Dual Correction Strategy for Ranking Distillation in Top-N Recommender System

Youngjune Lee[1], Kee-Eung Kim[1, 2]

[1]School of Computing, KAIST, Daejeon, Republic of Korea
[2]Graduate School of AI, KAIST, Daejeon, Republic of Korea
{dudwns511,kekim}@kaist.ac.kr

ABSTRACT

Knowledge Distillation (KD), which transfers the knowledge of a well-trained large model (teacher) to a small model (student), has become an important area of research for practical deployment of recommender systems. Recently, Relaxed Ranking Distillation (RRD) has shown that distilling the ranking information in the recommendation list significantly improves the performance. However, the method still has limitations in that **1)** it does not fully utilize the prediction errors of the student model, which makes the training not fully efficient, and **2)** it only distills the user-side ranking information, which provides an insufficient view under the sparse implicit feedback. This paper presents Dual Correction strategy for Distillation (DCD), which transfers the ranking information from the teacher model to the student model in a more efficient manner. Most importantly, DCD uses the *discrepancy* between the teacher model and the student model predictions to decide which knowledge to be distilled. By doing so, DCD essentially provides the learning guidance tailored to "correcting" what the student model has failed to accurately predict. This process is applied for transferring the ranking information from the user-side as well as the item-side to address sparse implicit user feedback. Our experiments show that the proposed method outperforms the state-of-the-art baselines, and ablation studies validate the effectiveness of each component.

CCS CONCEPTS

• **Information systems** → **Learning to rank**; *Collaborative filtering*; *Retrieval efficiency*.

KEYWORDS

Recommender System; Knowledge Distillation; Learning to Rank; Model Compression; Retrieval efficiency

1 INTRODUCTION

In this era of information explosion, Recommender Systems (RS) are widely used in various industries to provide personalized user experience [3, 12, 16, 18]. For achieving higher recommendation accuracy, the recommendation model has become very large to capture the complexity of personalized recommendations [11, 17, 19, 26, 28, 30]. However, large models incur correspondingly large computational cost as well as high latency for inference, which has become one of the major obstacles for real-time service [9, 11, 17].

To reduce the inference latency, early methods adopt hash techniques [13, 20, 21, 30] or tree-based data structure [1]. However, they have problems such as easily falling into a local optimum or applicable only to specific models [9, 28]. To address the problems, Knowledge Distillation (KD) has been actively studied for RS [9, 10, 17, 19, 26, 28]. KD is a model compression technique that improves the performance of a small student model by transferring the knowledge of a pre-trained large teacher model [2, 4, 6, 25]. During the distillation, the teacher model provides additional supervision which is not existent in the users' feedback, so the student model can achieve a higher recommendation accuracy compared to the student model trained only on the original feedback.

The state-of-the-art method, Relaxed Ranking Distillation (RRD) [9], formulates the distillation process as a *ranking matching problem* between the recommendation list of the teacher model and that of the student model. In other words, it utilizes the ranking orders among the items from the teacher model as additional supervision to guide the student model, and trains the student model to preserve the ranking orders of the teacher model. This ranking-distillation approach transfers the relative preference order among the user's preferred items, which is the key knowledge directly affecting top-N recommendation accuracy. As a result, it significantly improves performance over the previous methods [19, 26] that do not directly utilize the ranking information [9].

Still, there are limitations in RRD. First, it transfers the knowledge without consideration of the prediction errors of the student model. As the student model gets more accurate in matching the prediction of the teacher model, repeatedly distilling the ranking information that the student model already correctly predicts cannot effectively enhance the student model and makes the training inefficient. We argue that the knowledge to be distilled should be dynamically changed based on the student model's prediction error, enabling "correction" for what the student model has not yet predicted accurately. Second, it only transfers user-side ranking information, i.e., ranking orders among the items. Previous studies have pointed out that learning only with the user-side ranking information degrades the quality of user representation [7] and provides a view insufficient to fully understand the sparse implicit

feedback [10, 14]. Particularly in KD where the student model's capacity is limited, these problems can be further exacerbated and severely degrade the performance.

In this work, we propose a novel Dual Correction strategy for Distillation (DCD), which aims to address the aforementioned shortcomings. To this end, DCD first computes *discrepancy* between the ranking list of the teacher model and that of the student model, then decides what knowledge to be distilled based on the discrepancy. By doing so, DCD provides guidance tailored to correct what the student model has failed to correctly predict, which helps to find an effective path for the student model's training. This process is conducted for dual-side ranking, i.e., for the user-side and the item-side, providing a comprehensive view to better understand both users and items [7, 14]. We validate the superiority of the proposed method with extensive experiments on real-world datasets, and provide an ablation study showing the effectiveness of each proposed component.

2 METHODOLOGY

2.1 Problem formulation and Notations

We focus on top-N recommendation task for implicit feedback [8, 18]. Given implicit user-item interactions, a recommender system provides a ranked list of top-N unobserved items for each user. The distillation process is conducted as follows: First, we train a large model (teacher) using the implicit feedback. Then, we train a small model (student) with the same feedback data along with the ranking list predicted from the teacher. The ranking information reveals the detailed preference orders among the unobserved items, which helps the training of the student. Our goal is to design a distillation strategy that allows the student to effectively follow the teacher's ranking list.

We denote the teacher by T and the student by S. R_T^u and R_S^u denote the user-side ranking list for user u (i.e., the list of the unobserved items) predicted by the teacher and the student, respectively. $R_*^u(i)$ denotes the rank of item i in the ranking list where a lower value means a higher ranking position, i.e., $R_*^u(i) = 0$ is the highest ranking. For the item-side ranking list, we simply reverse the notation of the user-side. Concretely, R_T^i and R_S^i denote the item-side ranking list for item i (i.e., the list of the unobserved users) predicted by the teacher and the student, respectively, and $R_*^i(u)$ denotes the rank of user u in the ranking list.

2.2 Distilling the Ranking Information

The ranking distillation (RRD) [9] formulates the distillation as a ranking matching problem between the ranking list of the teacher and that of the student (i.e., R_T^u and R_S^u). Specifically, the method trains the student to preserve the orders of ranking in R_T^u by using a variant of ListMLE [29]. The core idea is to define a permutation probability based on the the student's ranking scores, and train the student to maximize the likelihood of the teacher's ranking R_T^u.

To make the student better focus on top-ranked items, we also adopt *relaxed permutation probability* [9] that ignores the low-ranked items' detailed orders. Formally, let R_T^u is decomposed to two sub-ranking lists $R_T^u = [\pi; \pi']$, where π includes a few top-ranked items and π' includes the remaining items (u and T are omitted for simplicity). The relaxed permutation probability of the ranked list

π for the student S is defined as follows:

$$p(\pi|S) = \prod_{k=1}^{|\pi|} \frac{\exp S(u, \pi_k)}{\sum_{i=k}^{|\pi|} \exp S(u, \pi_i) + \sum_{j=1}^{|\pi'|} \exp S(u, \pi'_j)}, \quad (1)$$

where π_k is the k-th item in π, $S(u, \pi_k)$ is the score of the user-item interaction predicted by the student. By maximizing the probability, the student learns the detailed ranking orders in π while lowering all the ranks of items in π' below the lowest rank of items in π, which allows the student to focus more on top-N ranking orders. The student is trained by the ranking knowledge distillation (RKD) loss as follows:

$$\mathcal{L}_{RKD} = \mathcal{L}_{RS} + \lambda_{RRD}\mathcal{L}_{RRD}, \quad (2)$$

where $\mathcal{L}_{RS}$ is the loss for training the base model using the implicit feedback data, and $\mathcal{L}_{RRD} = -\frac{1}{|B|} \sum_{u \in B} \log p(\pi^u|S)$ is the permutation loss defined for the users in mini-batch B.

2.3 Dual Correction Strategy

We present Dual Correction strategy for Distillation (DCD) that adaptively assigns more concentrations on training instances that the student fails to predict correctly, unlike the prior methods such as RRD that generate training instances solely based on the teacher's predictions. This correction is used for transferring the ranking information from the user-side as well as the item-side, providing a comprehensive view to understand both users and items.

2.3.1 ***Identifying discrepancy between Teacher model and Student model****.* DCD first identifies *discrepancy* between R_T^u and R_S^u to decides what knowledge to be distilled. We define two types of discrepancy: 1) underestimation error and 2) overestimation error. The underestimation error means that the student predicts a low ranking position whereas the teacher predicts a higher ranking position, i.e., $R_S^u(i) > R_T^u(i)$. Thus, the student needs to be corrected to give a lower rank value for (u, i). The overestimation error means the opposite, the student predicts a high ranking position whereas the teacher predicts a lower ranking position, i.e., $R_S^u(i) < R_T^u(i)$, which needs to be corrected to give a higher rank value. The user-side errors are computed as follows:

$$D_i^u(S, T) = tanh(max(\mu(R_S^u(i) - R_T^u(i)), 0)) \quad (3)$$

$$D_i^u(T, S) = tanh(max(\mu(R_T^u(i) - R_S^u(i)), 0)) \quad (4)$$

We use *tanh*, which is a saturated function, to treat the errors above a certain threshold equally, allowing the student to learn the teacher's knowledge on most of the discrepant predictions. μ is a hyperparameter that controls the sharpness of the tanh function. Using the computed errors, we identify the discrepant predictions that need to be corrected. Concretely, we sample M_l^u underestimated items and M_h^u overestimated items. Both sampling probabilities are proportional to the degree of a discrepancy.

$$p_l^u(i) \propto D_i^u(S, T), \quad p_h^u(i) \propto D_i^u(T, S) \quad (5)$$

where $p_l^u(i)$ is the sampling probability of underestimated items and $p_h^u(i)$ is the sampling probability of overestimated items for user u. These discrepant items are dynamically changed based on the prediction errors of the student during the training, and will be corrected by the correction loss (Sec. 2.3.2).

DCD also provides the corrections for discrepancy in terms of the item-side ranking. As pointed out in the previous work [7, 14], learning only the user-side ranking degrades the quality of user representation [7] and also provides a restricted view insufficient to understand the sparse implicit feedback [14]. Especially, in KD where the student's capacity is highly limited, these problems can be further exacerbated, which leads to degraded performance.

Similar to the user-side, we identify the discrepant predictions on the item-side. We sample M_l^i underestimated users and M_h^i overestimated users based on the discrepancy between R_T^i and R_S^i. The sampling probabilities are as follows:

$$p_l^i(u) \propto D_u^i(S, T), \quad p_h^i(u) \propto D_u^i(T, S), \quad (6)$$

where $p_l^i(u)$ is the probability of underestimated users and $p_h^i(u)$ is the probability of overestimated users for item i. Without loss of generality, $D_u^i(A, B) = tanh(max(\mu(R_A^i(u) - R_B^i(u)), \mathbf{0}))$.

*2.3.2 **Dual Correction Distillation Loss**.* Now, we correct the discrepant predictions in the user-side (summarized by M_l^u and M_h^u) and the item-side (summarized by M_l^i and M_h^i). From the points of the teacher, the underestimation errors contain the predictions that should be higher-ranked, whereas the overestimation errors contain the predictions that should be relatively lower-ranked compared to the former. As consistently shown in the existing distillation work [9, 19, 26], the student takes a huge benefit by learning the teacher's knowledge with a particular emphasis on the high-ranked items, because it directly affects the top-N recommendation accuracy. In this regard, we design the correction loss that corrects the ranks of the underestimation errors in detail and lowers the ranks of the overestimation errors overall.

Let ρ^u denote the sorted lists of M_l^u by the original order in R_T^u, and ρ^i denote the sorted lists of M_l^i by the order in R_T^i. The user-side correction distillation (UCD) for user u is conducted by maximizing the following relaxed permutation probability:

$$p(\rho^u|S) = \prod_{k=1}^{|\rho^u|} \frac{\exp S(u, \rho_k^u)}{\sum_{i=k}^{|\rho^u|} \exp S(u, \rho_i^u) + \sum_{j \in M_h^u} \exp S(u, j)}, \quad (7)$$

where ρ_k^u is the k-th item in ρ^u. UCD is applied for the users in mini-batch B, i.e., $\mathcal{L}_{UCD} = -\frac{1}{|B|} \sum_{u \in B} \log p(\rho^u|S)$. Analogously, item-side correction distillation (ICD) for item i is conducted by maximizing the following relaxed permutation probability:

$$p(\rho^i|S) = \prod_{k=1}^{|\rho^i|} \frac{\exp S(\rho_k^i, i)}{\sum_{j=k}^{|\rho^i|} \exp S(\rho_j^i, i) + \sum_{l \in M_h^i} \exp S(l, i)}. \quad (8)$$

ICD is also applied for correcting errors with respect to the items in the batch, i.e., $\mathcal{L}_{ICD} = -\frac{1}{|B|} \sum_{i \in B} \log p(\rho^i|S)$. Finally, the proposed DCD trains the student with the following loss function.

$$\min_{\theta_s} \mathcal{L}_{RKD} + \lambda_{UCD} \mathcal{L}_{UCD} + \lambda_{ICD} \mathcal{L}_{ICD}, \quad (9)$$

where θ_s is the learning parameters of the student. λ_{UCD} and λ_{ICD} are hyperparameters controlling the user-side and item-side corrections, respectively. Note that $\mathcal{L}_{RKD}$ is computed for the same ground-truths regardless of the discrepancy during the training. Finally, our dual correction loss provides dynamically changing guidance to correct the student errors for more effective training.

Table 1: The number of parameters and inference time for generating recommendation list for every user.

Dataset	Base Model	# Parameters	Time (GPU/CPU)
CiteULike	BPR (Teacher)	6.08M	5.90s / 59.80s
	BPR (Student)	0.61M	3.67s / 14.90s
	NeuMF (Teacher)	12.24M	17.80s / 261.37s
	NeuMF (Student)	1.22M	7.42s / 33.75s
Foursquare	BPR (Teacher)	9.61M	25.22s / 252.23s
	BPR (Student)	0.96M	15.22s / 64.22s
	NeuMF (Teacher)	19.30M	75.22s / 1086.44s
	NeuMF (Student)	1.92M	30.54s / 144.05s

3 EXPERIMENTS

3.1 Experiment Setup

We closely follow the setup of the state-of-the-art method, RRD [9]. Specifically, datasets, base models, evaluation protocol, and the metrics are the same as [9]. Due to the limited space, we omit the detailed explanations of the setup. Please refer to [9].

Datasets. We use CiteULike [27] and Foursquare [22] which are public real-world datasets. After the preprocessing [9], CiteULike has 5,220 users, 25,182 items, and 115,142 interactions. Foursquare has 19,466 users, 28,594 items, and 609,655 interactions.

Base models. We use two base models for the top-N recommendation: BPR [24] and NeuMF [5], which have different architectures and optimization strategies. For both models, the dimension of user/item representations are set to 200 for the teacher model, and 20 for the student model. Following [9], we denote the student model trained without distillation as "Student". Table 1 presents the number of parameters and inference time. The inferences are made using PyTorch with CUDA from TITAN Xp GPU and Intel i7-4770 CPU. It shows that the smaller model has lower inference latency.

Evaluation protocol and metrics. We use the leave-one-out evaluation protocol whereby two interacted items for each user are held out for test/validation, and the rest are used for training [9]. We adopt two top-N ranking evaluation metrics, namely Hit Ratio (H@N) and Mean Reciprocal Rank (M@N). We compute the average score of those two metrics for each user. Finally, we report the average of the five independent runs.

Baselines. We compare DCD with the state-of-the-art ranking distillation method, RRD [9]. Note that we do not include the previous methods distilling point-wise information (e.g., RD [26], CD [19]), because RRD already outperforms them by a huge margin [9].

Implementation details. We use PyTorch [23] for implementation and train all models with Adam optimizer [15]. For each base model and dataset, we tune the hyperparameters by grid search on the validation set. We tune learning rate and L2 regularizer $\in \{10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}\}$. In the case of RRD-specific hyperparameters, we tune them in the ranges suggested by the original paper. For the dual correction loss, we tune $\lambda_{UCD}, \lambda_{ICD} \in \{1, 10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}\}$, and conduct the sampling process every 5 epochs. The number of discrepant users/items (M_l^*, M_h^*) is set to 40, but it can be further tuned. Lastly, μ is set to 10^{-3}.

3.2 Results

*3.2.1 **Overall Evaluation**.* Table 2 presents top-N recommendation accuracy of the methods compared. DCD achieves significantly higher performance than RRD on both datasets and both

Table 2: Performance comparison. *improve.r* denotes the improvement of DCD over RRD and *improve.s* denotes the improvement of DCD over Student. * and ** indicate $p \leq 0.005$ and $p \leq 0.0005$ for the paired t-test of DCD vs. RRD on H@5.

		CiteULike			
Base Model	KD Method	H@5	M@5	H@10	M@10
BPR	Teacher	0.5282	0.3704	0.6328	0.3844
	Student	0.4570	0.3061	0.5673	0.3214
	RRD	0.4735	0.3200	0.5800	0.3343
	DCD *	**0.4989**	**0.3412**	**0.6082**	**0.3560**
	improve.r	5.36%	6.63%	4.86%	6.49%
	improve.s	9.17%	11.47%	7.21%	10.77%
NeuMF	Teacher	0.4840	0.3346	0.5827	0.3478
	Student	0.3805	0.2499	0.4817	0.2634
	RRD	0.4563	0.2952	0.5647	0.3092
	DCD **	**0.4700**	**0.3101**	**0.5742**	**0.3241**
	improve.r	3.00%	5.05%	1.68%	4.82%
	improve.s	23.52%	24.09%	19.20%	23.04%
		Foursquare			
Base Model	KD Method	H@5	M@5	H@10	M@10
BPR	Teacher	0.5623	0.3618	0.7068	0.3812
	Student	0.4982	0.3140	0.6498	0.3342
	RRD	0.5132	0.3259	0.6625	0.3454
	DCD **	**0.5468**	**0.3483**	**0.6958**	**0.3683**
	improve.r	6.55%	6.87%	5.03%	6.63%
	improve.s	9.76%	10.92%	7.08%	10.20%
NeuMF	Teacher	0.5459	0.3499	0.6897	0.3693
	Student	0.4793	0.2959	0.6312	0.3162
	RRD	0.5076	0.3119	0.6615	0.3313
	DCD **	**0.5287**	**0.3303**	**0.6734**	**0.3497**
	improve.r	4.16%	5.90%	1.80%	5.55%
	improve.s	10.31%	11.63%	6.69%	10.59%

base models. Also, in terms of the number of recommended items (N), DCD shows larger improvements for H@5/M@5 compared to H@10/M@10. Namely, DCD has a better performance at predicting the top-ranked items than RRD, which is practically advantageous for real-world RS, which gives the users the most preferred items.

3.2.2 **Ablation Study**. We provide ablation study of the key components of DCD in Table 3. We compare the following ablations: 1) **w/o Correction** transfers the user-side and item-side ranking information without the correction strategy, i.e., RRD + item-side RRD. 2) **w/o Item-side** and **w/o User-side** ablate ICD and UCD from DCD, respectively. 3) **w/o Sampling** deterministically selects items/users with the largest underestimation error and overestimation error without the sampling process (Sec. 2.3.1). We observe that each proposed component is indeed effective in distilling the ranking information. This result supports our claim that the supervision from the teacher model should be dynamically changed based on the student's errors (w/o Correction) and distilling the single-side ranking is insufficient (w/o Item-side and w/o User-side). Also, the comparison with "w/o Sampling" shows that a certain degree of flexibility is beneficial in choosing the discrepant predictions for the correction strategy.

3.2.3 **Further Analysis**. We provide further analysis on DCD. For the sake of the space, we report the results of BPR on CiteULike.

First, Figure 1a presents the average ranking-discrepancy of various methods. In specific, we compute the discrepancy as $|R_S^*(\cdot) -$

Table 3: Ablation analysis on Foursquare dataset.

Base Model	KD Method	H@5	M@5	H@10	M@10
BPR	DCD	**0.5468**	**0.3483**	**0.6958**	**0.3683**
	w/o Correction	0.5318	0.3408	0.6710	0.3594
	w/o Item-side	0.5399	0.3424	0.6908	0.3628
	w/o User-side	0.5377	0.3419	0.6870	0.3620
	w/o Sampling	0.5420	0.3426	0.6898	0.3621
NeuMF	DCD	**0.5287**	**0.3303**	**0.6734**	**0.3497**
	w/o Correction	0.5147	0.3214	0.6704	0.3424
	w/o Item-side	0.5180	0.3210	0.6716	0.3417
	w/o User-side	0.5225	0.3254	0.6681	0.3450
	w/o Sampling	0.5196	0.3230	0.6706	0.3433
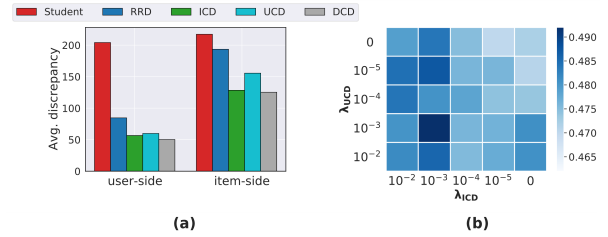
Figure 1: Effects of DCD. (a) The average discrepancy from Teacher, **(b)** H@5 with varying λ_{UCD} and λ_{ICD}.

$R_T^*(\cdot)|$ for the user-side (and for the item-side) top-50 recommendation list produced by the teacher model. We compute it for all users (and for all items), then report the average value. We observe that the proposed correction strategy effectively reduces the discrepancy between the teacher model and the student model. All the correction-based methods (i.e., ICD, UCD, and DCD) achieves lower discrepancy than RRD. Also, DCD achieves the lowest discrepancy in both user-side and item-side, which supports its superior recommendation performance. This also again shows the importance of the dual-side ranking correction. Lastly, Figure 1b shows the effects of λ_{UCD} and λ_{ICD}. Note that $\lambda_{UCD} = 0$ & $\lambda_{UCD} = 0$ corresponds to RRD. We again observe that both user-side and item-side corrections are indeed effective. The best performance is achieved when λ_{ICD} is around 10^{-2}-10^{-3} and λ_{UCD} is around 10^{-2}-10^{-3}.

4 CONCLUSION

We propose DCD, a dual correction strategy for ranking distillation in top-N RS. Unlike the existing method based on unilateral distillation, DCD provides guidance designed to correct the errors that the student model has failed to learn. By considering the prediction errors of the student model, DCD helps to find an effective path for the student model's training. DCD also considers the user-side ranking and item-side ranking simultaneously, providing a comprehensive view to understand both users and items. We validate the effectiveness of DCD with extensive experiments on real-world datasets. Also, we provide in-depth ablation study to ascertain the validity of each proposed component. For future work, we will investigate the effects of DCD on various base models.

REFERENCES

[1] Yoram Bachrach, Yehuda Finkelstein, Ran Gilad-Bachrach, Liran Katzir, Noam Koenigstein, Nir Nice, and Ulrich Paquet. 2014. Speeding up the xbox recommender system using a euclidean transformation for inner-product spaces. In *RecSys*.

[2] Guobin Chen, Wongun Choi, Xiang Yu, Tony Han, and Manmohan Chandraker. 2017. Learning efficient object detection models with knowledge distillation. In *NeurIPS*.

[3] Junsu Cho, SeongKu Kang, Dongmin Hyun, and Hwanjo Yu. 2021. Unsupervised Proxy Selection for Session-based Recommender Systems. In *SIGIR*.

[4] Tommaso Furlanello, Zachary C Lipton, Michael Tschannen, Laurent Itti, and Anima Anandkumar. 2018. Born again neural networks. *arXiv preprint arXiv:1805.04770* (2018).

[5] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *WWW*.

[6] Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the knowledge in a neural network. *NIPS* (2015).

[7] Jun Hu and Ping Li. 2018. Collaborative multi-objective ranking. In *CIKM*.

[8] Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative filtering for implicit feedback datasets. In *ICDM*.

[9] SeongKu Kang, Junyoung Hwang, Wonbin Kweon, and Hwanjo Yu. 2020. DE-RRD: A Knowledge Distillation Framework for Recommender System. In *CIKM*.

[10] SeongKu Kang, Junyoung Hwang, Wonbin Kweon, and Hwanjo Yu. 2021. Item-side Ranking Regularized Distillation for Recommender System. *Information Sciences* (2021). https://doi.org/10.1016/j.ins.2021.08.060

[11] SeongKu Kang, Junyoung Hwang, Wonbin Kweon, and Hwanjo Yu. 2021. Topology Distillation for Recommender System. In *KDD*.

[12] SeongKu Kang, Junyoung Hwang, Dongha Lee, and Hwanjo Yu. 2019. Semi-supervised learning for cross-domain recommendation to cold-start users. In *CIKM*.

[13] Wang-Cheng Kang and Julian McAuley. 2019. Candidate Generation with Binary Codes for Large-Scale Top-N Recommendation. In *CIKM*.

[14] Seunghyeon Kim, Jongwuk Lee, and Hyunjung Shim. 2019. Dual neural personalized ranking. In *WWW*.

[15] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[16] Wonbin Kweon, Seongku Kang, Junyoung Hwang, and Hwanjo Yu. 2020. Deep Rating Elicitation for New Users in Collaborative Filtering. In *WWW*.

[17] Wonbin Kweon, SeongKu Kang, and Hwanjo Yu. 2021. Bidirectional Distillation for Top-K Recommender System. In *WWW*.

[18] Dongha Lee, SeongKu Kang, Hyunjun Ju, Chanyoung Park, and Hwanjo Yu. 2021. Bootstrapping User and Item Representations for One-Class Collaborative Filtering. *SIGIR* (2021).

[19] Jaewoong Lee, Minjin Choi, Jongwuk Lee, and Hyunjung Shim. 2019. Collaborative Distillation for Top-N Recommendation. *ICDM* (2019).

[20] Defu Lian, Rui Liu, Yong Ge, Kai Zheng, Xing Xie, and Longbing Cao. 2017. Discrete Content-Aware Matrix Factorization. In *KDD*.

[21] Han Liu, Xiangnan He, Fuli Feng, Liqiang Nie, Rui Liu, and Hanwang Zhang. 2018. Discrete factorization machines for fast feature-based recommendation. *arXiv preprint arXiv:1805.02232* (2018).

[22] Yiding Liu, Tuan-Anh Nguyen Pham, Gao Cong, and Quan Yuan. 2017. An experimental evaluation of point-of-interest recommendation in location-based social networks. *Proceedings of the VLDB Endowment* (2017).

[23] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *arXiv preprint arXiv:1912.01703* (2019).

[24] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *UAI*.

[25] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. 2014. Fitnets: Hints for thin deep nets. In *arXiv*.

[26] Jiaxi Tang and Ke Wang. 2018. Ranking distillation: Learning compact ranking models with high performance for recommender system. In *KDD*.

[27] Hao Wang, Binyi Chen, and Wu-Jun Li. 2013. Collaborative topic regression with social regularization for tag recommendation. In *IJCAI*.

[28] Haoyu Wang, Defu Lian, and Yong Ge. 2019. Binarized collaborative filtering with distilling graph convolutional networks. *IJCAI* (2019).

[29] Fen Xia, Tie-Yan Liu, Jue Wang, Wensheng Zhang, and Hang Li. 2008. Listwise approach to learning to rank: theory and algorithm. In *ICML*.

[30] Hanwang Zhang, Fumin Shen, Wei Liu, Xiangnan He, Huanbo Luan, and Tat-Seng Chua. 2016. Discrete Collaborative Filtering. In *SIGIR*.