Learning to Augment for Casual User Recommendation

Jianling Wang*
Texas A&M University
College Station, TX, USA
jlwang@tamu.edu

Ya Le
Google, Inc.
Mountain View, CA, USA
elainele@google.com

Bo Chang
Google, Inc.
Mountain View, CA, USA
bochang@google.com

Yuyan Wang
Google, Inc.
Mountain View, CA, USA
yuyanw@google.com

Ed H. Chi
Google, Inc.
Mountain View, CA, USA
edchi@google.com

Minmin Chen
Google, Inc.
Mountain View, CA, USA
minminc@google.com

ABSTRACT

Users who come to recommendation platforms are heterogeneous in activity levels. There usually exists a group of core users who visit the platform regularly and consume a large body of content upon each visit, while others are casual users who tend to visit the platform occasionally and consume less each time. As a result, consumption activities from core users often dominate the training data used for learning. As core users can exhibit different activity patterns from casual users, recommender systems trained on historical user activity data usually achieve much worse performance on casual users than core users. To bridge the gap, we propose a model-agnostic framework *L2Aug* to improve recommendations for casual users through data augmentation, without sacrificing core user experience. *L2Aug* is powered by a data augmentor that learns to generate augmented interaction sequences, in order to fine-tune and optimize the performance of the recommendation system for casual users. On four real-world public datasets, *L2Aug* outperforms other treatment methods and achieves the best sequential recommendation performance for both casual and core users. We also test *L2Aug* in an online simulation environment with real-time feedback to further validate its efficacy, and showcase its flexibility in supporting different augmentation actions.

CCS CONCEPTS

• **Information systems → Recommender systems**;

KEYWORDS

Recommendation Systems, Data Augmentation, Policy Learning

ACM Reference Format:
Jianling Wang*, Ya Le, Bo Chang, Yuyan Wang, Ed H. Chi, and Minmin Chen. 2022. Learning to Augment for Casual User Recommendation. In *Proceedings of the ACM Web Conference 2022 (WWW '22), April 25–29, 2022, Virtual Event, Lyon, France.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3485447.3512147

*Work was done as an intern at Google.

WWW '22, April 25–29, 2022, Virtual Event, Lyon, France
© 2022 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-9096-5/22/04.
https://doi.org/10.1145/3485447.3512147

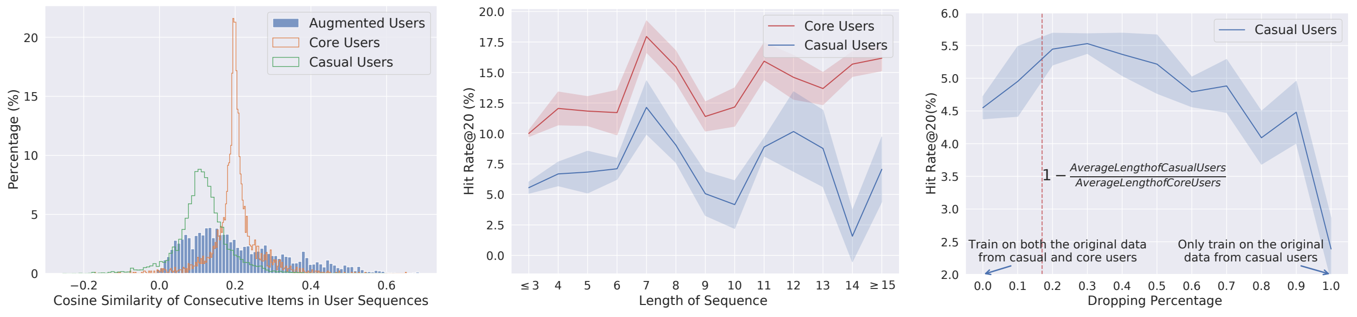
1 INTRODUCTION

Recommendation systems are ubiquitous. For example, streaming services rely on recommender systems to serve high-quality informational and entertaining content to their users, and e-commerce platforms recommend interesting items to assist customers in making shopping decisions. Sequential recommendation, which focuses on predicting the next item the user is interested in consuming based on their historical interaction sequences, has been extensively studied in many applications [19, 20, 23, 42, 56].

Users coming to the online platform are often heterogeneous in activity levels. There usually exists a set of ***core users*** who visit the platform regularly and consistently, while others are ***casual users*** who tend to visit the platform occasionally. The heterogeneity in activity levels can lead to distinct transitional patterns between these two groups of users [5, 7]. As shown in Figure 1(a), consecutively interacted items are less concentrated, and of lower similarity in casual users than core as they come to the platform less frequently.

Sequential recommenders trained predominantly on interaction data from core users often fail to capture the activity patterns of casual users and, as a result, provide less satisfactory recommendations for casual users. As shown in Figure 1(b), the self-attention based recommender (SASRec [23]) performs significantly worse on casual users than on core users in all sequence lengths. *How to improve the recommendation for casual users without sacrificing the performance on core users* is a critical challenge for building satisfactory recommendation services for all.

Existing methods such as cold-start recommendation [15, 32, 44, 60] or cross-domain approaches [14, 24, 25, 58, 61], mainly focus on addressing the data scarcity, but fail to handle the activity disparity between different types of users in the system. Although there are often more casual users than core users on the platforms, they left much fewer interactions in total compared to the core users. How to distill informative transition patterns from core users and efficiently adapt to casual users is the main research question we aim to tackle here. Inspired by the recent advances in data augmentation techniques [10, 27, 29, 50], we set out to generate augmented data sequences from core users to mimic the behavior patterns of casual users. While many augmentation techniques have been studied for continuous inputs such as images [8, 17], augmentation on user activity data consisting of sequential discrete item IDs is still under-explored. Meanwhile, compared with core users that tend to leave consistent and informative interaction sequences, casual users usually have noisier and more diverse behavior sequences



(a) (b) (c)

Figure 1: (a) Comparison of interest continuity (i.e., the similarity of items consumed consecutively) for different user groups. (b) Given a recommendation system (SASRec [23]) trained on interaction sequences from both core and casual users, for interaction sequences of the same length, it always performs worse on casual users than on core users, indicating the difficulty of making recommendations for casual users. (c) Data from core users is helpful for training a model for casual users. Furthermore, randomly dropping part of the interactions from core users can further improve the performance on casual users.

as shown in Figure 1(a). It is an open question to find an effective data augmentation method to generate augmented sequences that inherit informative transition patterns from core users and improve casual user recommendations.

To tackle the aforementioned challenges, we propose a model-agnostic "Learning to Augment" framework – *L2Aug*, which bridges the gap between casual and core users for sequential recommendation systems. Specifically, we develop a data augmentor which decides on a series of augmentation actions on the input sequence to generate augmented data sequences. From the "Learning to Augment" perspective, this data augmentor is trained to conduct effective data augmentation to maximize the performance of the *target model* (i.e., recommender). Framing this as learning a data augmentation policy, the data augmentor (agent) generates context/state and chooses the augmentation action. Meanwhile, the target model is updated with the augmented data sequences, and its performance improvement on a meta validation set is used as the reward to guide the learning of the augmentation policy. Through alternating between the data augmentation step using the recommender performance as the reward, and improving the recommender with the augmented data, the two modules reinforce each other and progressively improve both the data augmentation and recommendation quality. As a result, this builds an adaptive recommendation system, which can distill informative transition patterns from core users and adapt to casual users with dramatically different interaction patterns. Our contributions can be summarized as follows:

- We investigate the disparity between core and casual users in their transitional patterns within the interaction sequences, and study the feasibility of bridging the gap between sequential recommendation for core and casual users from the data augmentation perspective.
- We propose a model-agnostic framework *L2Aug* to learn a data augmentation policy using REINFORCE and improve the recommendation system using generated augmented data.
- We evaluate *L2Aug*, on top of various SOTA sequential recommendation models, on four real-world datasets, and show that it outperforms other treatment methods and achieves the best recommendation performance on both core and casual users.
- We also evaluate *L2Aug* in an online simulation environment, where the user responses on counterfactual recommendations are known. We also showcase its effectiveness as well as flexibility in supporting multiple augmentation actions.

2 MOTIVATION

In this section, we conduct an initial investigation with data sampled from the public Amazon review dataset [35], to explore the distinct behavior patterns between casual and core users, and then examine the feasibility of applying data augmentation in bridging the gap between them.

Firstly, to investigate the interest continuity in item consumption history of different users, we compute the *correlation between consecutive items* in their interaction sequences. We apply the bag-of-words model on item descriptions to obtain the item embeddings and then calculate the cosine similarity between the embeddings of consecutive items in the interaction sequences. In Figure 1(a), we can observe that the consecutive items consumed by core users are more similar. It confirms our hypothesis that core and casual users behave differently and the interests of casual users are less concentrated compared with core users.

Next, as interaction sequences of core users tend to be longer and denser than those of casual users, the most straightforward approach for data augmentation is to *randomly drop* part of the interactions from core users. We adopt this approach in this initial investigation and train a SASRec model [23] with the augmented data. In Figure 1(c), we visualize the performance for casual user recommendation by varying the percentage of dropped interactions. When the dropping percentage is equal to 0, none of the interactions from core users is dropped, thus the recommender is trained on the original data from both core and casual users. On the contrary, when the dropping percentage is equal to 1.0, all interactions from the core users are dropped, meaning that the recommender is trained only on the original data from casual users. It can be observed that the recommender system achieves improved performance on casual users when we start to drop interactions from the core users, which suggests that the synthetic data can help improve casual user recommendation. However, as the dropping percentage increases, discarding too much information negatively impacts casual user recommendation. These observations motivate us to search for more fine-grained and controlled augmentation policies.

3 PRELIMINARIES

In this section, we describe the problem setting of sequential recommendation and introduce the baseline recommendation model.

Problem Formulation. We use $\mathcal{U} = \{u_1, u_2, ..., u_{|\mathcal{U}|}\}$ and $\mathcal{I} = \{i_1, i_2, ..., i_{|\mathcal{I}|}\}$ to denote the set of $|\mathcal{U}|$ users and the set of $|\mathcal{I}|$ items on the platform. In this work, the users on the platform can be partitioned into two groups: casual users $\mathcal{U}_{casual}$ and core users $\mathcal{U}_{core}$ based on their activity levels (i.e., the frequency they visit the platform), s.t. $\mathcal{U}_{casual} \cap \mathcal{U}_{core} = \emptyset$ and $\mathcal{U}_{casual} \cup \mathcal{U}_{core} = \mathcal{U}$. Note that each item is mapped to a trainable embedding vector associated with its unique ID. In this work, we do not consider any auxiliary information for users and items. Given the sequence of items that user u has interacted with in chronological order $\mathbf{S}_u = [i_{u,1}, i_{u,2}, \ldots, i_{u,p}, \ldots, i_{u,n}]$, where $i_{u,p}$ represents the pth item u interacted with, the objective of a sequential recommendation model is to infer the next interesting item $i_{u,n+1}$ for user u. To simplify notations, we use $\mathbf{S}_{u,[1:p]}$ to denote the subsequence of items $[i_{u,1}, i_{u,2}, \ldots, i_{u,p}]$.

Target Model – Recommender. We learn a target model (recommender) f, which predicts the preference score for each candidate item in $\mathcal{I}$ given $\mathbf{S}_{u,[1:j-1]}$:

$$\hat{\mathbf{y}}_{u,j} = f(\mathbf{S}_{u,[1:j-1]}; \theta), \quad (1)$$

where θ represents the parameters of the function f and the prediction $\hat{\mathbf{y}}_{u,j} \in \mathcal{R}^{|\mathcal{I}|}$ denotes the predicted scores of user u at step j on all the candidates in $\mathcal{I}$. Then the items with the top preference scores based on $\hat{\mathbf{y}}_{u,j}$ are recommended to user u. Specifically, we adopt the cross-entropy loss as the objective function, that is:

$$\ell(\theta) = -\sum_{u \in \mathcal{U}} \sum_{j=1}^{|\mathbf{S}_u|} \left[\log(\sigma(\hat{\mathbf{y}}_{u,j}^{i_{u,j}})) + \sum_{k \notin \mathbf{S}_u} \log(1 - \sigma(\hat{\mathbf{y}}_{u,j}^{k})) \right], \quad (2)$$

where σ is the Sigmoid function and $\hat{\mathbf{y}}_{u,j}^{i_{u,j}}$ is the predicted preference score for the ground-truth item $i_{u,j}$ that the user u interacted with at step j. In our experiments, we sample a single $k \notin \mathbf{S}_u$ as the negative sample for each user at each step j.

4 LEARNING TO AUGMENT

In this section, we elaborate on the design of *L2Aug*, organized around two guiding research questions: 1) How to perform data augmentation on sequential data to support various operations (e.g., remove, keep or substitute) on items within the sequence? 2) How to learn an effective data augmentation policy to achieve the goal of improving causal user recommendation?

Framework Overview. We propose *L2Aug*, illustrated in Figure 2, to learn the data augmentation policy for improving casual user recommendation. There are two main components in the design: a *recommender* to make sequential recommendations and a *data augmentor* to generate synthetic interaction sequences by applying the learned data augmentation policy on the input sequences. These two components are alternately trained. Specifically, for each batch of sequences sampled from $\mathbf{S}_{core}$, the *data augmentor* learns to take a series of augmentation actions (e.g., remove, keep or substitute) so the generated synthetic sequences can improve the performance of the *recommender.* Framing this as learning a data augmentation policy, the data augmentor (agent) generates context/state and chooses the augmentation action. Meanwhile, the target model is updated with the augmented data sequences and its performance improvement on a meta validation set is used as the reward to guide the training of the augmentor. Through alternating between the data augmentation step using the recommender performance as the reward, and improving the recommender with the augmented data, the two modules reinforce each other and progressively improve both the data augmentation and recommendation quality.

4.1 Learning Augmentation Policy

Our goal is to *learn a discrete sequential data augmentation policy* for *maximizing the performance of recommendation systems* on casual users. Inspired by preliminary studies in Section 2, we set out to generate augmented (synthetic) data sequences to mimic the behavior patterns of casual users by editing the core user sequences. For each batch of interaction sequences sampled from $\mathcal{U}_{core}$, we consider as a *data augmentation task* that generates the synthetic sequences by taking a series of editing actions from {keep, drop, replace, ...} for items in the input sequences sequentially. Since the data augmentation process is non-diffentiable, we adopt the policy learning framework described below to enable the training:

- **Context/State**: Let $\mathbf{S}_u$ denote an interaction sequence from $\mathcal{U}_{core}$. When encountering item $i_{u,k}$ in sequence $\mathbf{S}_u$, a vector $\mathbf{h}_k$ that encodes the subsequence $\mathbf{S}_{[1:k]}$ is regarded as the context or state representation. The detailed model to obtain the state vector will be discussed in Section 4.2.
- **Action**: For simplicity, we use two actions – "Keep" and "Drop". The model could be extended to support multiple actions (i.e., more than two); more details can be found in Section 5.3.1. At step k, for item $i_{u,k}$ in sequence $\mathbf{S}_u$, we need to decide on the action $a_{u,k} \in \{0, 1\}$ to keep or drop item $i_{u,k}$ when generating the augmented sequence from sequence $\mathbf{S}_u$. Note that $a_{u,k} = 0$ indicates dropping the item and $a_{u,k} = 1$ means to keep it.
- **Meta Valication Set ($\mathcal{U}_{meta}$)**: To guide the training of the data augmentor, from the set of N casual users $\mathcal{U}_{casual}$, we randomly sample a small subset of M users ($M \ll N$) and construct the meta validation set $\mathcal{U}_{meta}$ with their interaction sequences. During the training process, the performance of the target model on $\mathcal{U}_{meta}$ is computed as the reward for learning the augmentation policy.
- **Reward**: The reward function aims to guide the agent to learn the augmentation policy in order to maximize the performance of the target model. Each batch of the augmented sequences is used to train the target model and leads to a performance change of the target model, which can be regarded as the reward to the data augmentor. In offline settings, it can be the *update of recommendation performance* on $\mathcal{U}_{meta}$ measured by offline metrics (e.g., NDCG, Hit rate and Mean Reciprocal Ranking).

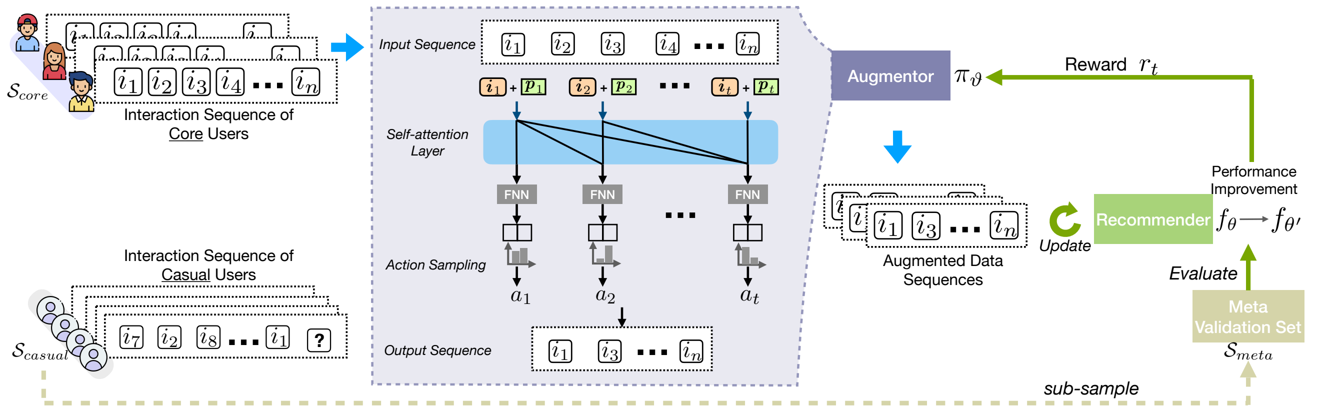


Figure 2: The proposed model-agnostic "Learning to Augment" framework – *L2Aug*. The *data augmentor* takes a series of augmentation *actions* on the input sequence to generate synthetic data sequences. Meanwhile the *recommender* is updated by the synthetic data sequences; its performance improvement on a meta validation set is used as the *reward* to guide the training of the augmentor. These two components reinforce each other and progressively improve both the data augmentation and recommendation quality.

In online settings where we can simulate user response on counterfactual recommendations, the reward can be the user response returned by the environment (e.g., engagement, rating or conversion). More details about the metrics can be found in Section 5.1. In our experiments, we use the performance gain on both NDCG and Hit Rate as the reward for offline experiment, and use the change of simulated rating as the reward for online experiment. Following [11, 54, 57], the policy network for data augmentation is updated on a delayed reward received after feeding the generated augmented data to the recommender.

4.2 Data Augmentor

Since most of the existing data augmentation methods are designed for continuous feature spaces [8, 17], they are not fit for handling sequence data consisting of discrete item IDs in our case. We propose the *Data Augmentor*, which generates a synthetic sequence by encoding the input sequence as the **context/state** representations, and deciding on the editing **actions** on the input sequence.

Given a sequence of interacted items $\mathbf{S}_u = [i_{u,1}, \dots, i_{u,k}, \dots, i_{u,t}]$ for a core user $u \in \mathcal{U}_{core}$, the Data Augmentor needs to decide on the editing action on each item. To make the decision on each item $i_{u,k}$, the agent needs to encode the subsequence $\mathrm{S}_{[1:k]}$, which in turn requires a representation of each item. We encode two pieces of information in the individual item representation: the item content itself and its position. In other words, for each item $i_{u,k}$ in $\mathbf{S}_u$, we have $\mathbf{e}_k = \mathbf{i}_k + \mathbf{p}_k$. Here, $\mathbf{i}_k$ is the embedding for item $i_{u,k}$ and $\mathbf{p}_k$ is the positional embedding of position k, which is used to retain the order information.

With the individual item representation, any sequential embedding model including RNN, Bidirectional-RNN or Transformers [21, 43] can be used to encode the subsequence $\mathrm{S}_{[1:k]}$ to produce a context/state representation. In this work, we adopt the self-attention model [43] to produce the state $\mathbf{h}_k$. Self-attention [43] is designed to match a sequence against itself and thus uses the same objects as the queries, keys, and values. We will transform the position-aware embeddings with $\mathbf{W}_Q$, $\mathbf{W}_K$, and $\mathbf{W}_V$ to generate the query vectors, key vectors, and value vectors, correspondingly. Then we can generate the state representation for each item $i_{u,k}$ by encoding its correlation with other items in the sequence via:

$$\mathbf{h}_k = \sum_{j \le k} \sigma_{kj} \mathbf{W}_V \mathbf{e}_j \quad \text{and} \quad \sigma_{kj} = \frac{S(\mathbf{W}_Q \mathbf{e}_k, \mathbf{W}_K \mathbf{e}_j)}{\sum_{j \le k} S(\mathbf{W}_Q \mathbf{e}_k, \mathbf{W}_K \mathbf{e}_j)}, \tag{3}$$

in which the function $S(\cdot, \cdot)$ is used to denote the similarity score between two elements. In particular, we use the *Scaled Dot-Product Attention* [23, 43] to calculate the scores: $S(\mathbf{a}, \mathbf{b}) = \mathbf{a}^T \mathbf{b} / \sqrt{D}$, where D is the dimension size and used for normalization when calculating the similarity scores. Once obtaining each individual ***context/state*** representation $\mathbf{h}_k$, the agent produces a policy π_k over the action space $\mathcal{A}$ through

$$\pi_{\boldsymbol{\vartheta}}(\cdot|\mathbf{h}_k) = \text{Softmax}(\mathbf{W}_A \cdot \mathbf{h}_k). \tag{4}$$

Here $\mathbf{W}_A \in \mathcal{R}^{|\mathcal{A}| \times |\mathbf{h}|}$ is a trainable weight matrix. We use $\boldsymbol{\vartheta}$ to denote all the parameters involved in building the augmentation policy network. Each dimension in $\pi_{\boldsymbol{\vartheta},k}$ represents the probability of a specific action at step k. The agent then decides on the editing ***action*** for the item by sampling with the probabilities.

4.3 Augmentation Policy Optimization

To optimize for the data augmentation policy, we aim to maximize expected rewards on improved recommendation quality, which can be defined as:

$$J(\boldsymbol{\vartheta}) = \max_{\boldsymbol{\vartheta}} \mathbb{E}_{\pi_{\boldsymbol{\vartheta}}} [r_t], \tag{5}$$

where r_t is the reward computed as the improvement in recommendation performance when feeding the augmented data generated from input batch t. In Algorithm 1 line 12, we use the Hit Rate HT to evaluate the recommendation performance and the difference of the two as the reward. Note that Hit Rate can be replaced with any performance metrics or their combination. We update the parameters $\boldsymbol{\vartheta}$ via policy gradient:

$$\boldsymbol{\vartheta} \leftarrow \boldsymbol{\vartheta} + \alpha \nabla_{\boldsymbol{\vartheta}} \tilde{J}(\boldsymbol{\vartheta}), \tag{6}$$

```
Algorithm 1: L2Aug Training Process
Input: Training sequences $\mathcal{S}_{core}$, $\mathcal{S}_{meta}$, $\mathcal{S}_{casual}$ and the
       pre-trained recommendation system $f_{\theta}$, update frequency $F$
Output: A fine-tuned recommendation system $f_{\theta}$
1  Random initialize the data augmentor $\pi_{\vartheta}$ and $i = 0$.
2  while not converge do
3      Sample $B$ samples $\mathcal{S}_B$ from $\mathcal{S}_{core}$
4      // Augmentation by Data Augmentor
5      for $u = 1 \rightarrow B$ do
6          for $k = 1 \rightarrow |s_u|$ do
7              Compute the state representation for the $k_{th}$ item
8              Sample the augmentation action
9          Obtain the augmented sequence
10     // Augmentation Policy Optimization
11     Fine-tune $f_{\theta}$ with the batch of augmented sequences to get $f_{\theta'}$
12     Calculate the reward $r_t = \mathrm{HT}(f_{\theta'}, \mathcal{S}_{meta}) - \mathrm{HT}(f_{\theta}, \mathcal{S}_{meta})$
13     Update $\pi_{\vartheta}$ according to Eq. (6) and (7)
14     // Replay and Update the Recommender
15     if $i$ mod $F == 0$ then
16         Sample $B$ samples $\mathcal{S}_{train}$ from $\mathcal{S}_{core}$
17         Generate synthetic samples $\mathcal{D}_{syn}$ from $\mathcal{S}_{train}$ with $\pi_{\vartheta}$
18         Sample $B$ samples $\mathcal{S}'_{train}$ from $\mathcal{S}_{casual}$ and $\mathcal{S}_{core}$
19         Update the recommender $f_{\theta}$ with $\mathcal{S}_{syn}$ and $\mathcal{S}'_{train}$
20     $i \leftarrow i + 1$
21 return The fine-tuned recommendation system $f_{\theta}$
```

in which α is the learning rate. With the obtained rewards, the gradient can be computed by:

$$\nabla_{\vartheta}\tilde{J}(\vartheta) = r_t \sum_{u \in \mathcal{S}_B} \sum_{k=1}^{|\mathbf{S}_u|} \nabla_{\vartheta} \log \pi_{\vartheta}(a_{u,k}|\mathbf{h}_k), \tag{7}$$

Details of the training process are shown in Algorithm 1. Note that the same meta reward over the batch is assigned to all the augmentation decisions taken within the batch. In essence, the obtained reward based on recommendation improvement computed on the meta validation set is used to guide the learning of the data augmentation policy. The augmented sequences in return are used to further improve the performance of the recommender. During the training process, the data augmentor and recommender system can reinforce each other, and progressively improve the data augmentation and recommendation quality.

Replay. To ensure that the model still achieves satisfying performance on core users, we adopt the replay strategy [39] to avoid forgetting. Besides the synthetic sequences, the recommendation system is also updated with the original data sequences from core users, leading to a recommender that has improved performance on casual users without sacrificing the performance on core users.

5 EXPERIMENTS

In this section, we report our experiments over multiple datasets to evaluate the performance of *L2Aug* and answer the following questions: (i) Whether *L2Aug* improves recommendations for casual users without sacrificing the performance on core users? (ii) Is the proposed framework flexible to support more augmentation actions and various recommendation setups?

Table 1: Summary statistics for the datasets.

	# Items	# Casual Users	# Core Users	Avg. # Interactions
Amazon_CDs	22,685	2,176	1,022	23.75
Amazon_Books	17,443	11,083	3,457	31.71
Amazon_Movies	11,079	10,020	2,808	14.06
Goodreads	65,864	11,836	5,017	130.03

5.1 Experimental Setup

Datasets. To examine the performance of the proposed method, we conduct experiments on four real-world datasets. Table 1 shows the summary statistics for each dataset. ***Amazon_CDs*** is adopted from the public Amazon review dataset [35], which includes reviews spanning May 1996 – July 2014 on products belonging to the "CDs and Vinyl" category on Amazon. Similarly, ***Amazon_Books*** and ***Amazon_Movies*** are from two of the largest categories – "Books" and "Movies" of the same Amazon review dataset. To further investigate the performance in other application scenarios, we include another public dataset from an idea-sharing community – ***Goodreads***, on which users can leave their reviews and ratings on books [45]. For all the datasets, all the items the user has *interacted* with (reviewed) form the user interaction sequence $\mathbf{S}_u$. We use the average time gap between their consecutive interactions to differentiate between casual and core users: core users are those with average time gaps of less than 30 days; others are labeled as casual users. More details on data prepossessing and splitting can be found in Appendices A.1.

Baselines. As the proposed method is model agnostic, we apply it to various sequential recommendation models and compare its performance with other model-agnostic treatment methods to examine its effectiveness. We select three major *sequential recommendation models* – **GRU4Rec** [20], **SASRec** [23] and **NextItNet** [56], which are built on top of Gated Recurrent Units (GRU), self-attention layers and stacked 1D dilated convolutional layers to capture the sequential patterns in user interaction sequences respectively. They are widely used in many applications and serve as the foundation for many advanced recommender systems.

Since there are no previous works focusing on improving casual user recommendations, for each sequential recommendation model, we compare the proposed *L2Aug* with the following *treatment methods* which are proved to alleviate the performance gap between different user groups.

- **Random**: It randomly drops the interactions of core users to obtain the synthetic data, which are combined with the original data (both core & casual users) for training the recommender.
- **Focused Learning** [2]: It treats the casual users $\mathcal{U}_{casual}$ as the focused set and performs a grid search for the best performing hyperparameters (i.e., the regularization) for improving recommendation accuracy on the focused set.
- **Adversarial Reweighting** [30]: It plays a minimax game between a recommender and an adversary. The adversary would

Table 2: Performance on casual user recommendation of various models on different datasets.

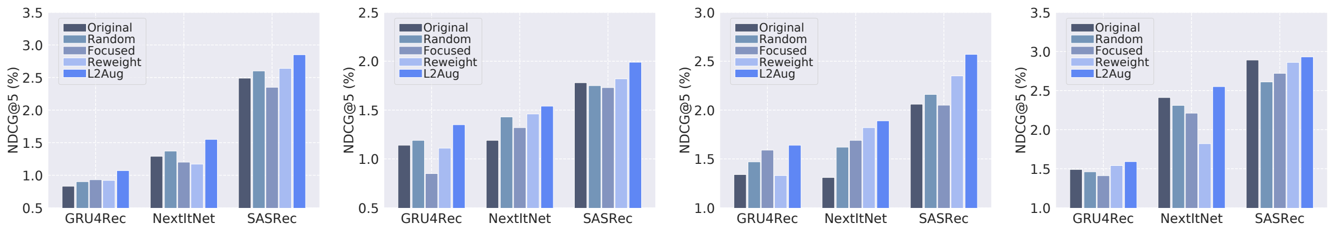
Method	Amazon_CDs		Amazon_Books		Amazon_Movies		Goodreads	
	NDCG@5 (%)	HT@5 (%)	NDCG@5 (%)	HT@5 (%)	NDCG@5 (%)	HT@5 (%)	NDCG@5 (%)	HT@5 (%)
GRU	0.64 ± 0.04	1.05 ± 0.09	0.66 ± 0.04	1.10 ± 0.04	1.11 ± 0.07	1.81 ± 0.08	1.32 ± 0.06	2.09 ± 0.08
w/ Random	0.66 ± 0.06	1.13 ± 0.11	0.71 ± 0.03	1.15 ± 0.05	1.22 ± 0.01	1.95 ± 0.12	1.44 ± 0.13	2.29 ± 0.14
w/ Focused	0.69 ± 0.03	1.17 ± 0.08	0.74 ± 0.02	1.19 ± 0.04	1.28 ± 0.06	2.08 ± 0.07	1.43 ± 0.05	2.37 ± 0.11
w/ Adversarial	0.73 ± 0.02	1.24 ± 0.06	0.73 ± 0.03	1.20 ± 0.05	1.34 ± 0.05	2.23 ± 0.06	1.46 ± 0.04	2.21 ± 0.08
w/ L2Aug	**0.80 ± 0.03**	**1.37 ± 0.07**	**0.75 ± 0.02**	**1.25 ± 0.03**	**1.43 ± 0.04**	**2.35 ± 0.05**	**1.53 ± 0.03**	**2.45 ± 0.05**
NextItNet	1.12 ± 0.16	1.69 ± 0.18	0.97 ± 0.03	1.56 ± 0.06	1.30 ± 0.05	2.13 ± 0.09	2.05 ± 0.11	2.97 ± 0.13
w/ Random	1.17 ± 0.18	1.88 ± 0.20	1.08 ± 0.05	1.71 ± 0.07	1.45 ± 0.08	2.26 ± 0.09	2.13 ± 0.10	3.11 ± 0.16
w/ Focused	1.22 ± 0.15	2.07 ± 0.16	1.13 ± 0.04	1.68 ± 0.03	1.47 ± 0.06	2.42 ± 0.08	2.25 ± 0.09	3.32 ± 0.13
w/ Adversarial	1.52 ± 0.11	2.39 ± 0.13	1.15 ± 0.04	1.75 ± 0.05	1.58 ± 0.06	2.56 ± 0.08	2.42 ± 0.06	3.43 ± 0.11
w/ L2Aug	**1.62 ± 0.09**	**2.44 ± 0.10**	**1.24 ± 0.05**	**1.87 ± 0.04**	**1.71 ± 0.05**	**2.74 ± 0.07**	**2.51 ± 0.07**	**3.62 ± 0.10**
SASRec	1.83 ± 0.10	2.77 ± 0.16	1.13 ± 0.05	1.78 ± 0.06	1.72 ± 0.05	2.71 ± 0.08	2.29 ± 0.07	3.49 ± 0.10
w/ Random	1.85 ± 0.15	2.81 ± 0.18	1.21 ± 0.05	1.86 ± 0.07	1.76 ± 0.11	2.73 ± 0.12	2.36 ± 0.11	3.54 ± 0.19
w/ Focused	1.88 ± 0.14	2.90 ± 0.13	1.24 ± 0.03	1.94 ± 0.05	1.81 ± 0.09	2.83 ± 0.11	2.42 ± 0.10	3.60 ± 0.14
w/ Adversarial	1.92 ± 0.13	3.03 ± 0.14	1.23 ± 0.02	1.93 ± 0.03	1.88 ± 0.06	2.86 ± 0.10	2.45 ± 0.08	3.72 ± 0.11
w/ L2Aug	**2.11 ± 0.11**	**3.26 ± 0.13**	**1.31 ± 0.04**	**1.99 ± 0.04**	**1.95 ± 0.05**	**3.00 ± 0.08**	**2.71 ± 0.09**	**3.93 ± 0.10**

Method	Amazon_CDs		Amazon_Books		Amazon_Movies		Goodreads	
	NDCG@10 (%)	HT@10 (%)	NDCG@10 (%)	HT@10 (%)	NDCG@10 (%)	HT@10 (%)	NDCG@10 (%)	HT@10 (%)
GRU	0.84 ± 0.07	1.74 ± 0.13	0.92 ± 0.03	1.87 ± 0.05	1.56 ± 0.06	3.22 ± 0.11	1.81 ± 0.07	3.61 ± 0.12
w/ Random	0.91 ± 0.12	1.92 ± 0.16	0.99 ± 0.04	2.02 ± 0.03	1.67 ± 0.08	3.34 ± 0.15	1.94 ± 0.10	3.84 ± 0.20
w/ Focused	0.93 ± 0.06	1.91 ± 0.14	1.03 ± 0.03	2.06 ± 0.05	1.77 ± 0.06	3.59 ± 0.13	1.91 ± 0.09	3.89 ± 0.15
w/ Adversarial	0.96 ± 0.04	1.97 ± 0.13	0.99 ± 0.04	2.02 ± 0.04	1.75 ± 0.08	3.51 ± 0.14	1.93 ± 0.08	3.68 ± 0.13
w/ L2Aug	**1.03 ± 0.03**	**2.11 ± 0.10**	**1.05 ± 0.04**	**2.16 ± 0.03**	**1.86 ± 0.05**	**3.65 ± 0.10**	**1.99 ± 0.07**	**3.96 ± 0.11**
NextItNet	1.59 ± 0.17	2.87 ± 0.20	1.29 ± 0.05	2.55 ± 0.08	1.79 ± 0.07	3.66 ± 0.16	2.52 ± 0.12	4.44 ± 0.18
w/ Random	1.57 ± 0.18	3.10 ± 0.24	1.34 ± 0.07	2.53 ± 0.09	2.05 ± 0.13	4.10 ± 0.18	2.66 ± 0.16	4.74 ± 0.19
w/ Focused	1.45 ± 0.12	2.80 ± 0.19	1.47 ± 0.05	2.74 ± 0.06	1.92 ± 0.11	3.80 ± 0.14	2.69 ± 0.15	4.73 ± 0.17
w/ Adversarial	1.85 ± 0.11	3.44 ± 0.15	1.48 ± 0.04	2.77 ± 0.05	2.11 ± 0.09	4.20 ± 0.11	2.89 ± 0.07	4.90 ± 0.13
w/ L2Aug	**2.11 ± 0.12**	**3.95 ± 0.14**	**1.55 ± 0.03**	**2.86 ± 0.05**	**2.22 ± 0.08**	**4.32 ± 0.09**	**2.98 ± 0.08**	**5.10 ± 0.15**
SASRec	2.38 ± 0.14	4.49 ± 0.21	1.60 ± 0.07	3.24 ± 0.12	2.26 ± 0.11	4.41 ± 0.19	3.02 ± 0.15	5.77 ± 0.19
w/ Random	2.25 ± 0.15	4.03 ± 0.26	1.61 ± 0.06	3.13 ± 0.14	2.34 ± 0.13	4.48 ± 0.18	3.05 ± 0.15	5.68 ± 0.21
w/ Focused	2.47 ± 0.11	4.73 ± 0.18	1.66 ± 0.04	3.32 ± 0.09	2.45 ± 0.10	4.83 ± 0.14	3.02 ± 0.12	5.47 ± 0.17
w/ Adversarial	2.35 ± 0.16	4.37 ± 0.21	1.62 ± 0.03	3.12 ± 0.10	2.43 ± 0.11	4.54 ± 0.16	3.15 ± 0.10	5.93 ± 0.21
w/ L2Aug	**2.61 ± 0.12**	**4.87 ± 0.19**	**1.69 ± 0.03**	**3.38 ± 0.08**	**2.44 ± 0.08**	**4.53 ± 0.11**	**3.32 ± 0.11**	**5.86 ± 0.15**

adversarially assign higher weights to regions where the recommender makes significant errors, in order to improve the recommender's performance in these regions.

Evaluation Metrics. We train a recommendation model on the training data and then test it on the test data, where each of the users has one ground-truth item. We adopt the widely used Top-K metrics, including Hit Rate (HT@K) and Normalized Discounted Cumulative Gain (NDCG@K) to evaluate the recommendation performance. For user u, based on the scores $\hat{\mathbf{y}}_u$ from Eq. (1), we obtain the ranking r_u of the ground-truth item. Hit rate indicates whether the ground-truth item appears in the Top-K list, i.e., $\text{HT}_u@K = 1$ if $r_u \leq K$ and $\text{HT}_u@K = 0$ otherwise. Note that hit rate is equal to recall in this case since there is only one item in the test set for each user. When computing NDCG, the Ideal Discounted Cumulative Gain (IDCG) is equal to 1 for all users. Therefore, we have $\text{NDCG}_u@K = \frac{1}{\log_2(1+r_u)}$ if $r_u \leq K$ and $\text{NDCG}_u@K = 0$ otherwise. In the following, we report the results averaged over all users in the test set with $K = 5$ and 10.

Implementation Details. All of the recommendation models are implemented in TensorFlow and optimized with Adam [26]. We utilized the code published by or suggested by authors of the original work and keep the settings unchanged. We set the maximum sequence length to 200 and the batch size to 512 for all the datasets. For a fair comparison, the item embedding dimension is 100 and the negative sampling rate is set to 1 in the training process for all the models. We also implement our data augmentor in TensorFlow and adopt Adam as the optimizer. In Table 2, we report the results based on the setting chosen on the validation set. In *L2Aug*, 10% of casual users among the training set are used to construct the meta validation set to compute reward for learning the data augmentation policy. The update frequency F is 5 for all the datasets. More



(a) **Amazon_CDs** (b) **Amazon_Books** (c) **Amazon_Movies** (d) **Goodreads**

Figure 3: Performance on <u>core</u> user recommendation of various models on different datasets.

details for parameter selection and model implementation can be found in Appendices A.2 and A.3.

5.2 Overall Performance

We conduct the experiments on the four datasets and run each experiment 10 times. The mean and variance of each metric are reported. We present quantitative results on recommendation performance metrics for both casual and core users, along with qualitative analyses, to showcase the efficacy of the proposed method.

5.2.1 Casual User Recommendation. We summarize the average performance of different baseline methods on all of the datasets in Table 2. When combined with various sequential models, the proposed L2Aug outperforms all of the other treatment methods and achieves the best recommendation for casual users. In the following, we present more in-depth observations and analyses:

- The simplest treatment of *randomly* dropping part of the interactions from core users helps to improve casual user recommendations compared to the model trained on original data. This observation verifies the hypothesis that data augmentation can help bridge the gap between core and casual users.
- By learning a recommendation model with a special focus on casual users, the *focused learning* treatment can help improve model performance on casual users. Meanwhile, since recommenders tend to make inaccurate predictions on casual users, *adversarial reweighting* can guide the recommender to improve its performance on casual users, leading to more accurate recommendations for them.
- In general, the proposed L2Aug significantly outperforms all the baseline treatments on improving casual user recommendations for various widely-used sequential recommendation models. Take the *Amazon_CDs* dataset as an example, we find that L2Aug achieves 9.59%, 6.58%, and 9.90% improvements for NDCG@5 with GRU, NextItNet and SASRec, respectively, compared against the best performing baseline treatment. We can conclude that L2Aug is effective in solving the challenging problem of improving casual user recommendations.

5.2.2 Core User Recommendation. Besides the performance on casual users, we also report the recommendation performance on core users in Figure 3, measured by NDCG@5; results by other metrics can be found in Appendix A.4. Although *focused learning* improves the recommendation on casual users, it loses its prediction power on core users. The *adversarial reweighting* treatment, aiming at improve challenging data samples rather than specific user groups, improves core user recommendations in some cases, but not always. In contrast, the proposed L2Aug improves core user recommendations with various sequential recommendation models and outperforms all the other baseline treatments. These results showcase its effectiveness in bridging the gap between recommendation for casual users and core users, leading to overall recommendation improvement.

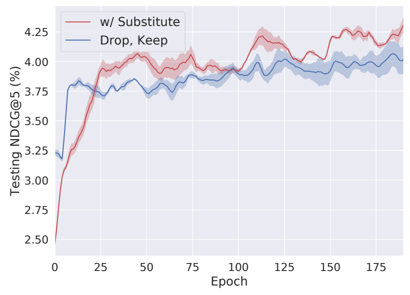


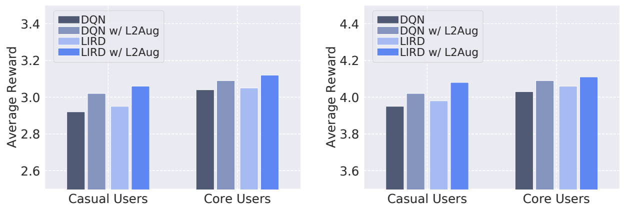
Figure 4: L2Aug is easily extended to support more actions (i.e., substitute) with improved casual user recommendation.

5.2.3 Qualitative Analyses of the Augmented Data. Figure 1(a) plots the synthetic sequential data (i.e., augmented users) produced by *L2Aug* along with the casual and core users from the Amazon review dataset. One can see the distribution of the interest continuity for the augmented users lies between that of the casual and core users from the original dataset, suggesting *L2Aug* successfully distilled more consistent patterns from core users while adapting to the patterns of casual users.

5.3 Flexibility Analysis

5.3.1 Expanding the Action Space. Recall that, so far, the augmentation policy can take two actions: "keep" and "drop". In this section, we examine the feasibility of expanding the action space of L2Aug, which makes the augmentor more versatile. As an initial study, we consider the "substitute" action, which replaces an item with its most correlated item. We adopt the inverse user frequency (i.e., $|N(i) \cap N(j)|/\sqrt{|N(i)||N(j)|}$) [4, 40] to define the correlation between two items (i.e., i and j), in which $N(i)$ is the set of users interacted with item i. Figure 4 shows that adding the "substitute" action leads to higher recommendation performance on casual users; it also takes more epochs for the model to converge. Similarly, the proposed L2Aug can also be extended to support other actions like "reorder" and "insert". Based on the observation, it can be conjectured that the proposed framework is capable of handling various editing actions for sequential data augmentation.

5.3.2 Online Experiments. The experiments so far are in offline settings, taking the observed user response on recommendations provided by the system as the ground-truth. Offline experiments have the drawback that we are not able to observe user response on counterfactual recommendations, i.e., items that were not shown to the users. To further evaluate the capability of the proposed model for real-world applications, we also conduct online experiments with simulation. We follow [59] to set up the online simulation environment. Given the user's historical interactions and any recommendation candidate, it simulates the user response based on memory matching. This allows us to evaluate models on real-time responses (i.e., ratings) obtained from the simulator instead of relying on offline metrics. In the online experiment, we adopt the public MovieLen 100K dataset and split it into 7 : 3 for training and testing, respectively. We treat users of the top 30% visiting frequency as core users and the rest as casual users. In Figure 5, DQN [36] is the deep Q-learning method and LIRD [59] is the state-of-the-art for list-wise recommendation. We use them as the recommenders (i.e., target model) in the L2Aug framework. Combined with L2Aug, both achieve improved performance on casual and core user recommendation under different list sizes, which further corroborates the efficacy of L2Aug under different recommendation scenarios.



(a) List Size = 1 (b) List Size = 4

Figure 5: *Online* test for List-wise Recommendation.

6 RELATED WORK

Sequential Recommendation. Instead of treating users as static, sequential recommendation aims to capture the sequential patterns from historical user interactions and infer the interesting items based on users' dynamic preferences. Early works propose to leverage Markov chains (MC) to model the transition among items and predict the subsequent interaction [18, 38]. Grounded on the recent advances of deep learning techniques, there are lots of efforts on Recurrent Neural Networks (RNNs) to investigate users' sequential interactions [19, 20, 47, 51]. Meanwhile, Convolutional Neural Networks (CNN)-based recommenders also show superior performance. Caser [42] is built on top of the 2D convolutional sequence embedding model and NextitNet [56] investigates 1D CNNs with the dilated convolution filters for better performance. With its success in handling the textual data, self-attention layer (transformer) [43] is adopted in SASRec [23] and Bert4Rec [41] to generate dynamic user embedding based on their interaction sequences. More recently, Graph Neural Networks have been exploited in [34, 49, 52] to encode the contextual information for more accurate user modeling in sequential recommendation.

Long-tailed and Cold-User Recommendation. Several research topics in recommender systems are related to improving casual user recommendations. Beutel et al. [2] define the focused learning problem and propose to find different optimal solutions for different user groups through hyperparameter optimization and a customized matrix factorization objective. There are other efforts trying to learn multiple recommendation models and then select the best performing one from the pool for each user [13, 33]. Another line of research focuses on domain transfer and improving the recommendation for users who have very few observed interactions [46, 48]. Most of these works rely on side information or contextual data [3, 22, 31, 32]. Yin et al. [55] propose to learn the transferable parameters between data-rich head users and data-ill tail users, leading to an improved recommendation for both groups. Instead, we aim to bridge the gap between users who visit more frequently and users who casually visit the service via data augmentation.

Data Augmentation. Data augmentation is a widely adopted strategy for training deep neural models in computer vision [10, 29], graph learning [12] and natural language processing [28, 50]. The idea is to generate more diverse data samples to improve model robustness. While most of the early work manually designs dataset-specific augmentation strategies or implementations, recent attention has been paid to automating the process of generating effective augmentation for the target dataset. Generative Adversarial Networks (GANs) have been commonly used to generate additional data by learning the distribution of the training data under the MinMax framework [1, 6, 9, 16]. As an alternative to GAN-based methods, AutoAugment [10] automatically searches for the best augmentation policy using reinforcement learning. However, there is still a research gap between generating additional images or textual data and augmenting sequential user interaction data; the latter is still understudied. Recently, CL4Rec [53] proposes to construct different views of an interaction sequence with simple data augmentation using the contrastive loss, resulting in a more robust recommendation system. We focus on learning to generate effective augmented interaction sequences that mimic the patterns of casual users while inheriting informative transition patterns of core users.

7 CONCLUSION AND FUTURE WORK

Users who come to the online platform are heterogeneous in activity levels. To bridge the gap between recommendation for core users and casual users, we propose a model-agnostic framework *L2Aug* to learn the data augmentation policy and improve the recommendation system with the generated data. With experiments on four real-world public datasets, the proposed *L2Aug* can outperform other treatment methods and improve casual user recommendation without sacrificing the recommendation for core users. Furthermore, *L2Aug* is flexible in supporting multiple augmentation actions and different experimental (i.e., offline and online) setups. For future work, we are interested in extending such a "learning to augment" concept to other application scenarios (e.g., cross-domain, cold-start) for improving the robustness and adaptivity of different recommendation systems. Moreover, we are interested in extending our augmentation policy from the bandit setup studied here to the reinforcement learning setup, where the agent chooses editing actions depending on its previous decisions.

REFERENCES

[1] Antreas Antoniou, Amos Storkey, and Harrison Edwards. 2017. Data augmentation generative adversarial networks. *arXiv preprint arXiv:1711.04340* (2017).

[2] Alex Beutel, Ed H Chi, Zhiyuan Cheng, Hubert Pham, and John Anderson. 2017. Beyond globally optimal: Focused learning for improved recommendations. In *TheWebConf*.

[3] Ye Bi, Liqiang Song, Mengqiu Yao, Zhenyu Wu, Jianming Wang, and Jing Xiao. 2020. DCDIR: A deep cross-domain recommendation system for cold start users in insurance domain. In *SIGIR*.

[4] John S Breese, David Heckerman, and Carl Kadie. 2013. Empirical analysis of predictive algorithms for collaborative filtering. *arXiv preprint arXiv:1301.7363* (2013).

[5] Francis Buttle. 2004. *Customer relationship management*. Routledge.

[6] Dong-Kyu Chae, Jin-Soo Kang, Sang-Wook Kim, and Jaeho Choi. 2019. Rating augmentation with generative adversarial networks towards accurate collaborative filtering. In *TheWebConf*.

[7] Minmin Chen, Yuyan Wang, Can Xu, Ya Le, Mohit Sharma, Lee Richardson, Su-Lin Wu, and Ed Chi. 2021. Values of User Exploration in Recommender Systems. In *RecSys*.

[8] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *ICML*. PMLR.

[9] Danilo Croce, Giuseppe Castellucci, and Roberto Basili. 2020. GAN-BERT: Generative adversarial learning for robust text classification with a bunch of labeled examples. In *ACL*.

[10] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. 2019. Autoaugment: Learning augmentation strategies from data. In *CVPR*.

[11] Kaize Ding, Dingcheng Li, Alexander Hanbo Li, Xing Fan, Chenlei Guo, Yang Liu, and Huan Liu. 2021. Learning to Selectively Learn for Weakly-supervised Paraphrase Generation. In *EMNLP*.

[12] Kaize Ding, Zhe Xu, Hanghang Tong, and Huan Liu. 2022. Data Augmentation for Deep Graph Learning: A Survey. *arXiv preprint arXiv:2202.08235* (2022).

[13] Michael Ekstrand and John Riedl. 2012. When recommenders fail: predicting recommender failure for algorithm selection and combination. In *RecSys*.

[14] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. 2016. Domain-adversarial training of neural networks. In *JMLR*.

[15] Zeno Gantner, Lucas Drumond, Christoph Freudenthaler, Steffen Rendle, and Lars Schmidt-Thieme. 2010. Learning attribute-to-feature mappings for cold-start recommendations. In *ICDM*.

[16] Min Gao, Junwei Zhang, Junliang Yu, Jundong Li, Junhao Wen, and Qingyu Xiong. 2021. Recommender systems based on generative adversarial networks: A problem-driven perspective. *Information Sciences* (2021).

[17] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. 2020. Momentum contrast for unsupervised visual representation learning. In *CVPR*.

[18] Ruining He and Julian McAuley. 2016. Fusing similarity models with markov chains for sparse sequential recommendation. In *ICDM*.

[19] Balázs Hidasi and Alexandros Karatzoglou. 2018. Recurrent neural networks with top-k gains for session-based recommendations. In *CIKM*.

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

[21] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

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

[23] Wang-Cheng Kang and Julian McAuley. 2018. Self-attentive sequential recommendation. In *ICDM*.

[24] Muhammad Murad Khan, Roliana Ibrahim, and Imran Ghani. 2017. Cross domain recommender systems: a systematic literature review. In *CSUR*.

[25] Taeksoo Kim, Moonsu Cha, Hyunsoo Kim, Jung Kwon Lee, and Jiwon Kim. 2017. Learning to discover cross-domain relations with generative adversarial networks. In *ICML*.

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

[27] Tom Ko, Vijayaditya Peddinti, Daniel Povey, and Sanjeev Khudanpur. 2015. Audio augmentation for speech recognition. In *Interspeech*.

[28] Sosuke Kobayashi. 2018. Contextual augmentation: Data augmentation by words with paradigmatic relations. In *NAACL-HLT*.

[29] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *NeurIPS*.

[30] Preethi Lahoti, Alex Beutel, Jilin Chen, Kang Lee, Flavien Prost, Nithum Thain, Xuezhi Wang, and Ed H Chi. 2020. Fairness without demographics through adversarially reweighted learning. In *NeurIPS*.

[31] Hoyeop Lee, Jinbae Im, Seongwon Jang, Hyunsouk Cho, and Sehee Chung. 2019. MeLU: Meta-Learned User Preference Estimator for Cold-Start Recommendation.

In *KDD*.
[32] Jingjing Li, Mengmeng Jing, Ke Lu, Lei Zhu, Yang Yang, and Zi Huang. 2019. From zero-shot learning to cold-start recommendation. In *AAAI*.
[33] Mi Luo, Fei Chen, Pengxiang Cheng, Zhenhua Dong, Xiuqiang He, Jiashi Feng, and Zhenguo Li. 2020. Metaselector: Meta-learning for recommendation with user-level adaptive model selection. In *TheWebConf*.
[34] Chen Ma, Liheng Ma, Yingxue Zhang, Jianing Sun, Xue Liu, and Mark Coates. 2020. Memory augmented graph neural networks for sequential recommendation. In *AAAI*.
[35] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. 2015. Image-based recommendations on styles and substitutes. In *SIGIR*.
[36] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602* (2013).
[37] Tom O'Malley, Elie Bursztein, James Long, François Chollet, Haifeng Jin, Luca Invernizzi, et al. 2019. KerasTuner. (2019).
[38] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2010. Factorizing personalized markov chains for next-basket recommendation. In *WWW*.
[39] David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy P Lillicrap, and Greg Wayne. 2018. Experience replay for continual learning. *arXiv preprint arXiv:1811.11682* (2018).
[40] Gerard Salton and Michael J McGill. 1983. *Introduction to modern information retrieval*. mcgraw-hill.
[41] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. 2019. BERT4Rec: Sequential Recommendation with Bidirectional Encoder Representations from Transformer. *CIKM*.
[42] Jiaxi Tang and Ke Wang. 2018. Personalized top-n sequential recommendation via convolutional sequence embedding. In *WSDM*.
[43] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NeurIPS*.
[44] Maksims Volkovs, Guangwei Yu, and Tomi Poutanen. 2017. Dropoutnet: Addressing cold start in recommender systems. In *NeurIPS*.
[45] Mengting Wan and Julian McAuley. 2018. Item recommendation on monotonic behavior chains. In *RecSys*.
[46] Bo Wang, Minghui Qiu, Xisen Wang, Yaliang Li, Yu Gong, Xiaoyi Zeng, Jun Huang, Bo Zheng, Deng Cai, and Jingren Zhou. 2019. A minimax game for instance based selective transfer learning. In *KDD*.
[47] Jianling Wang and James Caverlee. 2019. Recurrent Recommendation with Local Coherence. In *WSDM*.
[48] Jianling Wang, Kaize Ding, and James Caverlee. 2021. Sequential Recommendation for Cold-start Users with Meta Transitional Learning. In *SIGIR*.
[49] Jianling Wang, Kaize Ding, Liangjie Hong, Huan Liu, and James Caverlee. 2020. Next-item recommendation with sequential hypergraphs. In *SIGIR*.
[50] Jason Wei and Kai Zou. 2019. Eda: Easy data augmentation techniques for boosting performance on text classification tasks. *EMNLP* (2019).
[51] Chao-Yuan Wu, Amr Ahmed, Alex Beutel, Alexander J Smola, and How Jing. 2017. Recurrent recommender networks. In *WSDM*.
[52] Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan. 2019. Session-based recommendation with graph neural networks. In *AAAI*.
[53] Xu Xie, Fei Sun, Zhaoyang Liu, Shiwen Wu, Jinyang Gao, Bolin Ding, and Bin Cui. 2021. Contrastive Learning for Sequential Recommendation. *SIGIR*.
[54] Ziyu Yao, Jayavardhan Reddy Peddamail, and Huan Sun. 2019. Coacor: Code annotation for code retrieval with reinforcement learning. In *TheWebConf*.
[55] Jianwen Yin, Chenghao Liu, Weiqing Wang, Jianling Sun, and Steven CH Hoi. 2020. Learning Transferrable Parameters for Long-tailed Sequential User Behavior Modeling. In *KDD*.
[56] Fajie Yuan, Alexandros Karatzoglou, Ioannis Arapakis, Joemon M Jose, and Xiangnan He. 2019. A Simple Convolutional Generative Network for Next Item Recommendation. In *WSDM*.
[57] Kaitao Zhang, Chenyan Xiong, Zhenghao Liu, and Zhiyuan Liu. 2020. Selective weak supervision for neural information retrieval. In *TheWebConf*.
[58] Cheng Zhao, Chenliang Li, Rong Xiao, Hongbo Deng, and Aixin Sun. 2020. Catn: Cross-domain recommendation for cold-start users via aspect transfer network. In *SIGIR*.
[59] Xiangyu Zhao, Liang Zhang, Long Xia, Zhuoye Ding, Dawei Yin, and Jiliang Tang. 2019. Deep reinforcement learning for list-wise recommendations.
[60] Yujia Zheng, Siyi Liu, Zekun Li, and Shu Wu. 2020. Cold-start Sequential Recommendation via Meta Learner. In *AAAI*.
[61] Feng Zhu, Yan Wang, Chaochao Chen, Jun Zhou, Longfei Li, and Guanfeng Liu. 2021. Cross-domain recommendation: challenges, progress, and prospects. In *IJCAI*.

A APPENDIX

A.1 Datasets Details

Amazon. This is a public dataset[1] obtained from the E-commerce website – Amazon, which includes users' reviews and ratings on products from May 1996 to July 2014. The dataset is split based on the product categories. In the experiments, we focus on three of the most popular categories to exploit the efficiency of the models in E-commerce scenarios.

Gooreads. The public dataset[2] were collected in late 2017 from Goodreads, which is a online platforms for users to share their opinions on books. In the experiments, we focus on users' reviews and ratings as their interactions with books. We treat different ratings and reviews equally and convert the interactions into binary feedback (i.e., interacted or not-interacted).

For each of the datasets, we group the interactions by users and then sort them based on their timestamps. Firstly, we calculate the average time gap between their consecutive interactions of each user. Then, we classify users with average time gap less than 30 days as core users, otherwise we label them as casual users. Furthermore, to avoid information leakage during the training process, we split each dataset with a corresponding cutting timestamp T, such that the interactions before T are used for model training. And for each users, the first interaction after T is for parameter fine-tuning and the second one is for testing. We select the cutting timestamp T to be January 1, 2009 for all the datasets.

A.2 Implementation Details

Implementation of Baselines. For all the sequential recommendation models, we adopt their implementations in Tensorflow so that they are compatible with the treatment methods.

- **GRU4Rec** [20]: We adopt the Tensorflow implementation[3] and use one GRU layer with 100 hidden units as in the default setting.
- **SASRec** [23]: The implementation[4] is published with the original paper. As suggested by the authors, we set the number of heads and the number of blocks in self-attention layer as 1 and 2, correspondingly.
- **NextItNet** [56]: We utilize the code[5] published by the authors and keep the settings. unchanged.

As for the the baseline treatment methods, we combine them with various sequential recommendation models to examine its efficiency in improving casual user recommendation.

- **Random**: We randomly drops the interactions of core users to obtain the synthetic data sequences. The dropping rate is selected based on the performance on validation set.
- **Focused Learning** [2]: Following the idea in the paper, we grid search for the best performing hyperparameters with the hyperparameter optimization framework KerasTuner [37][6].

[1]http://jmcauley.ucsd.edu/data/amazon/links.html
[2]https://sites.google.com/eng.ucsd.edu/ucsdbookgraph/home
[3]https://github.com/Songweiping/GRU4Rec_TensorFlow
[4]https://github.com/kang205/SASRec
[5]https://github.com/fajieyuan/WSDM2019-nextitnet
[6]https://github.com/keras-team/keras-tuner

- **Adversarial Reweighting** [30]: We adapt the original implementations[7] to the setup of sequential recommendation via replacing the classifer with the recommendation models.

Note that to achieve the best performance for each model, we perform a grid search for the dropout rate in {0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8} for *Random*, the regularization weight λ in $\{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$ for *Focused Learning*, and the learning rate in $\{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$ for all methods.

A.3 Parameter Sensitivity

One key design in L2Aug is to elicit the signal from meta validation set to guide the meta-learner for learning an effective augmentation policy. Note that the meta validation set is a small subset sampled from the casual user interaction sequences. In this section, we conduct experiments to evaluate the influence of the size of the meta validation set. In Figure 6, we change the ratio of the meta validation set (i.e., $\frac{M}{N}$) and show the resulted recommendation performance on casual users with SASRec and L2Aug. We can observe that casual user recommendation would be improved with larger meta validation set, which demonstrates the effective guidance the meta validation set can provide to achieve our goal. However, sampling a meta validation set with size larger than 10% would not bring in positive effect to the recommendation. Thus in the experiments, we select the ratio to be 10% for the best performance.

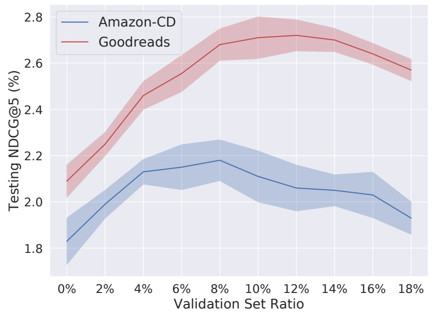
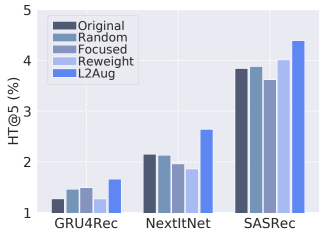


Figure 6: Test performance VS Meta validation ratio

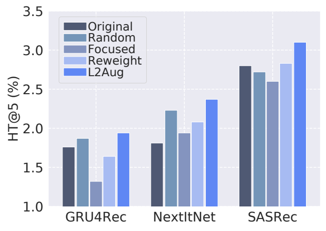
A.4 Additional Experimental Results

As supplement to Figure 3, we compare the performance of *L2Aug* with other treatments for *core users* under HT@5, NDCG@10 and HT@10 in Figure 7, 8 and 9. Though improving the original sequential recommendation models on casual users, the baseline treatment methods loss their power in modeling core users' transitional patterns and result in a decrease for the overall recommendation. For example, on Amazon_Books dataset, combining *Focused Learning* with GRURec will result in a 25.05% and 17.48% decrease in core user recommendation. However, the proposed *L2Aug* data augmentation is able to bridge the gap between core and casual users. Thus, we can observe that it can contribute in improvement in both core and casual user recommendation across different datasets.

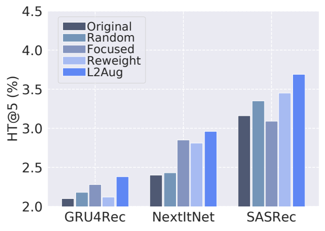
[7]https://github.com/google-research/google-research/tree/master/group_agnostic_fairness



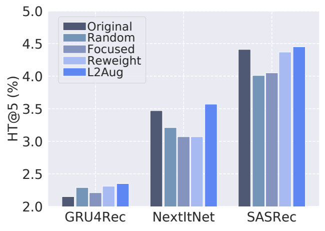
(a) **Amazon_CDs**



(b) **Amazon_Books**

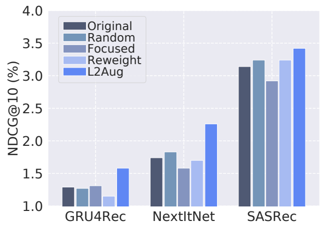


(c) **Amazon_Movies**

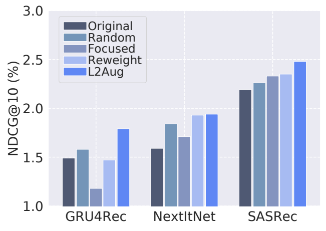


(d) **Goodreads**

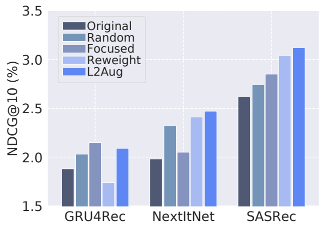
Figure 7: Performance (HT@5) in offline setup on Core User Recommendation of various models on different datasets.



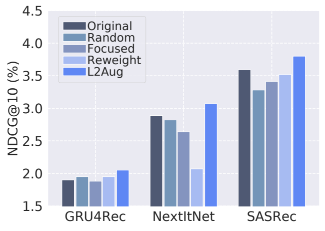
(a) **Amazon_CDs**



(b) **Amazon_Books**

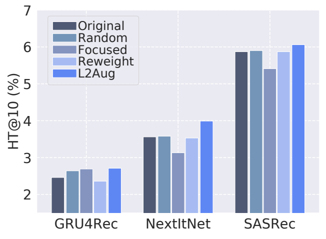


(c) **Amazon_Movies**

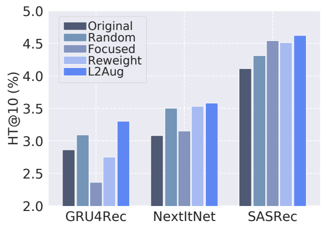


(d) **Goodreads**

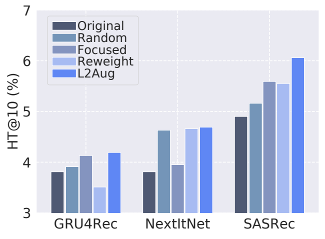
Figure 8: Performance (NDCG@10) in offline setup on Core User Recommendation of various models on different datasets.



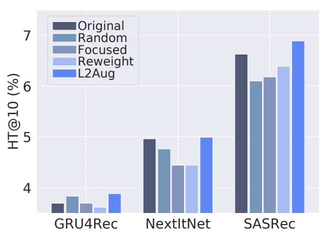
(a) **Amazon_CDs**



(b) **Amazon_Books**



(c) **Amazon_Movies**



(d) **Goodreads**

Figure 9: Performance (HT@10) in offline setup on Core User Recommendation of various models on different datasets.