SIGIR 2021

Intervention for Leveraging Popularity Bias in Recommendation

- Authors: Yang Zhang et al.
- Main Affiliation: University of Science and Technology of China

2025.07.24

서정호

DGIST DATA SCIENCE LAB

DGIST 대구경북과학기술원
Daegu Gyeongbuk Institute of Science & Technology

Contents

DGIST DATA SCIENCE LAB

DGIST 대구경북과학기술원
Daegu Gyeongbuk Institute of Science & Technology

1. Proposal

RS: Popularity Bias problem.

- Interaction frequencey [popularity] → User-Item interaction data: long-tailed distribution.
  - Few head items가 most of the interactions 차지.

▼ Previous works → Eliminate the effect of popularity bias

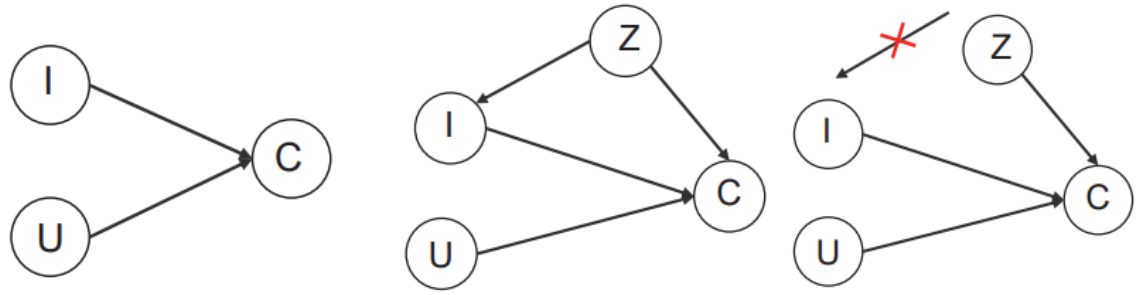
1. Inverse Propensity Scoring (IPS)
   a. Propensity estimation 어려움.
   b. High model variance.
2. Causal Embedding
   a. Bias-free uniform data 사용. → Discard item popularity → Expose randomly items to users → 구하기 어려움.
   → Small data → learning: unstable.
3. Ranking Adjustment
   a. Heuristically designed
   b. Lack of theoretical foundations

1. Proposal

Idea

- Higher popularity items
  - Better intrinsic quality
  - Representing current trends
  - → popularity bias를 모두 삭제하는 것은 별로 좋지 못함.
- 어떻게 Popularity Bias를 RS 성능을 위해 사용[leverage]할 수 있을까?
  1. Training 단계에서 Popularity Bias의 Bad impacts를 제거
  2. Inference 단계에서 desired popularity bias를 삽입. [top-K recommendations 생성.]

1. Proposal



(a) Causal graph of traditional methods. (b) Causal graph that considers item popularity. (c) We cut off $Z \rightarrow I$ for model training.

Figure 1: Causal graphs to describe the recommendation process. U: user, I: exposed item, C: interaction probability, Z: item popularity. We identify Z as the confounder between I and C, and propose deconfounded training with $P(C|do(U, I))$ as the interest matching.

U: User

I: Item **Exposure**

C: **Interaction** Prob.

Z: Item **Popularity**

- Cause → Effect: Directed Acyclic Graph [DAG]
- U, I, Z → C
- Z → I → C & Z→ C
  - Z: confounder, confounding effect를 가짐.
  - Z → I → C: Bad effect of popularity bias [negative impact to learn real user interest.]
  - Z → I 부분을 제거해야 함.
  - do-calculus 연산 사용. [de-confounded training]
    - $do(I)$ forces to remove the impact of $I's$ parent nodes

DGIST DATA SCIENCE LAB

DGIST 대구경북과학기술원 Daegu Gyeongbuk Institute of Science & Technology

1. Proposal

Proposal

- New training & inference paradigm: **Popularity-bias Deconfounding and Adjusting (PDA)**
  1. Training 단계에서 confounding popularity bias 제거
  2. Inference 단계에서, **Causal Intervention** → desired popularity bias → recommendation score 조정.

2. Primary Knowledge

Notation

1. U: user, I: item
2. Upper-case character: random variable. e.g. U, I
3. lower-case character: specific value. e.g. u, i
4. Calligraphic font: sample space. e.g. $\mathcal{U} = \{u_1, \ldots, u_{|\mathcal{U}|}\}, \mathcal{I} = \{i_1, \ldots, i_{|\mathcal{I}|}\}$
5. Probability dist.: $P(\cdot)$
6. $\mathcal{D}_t$: t번째 stage의 data → $\mathcal{D} = \cup_{t=1}^{T} \mathcal{D}_t$: historical data
7. D_i^t: Number of observed interactions for item i in $\mathcal{D}_t$

If whole training set, Global popularity

$$m_i^t = \frac{D_i^t}{\sum_{j \in \mathcal{I}} D_j^t} : \text{local popularity of item i on the stage t} \quad (1)$$

Popularity = Item's frequency

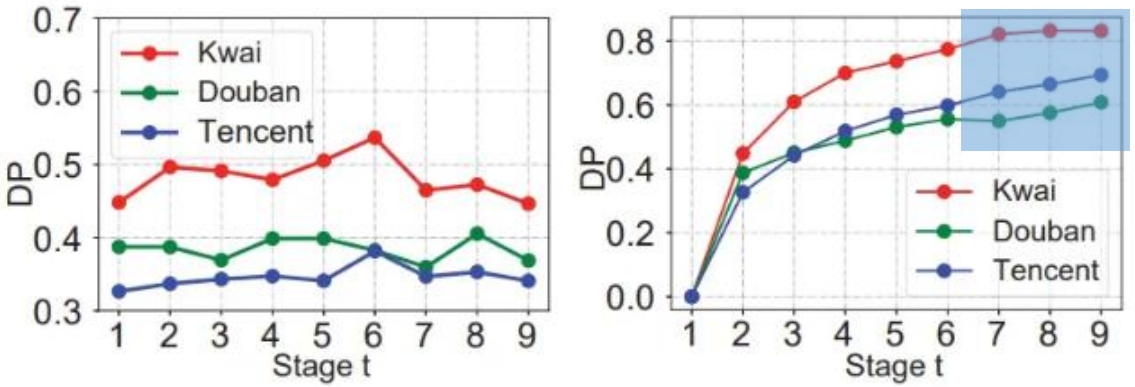
2. Primary Knowledge

Metrics

- **Drift of Popularity (DP)** between stage t and stage s [**시간이 흐를수록 popularity가 변화.**]

$$\mathrm{DP}(t,s) = \mathrm{JSD}\left(\left[m_1^t,\ldots,m_{\mathcal{I}}^t\right],\left[m_1^s,\ldots,m_{\mathcal{I}}^s\right]\right) \quad (2)$$

  - Jensen-Shannon **Divergence**(JSD): dis-similarity **between two stages**
  - **작을수록 분포 유사.**



(a) $DP(t, t+1)$ (b) $DP(1, t)$

Figure 2: Popularity drift between: (a) two successive stages $DP(t, t+1)$; (b) the first and present stage $DP(1, t)$.

Popularity 차이가 크다.

- 3 real-world dataset: Kwai, Douban, Tencent
- (a): dataset에 따라 추세가 다름.
- (b) 시간이 지남에 따라 점차 popularity의 변화가 누적됨.

3. Framework

Deconfounded Training

Traditional predictive model

- $P(C|U=u, I=i)$: (user, item)에 대하여 interaction할 확률. $P(c=1|u,i)$ =??
- 높은 순서대로 top-K recommendation.

Paper's predictive model

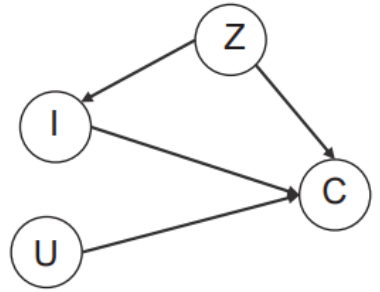
- $Z \rightarrow I$를 없애야 함. I를 그대로 사용하지 말고, $do(I)$를 사용.

Training: $P(C|U, do(I)) = P(C|do(U), do(I)) = P(C|do(U,I)) \rightarrow P(c=1|do(U=u, I=i))$ =??

- Figure 1-(b) = G, Figure 1-(c) = G'이라 하면,

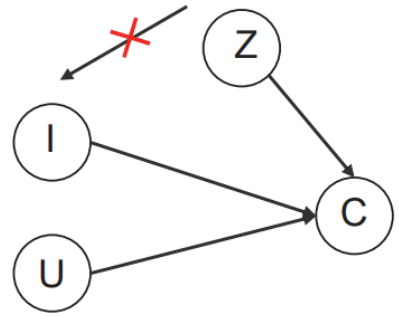
$$P(C|do(U,I)) = P_{G'}(C|U,I) = \sum_{z \in Z} P_{G'}(C,z|U,I) = \sum_{z \in Z} P_{G'}(C|U,I,z)\, P_{G'}(z|U,I)$$
$$= \sum_{z \in Z} P_{G'}(C|U,I,z)\, P_{G'}(z) = \sum_{z \in Z} P(C|U,I,z)\, P(z) \quad (3)$$

G



(b) Causal graph that considers item popularity.

G'



(c) We cut off $Z \rightarrow I$ for model training.

3. Framework

Deconfounded Training

Traditional predictive model

- $P(C|U=u, I=i)$: (user, item)에 대하여 interaction할 확률. $P(c=1|u,i)$ =??
- 높은 순서대로 top-K recommendation.

Paper's predictive model

- $Z \rightarrow I$를 없애야 함. I를 그대로 사용하지 말고, $do(I)$를 사용.

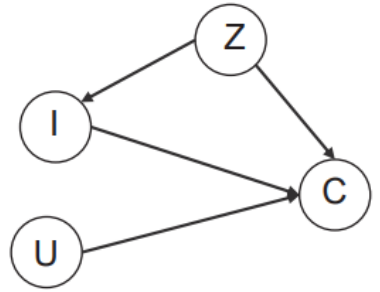
$do(I)$ forces to remove the impact of $I's$ parent nodes

Training: $P(C|U, do(I)) = P(C|do(U), do(I)) = P(C|do(U,I)) \rightarrow P(c=1|do(U=u, I=i))$ =??

- Figure 1-(b) = G, Figure 1-(c) = G'이라 하면,

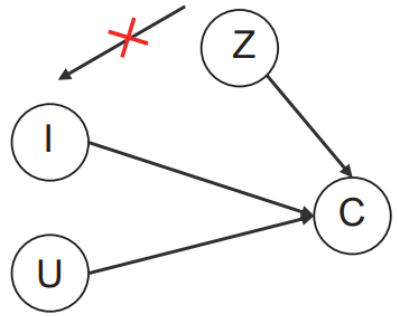
$$P(C|do(U,I)) = P_{G'}(C|U,I) = \sum_{z \in Z} P_{G'}(C,z|U,I) = \sum_{z \in Z} P_{G'}(C|U,I,z)\, P_{G'}(z|U,I)$$

$$= \sum_{z \in Z} P_{G'}(C|U,I,z)\, P_{G'}(z) = \sum_{z \in Z} P(C|U,I,z)\, P(z) \quad (3)$$



G

(b) Causal graph that considers item popularity.



G'

(c) We cut off $Z \rightarrow I$ for model training.

3. Framework

Deconfounded Training

- Step 1. $P\left(c=1|u,i,z\right)$ estimation. for $\hat{P}\left(c=1|do\left(u,i\right)\right)$
  - Θ: parameters of $P\left(c=1|U=u,I=i,z\right)$
  - Pair-wise BPR objective function with L_2 regularization term.

$$\mathcal{L}_{\text{BPR}} = -\sum_{(u,i,j)\in\mathcal{D}} \log\sigma\left(P_\Theta\left(c=1|u,i,m_i^t\right) - P_\Theta\left(c=1|u,j,m_j^t\right)\right) + \lambda\|\Theta\|^2 \quad (4')$$

$$\arg\min_\Theta \mathcal{L}_{\text{BPR}} = \hat{\Theta}$$

  - $U=u, I=i, Z=m_i^t$
  - j: negative sample for u → interaction이 없는 item
  - $\mathcal{L}_{\text{BPR}}\downarrow \quad \Leftrightarrow \quad P_\Theta\left(c=1|u,i,m_i^t\right) > P_\Theta\left(c=1|u,j,m_j^t\right)$: interaction이 존재하는 item의 interaction 확률이 더 크다. trivial.

3. Framework

Deconfounded Training

- (U, I) , Z 분리. → 장점은 명확하나 정당화 하는 논거가 없음. 가정으로 판단.

$$P_\Theta\left(c=1|u,i,m_i^t\right)=\mathrm{ELU}'\left(f_\Theta\left(u,i\right)\right)\times\left(m_i^t\right)^\gamma \quad (5)$$

1. f를 변화시켜 가며 extendable한 model 생성 가능
2. Inference에서 popularity bias를 조정하기 용이. $(m_i^t)^\gamma$
   a. γ: hyper-parameter for tuning.
   b. $\gamma\uparrow \Rightarrow$ higher impact of bias
   - In this paper, f : Matrix Factorization
   - ELU: Exponential Linear Unit [activation function]

$$\mathrm{ELU}'\left(x\right)=\begin{cases}e^{-x}, & \text{if } x\leq 0\\ x+1, & \text{else}\end{cases} \quad (6)$$

3. Framework

Deconfounded Training

▼ Step 2. $\sum_{z \in Z} P(c=1|u,i,z)\,P(z)$ estimation

$$P(c=1|do(u,i)) = \sum_{z \in Z} P(c=1|u,i,z)\,P(z) = \sum_{z \in Z} P_\Theta\left(c=1|u,i,m_i^t\right) P(z)$$
$$= \sum_{z \in Z} \mathrm{ELU}'\left(f_\Theta(u,i)\right) \times \left(m_i^t\right)^\gamma P(z) = \mathrm{ELU}'\left(f_\Theta(u,i)\right) \times \sum_{z \in Z} z^\gamma P(z)$$
$$= \mathrm{ELU}'\left(f_\Theta(u,i)\right) \times \mathbb{E}\left(Z^\gamma\right)$$

Popularity-bias Deconfounding (PD)

- $\mathbb{E}\left(Z^\gamma\right)$: constant. → Popularity term Z: 영향 사라짐. [Z→I path 사라짐.] → Bias의 negative effect 제거.
- Estimated parameters $\hat{\Theta} \rightarrow \mathrm{ELU}'\left(f_{\hat{\Theta}}(u,i)\right) \times \mathbb{E}\left(Z^\gamma\right) \rightarrow \hat{P}\left(c=1|do\left(U=u,I=i\right)\right)$

Training: (user, item) 만으로 학습. $\hat{\Theta} \rightarrow \mathrm{ELU}'\left(f_{\hat{\Theta}}(u,i)\right) \times \mathbb{E}\left(Z^\gamma\right) \rightarrow \hat{P}\left(c=1|do\left(U=u,I=i\right)\right)$

3. Framework

Adjusting Popularity Bias in Inference

▼ Adjusting Popularity Bias in Inference

- Popularity Bias의 better usage: promoting, i.e, ∃ target popularity bias $\tilde{z}$
- 따라서 target $\tilde{z}$ [desired popularity]를 inference에 반영할 수 있다.

$$P\left(c=1|do\left(U=u,I=i\right),do\left(Z=\tilde{z}\right)\right)=P_{\Theta}\left(c=1|u,i,\tilde{m}_i\right) \quad (7)$$

  - $\tilde{m}_i$: popularity value of $\tilde{z}$, modeling with the time-series forecasting method.

Popularity-bias Deconfounding and Adjusting (PDA)

$$\tilde{m}_i=m_i^T+\alpha\left(m_i^T-m_i^{T-1}\right) \quad (8)$$

- m_i^T: popularity value of the last stage T
- α: hyper-parameter, control the popularity drift

$$P_{\Theta}\left(c=1|u,i,m_i^t\right)=\mathrm{ELU}'\left(f_{\Theta}\left(u,i\right)\right)\times\left(m_i^t\right)^{\gamma} \quad (5)$$

$$\mathrm{PDA}_{u,i}=\mathrm{ELU}'\left(f_{\Theta}\left(u,i\right)\right)\times\left(\tilde{m}_i\right)^{\tilde{\gamma}} \quad (9)$$

- $\tilde{\gamma}$: hyper-parameter for the strength of popularity bias

💡 [Inference] $\mathrm{PDA}_{u,i}$: recommendation score → each user에 대하여 상위 K개의 item을 recommendation.

3. Framework

Adjusting Popularity Bias in Inference

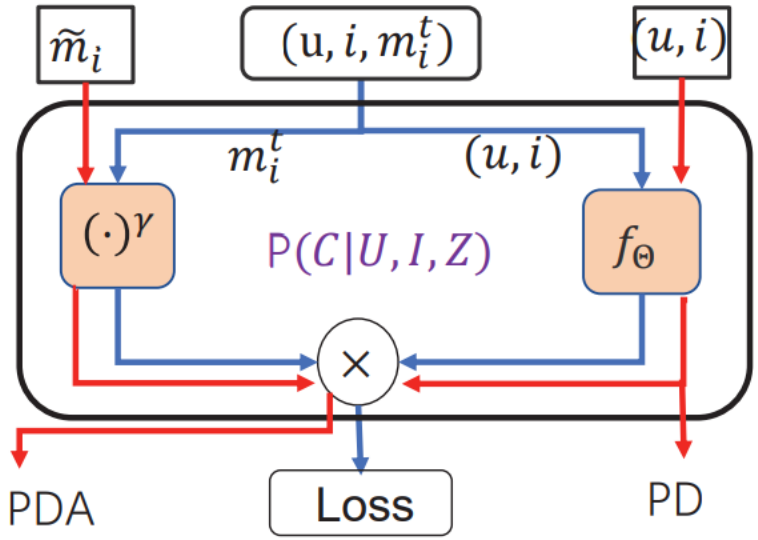


Figure 3: The workflow of our method. The blue arrows represent the training stage and the red arrows represent the inference stage.

Algorithm 1: PD/PDA

Input: dataset $\mathcal{D} = \{(u, i, m_i^t)\}$; hyper-parameter γ;
predicted popularity $\{\tilde{m}_i\}$; *mode*: PD or PDA

1 **while** *stop condition is not reached* **do**
2 Update model parameters Θ by optimizing Equation (4);
3 **if** *mode==PD* **then** validate model with $ELU'(f_\Theta(u, i))$;
4 **else** validate model with Equation (9) (simplify $\tilde{\gamma} = \gamma$);
5 **end**
6 **if** *mode==PD* **then** recommend items using $ELU'(f_\Theta(u, i))$;
7 **else** recommend items using Equation (9) (simplify $\tilde{\gamma} = \gamma$);

Inference: (user, item), target popularity bias $\tilde{z} \rightarrow \mathrm{PDA}_{u,i} = \mathrm{ELU}'\left(f_\Theta\left(u, i\right)\right) \times \left(\tilde{m}_i\right)^{\tilde{\gamma}} \rightarrow$ top-K recommendation.

3. Framework

Comparison with Correlation $P(C|U,I)$

- **Comparison with Correlation $P(C|U,I)$**

$$P(C|do(U,I)) = P_{G'}(C|U,I) = \sum_{z\in Z} P_{G'}(C,z|U,I) = \sum_{z\in Z} P_{G'}(C|U,I,z)\, P_{G'}(z|U,I)$$
$$= \sum_{z\in Z} P_{G'}(C|U,I,z)\, P_{G'}(z) = \sum_{z\in Z} P(C|U,I,z)\, P(z) \quad (3)$$

$$P(C|U,I) = P_G(C|U,I) = \sum_{z\in Z} P_G(C,z|U,I) = \sum_{z\in Z} P_G(C|U,I,z)\, P_G(z|U,I)$$
$$= \sum_{z\in Z} P_G(C|U,I,z)\, P_G(z|I) = \sum_{z\in Z} P(C|U,I,z)\, P(I|z)\, \frac{P(z)}{P(I)} \propto \sum_{z\in Z} P(C|U,I,z)\, P(I|z)\, P(z) \quad (10)$$

- $P(I|z) = P(I = i|Z = m_i^t)$ → Popularity Z가 training에 사용됨.
- $P(C|U,I)$: correlation-based training
- $P(C|do(U,I))$: causality-based training with causal intervention $do(\cdot)$

DGIST DATA SCIENCE LAB

DGIST 대구경북과학기술원 Daegu Gyeongbuk Institute of Science & Technology

4. Experiments

Experiments

- RQ1:
  1. **Does PD** achieve the goal of **removing the bad effect of popularity bias**?
  2. How is its **performance** compared with existing methods?
- RQ2:
  1. **Can PDA** effectively **inject desired popularity bias**?
  2. To what extent **leveraging popularity bias enhance** the recommendation **performance**.

Datasets

1. Kwai: clicking data
2. Douban Movie: user ratings for movies
3. Tencent: user interactions are "likes", which are reflective of user satisfaction but far more sparse than clicks.

4. Experiments

Baselines vs PD

Table 1: Recommendation performance after deconfounded training on the three datasets. "RI" refers to the relative improvement of PD over the corresponding baseline. The best results are highlighted in bold and sub-optimal results are underlined.

Dataset	Methods	Top 20					Top 50				
		Recall	Precision	HR	NDCG	RI	Recall	Precision	HR	NDCG	RI
Kwai	MostPop	0.0014	0.0019	0.0341	0.0030	632.4%	0.0040	0.0021	0.0802	0.0036	480.9%
	BPRMF	0.0054	0.0057	0.0943	0.0067	146.3%	0.0125	0.0053	0.1866	0.0089	121.0%
	xQuad	0.0054	0.0057	0.0948	0.0068	145.0%	0.0125	0.0053	0.1867	0.0090	120.3%
	BPR-PC	0.0070	0.0056	0.0992	0.0072	125.0%	0.0137	0.0046	0.1813	0.0092	123.7%
	DICE	0.0053	0.0056	0.0957	0.0067	147.8%	0.0130	0.0052	0.1872	0.0090	119.0%
	PD	**0.0143**	**0.0138**	**0.2018**	**0.0177**	-	**0.0293**	**0.0118**	**0.3397**	**0.0218**	-
Douban	MostPop	0.0218	0.0297	0.2373	0.0349	75.4%	0.0490	0.0256	0.3737	0.0406	55.9%
	BPRMF	0.0274	0.0336	0.2888	0.0405	47.0%	0.0581	0.0291	0.4280	0.0475	34.3%
	xQuad	0.0274	0.0336	0.2895	0.0391	48.3%	0.0581	0.0291	0.4281	0.0473	34.4%
	BPR-PC	0.0282	0.0307	0.2863	0.0381	51.6%	0.0582	0.0271	0.4260	0.0457	38.0%
	DICE	0.0273	0.0336	0.2845	0.0421	46.2%	0.0513	0.0273	0.4000	0.0460	44.5%
	PD	**0.0453**	**0.0454**	**0.3970**	**0.0607**	-	**0.0843**	**0.0362**	**0.5271**	**0.0686**	-
Tencent	MostPop	0.0145	0.0043	0.0684	0.0093	340.8%	0.0282	0.0035	0.1181	0.0135	345.8%
	BPRMF	0.0553	0.0153	0.2005	0.0328	27.1%	0.1130	0.0129	0.3303	0.0497	25.3%
	xQuad	0.0552	0.0153	0.2007	0.0326	27.3%	0.1130	0.0129	0.3302	0.0497	25.3%
	BPR-PC	0.0556	0.0153	0.2018	0.0331	26.5%	0.1141	0.0128	0.3322	0.0500	24.9%
	DICE	0.0516	0.0149	0.1948	0.0312	32.8%	0.1010	0.0132	0.3312	0.0486	29.0%
	PD	**0.0715**	**0.0195**	**0.2421**	**0.0429**	-	**0.1436**	**0.0165**	**0.3875**	**0.0641**	-

4. Experiments

Deconfounding Performance (RQ1)

Deconfounding Performance (RQ1)

Baselines vs PD

- Group 1-10: average popularity
- Group 1: 가장 인기 많음 ~ Group 10: 가장 인기 없음.

$$RR(g, algo) = \frac{\sum_{i \in Group_g} RC(i, algo)}{\sum_{i \in AllGroups} RC(i, algo)}$$

where $RC(i, algo)$ gives the number of times the algorithm *algo* recommends item i in top-K recommendation lists.

- **BPRMF**: Group 1에 많은 추천 → popularity와 추천이 비례 → popularity bias amplification.
- **DICE**: Group 5에 많은 추천 → 반반.
- **PD**: training 데이터의 RR 분포와 가장 유사 & 표준편차(std. dev.)가 가장 작음
  - → **bias amplification 제거 → popularity bias 영향 제거됨**

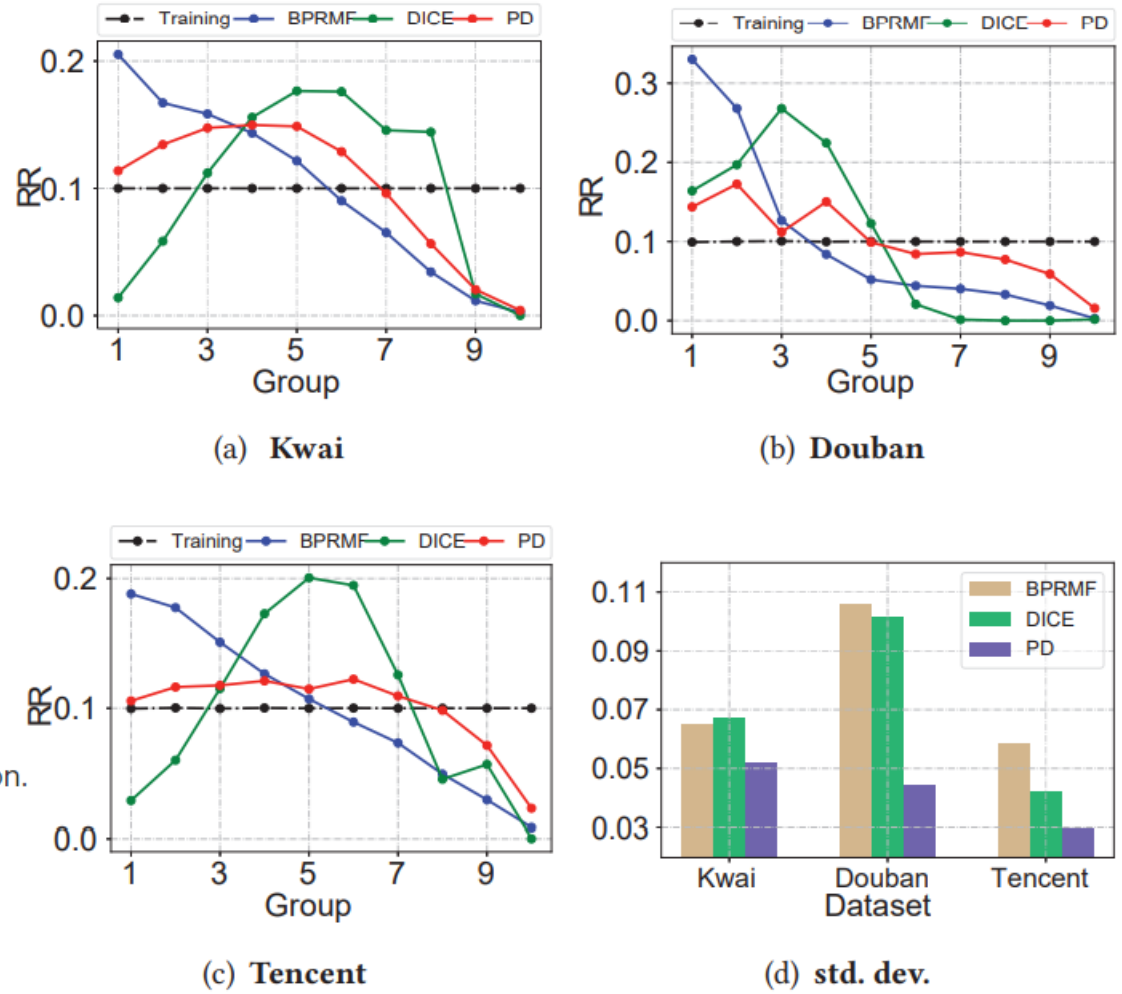
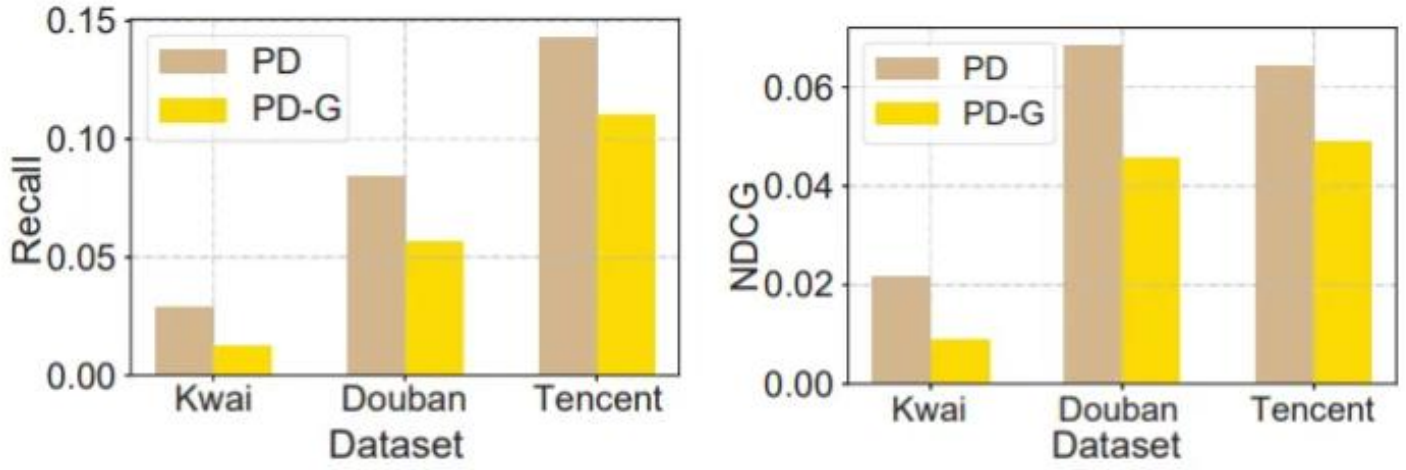


Figure 4: Recommendation rate(RR) over item groups.

4. Experiments

Deconfounding Performance (RQ1)

- Global Popularity V.S. Local Popularity



(a) Recall (b) NDCG

Figure 5: Comparisons between PD and PD-G on Recall@50 and NDCG@50. PD-G is a version of PD that computes item popularity over the total training set.

- 각 stage에 대응하는 Popularity 값을 쓰는 것이 더 성능 좋음.

4. Experiments

Performance of Adjusting Popularity (RQ2)

- (a): $\tilde{m}_i = m_i^T$
- (b): $\tilde{m}_i = m_i^T + \alpha\left(m_i^T - m_i^{T-1}\right)$ (8) → 더 성능 좋음.

desired [target] popularity

Table 2: Top-K recommendation performance with popularity adjusting on Kwai, Douban, and Tencent Datasets.

Datasets		Kwai				Douban				Tencent			
Methods		top 20		top 50		top 20		top 50		top 20		top 50	
		Recall	NDCG	Recall	NDCG	Recall	NDCG	Recall	NDCG	Recall	NDCG	Recall	NDCG
MostRecent		0.0074	0.0096	0.0139	0.011	0.0398	0.0582	0.0711	0.0615	0.0360	0.0222	0.0849	0.0359
BPRMF(t)-pop		0.0188	0.0241	0.0372	0.0286	0.0495	0.0682	0.0929	0.0760	0.1150	0.0726	0.2082	0.1001
BPRMF-A	(a)	0.0191	0.0249	0.0372	0.0292	0.0482	0.0666	0.0898	0.0744	0.1021	0.0676	0.1805	0.0905
	(b)	0.0201	0.0265	0.0387	0.0306	0.0486	0.0667	0.0901	0.0746	0.1072	0.0719	0.1886	0.0953
DICE-A	(a)	0.0242	0.0315	0.0454	0.0363	0.0494	0.0681	0.0890	0.0736	0.1227	0.0807	0.2161	0.1081
	(b)	0.0245	0.0323	0.0462	0.0370	0.0494	0.0680	0.0882	0.0734	0.1249	0.0839	0.2209	0.1116
PDA	(a)	0.0279	0.0352	0.0531	0.0413	0.0564	**0.0746**	**0.1066**	**0.0845**	0.1357	0.0873	0.2378	0.1173
	(b)	**0.0288**	**0.0364**	**0.0540**	**0.0429**	**0.0565**	0.0745	**0.1066**	0.0843	**0.1398**	**0.0912**	**0.2418**	**0.1210**

4. Experiments

Performance of Adjusting Popularity (RQ2)

More Refined Predictions for Popularity.

- Uniformly split the data in the last training stage into N sub-stages by time
- (a): $\tilde{m}_i = m_i^T$ 사용.
- N이 증가할수록 성능 증가하는 경향 → popularity를 더 정밀하게 예측할수록 성능 향상.
- *This experiment shows that the performance of PDA can be further improved with a more precise prediction of popularity.*

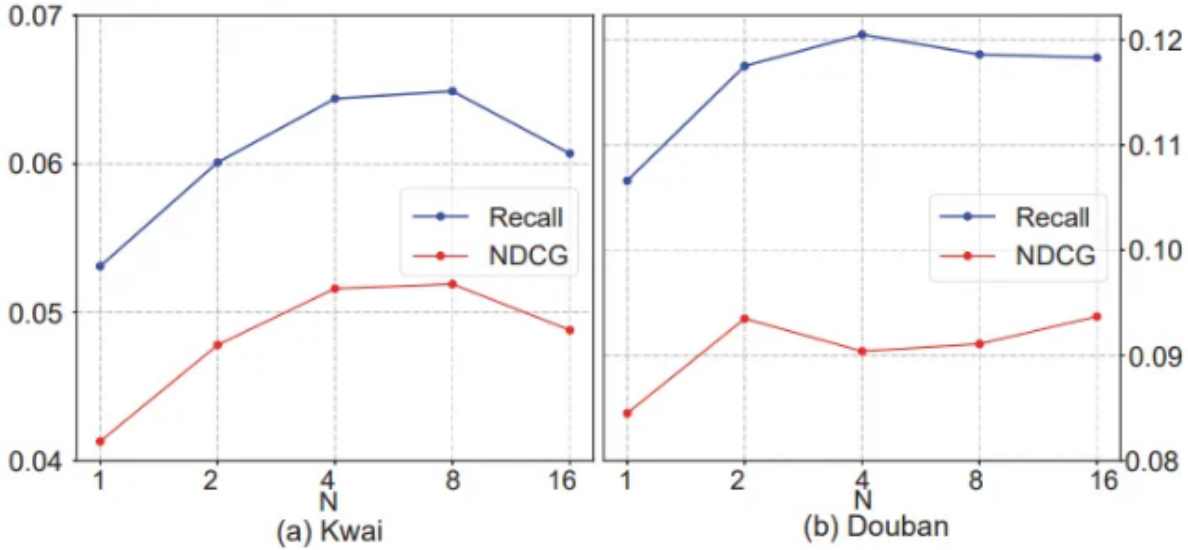


Figure 6: PDA recommendation performance regarding Recall@50 and NDCG@50 on Kwai and Douban, with the last training stage split into N sub-stages.

DGIST DATA SCIENCE LAB

4. Experiments

Performance of Adjusting Popularity (RQ2)

Total Improvements of PDA.

Table 3: The total relative improvements of PDA compared with the base recommendation model, *i.e.*, BPRMF. Each result is averaged on top 20 and top 50.

Dataset	Recall	Precision	HR	NDCG
Kwai	488%	441%	241%	532%
Douban	124%	68%	57%	97%
Tencent	133%	133%	79%	161%