Pre-trained Recommender Systems: A Causal Perspective

ZIQIAN LIN[*†], University of Wisconsin-Madison, USA
HAO DING[*], AWS AI Lab, USA
NGHIA HOANG[‡], Washington State University, USA
BRANISLAV KVETON, ANOOP DEORAS, and HAO WANG, AWS AI Labs, USA

ACM Reference Format:
Ziqian Lin, Hao Ding, Nghia Hoang, Branislav Kveton, Anoop Deoras, and Hao Wang. 2023. Pre-trained Recommender Systems: A Causal Perspective. In . ACM, 13 pages. https://doi.org/X.X.X.X

1 PRE-TRAINED RECOMMENDER SYSTEMS

Recent studies on pre-trained vision/language models such as BERT [6] and GPT [26] have demonstrated the benefit of a promising solution-building paradigm where models can be pre-trained on broad data describing a generic task space and then adapted successfully to solve a wide range of downstream tasks, even when training data of downstream task is limited. Inspired by such progress, we investigate the possibilities and challenges of adapting such a paradigm to the context of recommender systems, and propose a causal recommender model named PreRec. It captures generic interaction patterns by training on diverse user-item interaction data extracted from different domains, which can then be fast adapted to improve zero- and few-shot learning performance in unseen new domains (with no or limited data).

The key to learning a generalizable model lies in capturing knowledge grounded on a universal feature space. To bridge domain discrepancies, ZESRec [7] first proposed to use generic item textual description to produce the item universal embedding, while the user universal embedding is computed via a sequential model that aggregates item universal embeddings for items in the user history. Despite the promising results, ZESRec limits itself to pre-training on a single source domain and inference on a single target domain. As a follow-up work, UniSRec [12] further extends support for multi-domain pre-training and evaluate the pre-trained model on multiple target domains. However, UniSRec fails to consider the bias either within each domain or across domains during pre-training, while both may lead to drift in user interests, item properties, and user behavioral patterns.

In this work, we aim to design an universally generalizable recommender that can be pre-trained on multiple source domains and fine-tuned on different target domains. We start by identifying two types of bias: **In-domain bias** considers noises taking effects within each domain, and one example is the popularity bias that affects not only the item exposure rate, but also the user behavioral pattern due to users tend to follow the majority and interact with trending items. **Cross-domain bias** considers noises introduced by the unique domain properties. For instance, each domain has distinctive user community, which causes the shift in user interests across different domains.

[*]Both authors contributed equally to this research.
[†]Work done during the author's internship at AWS AI Labs.
[‡]Work done at AWS AI Labs.

Manuscript submitted to ACM

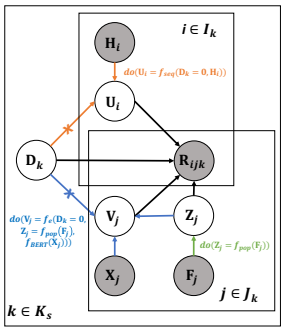

Fig. 1. The causal graphical model for PreRec. λ_u, λ_v, λ_d, λ_z are hyperparameters related to distribution variance of variables U_i, V_j, D_k, Z_j, which are omitted from the graph for simplicity.

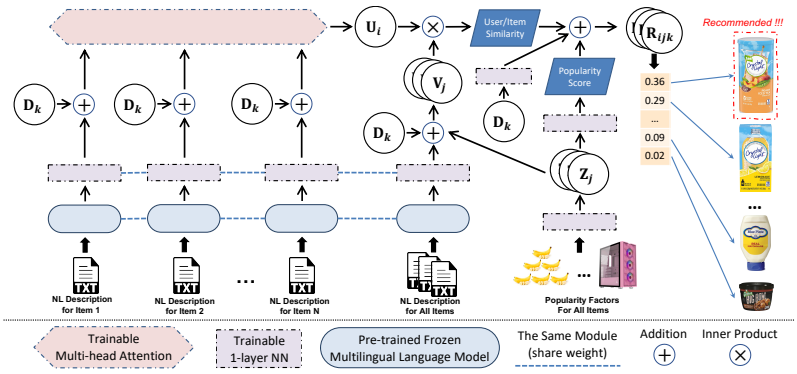

Fig. 2. The neural network implementation for our PreRec. Analogous to Fig. 2, f_{BERT} is implemented as a multilingual language model, f_{seq} is implemented as a transformer decoder, and f_{pop} is implemented as a linear layer with activation.

Fig. 2 illustrates the implementation from a neural network perspective. Fig. 1 shows our graphical model, specifically:

- Variable $\mathbf{D}_k \in \mathbb{R}^B$ represents the (latent) properties of domain k such as the user community. B is the embedding hidden dimension. We identify $\mathbf{D}_k$ as a *confounder* as it affects user $\mathbf{U}_i$, item $\mathbf{V}_j$, and interactions $\mathbf{R}_{ijk}$.
- Variables $\mathbf{U}_i \in \mathbb{R}^B$ and $\mathbf{H}_i \in \mathbb{R}^{N_u \times B}$ represent interests of user i and user i's interaction history in domain k, and N_u represents the number of items in the history of user i.
- Variable $\mathbf{F}_j \in \mathbb{R}^C$ represents the popularity factors of item j in domain k, including two prominent factors: (1) number of recent interactions of item j, (2) numbers of recent interactions of all items in domain k.
- Variables $\mathbf{X}_j, \mathbf{Z}_j \in \mathbb{R}^B$, and $\mathbf{V}_j \in \mathbb{R}^B$ are item j's description, popularity representation, and overall properties. We identify $\mathbf{Z}_j$ as a *confounder* affecting both item $\mathbf{V}_j$ and interactions $\mathbf{R}_{ijk}$. (See Appendix E for computing $\mathbf{Z}_j$.)
- Variable $\mathbf{R}_{ijk} \in \{0, 1\}$ is the interaction label denoting whether user i has interacted with item j in domain k.

Pre-training. For the pre-training stage of PreRec, we assume the user-item interactions $\mathbf{R}_{ijk}$ and item textual description $\mathbf{X}_j$ are observed for each domain, while the popularity factors $\mathbf{F}_j$ and user history $\mathbf{H}_i$ can be directly derived from the interaction $\mathbf{R}_{ijk}$. With hyperparameters λ_u, λ_v, λ_d, and λ_z and given the graphical model in Fig. 1, the maximum a posteriori (MAP) estimation on latent variables $(\mathbf{U}_i, \mathbf{V}_j, \mathbf{D}_k, \mathbf{Z}_j)$ can be decomposed as following:

$$P(\mathbf{U}_i, \mathbf{V}_j, \mathbf{D}_k, \mathbf{Z}_j | \mathbf{R}_{ijk}, \mathbf{H}_i, \mathbf{X}_j, \mathbf{F}_j, \lambda_u, \lambda_v, \lambda_d, \lambda_z) \propto$$
$$P(\mathbf{R}_{ijk} | \mathbf{U}_i, \mathbf{V}_j, \mathbf{D}_k, \mathbf{Z}_j) \cdot P(\mathbf{U}_i | \mathbf{H}_i, \mathbf{D}_k, \lambda_u) \cdot P(\mathbf{V}_j | \mathbf{X}_j, \mathbf{D}_k, \mathbf{Z}_j, \lambda_v) \cdot P(\mathbf{Z}_j | \mathbf{F}_j, \lambda_z) \cdot P(\mathbf{D}_k | \lambda_d). \quad (1)$$

We define the conditional probability over the observed interactions as:

$$P(\mathbf{R}_{ijk} | \mathbf{U}_i, \mathbf{V}_j, \mathbf{D}_k, \mathbf{Z}_j) = f_{softmax}(\mathbf{U}_i^T \mathbf{V}_j + \mathbf{D}_k \mathbf{W}_d + \mathbf{Z}_j \mathbf{W}_z). \quad (2)$$

We assume Gaussian distributions on all latent variables in Fig. 1, i.e., $\mathbf{U}_i$, $\mathbf{V}_j$, $\mathbf{D}_k$, and $\mathbf{Z}_j$, as follows:

$$P(\mathbf{U}_i | \mathbf{H}_i, \mathbf{D}_k, \lambda_u) = \mathcal{N}(\mathbf{U}_i; f_{seq}(\mathbf{D}_k, \mathbf{H}_i), \lambda_u^{-1} \mathbf{I}_B);\ P(\mathbf{V}_j | \mathbf{X}_j, \mathbf{D}_k, \mathbf{Z}_j, \lambda_v) = \mathcal{N}(\mathbf{V}_j; f_{item}(\mathbf{D}_k, \mathbf{Z}_j, f_e(\mathbf{X}_j)), \lambda_v^{-1} \mathbf{I}_B);$$
$$P(\mathbf{Z}_j | \mathbf{F}_j, \lambda_z) = \mathcal{N}(\mathbf{Z}_j; f_{pop}(\mathbf{F}_j), \lambda_z^{-1} \mathbf{I}_B);\ P(\mathbf{D}_k | \lambda_d) = \mathcal{N}(\mathbf{D}_k; \mathbf{0}, \lambda_d^{-1} \mathbf{I}_B), \quad (3)$$

where $\mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, \lambda^{-1} \mathbf{I}_B)$ denotes the probability density function (PDF) of a Gaussian distribution with mean $\boldsymbol{\mu}$ and diagonal covariance $\lambda^{-1} \mathbf{I}_B$ for the variable $\mathbf{x}$; $f_e(\cdot)$ and $f_{pop}(\cdot)$ are the learnable encoding functions for item embedding $\mathbf{V}_j$ and popularity embedding $\mathbf{Z}_j$, respectively. In our work, we adopt multi-layer perceptron (MLP) for $f_e(\cdot)$ and $f_{pop}(\cdot)$.

Maximizing the posterior probability is equivalent to minimizing the negative log likelihood (NLL) of Eqn. 1:

$$\mathcal{L} = -\sum_{k=1}^{K_s}\sum_{i=1}^{I_k}\sum_{j=1}^{J_k}\mathbf{R}_{ijk}\log(f_{softmax}(\mathbf{U}_i^T\mathbf{V}_j + \mathbf{D}_k\mathbf{W}_d + \mathbf{Z}_j\mathbf{W}_z)) + \frac{\lambda_z}{2}\sum_{j=1}^{J_k}||\mathbf{Z}_j - f_{pop}(\mathbf{F}_j)||^2 + \frac{\lambda_d}{2}\sum_{k=1}^{K_s}||\mathbf{D}_k||^2$$
$$+\frac{\lambda_v}{2}\sum_{k=1}^{K_s}\sum_{j=1}^{J_k}||\mathbf{V}_j - f_e(\mathbf{D}_k, \mathbf{Z}_j)||^2 + \frac{\lambda_u}{2}\sum_{k=1}^{K_s}\sum_{i=1}^{I_k}||\mathbf{U}_i - f_{seq}(\mathbf{D}_k, \mathbf{H}_i)||^2, \tag{4}$$

where $\mathbf{W}_d \in \mathbb{R}^B$, $\mathbf{W}_z \in \mathbb{R}^B$ are learnable weights. K_s denotes a set of source domains; for each domain k, we have I_k users and J_k items.

Causal Zero-shot Recommendation. Given the pre-trained model, we enforce intervention on $\mathbf{U}_i$, $\mathbf{V}_j$ and $\mathbf{Z}_j$ by performing the *do*-calculus [27] to eliminate the cross-domain bias while incorporate the in-domain bias in the target domain, as shown in Fig. 1. Specifically, we do not assume any domain properties and thus the posterior of $\mathbf{D}_k$ collapses to the prior $\mathbf{0}$; the popularity factors $\mathbf{F}_j$ are based on data statistics which can be derived on the fly in an online environment. Therefore, we compute $\mathbf{U}_i = f_{seq}(\mathbf{D}_k = \mathbf{0}, \mathbf{H}_i)$ and $\mathbf{V}_j = f_e(\mathbf{D}_k = \mathbf{0}, \mathbf{Z}_j = f_{pop}(\mathbf{F}_j), f_{BERT}(\mathbf{X}_j))$. Here we set $\mathbf{D}_k = \mathbf{0}$ for $\mathbf{U}_i$ and $\mathbf{V}_j$. This serves as an approximation to the output expectation over the distribution of $\mathbf{D}_k$ as input, and we found this approach achieved similar performance in practice. Following the *back-door formula* [27] we have:

$$P(\mathbf{R}_{ijk}|do(\mathbf{U}_i, \mathbf{V}_j, \mathbf{Z}_j))) = \int f_{softmax}(\mathbf{U}_i^T\mathbf{V}_j + \mathbf{D}_k\mathbf{W}_d + \mathbf{Z}_j\mathbf{W}_z)P(\mathbf{D}_k)d\mathbf{D}_k = \int \frac{exp(\mathbf{U}_i^T\mathbf{V}_j + \mathbf{Z}_j\mathbf{W}_z)exp(\mathbf{D}_k\mathbf{W}_d)}{\sum_j exp(\mathbf{U}_i^T\mathbf{V}_j + \mathbf{Z}_j\mathbf{W}_z)exp(\mathbf{D}_k\mathbf{W}_d)}P(\mathbf{D}_k)d\mathbf{D}_k$$
$$= \int \frac{exp(\mathbf{U}_i^T\mathbf{V}_j + \mathbf{Z}_j\mathbf{W}_z)}{\sum_j exp(\mathbf{U}_i^T\mathbf{V}_j + \mathbf{Z}_j\mathbf{W}_z)}P(\mathbf{D}_k)d\mathbf{D}_k = f_{softmax}(\mathbf{U}_i^T\mathbf{V}_j + \mathbf{Z}_j\mathbf{W}_z). \tag{5}$$

Note that for $f_{softmax}(\cdot)$ we only consider items in the same domain.

Fine-tuning. Once the interaction data in the target domain is available, we can then fine-tune all the parameters of PreRec end-to-end in this new domain. During fine-tuning, we optimize Eqn. 4 on the target domain data and re-estimate all latent variables along with corresponding encoding functions as stated in Eqn. 2 and Eqn. 3.

For inference after fine-tuning, With the re-estimated latent vectors $(\widehat{\mathbf{D}}_k, \widehat{\mathbf{Z}}_j, \widehat{\mathbf{V}}_j, \widehat{\mathbf{U}}_i)$ and all learnable parameters inside $f_{softmax}(\cdot)$, i.e., $\widehat{\mathbf{W}}_d$ and $\widehat{\mathbf{W}}_z$, we then perform causal inference by intervening on the domain bias $(\mathbf{D}_k)$, popularity bias $(\mathbf{Z}_j)$, item properties $(\mathbf{V}_j)$, and user interests $(\mathbf{U}_i)$:

$$P(\mathbf{R}_{ijk}|do(\mathbf{U}_i, \mathbf{V}_j, \mathbf{D}_k, \mathbf{Z}_j)) = f_{softmax}(\widehat{\mathbf{U}}_i^T\widehat{\mathbf{V}}_j + \widehat{\mathbf{D}}_k\widehat{\mathbf{W}}_d + \widehat{\mathbf{Z}}_j\widehat{\mathbf{W}}_z). \tag{6}$$

2 EXPERIMENTS

In this section, we introduce the zero-shot performance in Sec. 2.1, the fine-tuning performance in Sec. 2.2. We pre-train PreRec on three source domains (*"India"*, *"Spain"*, and *"Canada"* in the XMarket dataset [2]) and then perform zero-shot and finetuning evaluation against baselines on five target domains (*"Australia"*, *"Mexico"*, *"Germany"*, *"Japan"* in the XMarket dataset as Cross-Market and the Online Retail dataset as Cross-Platform) respectively. With the proposed metrics, the average performance over target domains is meaningful as the task difficulties are on the same level. Due to the page limitation, we put sections Datasets, Baselines, and Evaluation Metrics into Appendix C. Our code is available at https://github.com/myhakureimu/PreRec.

2.1 Zero-shot Experiments

The zero-shot performance comparison between varied methods is shown in Fig. 3. Among the non-learnable models, SBERT [28] (Inner product between the last interacted item embedding and all item embeddings. Item embeddings

Table 1. Zero-shot performance comparison of different methods. K%= 0.04% for Cross-Market and K%= 0.5% for Cross-Platform. We mark the best results with **bold face** and the second best results with underline. Please refer to Appendix C.3 for definitions of Recall@K% and r-NDCG@K%. These metrics are specifically designed to measure the average performance of multiple domains.

Scenario	Dataset	Metric	Test Type	Random	POP	SBERT	ZESRec	UniSRec	PreRec$_n$	PreRec
Cross -Market	Australia	Recall@K%	all	0.0004	0.0450	0.0552	0.0431	0.0404	0.0583	**0.1036**
			unseen	0.0004	0.0127	0.0619	0.0472	0.0452	0.0655	**0.0756**
		r-NDCG@K%	all	0.0002	0.0261	0.0380	0.0287	0.0281	0.0396	**0.0656**
			unseen	0.0002	0.0063	0.0420	0.0323	0.0316	0.0438	**0.0476**
	Mexico	Recall@K%	all	0.0004	0.0695	0.1509	0.1397	0.1521	0.1645	**0.2316**
			unseen	0.0004	0.0248	0.1403	0.1157	0.1386	**0.1487**	0.1469
		r-NDCG@K%	all	0.0002	0.0388	0.1095	0.0961	0.1134	0.1178	**0.1557**
			unseen	0.0002	0.0111	0.1009	0.0748	0.1000	**0.1027**	0.1017
	Germany	Recall@K%	all	0.0004	0.1514	0.2737	0.2639	0.2703	0.2827	**0.3526**
			unseen	0.0004	0.1016	0.2528	0.2295	0.2430	0.2506	**0.2750**
		r-NDCG@K%	all	0.0002	0.0936	0.2103	0.1886	0.2100	0.2103	**0.2576**
			unseen	0.0002	0.0559	0.1910	0.1703	0.1852	0.1923	**0.1954**
	Japan	Recall@K%	all	0.0004	0.0657	0.2926	0.2259	0.2579	0.2817	**0.3083**
			unseen	0.0004	0.0556	0.2737	0.2152	0.2662	0.2730	**0.2741**
		r-NDCG@K%	all	0.0002	0.0376	0.2016	0.1514	0.1776	0.1929	**0.2033**
			unseen	0.0002	0.0325	**0.2061**	0.1545	0.1967	0.1949	0.1980
Cross -Platform	Online Retail	Recall@K%	all/unseen	0.0050	0.0672	0.1383	0.0564	0.1147	0.1397	**0.1794**
		r-NDCG@K%	all/unseen	0.0025	0.0362	0.0848	0.0316	0.0698	0.0841	**0.1065**

are encoded based on item descriptions) outperforms Random and POP since it leverages additional item descriptions. Importantly, SBERT achieves comparable performance with the learnable models, ZESRec [7], UniSRec [12], and PreRec$_n$, which shows that without a proper pre-training strategy dealing with variance and bias in the pre-training data, the pre-training stage may not bring large performance gain. Among the learnable zero-shot models, UnisRec slightly outperforms ZESRec because it includes additional contrastive training objectives. Our PreRec significantly outperforms all baselines by a large margin in most cases, thanks to the causal mechanism for debiasing.

The improvement on unseen items is relatively lower than the overall improvement. With additional data analysis, we discovered that unseen items are usually unpopular items in target domains; this is also why POP's performance on "unseen" items is worse than that in "all" items.

Ablation Study on Causal Mechanism To demonstrate the benefit of the causal model in Fig. 1, we denote PreRec$_n$ as a vanilla version of the graphical model in Fig. 1. PreRec$_n$ does not consider in-domain and cross-domain bias, *i.e.*, without nodes $\mathbf{D}_k$ and $\mathbf{Z}_k$. Table 1 shows that in almost all cases PreRec outperforms PreRec$_n$ by a substantial margin.

2.2 Fine-tuning Experiments

Fig 3 in Appendix A.2 shows the results of incremental fine-tuning. Among all target domains, PreRec consistently outperforms all baselines, with the performance steadily improving as the number of target domain training samples increases, showing PreRec is progressively adapting to the target domain; the performance gap between PreRec and the baselines is gradually shrinking, suggesting diminishing complementary value of the distilled knowledge during pre-training. It is worth noting that the performance gap between PreRec and the baselines is still prominent even when there are as many as 10^4 target domain training samples, which are equivalent to several weeks or even months of effort in collecting data. Additionally, We put numerical results of full fine-tuning in the Appendix A.1.

REFERENCES

[1] Ye Bi, Liqiang Song, Mengqiu Yao, Zhenyu Wu, Jianming Wang, and Jing Xiao. 2020. A Heterogeneous Information Network based Cross Domain Insurance Recommendation System for Cold Start Users. In *SIGIR*. 2211–2220.

[2] Hamed Bonab, Mohammad Aliannejadi, Ali Vardasbi, Evangelos Kanoulas, and James Allan. 2021. Cross-market product recommendation. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 110–119.

[3] Jiawei Chen, Hande Dong, Xiang Wang, Fuli Feng, Meng Wang, and Xiangnan He. 2023. Bias and debias in recommender system: A survey and future directions. *ACM Transactions on Information Systems* 41, 3 (2023), 1–39.

[4] Junyoung Chung, **c**Caglar Gül**c**ehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. *CoRR* abs/1412.3555 (2014). arXiv:1412.3555 http://arxiv.org/abs/1412.3555

[5] Zeyu Cui, Jianxin Ma, Chang Zhou, Jingren Zhou, and Hongxia Yang. 2022. M6-Rec: Generative Pretrained Language Models are Open-Ended Recommender Systems. *CoRR* abs/2205.08084 (2022). https://doi.org/10.48550/arXiv.2205.08084 arXiv:2205.08084

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[7] Hao Ding, Yifei Ma, Anoop Deoras, Yuyang Wang, and Hao Wang. 2021. Zero-shot recommender systems. *arXiv preprint arXiv:2105.08318* (2021).

[8] Xiaoyu Du, Zike Wu, Fuli Feng, Xiangnan He, and Jinhui Tang. 2022. Invariant Representation Learning for Multimedia Recommendation. In *MM '22: The 30th ACM International Conference on Multimedia*. 619–628.

[9] Shijie Geng, Shuchang Liu, Zuohui Fu, Yingqiang Ge, and Yongfeng Zhang. 2022. Recommendation as Language Processing (RLP): A Unified Pretrain, Personalized Prompt & Predict Paradigm (P5). In *RecSys '22: Sixteenth ACM Conference on Recommender Systems, Seattle, WA, USA, September 18 - 23, 2022*. ACM, 299–315.

[10] Casper Hansen, Christian Hansen, Jakob Grue Simonsen, Stephen Alstrup, and Christina Lioma. 2020. Content-aware Neural Hashing for Cold-start Recommendation. In *SIGIR*. 971–980.

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

[12] Yupeng Hou, Shanlei Mu, Wayne Xin Zhao, Yaliang Li, Bolin Ding, and Ji-Rong Wen. 2022. Towards Universal Sequence Representation Learning for Recommender Systems. *arXiv preprint arXiv:2206.05941* (2022).

[13] Guangneng Hu, Yu Zhang, and Qiang Yang. 2018. CoNet: Collaborative Cross Networks for Cross-Domain Recommendation. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management, CIKM*. 667–676.

[14] Wang-Cheng Kang and Julian McAuley. 2018. Self-attentive sequential recommendation. In *2018 IEEE international conference on data mining (ICDM)*. IEEE, 197–206.

[15] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Yoshua Bengio and Yann LeCun (Eds.). http://arxiv.org/abs/1412.6980

[16] Jingjing Li, Mengmeng Jing, Ke Lu, Lei Zhu, Yang Yang, and Zi Huang. 2019. From zero-shot learning to cold-start recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 4189–4196.

[17] Pan Li and Alexander Tuzhilin. 2020. DDTCDR: Deep Dual Transfer Cross Domain Recommendation. In *WSDM '20: The Thirteenth ACM International Conference on Web Search and Data Mining, Houston, TX, USA, February 3-7, 2020*, James Caverlee, Xia (Ben) Hu, Mounia Lalmas, and Wei Wang (Eds.). ACM, 331–339. https://doi.org/10.1145/3336191.3371793

[18] Siqing Li, Liuyi Yao, Shanlei Mu, Wayne Xin Zhao, Yaliang Li, Tonglei Guo, Bolin Ding, and Ji-Rong Wen. 2021. Debiasing Learning based Cross-domain Recommendation. In *KDD '21: The 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 3190–3199.

[19] Xiaopeng Li and James She. 2017. Collaborative variational autoencoder for recommender systems. In *KDD*. 305–314.

[20] Tingting Liang, Congying Xia, Yuyu Yin, and Philip S. Yu. 2020. Joint Training Capsule Network for Cold Start Recommendation. In *SIGIR*. 1769–1772.

[21] Meng Liu, Jianjun Li, Guohui Li, and Peng Pan. 2020. Cross Domain Recommendation via Bi-directional Transfer Graph Collaborative Filtering Networks. In *CIKM '20: The 29th ACM International Conference on Information and Knowledge Management*. 885–894.

[22] Siwei Liu, Iadh Ounis, Craig Macdonald, and Zaiqiao Meng. 2020. A Heterogeneous Graph Neural Model for Cold-Start Recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2029–2032.

[23] Yifei Ma, Balakrishnan Narayanaswamy, Haibin Lin, and Hao Ding. 2020. Temporal-contextual recommendation in real-time. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2291–2299.

[24] Yifei Ma, Balakrishnan (Murali) Narayanaswamy, Haibin Lin, and Hao Ding. 2020. Temporal-Contextual Recommendation in Real-Time. In *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, CA, USA, August 23-27, 2020*, Rajesh Gupta, Yan Liu, Jiliang Tang, and B. Aditya Prakash (Eds.). ACM, 2291–2299. https://doi.org/10.1145/3394486.3403278

[25] Andriy Mnih and Russ R Salakhutdinov. 2007. Probabilistic matrix factorization. *Advances in neural information processing systems* 20 (2007).

[26] OpenAI. 2023. GPT-4 Technical Report. arXiv:2303.08774 [cs.CL]

[27] Judea Pearl. 2009. *Causality*. Cambridge university press.

[28] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics. http://arxiv.org/abs/1908.10084

[29] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2012. BPR: Bayesian personalized ranking from implicit feedback. *arXiv preprint arXiv:1205.2618* (2012).

[30] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).

[31] Hao Wang, Xingjian Shi, and Dit-Yan Yeung. 2016. Collaborative recurrent autoencoder: Recommend while learning to fill in the blanks. In *NIPS*. 415–423.

[32] Hao Wang, Naiyan Wang, and Dit-Yan Yeung. 2015. Collaborative deep learning for recommender systems. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*. 1235–1244.

[33] Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu, and Tat-Seng Chua. 2019. Kgat: Knowledge graph attention network for recommendation. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. 950–958.

[34] Yichao Wang, Huifeng Guo, Bo Chen, Weiwen Liu, Zhirong Liu, Qi Zhang, Zhicheng He, Hongkun Zheng, Weiwei Yao, Muyu Zhang, et al. 2022. CausalInt: Causal Inspired Intervention for Multi-Scenario Recommendation. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 4090–4099.

[35] Zimu Wang, Yue He, Jiashuo Liu, Wenchao Zou, Philip S. Yu, and Peng Cui. 2022. Invariant Preference Learning for General Debiasing in Recommendation. In *KDD '22: The 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 1969–1978.

[36] Le Wu, Yonghui Yang, Lei Chen, Defu Lian, Richang Hong, and Meng Wang. 2020. Learning to Transfer Graph Embeddings for Inductive Graph based Recommendation. In *SIGIR*. 1211–1220.

[37] Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan. 2019. Session-based recommendation with graph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 33. 346–353.

[38] Fajie Yuan, Xiangnan He, Alexandros Karatzoglou, and Liguang Zhang. 2020. Parameter-Efficient Transfer from Sequential Behaviors for User Modeling and Recommendation. In *SIGIR*. 1469–1478.

[39] Yuhui Zhang, HAO DING, Zeren Shui, Yifei Ma, James Zou, Anoop Deoras, and Hao Wang. 2021. Language Models as Recommender Systems: Evaluations and Limitations. In *I (Still) Can't Believe It's Not Better! NeurIPS 2021 Workshop*. https://openreview.net/forum?id=hFx3fY7-m9b

[40] Yang Zhang, Fuli Feng, Xiangnan He, Tianxin Wei, Chonggang Song, Guohui Ling, and Yongdong Zhang. 2021. Causal Intervention for Leveraging Popularity Bias in Recommendation. In *SIGIR '21: The 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, Virtual Event, Canada, July 11-15, 2021*, Fernando Diaz, Chirag Shah, Torsten Suel, Pablo Castells, Rosie Jones, and Tetsuya Sakai (Eds.). ACM, 11–20. https://doi.org/10.1145/3404835.3462875

[41] Yang Zhang, Fuli Feng, Xiangnan He, Tianxin Wei, Chonggang Song, Guohui Ling, and Yongdong Zhang. 2021. Causal intervention for leveraging popularity bias in recommendation. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 11–20.

[42] Cheng Zhao, Chenliang Li, and Cong Fu. 2019. Cross-Domain Recommendation via Preference Propagation GraphNet. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management, CIKM*. 2165–2168.

[43] Yu Zheng, Chen Gao, Xiang Li, Xiangnan He, Yong Li, and Depeng Jin. 2020. Disentangling User Interest and Popularity Bias for Recommendation with Causal Embedding. *CoRR* abs/2006.11011 (2020). arXiv:2006.11011 https://arxiv.org/abs/2006.11011

[44] Yu Zheng, Chen Gao, Xiang Li, Xiangnan He, Yong Li, and Depeng Jin. 2021. Disentangling user interest and conformity for recommendation with causal embedding. In *Proceedings of the Web Conference 2021*. 2980–2991.

[45] Feng Zhu, Yan Wang, Chaochao Chen, Jun Zhou, Longfei Li, and Guanfeng Liu. 2021. Cross-Domain Recommendation: Challenges, Progress, and Prospects. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021, Virtual Event / Montreal, Canada, 19-27 August 2021*, Zhi-Hua Zhou (Ed.). ijcai.org, 4721–4728. https://doi.org/10.24963/ijcai.2021/639

[46] Ziwei Zhu, Shahin Sefati, Parsa Saadatpanah, and James Caverlee. 2020. Recommendation for New Users and New Items via Randomized Training and Mixture-of-Experts Transformation. In *SIGIR*. 1121–1130.

A ADDITIONAL EXPERIMENT

For the fine-tuning experiments, we test under two settings: (1) *full fine-tuning* where we train or fine-tune models on all available training data in the target domain, (i.e., the training split of the target domain), and (2) *incremental fine-tuning* where we train or fine-tune models on a set of incrementally larger training data in the target domain. All the models are evaluated on the test set of the target domain. There are three sets of baselines: (1) **non-learnable zero-shot model**: Random, POP, SBERT; (2) **In-domain model**: GRU4Rec, SASRec; (3) **Pre-trained model**: UniSRec.

Scenario	Dataset	Metric	Random	POP	SBERT	GRU4Rec*	SASRec*	UniSRec	PreRec
Cross -Market	Australia	Recall@K%	0.0004	0.0450	0.0552	0.0546	0.0735	0.0600	**0.1130**
		r-NDCG@K%	0.0002	0.0261	0.0380	0.0349	0.0495	0.0416	**0.0715**
	Mexico	Recall@K%	0.0004	0.0695	0.1509	0.2315	0.2475	0.2478	**0.2646**
		r-NDCG@K%	0.0002	0.0388	0.1095	0.1683	0.1813	**0.1832**	0.1825
	Germany	Recall@K%	0.0004	0.1514	0.2737	0.3344	0.3428	0.3235	**0.3935**
		r-NDCG@K%	0.0002	0.0936	0.2103	0.2653	0.2726	0.2697	**0.2964**
	Japan	Recall@K%	0.0004	0.0657	0.2926	0.3428	0.3645	0.3587	**0.3813**
		r-NDCG@K%	0.0002	0.0376	0.2016	0.2475	0.2554	0.2549	**0.2601**
Cross -Platform	Online Retail	Recall@K%	0.0050	0.0672	0.1383	0.2524	0.2728	0.2513	**0.2992**
		r-NDCG@K%	0.0025	0.0362	0.0848	0.1545	0.1665	0.1589	**0.1769**

Table 2. Full fine-tuning results of different models. K%= 0.04% for Cross-Market and K%= 0.5% for Cross-Platform. The notation * indicates the model is trained from scratch. We mark the best results with **bold face** and the second best results with underline.

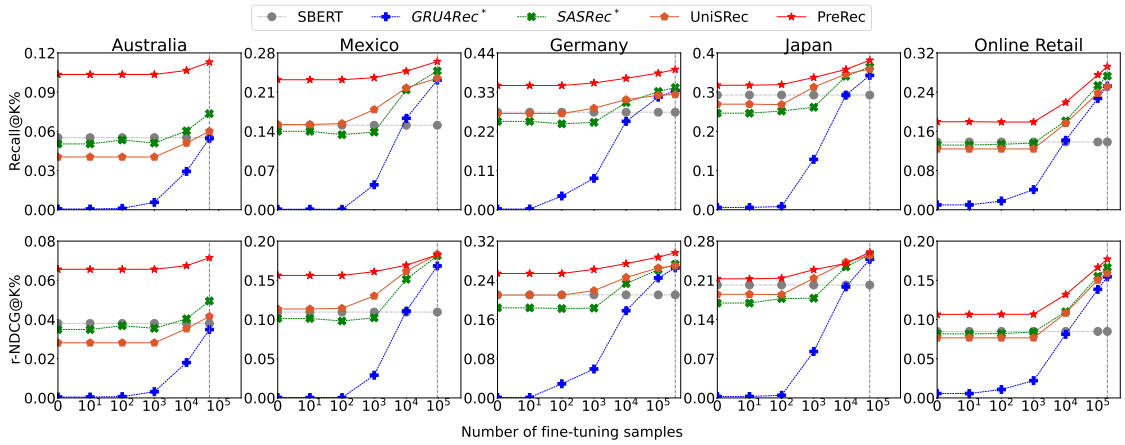

Fig. 3. The solid line indicates the model is pre-trained while the dashed line indicates the model is trained from scratch. The last point for each line corresponds to the full fine-tuning with all available training data in the target domain.

A.1 Full Fine-tuning Experiments

Table 2 shows the full fine-tuning performance of different methods. All in-domain models (GRU4Rec and SASRec) achieve better performance than SBERT, which is the best non-learnable zero-shot model, due to training on the target domain data. The pre-trained model UniSRec achieves performance similar to the in-domain models', showing that UniSRec brings no additional gain from pre-training on other domains if there is sufficient data in target domains. PreRec outperforms all baselines on almost all cases under both cross-market and cross-platform scenarios by a large margin after fine-tuning, demonstrating that: (1) PreRec distills ample generic knowledge in the pre-training stage, and it still has complementary value even after fine-tuning on target domain data; (2) PreRec is capable of rapidly adapting to the new domain without forgetting the distilled knowledge during pre-training (i.e., robust to catastrophic forgetting).

A.2 Incremental Fine-tuning Experiments

Fig 3 shows the incremental fine-tuning results as described in Sec. 2.2.

B RELATED WORK

In-domain Recommendation There is a rich literature on in-domain recommendation, i.e., training and testing the recommender systems on the same domain. Collaborative filtering methods such as PMF [25] and BPR [29] are first proposed to approach this problem. Later the deep learning methods such as GRU [4], Transformer [30] and Graph Neural Network [37] were proposed and demonstrated superiority on a variety of tasks, works such as GRU4Rec [11], SAS4Rec [14] and KGAT [33] adopt these latest neural network architectures and achieve great success in the recommendation regime. These methods assume only rating/interaction data is available without any content information (e.g., text description for items). Such an assumption precluded their direct application to pretraining-based recommenders. A new line of deep learning recommenders, pioneered by collaborative deep learning (CDL) [32] and its variants [19, 31], seamlessly incoporate content information into deep recommenders, thereby opening up the possibility of pretraining-based recommenders and significantly allevating cold-start problems in recommender systems.

Cross-domain Recommendation. Beyond in-domain recommendation, there are other works that utilize data from source domains to boost recommendation performance in target domains with either common users or common items [1, 10, 16, 20, 22, 36, 38, 46]. Another line of work, usually referred to as Dual-Target Cross-Domain Recommendation, tries to improve performance of both source and target domains [13, 17, 21, 42] (A more comprehensive survey of cross-domain recommender systems can be found in [45]). Note that these aforementioned cross-domain recommendation methods are fundamentally different from pre-trained recommendation. The former's training phase involves data from *both the source and target domains*; in contrast, the training phase of pre-trained recommendation uses data *only from the source domain*.

Bias in Recommendation While recommender systems are analyzed through in-domain and cross-domain perspectives, there is also an emphasis on understanding them through the lens of bias [3]. Within a single domain, the topic of item popularity debiasing has garnered significant attention. Studies such as [24, 40, 43] have demonstrated that addressing item popularity bias can enhance recommendation quality. In the broader context of cross-domain bias mitigation, some researchers have made noteworthy contributions. For instance, [18] has adapted traditional Inverse-Propensity-Score (IPS) methodologies to fit cross-domain contexts, while [8] employs invariant representations to reduce the impact of spurious correlations. However, the method to effectively leverage item popularity knowledge from pre-training domains to new domains remains an open question.

Causality in Recommendation Closely related to debiasing, studies have also incorporated causality into recommender systems. [34, 35] explore the extraction of invariant user-item embeddings across varying scenarios, aiming to minimize the influence of potentially misleading features. Both [41] and [44] integrate the influence of popularity into their causal analyses within a specific domain. Nonetheless, the application of causality on a broader scale, specifically within pre-trained recommender systems, remains largely unexplored.

Difference between Pre-trained and Traditional Recommender System Traditional recommendation approaches, including collaborative filtering-based methods [25, 32] and session-based methods [11, 14, 23], typically learn item embeddings indexed by domain-specific item IDs (also known as *item ID embeddings*) through interaction data. Such item ID embeddings are transductive and are not generalizable to unseen items, which causes the in-domain item cold-start issue and also becomes a blocker for transferring knowledge from one domain to another. Of course, we can infer the embedding of the unseen item based on its metadata, however different domains are likely to possess different sets of item metadata. For example, the item metadata in domain m is *{director, actor}* while in domain n is

Data	Pre-train			Zero-shot/Fine-tune				
	India	Spain	Canada	Australia	Mexico	Germany	Japan	Online Retail
Laguage	English	Spain	English	English	Spain	English	Japanese	English
#Items	45893	39675	99376	42094	43095	70527	22591	4223
#Users	507581	400883	992366	86975	249229	997555	277570	24446
#Inters	748607	708103	1971956	213086	483660	1840912	465746	540455
#Unseen	/	/	/	14525	32477	60637	12831	516009

Table 3. Dataset statistics for different domains of XMarket and Online Retail. Language indicates the default language used on Online Retail and the Amazon website for that domain (country). #Inters is the number of interactions, #Unseen is the number of target-domain samples that are non-overlapping with source domains.

{genre, rating}, the knowledge that the model learned based on item metadata in domain m can never be transferred to domain n and vice versa.

Usage of Large Language Model in Recommender System Several studies, such as [5, 9, 39], utilize Large Language Models (LLMs) for recommender systems. These works leverage LLMs for a variety of downstream tasks by developing customized prompts. Recently, works such as [7, 12] shed light on the path of pre-trained recommender systems, which capture generic knowledge grounded on the common feature space during pre-training. However, in contrast to our model PreRec, these approaches neglect to take into account the potential variance or bias present within the expansive multi-domain data.

C DATASETS, BASELINES, EVALUATION METRICS, AND IMPLEMENTATION DETAILS

C.1 Dataset Processing and Statistics

We consider two datasets: (1) **XMarket** dataset[1] [2], which is a large-scale real-world dataset covering 18 local markets (countries) on 16 different product categories; and (2) **Online Retail**[2], which contains data collected from a UK-based online retail platform. To ensure a sufficient number of non-overlapping users/items to evaluate zero-shot and fine-tuning performance rigorously while guaranteeing enough user-item interactions for training and evaluation, we set *"India", "Spain", "Canada"* in the XMarket dataset as the **pre-trained datasets**, set *"Australia", "Mexico", "Germany", "Japan"* in the XMarket dataset as the **cross-market datasets**, and choose the online retail dataset as the **cross-platform dataset**. For the experiment, we pre-train PreRec on the pre-trained datasets and evaluate on both cross-market datasets and cross-platform dataset.

Domain and Dataset Split. For each domain/dataset, we randomly split users into training/validation/test sets with the ratio of 4:3:3. The test splits of target domains are used for evaluation under both the zero-shot setting and the fine-tuning setting. Furthermore, to evaluate the zero-shot performance rigorously, we further filter out the overlapping users and items from the test splits; we name the remaining split **unseen** test set. See the Appendix for the dataset statistics of each target domain.

C.2 Baselines

We consider the following methods during evaluation:

- **Random**: Recommend items by random selection from the whole item catalogue without replacement.
- **POP**: Recommend items based on the items' popularity in the last stage (i.e., the last 15 days in our experiments).

[1] https://xmrec.github.io/
[2] https://www.kaggle.com/datasets/carrie1/ecommerce-data

- **SBERT**: Apply the pre-trained SBERT [28] on the item's description and generate the textual embedding as the item embedding, use the item embedding of the last interacted item in the user sequence as the user embedding, and recommend the next item to the user based on the inner product between the user embedding and the item embedding.
- **GRU4Rec** [11]: Use GRU to model the user interaction sequence for session-based recommendation. For fair comparison, we use the SBERT embedding generated from item's description as the input item representation.
- **SASRec** [14]: A self-attention based sequential model for session-based recommendation. For fair comparison, we use the SBERT embedding generated from item's description as the input item representation.
- **ZESRec** [7]: A flexible sequential framework that can be pre-trained on a source domain and directly applied to a target domain. It generates the universal item embedding via pre-trained language models (in our experiments we adopt SBERT) and generates universal user embeddings by aggregating universal item embeddings in the user sequence. In the experiments, we employ the GRU variant of ZESRec.
- **UniSRec** [12]: A follow-up work of ZESRec which further enables pre-training on multiple source domains. It adopts a self-attention based sequential model, equipping it with a MoE-enhanced Adaptor and additional contrastive training objectives, to assist with domain fusion and improve performance. As [12] claims, UniSRec is a state-of-the-art cross-domain model.
- **PreRec**$_n$: A simplified version of PreRec without the causal debiasing mechanism; in other words it does not take the cross-domain bias and the in-domain bias into the consideration during pre-training.

C.3 Evaluation Metrics

We use Recall@K% and our proposed r-NDCG@K% to evaluate model performance. For pre-training on multiple source domains, it is not ideal to use average Recall@K or NDCG@K to measure the performance and to do early-stop; this is because different domains have different numbers of items, and domains with fewer items tend to have larger NDCG@K or Recall@K, consequently dominating the evaluation. We therefore use Recall@K% and our proposed r-NDCG@K% ('r' stands for 'revised') to do early-stop and model evaluation. Denoting the number of items in a domain as N, with the next interacted item as the single target, r-NDCG@K% is defined as follows (note that there is no summation in the metrics because there is only a single target):

$$\text{r-DCG} = 1/\log(a + \tfrac{b*(rank-1)}{N}), \tag{7}$$

$$\text{r-iDCG} = 1/\log(a + \tfrac{b*(1-1)}{N}) = 1/\log(a), \tag{8}$$

$$\text{r-NDCG} = \tfrac{\text{r-DCG}}{\text{r-iDCG}} = \frac{\log(a)}{\log(a+\frac{b*(rank-1)}{N})}. \tag{9}$$

Here a and b are hyperparameters. Note that r-NDCG becomes the commonly used NDCG if $a = 2$ and $b = N$. In our experiments, we set $a = 2$ and $b = 15000$. Meanwhile, we use @K% instead of @K, with @K% indicating that only the top K% recommended items will contribute to the Recall@K% and r-NDCG@K%. Empirically, we set K%=0.04% for the cross-market scenario and set K%=0.5% for the cross-platform scenario. We use the average r-NDCG@K% on the validation set to early-stop training; we use both r-NDCG@K% and Recall@K% on the test set to evaluate a model.

D THE CHOICE OF K% IN R-NDCG@K% AND RECALL@K%

We choose $K\% = 0.04\%$ due to under the number of samples N from Table 3 we have $10 \leq N \times K\% \leq 50$. This provides the results between NDCG@10 and NDCG@50, which aligns with common evaluation setting including NDCG@10, NDCG@20, or NDCG@50. For the same reason, when evaluating Cross-Platform, we choose $K\% = 0.5\%$.

D.1 Implementation Details

Here we note down the details for model implementations and the training process.

Models. All methods except Random and POP use Sentence-BERT [28] to extract item BERT embeddings. For a fair comparison, all methods use 256-dimensional item embeddings and user embeddings; for sequential models based on GRU, the number of layers of GRU is set to 2; for self-attention based sequential models, the number of multi-head attention layers is set to 2.

Training. We use Adam [15] for both pre-training and fine-tuning with the learning rate of 0.0003. L2 regularization is implemented with weight decay in PyTorch's Adam. We use the average r-NDCG@K% on the validation set to early-stop pre-training, after which the pre-trained model is used for fine-tuning on each target domain. See the Appendix for more details. **Evaluation Setup.** During evaluation, we simulate an online environment where models *can only access the target domain data during inference*, and all the models are evaluated on the test set of the target domain. Besides measuring performance on the whole test set, we also separately measure performance on the *unseen* test set (see Sec. C.1). We utilize Recall@K% and r-NDCG@K% as evaluation metrics since they are less sensitive to differences in task difficulty across different domains. This is to ensure that performance for different domains is directly comparable, which is reflected by the consistent performance of the Random model across four target domains within the XMarket dataset (Recall@K% and r-NDCG@K% are constantly 0.004 and 0.002, respectively). The baselines can be broadly categorized into (1) **non-learnable zero-shot models**: Random, POP, SBERT, and (2) **learnable zero-shot models**: ZESRec, UnisRec, PreRec$_n$. Table 1 shows the zero-shot performance of different models.

E THE DESIGN RATIONALE OF COMPUTING POPULARITY PROPERTIES

To calculate the popularity score in a domain k, assuming a set of items J_k, and the number of interactions of an item j in time slot T is c_j^T. Naively, one could compute popularity factors $\mathbf{F}_j$ via the following two options:

- **Method A:** Directly using $\log c_j^T$ or c_j^T as the popularity factors $\mathbf{F}_j$ for item j at time interval T.
- **Method B:** Computing the percentage of the traffic volume, i.e., using $c_j^T / \sum_{i \in J_k} c_i^T$ or $(c_j^T / \sum_{i \in J_k} c_i^T)^p$ as the popularity factors $\mathbf{F}_j$ for item j at time interval t, where p is a pre-defined scalar.

However, these two methods are not transferable across multiple domains:

- For **Method A**, assuming domain m has more interactions than domain n and both domains have the same number of items. In this case, even though item j in domain m has more interactions than item i in domain n, it does not mean that item j in domain m is more popular than item i in domain n.
- For **Method B**, assuming domain m has more items than domain n. In this case, even if item j in domain m has the same percentage of traffic volume (e.g. 5%) as item i in domain n, item j may still be more popular in domain m than item i in domain n.

To address the abovementioned issue, we design a method inspired by the half Gaussian distribution. Assuming in any domain, the number of interactions of each item follows a half Gaussian distribution. The popularity factor $\mathbf{F}_j$ of

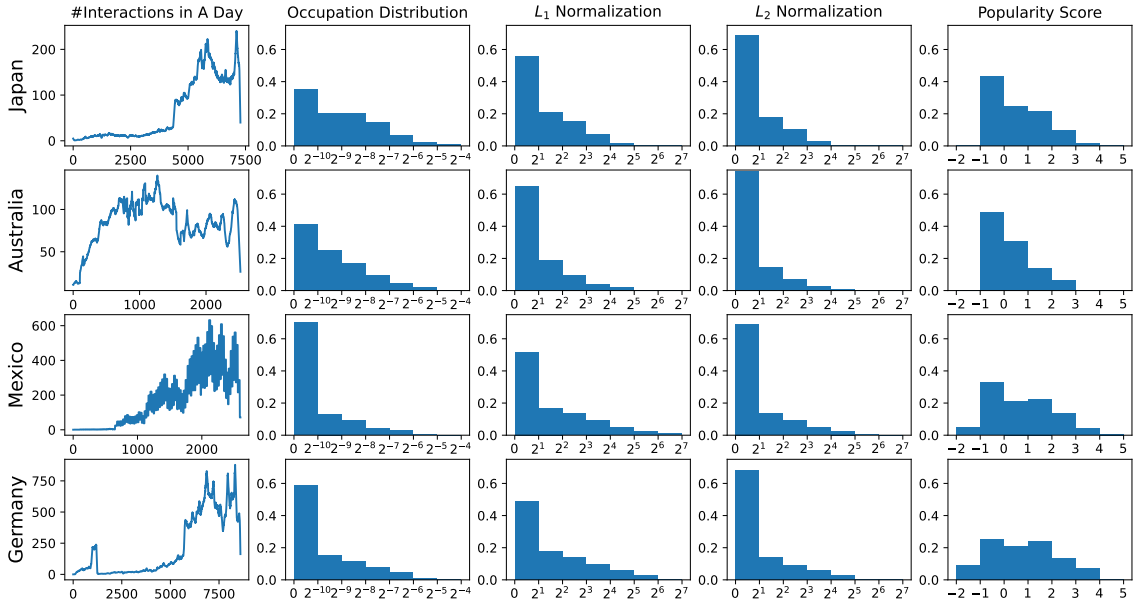

Fig. 4. The illustration of how popularity score works. The first column shows different domains have different traffic volumes and the second column shows occupation is not comparable among different domains since different domains have different numbers of available items. The third and fourth columns show the distribution of the first and the second dimension of the popularity factor $\mathbf{F}$ in each domain, which is a normalized frequency proposed in this paper, and shown to be more comparable among different domains. After pertaining, the popularity factor $\mathbf{F}$ is further mapped to the popularity score. As shown in the last column, the popularity score is comparable among zero-shot domains, indicating the generalizability of our proposed method for popularity.

item j can be measured as:

$$\mathbf{F}_j = c_j^T / \sqrt{\frac{\sum_{i \in J_k} (c_i^T)^2}{|J_k|}}, \tag{10}$$

where the popularity factor $\mathbf{F}_j$ is transferable from domain to domain.

We check the distribution over the number of interactions of items for each real-world dataset and discover that they may not precisely follow the half-Gaussian distribution. Therefore, we propose to consider different orders of norms for the denominator in Eqn. 10 and compute a set of popularity factors as $\mathbf{F}_j$:

$$\mathbf{F}_j = [\frac{c_j^T}{s_1^T}, \frac{c_j^T}{s_2^T}, ..., \frac{c_j^T}{s_w^T}],$$

where s_w^T is a normalization term calculated as:

$$s_w^T = (\sum_{j \in J_k} (c_j^T)^w / |J_k|)^{\frac{1}{w}}$$

with varied k as the popularity property, and use a trainable neural network to get the popularity score. The final w ranges from 1 to 4, i.e., the popularity property factors $\mathbf{F}_j$ is a 4-dimension vector. Fig. 4 illustrates how the popularity score works on four zero-shot domains.

hyperparameter	candidate value
learning rate	0.00003, **0.0003**, 0.003, 0.03
batch size	32, 64, 128, **256**, 512
# of negtive samples	64, 128, **256**, 512
# of gru layers	1, **2**, 3, 4, 6, 8, 12
# of attention layers	1, **2**, 3, 4, 6, 8, 12
# of attention heads	1, **2**, 3, 4, 6, 8, 12
# of embedding dimensions for user/item	64, 128, **256**, 512, 1024
domain bias regularization	0.03, **0.3**, 3, 30, 300
item bias regularization	**100**

Table 4. The hyperparameter tuning setting. The selected ones are marked in **bold face**.

F ADDITIONAL TRAINING DETAILS

We use one Tesla V100 GPU for training the model. The pre-training phase takes around 25 minutes per epoch, and the maximum number of training epochs is 30.

For contrastive learning, UniSRec uses in-batch negative. Thus, the negative items are from all domains. PreRec adopts a more complex implementation, sampling negative items randomly and uniformly from the same domain as the positive item.

For hyperparameter tuning, we consider candidates as shown in Table 4.