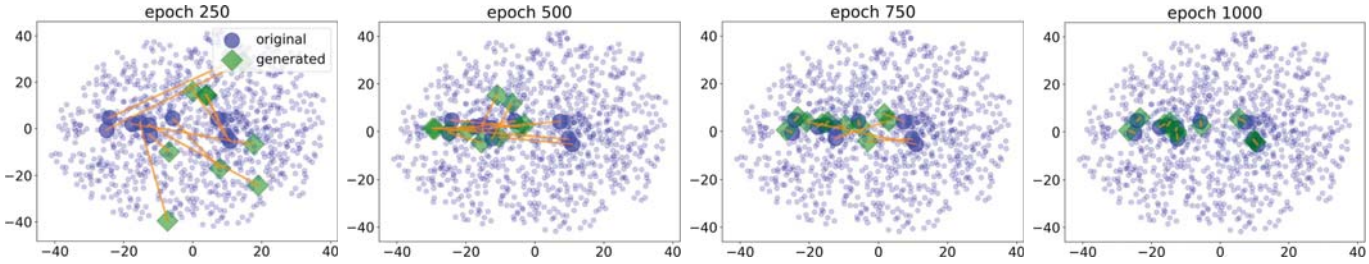


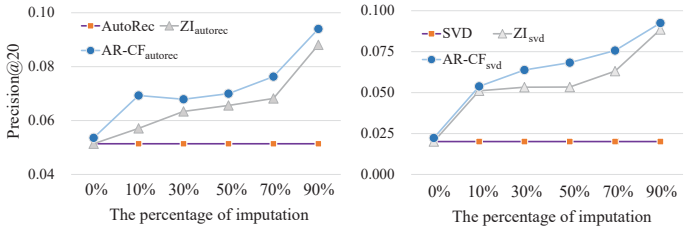
Figure 6: Visualization of the real users and virtual neighbors at specific epochs. Some selected users and their corresponding virtual neighbors are connected each other and are represented by the large circles and rhombuses, respectively.

Table 3: Comparison results. The best accuracy on each metric among competing methods is highlighted with bold face.

datasets	Movielens 100K				Movielens 1M				Watcha				CiaoDVD			
metrics	Pre.	Rec.	nDCG	MRR	Pre.	Rec.	nDCG	MRR	Pre.	Rec.	nDCG	MRR	Pre.	Rec.	nDCG	MRR
SVD	.0529	.0284	.0644	.1389	.0507	.0142	.0466	.0777	.0664	.0536	.0777	.1490	.0068	.0274	.0180	.0169
AutoRec	.0897	.0432	.0999	.1876	.0568	.0206	.0651	.1380	.0636	.0539	.0754	.1433	.0069	.0278	.0181	.0169
PureSVD	.1453	.1019	.1673	.2855	.0953	.0311	.0997	.1765	.1187	.0893	.1387	.2610	.0081	.0330	.0230	.0217
CDAE	.1411	.0843	.1561	.2608	.0856	.0410	.0922	.1722	.0968	.0685	.1057	.1981	.0069	.0286	.0188	.0176
CFGAN	.1451	.0826	.1610	.2755	.0904	.0398	.0972	.1785	.1015	.0696	.1100	.2005	.0072	.0296	.0188	.0173
NGCF	.1462	.0924	.1630	.2762	.0911	.0433	.0967	.1742	.1056	.0735	.1177	.2169	.0086	.0352	.0242	.0230
$\text{MisGAN}_{\text{svd}}$	.1643	.1032	.1891	.3131	.1028	.0390	.1096	.1953	.1231	.0922	.1428	.2634	.0071	.0295	.0188	.0173
$\text{MisGAN}_{\text{autorec}}$	.1627	.1034	.1872	.3125	.0994	.0389	.1059	.1898	.1269	.0951	.1456	.2633	.0071	.0318	.0210	.0189
GAIN_{svd}	.1614	.1050	.1864	.3109	.0991	.0372	.1054	.1895	.1210	.0908	.1398	.2551	.0089	.0355	.0245	.0229
$\text{GAIN}_{\text{autorec}}$	.1592	.0987	.1839	.3108	.0943	.0331	.1016	.1895	.1273	.0973	.1459	.2636	.0085	.0350	.0239	.0221
ZI_{svd}	.1689	.1124	.1936	.3181	.1039	.0378	.1103	.1969	.1248	.0938	.1463	.2700	.0093	.0371	.0261	.0250
$\text{ZI}_{\text{autorec}}$	.1656	.1084	.1900	.3173	.1037	.0397	.1098	.1957	.1260	.0924	.1472	.2681	.0097	.0405	.0274	.0252
$\text{RAGAN}_{\text{svd}}$	.1596	.1003	.1855	.3141	.1015	.0424	.1091	.1951	.1272	.0946	.1479	.2725	.0099	.0377	.0269	.0263
$\text{RAGAN}_{\text{autorec}}$	.1663	.1045	.1911	.3140	.1043	.0470	.1116	.1973	.1231	.0927	.1452	.2681	.0096	.0392	.0257	.0236
$\text{AR-CF}_{\text{svd}}$	.1678	.1077	**.1971**	**.3332**	**.1052**	**.0482**	**.1133**	**.2038**	.1319	.0973	.1527	.2738	.0103	.0417	.0288	**.0274**
$\text{AR-CF}_{\text{autorec}}$	**.1707**	**.1127**	.1954	.3254	.1047	.0455	.1120	.2005	**.1358**	**.1016**	**.1548**	**.2767**	**.0106**	**.0448**	**.0290**	.0267

is especially beneficial for the 10% and 20% cold-start users (i.e., "extremely" cold-start users). We believe such results came from the idea of our AR-CF that carefully generates the realistic neighbors of cold-start users to help the CF models understand them better, rather than simply filling the missing cells in $\mathbb{R}$ as other imputation methods did.

Here, we further examined whether our CGANs were well-trained so that the generated neighbors are highly plausible. Figure 6 visualizes the users (i.e., user rating vectors) in Movielens 100K by employing UMAP (*Uniform Manifold Approximation and Projection*) [25], one of the most popular visualization tools. In each plot, all the real users are projected on a shared 2D space at a specific epoch. Also, we chose 10 real users and visualized them and their virtual neighbors as well. As shown in the plot at epoch 250, our CGANs seemed to be under-trained because the generated virtual neighbors are quite far from their corresponding real users. However, we can observe that the distance between a pair of users tends to get closer as our CGANs are trained more. Finally, at 1000 epoch, we see that each pair of real and virtual users are placed very closely to each other in the space. Owing to these virtual (but plausible) neighbors, the CF models could understand the cold-start users much better and provide satisfactory recommendation to them. We note that we obtained very similar visualizations when training the CGANs



(a) AutoRec, $\text{ZI}_{\text{autorec}}$, and $\text{AR-CF}_{\text{autorec}}$ (b) SVD, ZI_{svd}, and $\text{AR-CF}_{\text{svd}}$

Figure 7: Accuracy w.r.t. the amount of imputation.

for items and also on the other datasets, but omit them due to the limited space.

5.2.2 Q2: Accuracy comparisons with state-of-the-arts for all users and items. The results shown in Table 3 aim to confirm that AR-CF is also effective in the general recommendation setting. We only show top-5 recommendation results and omit the top-20 results due to space limitation; they exhibited very similar trend. We observed that (1) even though NGCF, CFGAN, and CDAE relied on the limited information (i.e., binarized $\mathbb{R}$), they performed better than expected; especially NGCF outperformed several imputation-based methods

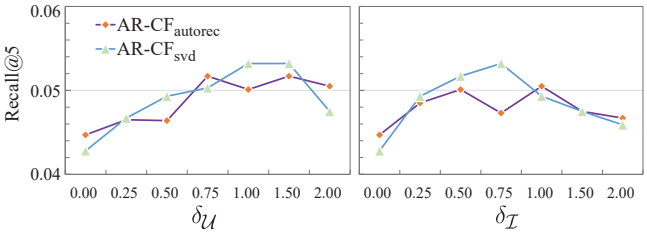


Figure 8: Sensitivity analysis on $\delta_{\mathcal{U}}$ and $\delta_{\mathcal{I}}$.

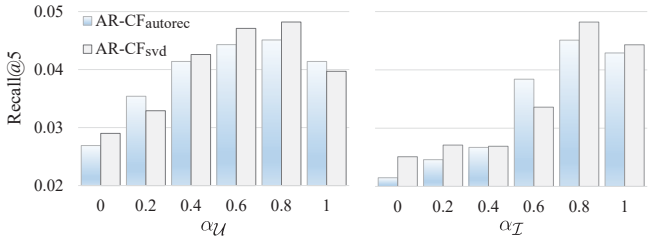


Figure 9: Impact of $\alpha_{\mathcal{U}}$ and $\alpha_{\mathcal{I}}$ on accuracy.

on Ciao. (2) The imputation-based methods performed much better than the CF models without imputation. This is because the data imputation relieves the data sparsity problem in $\mathbb{R}$ so that it makes CF models provide more qualified recommendation results. (3) Our AR-CF universally and consistently provided the best accuracy compared with all the imputation methods as well as the basic CF models, and we believe that the benefits are credited to AR-CF's characteristics that it can take the advantage of the performance gains coming from the data imputation and can exploit the virtual (but plausible) users and items as (additional) qualified training data. Through this observation, we can confirm that AR-CF is not only tailored to remedying the cold-start users and items, but is also effective in improving entire users' overall satisfaction with recommendations.

Also, we further examined the performance of ours with ZI in more detail, since we ran our framework on top of it. Figure 7 reports the accuracy (the y-axis) of ZI and ours equipped with AutoRec and SVD, according to the different percentages of missing cells in $\mathbb{R}$ imputed by ZI (the x-axis). As shown in the figure, ZI significantly improves original AutoRec and SVD; while enjoying this significant improvement by ZI, our AR-CF further improves the accuracy by augmenting additional users and items to $\mathbb{R}$. These results are consistent regardless of how may missing cells are imputed and which CF models are equipped.

5.2.3 Q3: Impact of key hyper-parameters. All the hyper-parameters used in AR-CF are shown in Section 5.1.2, each of which would affect the recommendation accuracy. Among them, this subsection investigates the impact of the following hyper-parameters, which are unique in our AR-CF and the most important in each of AR-step and CF-step: (1) $\delta_{\mathcal{U}}$ and $\delta_{\mathcal{I}}$, the numbers of virtual users and items to be generated, and (2) $\alpha_{\mathcal{U}}$ and $\alpha_{\mathcal{I}}$, the importance parameters used in the rating refinement process. The results of only *recall@5* on Movielens 1M are shown due to space limitations: those of the other metrics and those on the other datasets showed similar trend.

Figure 8 reports the accuracy of AR-CF$_{\text{autorec}}$ with regards to varying $\delta_{\mathcal{U}}$ and $\delta_{\mathcal{I}}$. Here, we fixed $\alpha_{\mathcal{U}} = \alpha_{\mathcal{I}} = 0.8$. We empirically found that the moderate range of $\delta_{\mathcal{U}}$ and $\delta_{\mathcal{I}}$ values would be in $[0.5, 1.5]$. If they are set with smaller values (e.g., less than 0.5), there won't be enough numbers of virtual users and items, so they are not that much influential. In contrast, if they are set with larger values (e.g., more than 1.5), CF models would pay too much attention to understanding the virtual users and items, rather than achieving its original goal of capturing preferences of real users on real items. Hence, $\delta_{\mathcal{U}}$ and $\delta_{\mathcal{I}}$ are set to 1.0 and 0.75, respectively; we believe it would provide a good balance in focus between virtual and real users (resp. items).

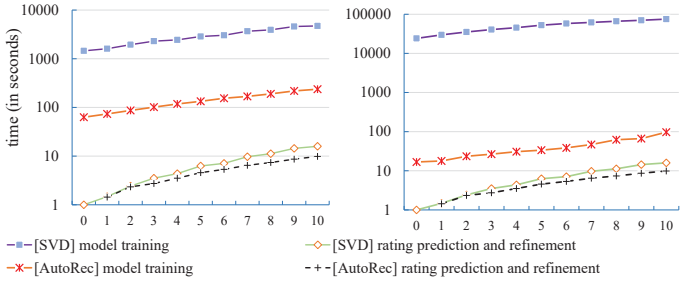
Figure 9 shows the accuracy over varying $\alpha_{\mathcal{U}}$ and $\alpha_{\mathcal{I}}$. Here, the two extreme cases (i.e., $\alpha_{\mathcal{U}} = \alpha_{\mathcal{I}} = 1$ and $\alpha_{\mathcal{U}} = \alpha_{\mathcal{I}} = 0$) provided lower accuracy, which implies the necessity of using both the real ratings as well as the generated ratings in our rating refinement step. We observe that the accuracy grows until $\alpha_{\mathcal{U}}$ and $\alpha_{\mathcal{I}}$ reach around 0.8. These results show that focusing more on a real (target) user's ratings is the most important in recommending items to her, but her neighbors' ratings as well as each target item's neighbors' ratings are still important and are worth referring to.

5.2.4 Q4: Scalability analysis. We analyze the scalability of our AR-CF because it involves the generated users and items that require additional computations in model training, rating prediction, and rating refinement. We tested it on CiaoDVD and Movielens 1M since CiaoDVD has the largest numbers of users and items and Movielens 1M has the largest number of ratings. For each dataset, we generated 500 users and 500 items and augmented them to $\mathbb{R}$. Then, we trained AutoRec and SVD to predict the unknown ratings and finally went through the rating refinement process. We repeated this experiment 10 times, where each iteration *doubles* the number of users and items to be generated and augmented. We measured the time taken by the two major computations (i.e., (1) model training and (2) rating prediction and refinement after the model training) in each iteration. Our experimental results are summarized in Figure 10. In each graph, the x-axis indicates each iteration, and the y-axis does computation time taken by the aforementioned tasks (in log scale). We ran all our experiments on a single machine equipped with an i9 7700K Intel CPU, 64GB RAM, and NVIDIA TITAN XP GPU.

We can observe that the rating prediction and refinement were performed very fast compared with the model training. In both datasets, the model training taken by each iteration increases *linearly* according to the increasing number of users and items augmented. Meanwhile, AutoRec performed much faster than SVD because AutoRec's training can be easily parallelized by fast GPU. The training of SVD took much more time in Movielens 1M since it has a much more number of ratings than CiaoDVD. We can conclude that AR-CF is *scalable* to the augmented users and items, requiring linear time in the number of users and items generated. Also, using AutoRec or any DNN-based model as CF would require a smaller time for additional computations thanks to the effective parallelism by using the GPU.

6 CONCLUDING REMARKS

In this paper, we proposed AR-CF, a novel framework for addressing the well-known cold-start problems. The AR-step in our framework



(a) CiaoDVD (b) Movielens 1M

Figure 10: Scalability analysis.

carefully trains four different CGANs to generate virtual, but realistic neighbors of cold-start users and items. The generated neighbors are augmented to the original rating matrix as additional rows (i.e., users) and columns (i.e., items). Then, in our CF-step, we proposed a rating refinement process that exploits virtual neighbors' ratings to adjust their corresponding real users/items' ratings. Through our extensive experiments, we demonstrated that AR-CF is really effective in dealing with the cold-start problems: it significantly improved the accuracy of recommendation to the cold-start users, and also made a meaningful number of the cold-start items to be displayed in top-N lists of users. Moreover, it performed the best among the recently proposed, state-of-the-art data imputation methods. Its sensitivity to several key-hyper parameters and scalability to the generated users and items are also analyzed accordingly. Indeed, AR-CF opens up a novel way to deal with the cold-start problems effectively via going outside the box of conventional data imputation approaches and working on top of them.

ACKNOWLEDGMENTS

This research was supported by (1) Next-Generation Information Computing Development Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT (NRF-2017M3C4A7069440), (2) the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. NRF-2020R1A2B5B03001960), and (3) 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).

REFERENCES

[1] R. Berg, T. N. Kipf, and M. Welling. 2017. Graph convolutional matrix completion. *arXiv preprint arXiv:1706.02263* (2017).
[2] R. Burke. 2002. Hybrid recommender systems: Survey and experiments. *User modeling and user-adapted interaction* 12, 4 (2002), 331–370.
[3] D.-K. Chae, S.-W. Kim, and J.-T. Lee. 2019. Autoencoder-based personalized ranking framework unifying explicit and implicit feedback for accurate top-N recommendation. *Knowledge-Based Systems* 176 (2019), 110–121.
[4] D.-K. Chae, J. A. Shin, and S.-W. Kim. 2019. Collaborative adversarial autoencoders: An effective collaborative filtering model under the GAN framework. *IEEE Access* 7 (2019), 37650–37663.
[5] D.-K. Chae et al. 2018. CFGAN: A generic collaborative filtering framework based on generative adversarial networks. In *ACM CIKM*. 137–146.
[6] D.-K. Chae et al. 2018. On identifying k-nearest neighbors in neighborhood models for efficient and effective collaborative filtering. *Neurocomputing* 278 (2018), 134–143.
[7] D.-K. Chae et al. 2019. Rating augmentation with generative adversarial networks towards accurate collaborative filtering. In *WWW*. 2616–2622.
[8] K.-J. Cho et al. 2019. No, that's not my feedback: TV show recommendation using watchable interval. In *IEEE ICDE*. 316–327.
[9] E. Choi et al. 2017. Generating multi-label discrete electronic health records using generative adversarial networks. *arXiv preprint arXiv:1703.06490* (2017).
[10] P. Cremonesi, Y. Koren, and R. Turrin. 2010. Performance of recommender algorithms on top-n recommendation tasks. In *ACM RecSys*. 39–46.
[11] C. Donahue, J. McAuley, and M. Puckette. 2018. Synthesizing audio with generative adversarial networks. *arXiv preprint arXiv:1802.04208* (2018).
[12] I. Goodfellow et al. 2014. Generative adversarial nets. In *NIPS*. 2672–2680.
[13] R. He and J. McAuley. 2016. VBPR: visual bayesian personalized ranking from implicit feedback. In *AAAI*.
[14] X. He et al. 2016. Fast matrix factorization for online recommendation with implicit feedback. In *ACM SIGIR*. 549–558.
[15] J. M. Hernández-Lobato, N. Houlsby, and Z. Ghahramani. 2014. Probabilistic matrix factorization with non-random missing data. In *ICML*. 1512–1520.
[16] D.-G. Hong et al. 2019. CrowdStart: Warming up cold-start items using crowdsourcing. *Expert Systems with Applications* 138 (2019).
[17] W.-S. Hwang et al. 2016. "Told you i didn't like it": Exploiting uninteresting items for effective collaborative filtering. In *IEEE ICDE*. 349–360.
[18] Y. Koren, R. Bell, and C. Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 42, 8 (2009).
[19] A. Krizhevsky, I. Sutskever, and G. E. Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *NIPS*. 1097–1105.
[20] Y.-C. Lee, S.-W. Kim, and D. Lee. 2018. gOCCF: Graph-theoretic one-class collaborative filtering based on uninteresting items. In *AAAI*.
[21] J. Lee et al. 2016. Improving the accuracy of top-N recommendation using a preference model. *Information Sciences* 348 (2016), 290–304.
[22] J. Lee et al. 2019. l-Injection: Toward effective collaborative filtering using uninteresting items. *IEEE TKDE* 31, 1 (2019), 3–16.
[23] Y. Lee et al. 2018. How to impute missing ratings?: Claims, solution, and its application to collaborative filtering. In *WWW*. 783–792.
[24] S. C.-X. Li, B. Jiang, and B. Marlin. 2019. MisGAN: Learning from incomplete data with generative adversarial networks. *arXiv preprint arXiv:1902.09599* (2019).
[25] L. McInnes et al. 2018. UMAP: Uniform manifold approximation and projection. *Journal of Open Source Software* 3, 29 (2018), 861.
[26] M. Mirza and S. Osindero. 2014. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784* (2014).
[27] R. Pan et al. 2008. One-class collaborative filtering. In *IEEE ICDM*. 502–511.
[28] C. Park et al. 2016. Improving top-k recommendation with truster and trustee relationship in user trust network. *Information Sciences* 374 (2016), 100–114.
[29] A. Radford, L. Metz, and S. Chintala. 2015. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434* (2015).
[30] Y. Ren et al. 2012. The efficient imputation method for neighborhood-based collaborative filtering. In *ACM CIKM*. 684–693.
[31] Y. Ren et al. 2013. AdaM: Adaptive-maximum imputation for neighborhood-based collaborative filtering. In *IEEE/ACM ASONAM*. 628–635.
[32] S. Sedhain et al. 2015. Autorec: Autoencoders meet collaborative filtering. In *WWW*. 111–112.
[33] S. Sedhain et al. 2016. On the effectiveness of linear models for one-class collaborative filtering.. In *AAAI*. 229–235.
[34] B. Settles. 2009. *Active learning literature survey*. Technical Report. University of Wisconsin-Madison Department of Computer Sciences.
[35] H. Steck. 2010. Training and testing of recommender systems on data missing not at random. In *KDD*. 713–722.
[36] F. Strub and J. Mary. 2015. Collaborative filtering with stacked denoising autoencoders and sparse inputs. In *NIPS workshop on machine learning for eCommerce*.
[37] X. Su and T. M. Khoshgoftaar. 2009. A survey of collaborative filtering techniques. *Advances in artificial intelligence* 2009 (2009), 4.
[38] J. Tang, H. Gao, and H. Liu. 2012. mTrust: Discerning multi-faceted trust in a connected world. In *WSDM*. 93–102.
[39] M. Wan et al. 2017. Modeling consumer preferences and price sensitivities from large-scale grocery shopping transaction logs. In *WWW*. 1103–1112.
[40] H. Wang, N. Wang, and D.-Y. Yeung. 2015. Collaborative deep learning for recommender systems. In *KDD*. 1235–1244.
[41] Q. Wang et al. 2019. Enhancing collaborative filtering with generative augmentation. In *KDD*. 548–556.
[42] X. Wang et al. 2019. Neural graph collaborative filtering. In *ACM SIGIR*. 165–174.
[43] Y. Wu et al. 2016. Collaborative denoising auto-encoders for top-N recommender systems. In *WSDM*. 153–162.
[44] J. Yoon, J. Jordon, and M. Schaar. 2018. GAIN: Missing data imputation using generative adversarial nets. In *ICML*. 5675–5684.
[45] J. Yu et al. 2019. Generating reliable friends via adversarial training to improve social recommendation. In *IEEE ICDM*. 768–777.