A Pre-trained Zero-shot Sequential Recommendation Framework via Popularity Dynamics

Junting Wang*
junting3@illinois.edu
University of Illinois at Urbana-Champaign
Urbana, Illinois, USA

Praneet Rathi*
prathi3@illinois.edu
University of Illinois at Urbana-Champaign
Urbana, Illinois, USA

Hari Sundaram
hs1@illinois.edu
University of Illinois at Urbana-Champaign
Urbana, Illinois, USA

ABSTRACT

This paper proposes a novel pre-trained framework for zero-shot cross-domain sequential recommendation without auxiliary information. While using auxiliary information (*e.g.,* item descriptions) seems promising for cross-domain transfer, a cross-domain adaptation of sequential recommenders can be challenging when the target domain differs from the source domain—item descriptions are in different languages; metadata modalities (*e.g.,* audio, image, and text) differ across source and target domains. If we can learn universal item representations independent of the domain type (*e.g.,* groceries, movies), we can achieve zero-shot cross-domain transfer without auxiliary information. Our critical insight is that user interaction sequences highlight shifting user preferences via the popularity dynamics of interacted items. We present a pre-trained sequential recommendation framework: **PrepRec**, which utilizes a novel popularity dynamics-aware transformer architecture. Through extensive experiments on five real-world datasets, we show that PrepRec, without any auxiliary information, can zero-shot adapt to new application domains and achieve competitive performance compared to state-of-the-art sequential recommender models. In addition, we show that PrepRec complements existing sequential recommenders. With a simple post-hoc interpolation, PrepRec improves the performance of existing sequential recommenders on average by 11.8% in Recall@10 and 22% in NDCG@10. We provide an anonymized implementation of PrepRec at https://github.com/CrowdDynamicsLab/preprec.

CCS CONCEPTS

• **Information systems** → **Recommender systems**.

KEYWORDS

Recommender System, Zero-shot Sequential Recommendation

ACM Reference Format:
Junting Wang, Praneet Rathi, and Hari Sundaram. 2024. A Pre-trained Zero-shot Sequential Recommendation Framework via Popularity Dynamics. In *18th ACM Conference on Recommender Systems (RecSys '24), October 14–18, 2024, Bari, Italy.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3640457.3688145

*Both authors contributed equally to this research.

RecSys '24, October 14–18, 2024, Bari, Italy

ACM ISBN 979-8-4007-0505-2/24/10
https://doi.org/10.1145/3640457.3688145

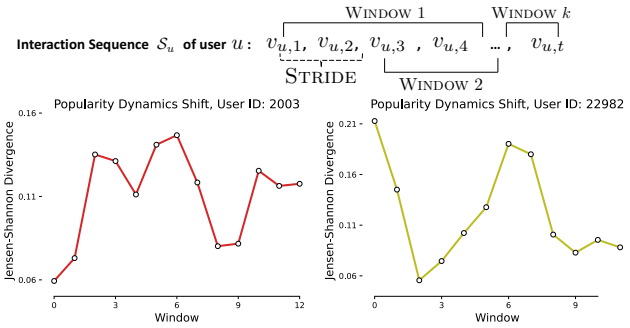

Figure 1: Jensen-Shannon divergence between two consecutive windows (k, $k+1$), where we measure the change in popularity of items in the user's sequence. The sampled users are from the Amazon Office dataset. This shows that there exist temporal item popularity shifts in user interaction sequences. We set the window size to 10 and stride to 5.

1 INTRODUCTION

Modeling sequential user behavior is critical to the success of online applications such as e-commerce, video streaming, and social media. Despite essential innovations for tackling the sequential recommendation task [16, 22, 27, 39, 45, 47], these systems have some limitations. Firstly, they must be trained from scratch for each application domain because they learn domain-specific item representations [22, 45], which is resource-consuming and limits model reuse across domains. Even within a domain, they must be retrained when there is a large influx of users or items to maintain performance. Prior work tackles these limitations by incorporating auxiliary information [7, 12, 19], *e.g.,* item descriptions. However, using auxiliary information can be problematic for cross-domain transfer if item descriptions are in different languages (*e.g.,* English and Chinese), or if the metadata modalities (*e.g.,* , audio, image, and text) differ across domains.

This paper tackles a challenging cross-domain transfer setting where we assume no access to auxiliary information. Thus, we ask: *can we build a pre-trained sequential recommender system capable of cross-domain and cross-application transfer without ny auxiliary information?* (*e.g.,* using the model trained for online shopping in the US to predict the next movie a user in China will watch). Our work is a performance baseline for cross-domain tasks; using compatible (*i.e.,* same language/modality) auxiliary information across domains, can only improve the performance.

At first glance, developing pre-trained sequential recommenders for cross-domain inference seems impossible. While we see pre-trained language [5, 6, 30, 35, 37] and vision models [9, 13, 36] show excellent generalizability across datasets and applications, being

able to achieve state-of-the-art performance by just a few fine-tuning steps [6, 30] or even without any training [5, 35] (*i.e.,* zero-shot transfer), there are essential differences. The representations learned by the pre-trained language model seem universal since the training domain and the application domain (*e.g.,* text prediction and generation) share the same language and vocabulary, supporting the effective reuse of the word representations. However, in the cross-domain recommendation, the items are distinct across domains in recommendation datasets (*e.g.,* grocery items vs movies). Therefore, forming such generalizable correspondence is nearly impossible if we learn representations for each item *within* each domain. Recent work explores pre-trained models for sequential recommendation [7, 12, 19] within the same application (*e.g.,* online retail). However, they assume access to metadata of items (*e.g.,* item description), which is domain-dependent and is often not generalizable to other domains. These models cannot learn universal representations of items; instead, they bypass the representation learning problem by using additional item-side information.

Our Insight: There exist item popularity shifts in the user's sequence, as indicated in Figure 1. The item popularity shifts can be explained as temporal shifts in the user's preferences. For example, a user might be interested in buying some common office goods such as pens, papers, and notebooks, but afterward, they might look for other less common office goods such as a whiteboard or a desk. Previous works try to learn users' preference from the past sequence but *ignore the crucial aspect of item popularity dynamics*, which could indicate the user's changing preferences. We know that the marginal distribution of user and item activities are heavy-tailed across datasets, supported by prior work in network science [3, 4] and by experiments in recommender systems [40]. In addition, recent work in recommender systems suggests that the popularity dynamics of items are also crucial for predicting users' behaviors [21].

Present Work: In this paper, we develop universal, transferable item representations for the zero-shot, cross-domain setting based on the popularity dynamics of items. We explicitly model the popularity dynamics of items and propose a novel pre-trained sequential recommendation framework: PrepRec. We learn universal item representations based on their popularity dynamics instead of their explicit item IDs or auxiliary information. We encode the relative time interval between two consecutive interactions via relative-time encoding and the position of each interaction in the sequence by positional encoding. Using physical time ensures that the predictions are not anti-causal, *i.e.,* using the future interactions to predict the present. We propose a popularity dynamics-aware transformer architecture for learning universal sequence representations. We show that it is possible to build a pre-trained sequential recommender system capable of cross-domain and cross-application transfer without any auxiliary information. Our **key contributions** are as follows:

Universal item and sequence representations: We are the first to learn universal item and sequence representations for sequential recommendation *without any auxiliary information* by exploiting item popularity dynamics. In contrast, prior research learns item representations for each item ID or through item auxiliary information. We learn universal item representations by modeling item popularity dynamics of two temporal resolutions: coarse and fine-grained. We learn universal sequence representations using a carefully designed popularity dynamics-aware transformer architecture. These universal item and sequence representations make possible pre-trained sequential recommender systems capable of cross-domain and cross-application transfer without any auxiliary information.

Zero-shot transfer without auxiliary information: We propose a new challenging setting for pre-trained sequential recommender systems: zero-shot cross-domain and cross-application transfer without any auxiliary information. In contrast, previous pre-trained sequential recommenders requires overlapping users [61], application-dependent auxiliary information [7, 12, 18, 19, 56], and are few-shot adapted to related domains within the same application [18, 19, 56]. Our work establishes a performance baseline for cross-domain sequential recommenders that use compatible (*i.e.,* same language/modality) auxiliary information across domains, as such metadata can only improve the performance of cross-domain transfer.

With extensive experiments, we empirically show that PrepRec has excellent generalizability across domains and applications. Remarkably, had we trained a state-of-the-art model from scratch for the target domain, *instead of zero-shot transfer using* PrepRec, the maximum performance gain over PrepRec would have been only 4%. In addition, we show that PrepRec is complementary to state-of-the-art sequential recommenders and with a post-hoc interpolation, PrepRec can outperform the state-of-the-art sequential recommender system on average by 11.8% in Recall@10 and 22% in NDCG@10. We attribute the improvements to the performance gains over long-tail items, which we show in the qualitative analysis. With this work, we set a baseline for pre-trained sequential recommenders and show that popularity dynamics not only enable us to build a pre-trained sequential recommender system capable of zero-shot transfer but also significantly boost the performance of sequential recommendation.

2 RELATED WORK

Sequential Recommendation: Sequential recommenders model user behavior as a sequence of interactions, and aim to predict the next item that a user will interact with. Early sequential recommenders adopt Markov chains [39, 42] and basic neural network architectures [16, 17, 47, 50, 51]. With the success of Transformer [52] in modeling sequential data [22, 27, 45]adopt the transformer architecture for sequential recommendation. Additionally, [27] considers the timestamps of each interaction and proposes a time-aware attention mechanism. [32, 46, 60] separate interaction sequences and categorize them to show the long-term and short-term interests of users. Temporal sequential recommenders [25, 58, 62] models the change in users' preferences. These works, while achieving state-of-the-art performance, only focus on the regular sequential recommendation and cannot transfer to other domains.

Cross-domain Recommendation: Cross-domain recommendation literature leverages the information-rich domain to improve the recommendation performance on the data-sparse domain [20, 28, 33]. However, most of these works assume user or item overlap [20, 28, 33, 63, 64] for effective knowledge transfer. Other cross-domain literature focuses on the cold-start problem [8, 10, 11, 26, 29, 31, 53, 57, 64]. In addition, multi-domain recommenders [1, 43]

leverage multi-domain data to gain insights into user preferences and item characteristics.

Pre-trained Sequential Recommenders: Recently, pre-trained recommenders have caught the attention of the community. ZESRec [7] is capable of zero-shot sequential recommendations. However, it only works for closely related domains and requires item metadata. PeterRec [61] requires overlapping users in both domains. On the other hand, finetuning-based models, *e.g.,* MISSRec [56], UnisRec [19], and VQ-Rec [18], are not designed for zero-shot sequential recommendation and works within the same application (e-commerce), and they rely on application-dependent auxiliary information. [54] investigates the joint and marginal activity distribution of users and items, but are not suitable for the sequential recommendation task.

To summarize, prior works on sequential recommendation focus on learning high-quality representations for *each* item in the training set and are not generalizable across domains. Pre-trained sequential recommenders are evaluated on closely related domains and platforms and rely heavily on application-dependent auxiliary information of items.

3 PROBLEM DEFINITION

In this section, we formally define the research problems this paper addresses (*i.e.,* regular sequential recommendation and zero-shot sequential recommendation) and introduce our notations.

In sequential recommendation, denote $\boldsymbol{M}$ as the implicit feedback matrix, $\mathcal{U} = \{u_1, u_2, ..., u_{|\mathcal{U}|}\}$ as the set of users, $\mathcal{V} = \{v_1, v_2, ..., v_{|\mathcal{V}|}\}$ as the set of items. The goal of sequential recommendation is to learn a scoring function, that predicts the next item $v_{u,t}$ given a user u 's interaction history $\mathcal{S}_u = \{v_{u,1}, v_{u,2}, ..., v_{u,t-1}\}$. Note that in this paper, since we model time explicitly, we assume access to the timestamp of each interaction, including the next item interaction. We argue that this is a reasonable assumption since the timestamp of the next interaction is always available in practice. For example, if Alice logs in to Netflix, Netflix will always know when Alice logs in and can predict the next movie for Alice. Formally, we define the scoring function as $\mathcal{F}(v^t|\mathcal{S}_u, \boldsymbol{M})$, where t is the time of the prediction.

Zero-shot Sequential Recommendation: Given two domains $\boldsymbol{M}$ and $\boldsymbol{M}'$ over $\mathcal{U}, \mathcal{V}$ and $\mathcal{U}', \mathcal{V}'$ respectively, we study the zero-shot recommendation problem in the scenario where the domains are different ($\boldsymbol{M} \cap \boldsymbol{M}' = \emptyset$), users are disjoint ($\mathcal{U} \cap \mathcal{U}' = \emptyset$), and item sets are unique ($\mathcal{V} \cap \mathcal{V}' = \emptyset$). The goal is to produce a scoring function $\mathcal{F}'$ without training on $\boldsymbol{M}'$ directly. In other words, the scoring function $\mathcal{F}'$ has to be trained on a different interaction matrix $\boldsymbol{M}$. Furthermore, we assume there is no metadata associated with users or items, which makes the problem particularly challenging but crucial to study. We want to set a baseline for pre-trained sequential recommenders, using metadata can only improve and simply the problem.

4 PREPREC FRAMEWORK

We first introduce the model architecture of PrepRec (§ 4.1) then the training procedure (§ 4.2). Finally, we formally define the zero-shot inference process (§ 4.3).

4.1 Model Architecture

The first step of building a pre-trained sequential recommender is to learn universal item representations. Our solution is to exploit the item popularity statistics to learn universal item representations. We learn to represent items at a given timestamp through the changes in their popularity histories over different periods, *i.e.,* popularity dynamics. We propose a popularity dynamics-aware Transformer architecture that obtains the representation of users' behavior sequences through item popularity dynamics.

4.1.1 Item Popularity Encoder. We learn to represent items based on their popularity dynamics, *i.e.,* changes in their popularity histories. Intuitively, popularity can be calculated by two horizons: long-term and short-term. Long-term horizons reflect the overall popularity of items, whereas short-term horizons should capture the recent trends in the domain. For example, the long-term popularity of a winter coat measures how popular is the coat in general, while its short-term popularity reflects more temporal changes, *e.g.,* season, weather conditions, and fashion trends. Therefore, consider an item v_j that has interaction at time t, denoted as v_j^t, we define two popularity representations for v_j^t: popularity $\mathbf{p}_j^t \in \mathbb{R}^k$ over a coarse period (*e.g.,* month) and popularity $\mathbf{h}_j^t \in \mathbb{R}^k$ over a fine period (*e.g.,* week).

To calculate $\mathbf{p}_j^t$ and $\mathbf{h}_j^t$, we first calculate the popularities of v_j^t over the two horizons, denoted as $a_j^t \in \mathbb{R}^+$ (coarse period number of interactions) and $b_j^t \in \mathbb{R}^+$ (fine period number of interactions). Specifically, we calculate them as:

$$a_j^t = \sum_{m=1}^{t} \gamma^{t-m} c_a(v_j^m), \quad b_j^t = c_b(v_j^t) \tag{1}$$

where $\gamma \in \mathbb{R}^+$ is a pre-defined discount factor and $c_a(v_j^m)$ is the number of interactions of v_j over a coarse time period m. Similarly, $c_b(v_j^t)$ denotes the number of interactions of v_j over a fine period t. We do not impose the discounting factor when computing b_j^t since we want it to capture the current popularity information, whereas a_j^t captures the cumulative popularity of an item over a longer horizon.

To make item popularity comparable across domains, we calculate the percentiles of a_j^t and b_j^t relative to their corresponding coarser and finer popularity distributions over all items at time t, denoted as $P(a_j^t) \in \mathbb{R}^+$ and $P(b_j^t) \in \mathbb{R}^+$, respectively.

We now encode the popularity percentiles $P(a_j^t)$ and $P(b_j^t)$ into k dimensional vector representations $\mathbf{p}_j^t$ and $\mathbf{h}_j^t$ respectively. Denote the popularity encoder as $E_p : \mathbb{R}^+ \to \mathbb{R}^k$, which takes in a percentile value. Suppose given the popularity percentile $P(a_j^t) \in \mathbb{R}^+$ over a coarse time period t, the coarse level popularity vector representation $\mathbf{p}_j^t \in \mathbb{R}^k$ is computed as follows:

$$\mathbf{p}_j^t = E_p(P(a_j^t))$$

$$(\mathbf{p}_j^t)_i = \begin{cases} 1 - \{\frac{P}{k-1}\}, & \text{if } i = \lfloor \frac{P}{k-1} \rfloor \\ \{\frac{P}{k-1}\}, & \text{if } i = \lfloor \frac{P}{k-1} \rfloor + 1 \\ 0, & \text{otherwise} \end{cases}$$

where $\lfloor \cdot \rfloor$ denotes the floor, $\{\cdot\}$ denotes the fractional part of a number, and $(\mathbf{p}_j^t)_i$ denotes the i-th index of $\mathbf{p}_j^t$. For example, if $k =$

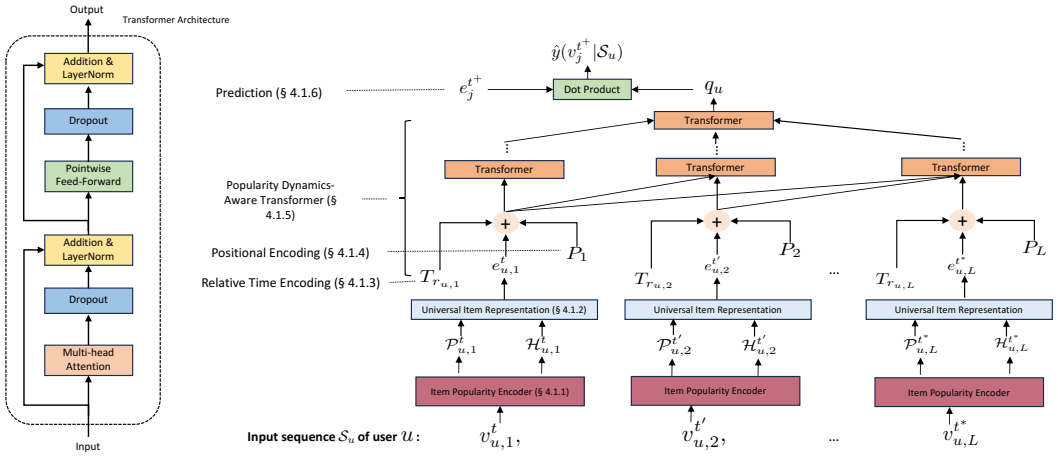

Figure 2: Model Architecture of PrepRec

11, $E_p(40.1) = [0, 0, 0, 0, 0.99, 0.01, 0, 0, 0, 0, 0]$. The interpretation of this would be considering the 10 deciles for $i \in \{0, 1, \ldots, 9, 10\}$ as basis vectors, and this popularity encoding as a linear combination of the nearest (in percentile space) two basis vectors. The fine level popularity vector is calculated identically, *i.e.,* $\mathbf{h}_j^t = E_p(P(b_j^t))$. In this example, we've fixed the vector representation size to be 11, but this approach is fully generalizable to other sizes and would just require changing the multipliers in the encoding function. We also experimented with sinuoisal encodings of the same size, but found that the linear encoding empirically performed better.

4.1.2 Universal Item Representation. We now define the popularity dynamics of v_j at time t over the coarse period (long-term horizon) to be $\mathcal{P}_j^t = \{\mathbf{p}_j^1, \mathbf{p}_j^2, ..., \mathbf{p}_j^{t-1}\}$, and over the fine period (short-term horizon) as $\mathcal{H}_j^t = \{\mathbf{h}_j^1, \mathbf{h}_j^2, ..., \mathbf{h}_j^{t-1}\}$. We use $t-1$ to constrain access to future interactions and prevent information leakage, *i.e.,* we do not have access to the popularity statistics of v_j at time t if we are at time t. For example, say an interaction happens on the second Wednesday in February, we consider the coarser and finer time period up until the end of January and the end of the first week in February respectively. To limit computation, we constrain window sizes m, n for $\mathcal{P}$ and $\mathcal{H}$ respectively. Formally, the coarse popularity dynamics of v_j at time t is $\mathcal{P}_j^t = \{\mathbf{p}_j^{t-m}, \mathbf{p}_j^{t-m+1}, ..., \mathbf{p}_j^{t-1}\}$, and the fine popularity dynamics of v_j at time t is $\mathcal{H}_j^t = \{\mathbf{h}_j^{t-n}, \mathbf{h}_j^{t-n+1}, ..., \mathbf{h}_j^{t-1}\}$.

Finally, we compute the embedding of item v_j at time t via the universal item representation encoder, defined as a function $\mathcal{E}(\mathcal{P}_j^t, \mathcal{H}_j^t)$ that learns to encode the popularity dynamics $\mathcal{P}_j^t$ and $\mathcal{H}_j^t$ into a d dimension vector representation $\mathbf{e}_j^t$. Specifically, we have:

$$\mathbf{e}_j^t = \mathcal{E}(\mathcal{P}_j^t, \mathcal{H}_j^t) = \boldsymbol{W}_p[(\|_{i=t-m}^{t-1}\mathbf{p}_j^i)\|(\|_{i=t-n}^{t-1}\mathbf{h}_j^i)] \tag{2}$$

where $\|$ denotes the concatenation operation, and $\boldsymbol{W}_p \in \mathbb{R}^{d \times k(m+n)}$ is a learnable weight matrix.

In addition, we define $\|_{i=t-m}^{t-1}\mathbf{p}_j^i := \mathbf{p}_j^{t-m}\|\mathbf{p}_j^{t-m+1}\|...\|\mathbf{p}_j^{t-1}$ and $\|_{i=t-n}^{t-1}\mathbf{h}_j^i := \mathbf{h}_j^{t-n}\|\mathbf{h}_j^{t-n+1}\|...\|\mathbf{h}_j^{t-1}$. The item popularity dynamics encoder can effectively capture the popularity change of items over different time periods. Most importantly, it does not take explicit item IDs or auxiliary information as input to compute the item embeddings. Instead, it learns to represent items through their popularity dynamics, which is universal across domains and applications.

4.1.3 Relative Time Interval. We also consider the time interval between two consecutive interactions when modeling sequences. Differences in time intervals might indicate differences in the users' behaviors. While previous works explore absolute time intervals [27], different domains exhibit diverse time scales, thus making modeling absolute time intervals ungeneralizable. Therefore, we propose to encode relative time intervals into modeling sequences. Given an interaction sequence $\mathcal{S}_u = \{v_{u,1}, v_{u,2}, ..., v_{u,L}\}$ of user u, we define the time interval between $v_{u,j}$ and $v_{u,j+1}$ as $t_{u,j} = t(v_{u,j+1}) - t(v_{u,j})$, where $t(v_{u,j})$ is the time that user u interacts with item $v_{u,j}$. We then rank the time intervals of user u. Define the rank of relative time interval of $t_{u,j}$ as $r_{u,j} = \text{rank}(t_{u,j})$. The relative time interval encoding of interval $t_{u,j}$ is then defined as $\boldsymbol{T}_{r_{u,j}} \in \mathbb{R}^d$, where $\boldsymbol{T} \in \mathbb{R}^{L \times d}$, following the same setup in [52], is a fixed sinusoidal encoding matrix defined as:

$$\boldsymbol{T}_{i,2j} = \sin(\frac{i}{L^{2j/d}}), \quad \boldsymbol{T}_{i,2j+1} = \cos(\frac{i}{L^{2j/d}}) \tag{3}$$

We also tried a learnable time interval encoding, but it yielded worse performance. We hypothesize that the sinusoidal encoding is more generalizable across domains and the learnable encoding is more prone to overfitting.

4.1.4 Positional Encoding. As we will see in § 4.1.5, the self-attention mechanism does not take the positions of the items into account. Therefore, following [52], we also inject a fixed positional encoding for each position in a user's sequence. Denote the positional embedding of a position l as $\boldsymbol{P}_l \in \mathbb{R}^d$, where $\boldsymbol{P} \in \mathbb{R}^{L \times d}$. We compute $\boldsymbol{P}$ using the same formula in Equation (3). Again, we also tried a learnable positional encoding as presented in [22, 45], but it yielded worse results.

4.1.5 Popularity Dynamics-Aware Transformer. We follow previous works in sequential recommendation [22, 27, 45] and propose an extension to the self-attention mechanism by incorporating universal item representations (§ 4.1.2), relative time intervals(§ 4.1.3), and positional encoding (§ 4.1.4).

Firstly, we transform the user sequence $\{v_{u,1}, v_{u,2}, ..., v_{u,|\mathcal{S}_u|}\}$ for each user u into a fixed-length sequence $\mathcal{S}_u = \{v_{u,1}, v_{u,2}, ..., v_{u,L}\}$ via truncating the oldest interactions or padding, where L is a pre-defined hyper-parameter controlling the maximum length of the sequence. Given a user sequence $\mathcal{S}_u = \{v_{u,1}, v_{u,2}, ..., v_{u,L}\}$, we compute its input matrix $\boldsymbol{E}_u$ as:

$$\boldsymbol{E}_u = \begin{bmatrix} \mathbf{e}^{t}_{u,1} + \boldsymbol{T}_{r_{u,1}} + \boldsymbol{P}_1 \\ \mathbf{e}^{t'}_{u,2} + \boldsymbol{T}_{r_{u,2}} + \boldsymbol{P}_2 \\ \vdots \\ \mathbf{e}^{t*}_{u,L} + \boldsymbol{T}_{r_{u,L}} + \boldsymbol{P}_L \end{bmatrix} \tag{4}$$

$\mathbf{e}^{t}_{u,1}, \mathbf{e}^{t'}_{u,2}, ..., \mathbf{e}^{t*}_{u,L}$ is computed from Equation (2), $\boldsymbol{T}_{r_{u,1}}, \boldsymbol{T}_{r_{u,2}}, ..., \boldsymbol{T}_{r_{u,L}}$ and $\boldsymbol{P}_1, \boldsymbol{P}_2, ..., \boldsymbol{P}_L$ are computed following the procedure in § 4.1.3 and § 4.1.4 respectively.

Multi-Head Self-Attention. We adopt a widely used multi-head self-attention mechanism [52], *i.e.,* Transformers. Specifically, it consists of multiple multi-head self-attention layers (denoted as MHAttn(·)), and point-wise feed-forward networks (FFN(·)). The multi-head self-attention mechanism is defined as:

$$\begin{aligned} \mathbf{z}_u &= \text{MHAttn}(\boldsymbol{E}_u) \\ \text{MHAttn}(\boldsymbol{E}_u) &= \text{Concat}(\text{head}_1, ..., \text{head}_h)\boldsymbol{W}^O \\ \text{head}_i &= \text{Attn}(\boldsymbol{E}_u\boldsymbol{W}_i^Q, \boldsymbol{E}_u\boldsymbol{W}_i^K, \boldsymbol{E}_u\boldsymbol{W}_i^V) \end{aligned} \tag{5}$$

where $\boldsymbol{E}_u$ is the input matrix computed from Equation (4), h is a tunable hyper-parameter indicating the number of attention heads, $\boldsymbol{W}_i^Q, \boldsymbol{W}_i^K, \boldsymbol{W}_i^V \in \mathbb{R}^{d \times d/h}$ are the learnable weight matrices, and $\mathbf{W}^O \in \mathbb{R}^{d \times d}$ is also a learnable weight matrix. Attn is the attention function and is formally defined as:

$$\text{Attn}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d/h}})\mathbf{V} \tag{6}$$

The scale factor $\sqrt{d/h}$ is used to avoid large values of the inner product, which can lead to numerical instability.

Causality: In sequential recommendation, the prediction of the $L+1$ item should only depend on the first L items that the user has interacted with in the past. However, the L-th output of the multi-head self-attention layer contains all the input information. Therefore, as in [22, 27], we do not let the model attend to the future items by forbidding links between Q_i and K_j $(j > i)$ in the attention function.

Point-Wise Feed-Forward Network: To add nonlinearity and interactions between different embedding dimensions, we follow previous works in sequential recommendation [22, 27, 45] and apply the same point-wise feed-forward network to the output of each multi-head self-attention layer. Formally, suppose the output of the multi-head self-attention layer is $\mathbf{z}_u$, the point-wise feed-forward network is defined as:

$$\text{FFN}(\mathbf{z}_u) = \text{ReLU}(\mathbf{z}_u\boldsymbol{W}_1 + \mathbf{b}_1)\boldsymbol{W}_2 + \mathbf{b}_2 \tag{7}$$

where $\boldsymbol{W}_1 \in \mathbb{R}^{d \times d}$ and $\boldsymbol{W}_2 \in \mathbb{R}^{d \times d}$ are learnable weight matrices, and $\mathbf{b}_1 \in \mathbb{R}^d$ and $\mathbf{b}_2 \in \mathbb{R}^d$ are learnable bias vectors.

Stacking Layers: As shown in previous works [22], stacking multiple multi-head self-attention layers and point-wise feed-forward networks can potentially lead to overfitting and instability during the training. Therefore, we follow previous works [22, 27, 45] and apply layer normalization [2] and residual connections to each multi-head self-attention layer and point-wise feed-forward network. Formally, we have:

$$g(\mathbf{x}) = \mathbf{x} + \text{Dropout}(g(\text{LayerNorm}(\mathbf{x}))) \tag{8}$$

$g(\mathbf{x})$ is either the multi-head self-attention layer or the point-wise feed-forward network. Therefore, for every multi-head self-attention layer and point-wise feed-forward network, we first apply layer normalization to the input, then apply the multi-head self-attention layer or point-wise feed-forward network, and finally apply dropout and add the input $\mathbf{x}$ to the layer output. The LayerNorm function is defined as:

$$\text{LayerNorm}(\mathbf{x}) = \alpha \odot \frac{\mathbf{x} - \mu}{\sqrt{\sigma^2 + \epsilon}} + \beta \tag{9}$$

where $\odot$ denotes the element-wise product, μ and σ are the mean and standard deviation of $\mathbf{x}$, α and β are learnable parameters, and ϵ is a small constant to avoid numerical instability.

4.1.6 Prediction. Given a sequence $\mathcal{S}_u$ of user u as input, we denote $\mathbf{q}_u$ as the output of the popularity dynamics-aware transformer. Suppose at time t^+, we want to predict the likelihood of v_j being the next item in the sequence, we first compute the item representation $\mathbf{e}^{t^+}_j$ from § 4.1.2. Then, we predict the score as the inner product of $\mathbf{q}_u$ and $\mathbf{e}^{t^+}_j$, formally:

$$\hat{y}(v^t_j|\mathcal{S}_u) = < \mathbf{q}_u, \mathbf{e}^{t^+}_j > \tag{10}$$

Note that there is no information leakage in the prediction process, *i.e.,* we do not assume access to the popularity statistics of v_j at time t^+ if we are at time t^+ (§ 4.1.2).

Dataset	#users	#items	#actions	avg length	density
Office	101,133	27,500	0.74M	7.3	0.03%
Tool	240,464	73,153	1.96M	8.1	0.01%
Movie	70,404	40,210	11.55M	164.2	0.41%
Music	20,539	10,121	0.66M	32.2	0.32%
Epinions	30,989	20,382	0.54M	17.5	0.09%

Table 1: Dataset statistics

4.2 Training Procedure

Now we present how to train the PrepRec model. Similar to [22], we adopt the binary cross entropy loss as the objective function, formally:

$$\begin{aligned} \mathcal{L} = -\sum_{\mathcal{S}_u \in \mathcal{S}} \sum_{z \in [1,2,...,L-1]} & [\log \sigma(\hat{y}(v^t_{z+1}|\mathcal{S}_{u,:z})) \\ & + \sum_{j' \notin \mathcal{S}_u} \log \sigma(1 - \hat{y}(v^t_{j'}|\mathcal{S}_{u,:z}))] \end{aligned} \tag{11}$$

where $\mathcal{S}_{u,:z} := \{v_{u,1}, v_{u,2}, ..., v_{u,z}\}$. v^t_{z+1} represents the $z+1$-th item in the sequence that happened at time t. We use Adam [23] as the optimizer and train the model end-to-end. Note that compared to previous sequential recommenders, we do not have any parameters modeling item IDs. Essentially, we are only optimizing $\boldsymbol{W}_p$ and parameters related to the multi-head self-attention mechanism.

S→T	Office		Tool		Movie		Music		Epinions	
Metric	R@10	N@10	R@10	N@10	R@10	N@10	R@10	N@10	R@10	N@10
Reference: Regular Sequential Recommendation										
MostPop	0.450	0.272	0.459	0.274	0.586	0.361	0.519	0.327	0.438	0.296
BERT4Rec [45]	0.541*	0.358*	0.544	0.350	0.900	0.728	0.816*	0.602*	0.702	0.512
PrepRec	0.536	0.344	0.551*	0.359*	0.908*	0.738*	0.782	0.573	0.795*	0.580*
Zero-shot Sequential Recommendation, Source→Target										
Office →	—	—	**0.540**	**0.326**	0.838	0.624	0.755	0.542	0.724	0.512
Tool →	**0.543**	**0.332**	—	—	**0.881**	**0.659**	0.749	0.536	0.717	0.510
Movie→	0.520	0.320	0.508	0.302	—	—	**0.811**	**0.600**	**0.751**	**0.537**
Music→	0.503	0.310	0.496	0.312	0.836	0.636	—	—	0.739	0.518
Epinions→	0.517	0.317	0.470	0.302	0.872	0.656	0.774	0.517	—	—

Table 2: *Zero-shot recommendation results.* Results for cross-domain, cross-application zero-shot transfer. S→T means we pre-train PrepRec using S's data (columns) and evaluate on T's data (rows). We follow the zero-shot inference setting in § 4.3. Reference models are trained from scratch on the target dataset. The best-performing zero-shot transfer results of each dataset are in bold. We empirically show PrepRec achieves remarkable zero-shot generalization performance across domains.

4.3 Zero-shot Inference

Suppose we are given a pre-trained model $\mathcal{F}$ trained on $\boldsymbol{M}$, where $\mathcal{F}$ is the scoring function learned from source domain $\boldsymbol{M}$. Denote the interaction matrix of the target domain as $\boldsymbol{M'}$. We first compute the popularity dynamics of each item in $\boldsymbol{M'}$ over a coarser period and a finer period. Then, we apply the pre-trained model $\mathcal{F}$ to $\boldsymbol{M'}$ and compute the prediction score as:

$$\hat{y}(v^t_{j'}|\mathcal{S}'_u) = \mathcal{F}(v^t_{j'}|\mathcal{S}'_u, \boldsymbol{M'}) \tag{12}$$

Note that in this procedure, we use the pre-trained model $\mathcal{F}$ trained on domain $\boldsymbol{M'}$ to predict the next item $v^t_{j'}$ that user u' will interact with in domain $\boldsymbol{M'}$. We do not use any auxiliary information in either domain. In addition, none of the parameters in $\mathcal{F}$ are updated during the zero-shot inference process.

To summarize, in this section, we showed how to develop a pre-trained sequential recommender system based on the popularity dynamics of items. We enforce the structure of each interaction in the sequence by the positional encoding and introduce a relative time encoding for modeling time intervals between two consecutive interactions. In addition, we showed the training process and formally defined the zero-shot inference procedure. In the next section, we present experiments to evaluate PrepRec .

5 EXPERIMENTS

We present extensive experiments on five real-world datasets to evaluate the performance of PrepRec, following the problem settings in § 3. We introduce the following research questions (RQ) to guide our experiments: (RQ1) How well can PrepRec perform on zero-shot cross-domain and cross-application transfer? (RQ2) Why should we model popularity dynamics in sequential recommendation? (RQ3) What affects the performance of PrepRec ?

5.1 Datasets and Preprocessing

We evaluate our proposed method on five real-world datasets across different applications, with varying sizes, and density levels.

Amazon [34] is a series of product ratings datasets obtained from Amazon.com, split by product categories. We consider the *Office* and *Tool* product domains in our study. **Douban** [44] consists of three datasets across different domains, collected from Douban.com, a Chinese review website. We work with the *Movie* and *Music* datasets.

Epinions [48, 49] is a dataset crawled from product review site Epinions. We utilize the ratings dataset for our study.

We present dataset statistics in Table 1. We compute the density as the ratio of the number of interactions to the number of users times the number of items. Douban datasets (*i.e.,* movie and music) are the densest and have no auxiliary information available, while the Amazon review datasets (*i.e.,* office and tool) are the sparsest.

For fair evaluation, we follow the same preprocessing procedure as previous works [22, 45], *i.e.,* we binarize the explicit ratings to implicit feedback. In addition, for each user, we sort interactions by their timestamp and use the second most recent action for validation, the most recent action for testing, and the rest for training.

5.2 Baselines and Experimental Setup

Baselines: Our baselines (supplementary materials contain detailed descriptions) include classic general recommendation models (*e.g.,* MostPop, BPR [38], NCF [15], LightGCN [14]) and state-of-the-art sequential recommendation models (*e.g.,* Caser [50], SasRec [22], BERT4Rec [45], TiSasRec [27], CL4SRec [59]).

Following previous works [15, 22, 24, 45], we adopt the *leave-one-out* evaluation method: for each user, we pair the test item with 100 unobserved items according to the user's interaction history. Then we rank the test item for the user among the 101 total items. We use two standard evaluation metrics for top-k recommendation: Recall@k (R@k) and Normalized Discounted Cumulative Gain@k (N@k). Our model explicitly utilizes popularity information. Therefore, we also present results where we sample the negatives based on their popularities, *i.e.,* popular items have higher probabilities of being sampled as negatives. We report the average of R@k and N@k over all the test interactions.

We use publicly available implementations for the baselines. For fair evaluation, we set dimension size d to 50, max sequence length L to 200, and batch size to 128 in all models. We use an Adam optimizer and tune the learning rate in the range $\{10^{-4}, 10^{-3}, 10^{-2}\}$ and set the weight decay to 10^{-5}. We use the dropout regularization rate of 0.3 for all models. We set $\gamma = 0.5$ in Equation (1), whose reason we will discuss the reason in supplement materials. We define the coarse and fine period to be 10 and 2 days respectively, and we fix the window size to be $m = 12$ and $n = 4$ for all datasets (§ 4.1.2). We train PrepRec for a maximum of 80 epochs. All experiments are conducted on a Tesla V100 using PyTorch. We repeat each

Dataset	Office		Tool		Movie		Music		Epinions	
Metric	R@10	N@10	R@10	N@10	R@10	N@10	R@10	N@10	R@10	N@10
General Recommender Systems										
MostPop	0.450	0.272	0.459	0.274	0.586	0.361	0.519	0.327	0.438	0.296
BPR [38]	0.457	0.289	0.363	0.216	0.747	0.477	0.646	0.434	0.568	0.397
NCF [15]	0.446	0.266	0.388	0.239	0.784	0.505	0.652	0.437	0.570	0.396
LightGCN [14]	0.465	0.293	0.463	0.275	0.793	0.512	0.665	0.447	0.575	0.396
Sequential Recommender Systems										
Caser [50]	0.512	0.334	0.496	0.297	0.891	0.701	0.796	0.576	0.674	0.475
SasRec [22]	0.539	0.354	0.536	0.337	0.918	0.749	0.816	0.599	0.705*	0.501
BERT4Rec [45]	0.541	0.358	0.544	0.350	0.900	0.728	0.816*	0.602*	0.702	0.512*
TiSasRec [27]	0.531	0.349	0.539	0.341	0.918*	0.752*	0.809	0.523	0.701	0.499
CL4SRec [59]	0.550*	0.358*	0.548*	0.352*	0.899	0.725	0.813	0.597	0.662	0.481
PrepRec	0.536	0.344	0.551	0.359	0.908	0.738	0.782	0.573	0.795	0.580
PrepRec Δ	-2.5%	-3.9%	+1.2%	+2.0%	-1.1%	-1.8%	-1.9%	-4.8%	+12.7%	+13.3%
Interp	**0.648**	**0.483**	**0.659**	**0.482**	**0.929**	**0.769**	**0.851**	**0.653**	**0.816**	**0.640**
Interp Δ	+17.8%	+34.9%	+20.3%	+35.0%	+1.1%	+2.3%	+4.3%	+8.5%	+15.7%	+25.0%

Table 3: *Regular sequential recommendation results, RQ2, (§ 5.4.1).* We make bold the best results and mark the best baseline results with '∗'. Interp represents the interpolation results between PrepRec and BERT4Rec. PrepRec Δ denotes the performance difference between PrepRec and the best results among the selected baselines, similar for Interp Δ. PrepRec achieves comparable performance to the state-of-the-art sequential recommenders, with only on average 0.2% worse than the best performing sequential recommenders in R@10 while having only a fraction of the model size (Table 5). After a simple post-hoc interpolation, we outperform the state-of-the-art sequential recommenders by 11.8% in R@10 on average.

experiment 5 times with different random seeds and report the average performance.

5.3 Zero-shot Transferability (RQ1)

5.3.1 Zero-shot Transfer Results. We follow the zero-shot inference setting introduced in § 4.3 and report the results in Table 2. We also include the results of PrepRec and the best-performing sequential recommenders trained on the target dataset for reference. In the zero-shot setting, PrepRec shows minimal performance reduction in the target datasets (*i.e.,* 6% maximum and 2% average reduction in R@10). The best zero-shot transfer results from PrepRec only fall short against the selected sequential recommendation baselines by up to 4% and even outperform (by up to 6.5%) them on the Epinions and Office. We found that PrepRec trained on douban-movie and amazon-tools show the highest generalizability, even outperforming the target-trained models on Music (0.811 *vs.* 0.782 on R@10). We conjecture that this is because Movie is the largest dataset in terms of the number of interactions. Overall, these results show PrepRec 's effectiveness in zero-shot transfer without any training on interaction data or side information. In addition, this experiment

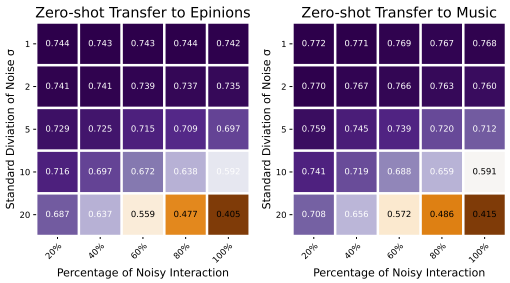

Figure 3: Zero-shot Transfer Results (R@10) with Gaussian noise added to the item popularity statistics (§ 5.3.2). We find that PrepRec is relatively robust to noise.

also demonstrates that the popularity dynamics-based item and sequence representations are generalizable across domains.

5.3.2 Robustness to Noise. We further investigate the robustness of PrepRec to possible noise in zero-shot transfer by adding Gaussian noise $\sim \mathcal{N}(0, \sigma)$ to the item popularity statistics and evaluate the zero-shot transfer performance on Douban-Music and Epinions from Douban-Movie. We randomly choose some percentage of items in the sequence to add noise, as indicated in Figure 3. We find that PrepRec is relatively robust to noise, maintaining robust performance across different noise levels at 20% noised interaction. We attribute this to the model's ability to learn from the overall popularity dynamics, which is less affected by noise in individual item popularity statistics. In addition, when the noise level is relatively low, *e.g.,* $\sigma \leq 5$, even if 100% of the sequence is noised, PrepRec still holds the performance, indicating significant item popularity shifts exist in the sequence (Figure 1).

Dataset	Music		Office		Epinions	
Metric	R@10	N@10	R@10	N@10	R@10	N@10
MostPop	0.197	0.139	0.099	0.046	0.163	0.110
SasRec [22]	**0.749**	0.519	0.453	0.291	0.658	0.442
BERT4Rec [45]	0.747	0.519	**0.461**	**0.299**	0.655	0.456
PrepRec	0.739	**0.523**	0.443	0.280	**0.762**	**0.551**
PrepRec Δ	-1.3%	+0.7%	-2.2%	-6.3%	+15.8%	+17.2%

Table 4: *Regular sequential recommendation results (§ 5.4.1) with popularity-based negative sampling.* PrepRec can learn discriminative item and sequence representations despite depending only on popularity statistics.

5.4 Why Popularity Dynamics? (RQ2)

PrepRec shows excellent performance in zero-shot sequential recommendation. Therefore, we ask: what is the role of popularity

Dataset	Office	Tool	Movie	Music	Epinions
SasRec	1.331M	3.581M	2.044M	0.542M	1.054M
BERT4Rec	2.687M	7.233M	4.126M	1.094M	2.127M
TiSasRec	1.367M	3.617M	2.127M	0.578M	1.09M
PrepRec	0.045M	0.045M	0.045M	0.045M	0.045M

Table 5: Comparison of model sizes (*i.e.,* number of learnable parameters in millions) over different datasets. PrepRec is 12 to 90x smaller.

dynamics in sequential recommendation, and how much does it explain the performance of state-of-the-art sequential recommenders? Therefore, we propose the following experiments to investigate the importance of popularity dynamics in sequential recommendation.

5.4.1 Regular Sequential Recommendation (RQ2). We show comparisons of PrepRec against state-of-the-art sequential recommenders in the regular sequential recommendation tasks (Table 3), *i.e.,* all models are trained from scratch. PrepRec achieves competitive performance—within 2% in R@10 and 5% in N@10, with the state-of-the-art baselines. On Epinions, PrepRec even outperforms all baselines by 7.3%, particularly impressive since PrepRec has significantly fewer model parameters (Table 5).

PrepRec explicitly models popularity information and the MostPop demonstrates decent performance compared to the remaining baselines, thus we conduct an additional experiment (Table 4) where we sample the unobserved (negative) items based on their popularity [45]. As shown in Table 4, MostPop's performance dropped significantly, while PrepRec shows even more competitive performance on some datasets (*e.g.,* Music and Epinions). This suggests PrepRec learns discriminative item and sequence representations.

PrepRec learns item representations through popularity dynamics, which is conceptually different from learning representations specific to each item ID. Therefore, we propose a simple post-hoc interpolation to investigate how much can popularity dynamics explain the performance of state-of-the-art sequential recommenders. We interpolate the scores from PrepRec with the scores from BERT4Rec as follows: $\hat{y}_{intp}(v_j^t|\mathcal{S}_u) = \alpha * \hat{y}_O(v_j^t|\mathcal{S}_u) + (1-\alpha) * \hat{y}_S(v_j|\mathcal{S}_u)$, where $\hat{y}_O(v_j^t|\mathcal{S}_u)$ and $\hat{y}_S(v_j|\mathcal{S}_u)$ are the scores from PrepRec (Equation (10)) and BERT4Rec, respectively. We set $\alpha = 0.5$ for all datasets. After interpolation, the performance significantly boosts by up to 34.9% in N@10. Gains are largest in the medium and low-density datasets (Epinions, Amazon), indicating that our model complements existing methods in sparse datasets where item embeddings are less informative. Therefore, it is crucial to consider popularity dynamics to maximize performance.

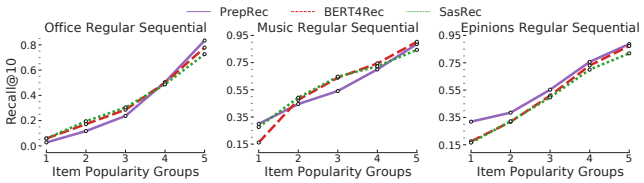

Figure 4: Recommendation results for different item popularity groups (§ 5.4.2), where Group 1 represents the least popular items, and Group 5 represents the most popular items. PrepRec achieves better performance on long-tail items while having competitive performance on popular items.

5.4.2 Qualitative Analysis on Regular Sequential Recommendation. We analyze the performance of PrepRec in detail. We separate test items into equally sized groups based on their popularity in the training set then compute the average R@10 and N@10 for each group (Figure 4). PrepRec achieves better performance on item group with the least interactions, *i.e.,* long-tail items, while the SasRec and BERT4Rec show stronger performance on popular items. Long-tail item recommendation is a particularly challenging task explored by many previous works [41] and requires recommenders able to learn high-quality representations with just a few interactions. This corresponds to our observation that PrepRec is more robust to data sparsity and can learn discriminative item and sequence representations (§ 5.4.1), showing that long-tail item recommendation can benefit from PrepRec 's popularity dynamics-based item representations.

5.5 What affects PrepRec performance? (RQ3)

Dataset	Music		Office		Epinions	
Metric	R@10	N@10	R@10	N@10	R@10	N@10
PrepRec	0.782	0.573	0.536	**0.344**	**0.795**	**0.580**
w/o Relative Time $\boldsymbol{T}$ (§ 4.1.3)	0.734	0.514	**0.541**	0.334	0.782	0.562
w/o Positional $\boldsymbol{P}$ (§ 4.1.4)	0.765	0.544	0.530	0.332	0.772	0.554
w/o Popularity Dynamics $\mathcal{P}$	**0.800**	**0.594**	0.530	0.341	0.761	0.560
w/o Popularity Dynamics $\mathcal{H}$	0.705	0.582	0.525	0.337	0.730	0.533
Sinuoisal Popularity Encoding	0.779	0.570	0.529	0.340	0.772	0.561

Table 6: Ablation study of PrepRec 's different variants.

5.5.1 Ablation Study. Here, we assess the importance of different components crucial to PrepRec , *i.e.,* relative time encoding (§ 4.1.3), positional encoding (§ 4.1.4), popularity encoder E_p (§ 4.1.1), and resolutions for popularity dynamics (§ 4.1.2). We find that removing relative time encoding $\boldsymbol{T}$ results in the largest performance drop on both the Music and Office datasets. This suggests that the relative time encoding is crucial for effectively capturing the popularity dynamics. Removing positional encoding $\boldsymbol{P}$ results in a maximum of 2.2% drop in R@10 on the Office dataset, indicating positional encoding is important for capturing sequential information. In addition, changing E_p to the non-linear sunusoidal encoding shows worse performance on all datasets, meaning that the linear encoding is more suitable for capturing the popularity dynamics. On the music dataset, removing coarse popularity encoding $\mathcal{P}$ results improves the performance by 2% in R@10, while removing fine popularity encoding $\mathcal{H}$ results in a 7.5% drop in R@10. This suggests that the music domain is more sensitive to recent trends in popularity. Coarse and fine popularity encodings complement each other on other datasets.

5.5.2 Effect of discounting factor γ. We examine the effect of different preprocessing weights γ used in popularity calculation (§ 4.1.1). In particular, $\gamma = 1$ corresponds to the cumulative popularity, or in other words, at a given time period t, the overall number of interactions up to period t. On the other hand, $\gamma = 0$ corresponds to the current popularity, or percentiles are calculated over interactions just in t, same as b_j^t in Equation (1). When $\gamma = 0.5$, it can be interpreted as interactions being exponentially weighted by time, with a half-life of 1 time period. We find that $\gamma = 0.5$ outperforms the other two settings, with the largest gains of around 12% R@10

Dataset	Music		Office		Epinions	
Metric	R@10	N@10	R@10	N@10	R@10	N@10
$\gamma = 0$ (Curr-pop)	0.749	0.542	0.512	0.328	0.689	0.496
$\gamma = 0.25$	0.764	0.529	0.538	0.338	0.761	0.562
$\gamma = 0.5^*$ (weighted -pop)	**0.782**	**0.573**	**0.536**	**0.344**	**0.795**	**0.580**
$\gamma = 0.75$	0.755	0.520	0.543	0.336	0.747	0.519
$\gamma = 1$ (cumul-pop)	0.695	0.452	0.530	0.330	0.733	0.505

Table 7: Recommendation results for varying the discounting factor γ in § 4.1.2. $\gamma = 0.5$ is the default setting, denoted by $'*'$. We find that $\gamma = 0.5$ generally outperforms the other two settings

and 27% N@10 over cumul-pop in the dense Music dataset, and the largest gains over curr-pop in the sparser Office (5% N@10 and 4% N@10) and Epinions (15% R@10 and 17%N@10) datasets. We suspect this is due to cumulative measures in denser datasets failing to capture recent trends due to the large historical presence, while current-only measures in sparser datasets convey too little or noisy information and lose the information of long-term trends. curr-pop shows decent performance on the Music dataset, suggesting that Music trends might be more cyclical and thus the current popularity is more informative.

Dataset	Music		Office		Epinions	
Metric	R@10	N@10	R@10	N@10	R@10	N@10
Fine:2 days; Coarse:10 days *	**0.782**	**0.573**	0.536	**0.344**	**0.795**	**0.580**
Fine:4 days; Coarse:15 days	0.778	0.553	**0.537**	0.341	0.790	0.574
Fine:7 days; Coarse:30 days	0.760	0.509	0.526	0.334	0.757	0.543

Table 8: Recommendation results for varying time horizons. Fine and coarse time horizons are used for short-term and long-term popularity dynamics respectively (§ 4.1.1).

5.5.3 Effect of Different Time Horizons. We study the effect of different time horizons to PrepRec . We found that in general, long-term horizons of 30 days and short-term horizons of 7 days perform worse than the other settings. This is likely because the long-term horizon might lead to the lack of resolutions in popularity statistics. We also find that depending on the dataset, the effect of different time horizons also varies. For example, both Music and Epinions show larger performance decrease from short to long-term horizons than Office. This could be because Music and Epinions are more sensitive to recent trends than Office, or their data are denser in terms of time granularity.

5.6 Fine-tune Capability

Dataset	Movie→Music		Tool→Office		Tool→Epinions	
Metric	R@10	N@10	R@10	N@10	R@10	N@10
PrepRec	0.803	0.591	**0.472**	**0.300**	**0.489**	**0.264**
SasRec	0.815	0.599	0.437	0.290	0.433	0.245
BERT4Rec	**0.816**	**0.602**	0.407	0.249	0.433	0.255

Table 9: Recommendation results for fine-tuning PrepRec. We fine-tune PrepRec and retrain the baselines from scratch on the target dataset.

We also investigate PrepRec's fine-tune capability. To ensure target datasets are smaller than the source, we further process the target datasets such that they are no more than 10% of the source datasets' total interactions. After further processing, we follow the same experimental setup in § 5.2. We fine-tune PrepRec and retrain the baselines from scratch on the target dataset and report the results in Table 9. We find that PrepRec , after fine-tuning, outperforms the selected baselines on Office and Epinions by up to 12.9%, indicating that PrepRec is capable of learning from the limited data and can be further fine-tuned to achieve better performance.

5.7 Discussion

PrepRec demonstrates the strong ability for zero-shot transfer. We argue that PrepRec is particularly useful in the following scenarios: 1) initial sequential model when the data in the domain is sparse; 2) backbone for developing more complex sequential recommenders (*i.e.,* prediction interpolation) 3) online recommendation settings.

PrepRec captures the popularity shifts in the sequence and is complementary to state-of-the-art sequential recommenders. It is worth noting that item popularity dynamics might not capture everything in users' preferences, but we believe they are orthogonal components towards capturing user preferences, which could explain why the interpolation results substantially outperform both PrepRec and the selected state-of-the-art baselines (Table 3).

Additionally, time granularity is also crucial for popularity dynamics, and sequence analysis requires careful consideration of the time horizon. Intuitively, when the dataset time precision is less accurate, i.e., weeks or days, we expect the performance to decrease as the sequential information and popularity dynamics become muddled. If the time precision in the training data increases, we can expect more accurate user sequences and more accurate measures of popularity dynamics. In general, time precision will not significantly impact the performance of PrepRec in most scenarios as in practice, online platforms can record precise time data for each user-item interaction. We will include more discussion in the arXiv version of this paper [55].

6 CONCLUSION

In this paper, using the critical insight of popularity dynamics in the user's sequence, we developed a novel pre-trained sequential recommendation framework, PrepRec, for the zero-shot, cross-domain setting without any auxiliary information. PrepRec learned transferable, universal item representations via popularity dynamics-aware transformers. We empirically showed that PrepRec can achieve excellent zero-shot transfer to a target domain, comparable to state-of-the-art sequential recommenders trained on the target domain. With extensive within-domain experiments, we found performance gains of 11.8% when we interpolated PrepRec 's results with state-of-the-art sequential recommenders, indicating that PrepRec is learning complementary information. We posit that popularity dynamics are crucial for developing generalizable sequential recommenders.

As part of future work, we plan to investigate: 1) developing more complex sequential recommenders by using PrepRec as a backbone (*i.e.,* prediction interpolation and auxiliary information), and 2) exploring online recommendation settings.

ACKNOWLEDGMENTS

This work was generously supported by the National Science Foundation (NSF) under grant number 2312561. We also would like to thank the anonymous reviewers for their valuable feedback.

REFERENCES

[1] Alejandro Ariza-Casabona, Bartlomiej Twardowski, and Tri Kurniawan Wijaya. 2023. Exploiting graph structured cross-domain representation for multi-domain recommendation. In *European Conference on Information Retrieval*. Springer, 49–65.

[2] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450* (2016).

[3] Albert-Laszlo Barabasi. 2005. The origin of bursts and heavy tails in human dynamics. *Nature* 435, 7039 (2005), 207–211.

[4] Albert-László Barabási and Réka Albert. 1999. Emergence of Scaling in Random Networks. *Science* 286, 5439 (1999), 509. http://search.ebscohost.com/login.aspx?direct=true&db=tfh&AN=2405932&site=ehost-live

[5] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems* 33 (2020), 1877–1901.

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina N. Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. https://arxiv.org/abs/1810.04805

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

[8] Manqing Dong, Feng Yuan, Lina Yao, Xiwei Xu, and Liming Zhu. 2020. Mamo: Memory-augmented meta-optimization for cold-start recommendation. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*. 688–697.

[9] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929* (2020).

[10] Xiaoyu Du, Xiang Wang, Xiangnan He, Zechao Li, Jinhui Tang, and Tat-Seng Chua. 2020. How to learn item representation for cold-start multimedia recommendation?. In *Proceedings of the 28th ACM International Conference on Multimedia*. 3469–3477.

[11] Philip J Feng, Pingjun Pan, Tingting Zhou, Hongxiang Chen, and Chuanjiang Luo. 2021. Zero shot on the cold-start problem: Model-agnostic interest learning for recommender systems. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 474–483.

[12] Bowen Hao, Jing Zhang, Hongzhi Yin, Cuiping Li, and Hong Chen. 2021. Pre-training graph neural networks for cold-start users and items representation. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*. 265–273.

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

[14] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation. *CoRR* abs/2002.02126 (2020). arXiv:2002.02126 https://arxiv.org/abs/2002.02126

[15] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *WWW*. 173–182.

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

[17] Balázs Hidasi, Massimo Quadrana, Alexandros Karatzoglou, and Domonkos Tikk. 2016. Parallel recurrent neural network architectures for feature-rich session-based recommendations. In *Proceedings of the 10th ACM conference on recommender systems*. 241–248.

[18] Yupeng Hou, Zhankui He, Julian McAuley, and Wayne Xin Zhao. 2023. Learning vector-quantized item representation for transferable sequential recommenders. In *Proceedings of the ACM Web Conference 2023*. 1162–1171.

[19] Yupeng Hou, Shanlei Mu, Wayne Xin Zhao, Yaliang Li, Bolin Ding, and Ji-Rong Wen. 2022. Towards universal sequence representation learning for recommender systems. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 585–593.

[20] 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*. 667–676.

[21] Yitong Ji, Aixin Sun, Jie Zhang, and Chenliang Li. 2020. A re-visit of the popularity baseline in recommender systems. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1749–1752.

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

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

[24] Yehuda Koren. 2008. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. 426–434.

[25] Yehuda Koren. 2009. Collaborative filtering with temporal dynamics. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. 447–456.

[26] Hoyeop Lee, Jinbae Im, Seongwon Jang, Hyunsouk Cho, and Sehee Chung. 2019. MeLU: Meta-Learned User Preference Estimator for Cold-Start Recommendation. In *KDD*. 1073–1082.

[27] Jiacheng Li, Yujie Wang, and Julian McAuley. 2020. Time interval aware self-attention for sequential recommendation. In *Proceedings of the 13th international conference on web search and data mining*. 322–330.

[28] Pan Li and Alexander Tuzhilin. 2020. Ddtcdr: Deep dual transfer cross domain recommendation. In *Proceedings of the 13th International Conference on Web Search and Data Mining*. 331–339.

[29] Weiming Liu, Jiajie Su, Chaochao Chen, and Xiaolin Zheng. 2021. Leveraging distribution alignment via stein path for cross-domain cold-start recommendation. *Advances in Neural Information Processing Systems* 34 (2021), 19223–19234.

[30] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692* (2019).

[31] Yuanfu Lu, Yuan Fang, and Chuan Shi. 2020. Meta-learning on heterogeneous information networks for cold-start recommendation. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1563–1573.

[32] Fuyu Lv, Taiwei Jin, Changlong Yu, Fei Sun, Quan Lin, Keping Yang, and Wilfred Ng. 2019. SDM: Sequential deep matching model for online large-scale recommender system. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. 2635–2643.

[33] Tong Man, Huawei Shen, Xiaolong Jin, and Xueqi Cheng. 2017. Cross-domain recommendation: An embedding and mapping approach.. In *IJCAI*, Vol. 17. 2464–2470.

[34] Jianmo Ni, Jiacheng Li, and Julian McAuley. 2019. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*. 188–197.

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

[36] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. 2021. Learning transferable visual models from natural language supervision. In *International conference on machine learning*. PMLR, 8748–8763.

[37] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research* 21, 1 (2020), 5485–5551.

[38] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *UAI*. AUAI Press, 452–461.

[39] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2010. Factorizing personalized markov chains for next-basket recommendation. In *Proceedings of the 19th international conference on World wide web*. 811–820.

[40] Matthew J. Salganik, Peter Sheridan Dodds, and Duncan J. Watts. 2006. Experimental Study of Inequality and Unpredictability in an Artificial Cultural Market. *Science* 311, 5762 (2006), 854–856. http://www.jstor.org/stable/3843620

[41] Aravind Sankar, Junting Wang, Adit Krishnan, and Hari Sundaram. 2021. ProtoCF: Prototypical Collaborative Filtering for Few-Shot Recommendation. In *Proceedings of the 15th ACM Conference on Recommender Systems* (Amsterdam, Netherlands) *(RecSys '21)*. Association for Computing Machinery, New York, NY, USA, 166–175. https://doi.org/10.1145/3460231.3474268

[42] Guy Shani, David Heckerman, Ronen I Brafman, and Craig Boutilier. 2005. An MDP-based recommender system. *Journal of Machine Learning Research* 6, 9 (2005).

[43] Xiang-Rong Sheng, Liqin Zhao, Guorui Zhou, Xinyao Ding, Binding Dai, Qiang Luo, Siran Yang, Jingshan Lv, Chi Zhang, Hongbo Deng, et al. 2021. One model to serve all: Star topology adaptive recommender for multi-domain ctr prediction. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 4104–4113.

[44] Weiping Song, Zhiping Xiao, Yifan Wang, Laurent Charlin, Ming Zhang, and Jian Tang. 2019. Session-based social recommendation via dynamic graph attention networks. In *Proceedings of the Twelfth ACM international conference on web search and data mining*. 555–563.

[45] 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. In *Proceedings of the 28th ACM international conference on information and knowledge management*. 1441–1450.

[46] Qiaoyu Tan, Jianwei Zhang, Ninghao Liu, Xiao Huang, Hongxia Yang, Jingren Zhou, and Xia Hu. 2021. Dynamic memory based attention network for sequential recommendation. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 35. 4384–4392.

[47] Yong Kiam Tan, Xinxing Xu, and Yong Liu. 2016. Improved recurrent neural networks for session-based recommendations. In *Proceedings of the 1st workshop on deep learning for recommender systems*. 17–22.

[48] H. Tang, J.and Gao, H. Liu, and A. Das Sarma. 2012. eTrust: Understanding trust evolution in an online world. , 253–261 pages.

[49] J. Tang, H. Gao, and H. Liu. 2012. mTrust: Discerning multi-faceted trust in a connected world. In *Proceedings of the fifth ACM international conference on Web search and data mining*. ACM, 93–102.

[50] Jiaxi Tang and Ke Wang. 2018. Personalized top-n sequential recommendation via convolutional sequence embedding. In *Proceedings of the eleventh ACM international conference on web search and data mining*. 565–573.

[51] Trinh Xuan Tuan and Tu Minh Phuong. 2017. 3D convolutional networks for session-based recommendation with content features. In *Proceedings of the eleventh ACM conference on recommender systems*. 138–146.

[52] 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).

[53] Maksims Volkovs, Guangwei Yu, and Tomi Poutanen. 2017. Dropoutnet: Addressing cold start in recommender systems. In *Advances in neural information processing systems*. 4957–4966.

[54] Junting Wang, Adit Krishnan, Hari Sundaram, and Yunzhe Li. 2023. Pre-trained Neural Recommenders: A Transferable Zero-Shot Framework for Recommendation Systems. arXiv:2309.01188 [cs.IR]

[55] Junting Wang, Praneet Rathi, and Hari Sundaram. 2024. A Pre-trained Sequential Recommendation Framework: Popularity Dynamics for Zero-shot Transfer. arXiv:2401.01497 [cs.IR] https://arxiv.org/abs/2401.01497

[56] Jinpeng Wang, Ziyun Zeng, Yunxiao Wang, Yuting Wang, Xingyu Lu, Tianxiang Li, Jun Yuan, Rui Zhang, Hai-Tao Zheng, and Shu-Tao Xia. 2023. MISSRec: Pre-training and Transferring Multi-modal Interest-aware Sequence Representation for Recommendation. In *Proceedings of the 31st ACM International Conference on Multimedia* (, Ottawa ON, Canada,) *(MM '23)*. Association for Computing Machinery, New York, NY, USA, 6548–6557. https://doi.org/10.1145/3581783.3611967

[57] Yinwei Wei, Xiang Wang, Qi Li, Liqiang Nie, Yan Li, Xuanping Li, and Tat-Seng Chua. 2021. Contrastive learning for cold-start recommendation. In *Proceedings of the 29th ACM International Conference on Multimedia*. 5382–5390.

[58] Chao-Yuan Wu, Amr Ahmed, Alex Beutel, Alexander J Smola, and How Jing. 2017. Recurrent recommender networks. In *Proceedings of the tenth ACM international conference on web search and data mining*. 495–503.

[59] Xu Xie, Fei Sun, Zhaoyang Liu, Shiwen Wu, Jinyang Gao, Jiandong Zhang, Bolin Ding, and Bin Cui. 2022. Contrastive learning for sequential recommendation. In *2022 IEEE 38th international conference on data engineering (ICDE)*. IEEE, 1259–1273.

[60] Haochao Ying, Fuzhen Zhuang, Fuzheng Zhang, Yanchi Liu, Guandong Xu, Xing Xie, Hui Xiong, and Jian Wu. 2018. Sequential recommender system based on hierarchical attention network. In *IJCAI International Joint Conference on Artificial Intelligence*.

[61] Fajie Yuan, Xiangnan He, Alexandros Karatzoglou, and Liguang Zhang. 2020. Parameter-Efficient Transfer from Sequential Behaviors for User Modeling and Recommendation. *Proceedings of the 42nd international ACM SIGIR conference on Research and development in Information Retrieval* (2020).

[62] Chenyi Zhang, Ke Wang, Hongkun Yu, Jianling Sun, and Ee-Peng Lim. 2014. Latent factor transition for dynamic collaborative filtering. In *Proceedings of the 2014 SIAM international conference on data mining*. SIAM, 452–460.

[63] 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 *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 229–238.

[64] Yongchun Zhu, Kaikai Ge, Fuzhen Zhuang, Ruobing Xie, Dongbo Xi, Xu Zhang, Leyu Lin, and Qing He. 2021. Transfer-meta framework for cross-domain recommendation to cold-start users. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1813–1817.