NISER: Normalized Item and Session Representations to Handle Popularity Bias

Priyanka Gupta, Diksha Garg, Pankaj Malhotra, Lovekesh Vig, Gautam Shroff

priyanka.g35@tcs.com,diksha.7@tcs.com,malhotra.pankaj@tcs.com,lovekesh.vig@tcs.com,gautam.shroff@tcs.com

TCS Research, New Delhi, India

ABSTRACT

The goal of session-based recommendation (SR) models is to utilize the information from past actions (e.g. item/product clicks) in a session to recommend items that a user is likely to click next. Recently it has been shown that the sequence of item interactions in a session can be modeled as graph-structured data to better account for complex item transitions. Graph neural networks (GNNs) can learn useful representations for such session-graphs, and have been shown to improve over sequential models such as recurrent neural networks [14]. However, we note that these GNN-based recommendation models suffer from popularity bias: the models are biased towards recommending popular items, and fail to recommend relevant long-tail items (less popular or less frequent items). Therefore, these models perform poorly for the less popular new items arriving daily in a practical online setting. We demonstrate that this issue is, in part, related to the magnitude or norm of the learned item and session-graph representations (embedding vectors). We propose a training procedure that mitigates this issue by using normalized representations. The models using normalized item and session-graph representations perform significantly better: i. for the less popular long-tail items in the offline setting, and ii. for the less popular newly introduced items in the online setting. Furthermore, our approach significantly improves upon existing state-of-the-art on three benchmark datasets.

KEYWORDS

Session-based Recommendation; Graph Neural Networks; Item and Session Representations; Popularity Bias
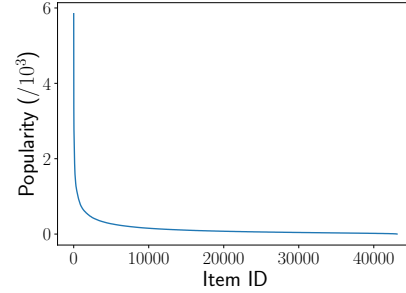
1 INTRODUCTION

The goal of session-based recommendation (SR) models is to recommend top-K items to a user based on the sequence of items clicked so far. Recently, several effective models for SR based on deep neural networks architectures have been proposed [6, 8, 13, 14]. These approaches consider SR as a multi-class classification problem where

Figure 1: Typical popularity distribution of items depicting the long tail.

input is a sequence of items clicked in the past in a session and target classes correspond to available items in the catalog. Many of these approaches use sequential models like recurrent neural networks considering a session as a sequence of item click events [4, 5, 13]. On the other hand, approaches like STAMP [8] consider a session to be a set of items, and use attention models while learning to weigh (attend to) items as per their relevance to predict the next item. Approaches like NARM [6] and CSRM [13] use a combination of sequential and attention models.

An important building block in most of these deep learning approaches is their ability to learn representations or embeddings for items and sessions. Recently, an alternative approach, namely SR-GNN [14], has been proposed to model the sessions as graph-structured data using GNNs [7] rather than as sequences or sets, noting that users tend to make complex *to-and-fro* transitions across items within a session: for example, consider a session $s = i_1, i_2, i_1, i_3, i_4$ of item clicks by a user. Here, the user clicks on item i_1, then clicks on item i_2 and then re-clicks on item i_1. This sequence of clicks induces a graph where nodes and edges correspond to items and transitions across items, respectively. For session s in the above example, the fact that i_2 and i_3 are neighbors of i_1 in the induced session-graph, the representation of i_1 can be updated using representations of its neighbors, i.e. i_2 and i_3, and thus obtain a more context-aware and informative representations. It is worth noting that this way of capturing neighborhood information has also been found to be effective in neighborhood-based SR methods such as SKNN [5] and STAN [3].

It is well-known that more popular items are presented and interacted-with more often on online platforms. This results in a skewed distribution of items clicked by users [1, 10, 15], as illustrated in Fig. 1. The models trained using the resulting data tend to have *popularity bias*, i.e. they tend to recommend more popular items over rarely clicked items.

We note that SR-GNN (referred to as GNN hereafter) also suffers from popularity bias. This problem is even more severe in a practical

online setting where new items are frequently added to the catalog, and are inherently less popular during initial days. To mitigate this problem, we study GNN through the lens of an *item and session-graph representation learning mechanism*, where the goal is to obtain a session-graph representation that is *similar* to the representation of the item likely to be clicked next. We motivate the advantage of restricting the item and session-graph representations to lie on a unit hypersphere both during training and inference, and propose **NISER**: **N**ormalized **I**tem and **Se**ssion **R**epresentations model for SR. We demonstrate the enhanced ability of NISER to deal with popularity bias in comparison to a vanilla GNN model in the offline as well as online settings. We also extend NISER to incorporate the sequential nature of a session via position embeddings [11], thereby leveraging the benefits of both sequence-aware models (like RNNs) and graph-aware models.

2 RELATED WORK

Recent results in computer vision literature, e.g. [12, 16], indicate the effectiveness of normalizing the final image features during training, and argue in favor of cosine similarity over inner product for learning and comparing feature vectors. [16] introduces the ring loss for soft feature normalization which eventually learns to constrain the feature vectors on a unit hypersphere. Normalizing words embeddings is also popular in NLP applications, e.g. [9] proposes penalizing the L_2 norm of word embeddings for regularization. However, to the best of our knowledge, the idea of normalizing item and session-graph embeddings or representations has not been explored.

In this work, we study the effect of normalizing the embeddings on popularity bias which has not been established and studied so far. Several approaches to deal with popularity bias exist in the collaborative filtering settings, e.g. [1, 10, 15]. To deal with popularity bias, [1] introduces the notion of flexible regularization in a learning-to-rank algorithm. Similarly [10, 15] uses the power-law of popularity where the probability of recommending an item is a smooth function of the items' popularity, controlled by an exponent factor. However, to the best of our knowledge, this is the first attempt to study and address popularity bias in DNN-based SR using SR-GNN [14] as a working example. Furthermore, SR-GNN does not incorporate the sequential information explicitly to obtain the session-graph representation. We study the effect of incorporating position embeddings [11] and show that it leads to minor but consistent improvement in recommendation performance.

3 PROBLEM DEFINITION

Let $\mathcal{S}$ denote all past sessions, and $\mathcal{I}$ denote the set of m items observed in the set $\mathcal{S}$. Any session $s \in \mathcal{S}$ is a chronologically ordered tuple of item-click events: $s = (i_{s,1}, i_{s,2}, \ldots, i_{s,l})$, where each of the l item-click events $i_{s,j}$ ($j = 1 \ldots l$) corresponds to an item in $\mathcal{I}$, and j denotes the position of the item $i_{s,j}$ in the session s. A session s can be modeled as a graph $\mathcal{G}_s = (\mathcal{V}_s, \mathcal{E}_s)$, where $i_{s,j} \in \mathcal{V}_s$ is a node in the graph. Further, $(i_{s,j}, i_{s,j+1}) \in \mathcal{E}_s$ is a directed edge from $i_{s,j}$ to $i_{s,j+1}$. Given s, the goal of SR is to predict the next item $i_{s,l+1}$ by estimating the m-dimensional probability vector $\hat{y}_{s,l+1}$ corresponding to the relevance scores for the m items. The K items with highest scores constitute the top-K recommendation list.

4 LEARNING ITEM AND SESSION REPRESENTATIONS

Each item is mapped to a d-dimensional vector from the trainable embedding look-up table or matrix $\mathbf{I} = [\mathbf{i}_1, \mathbf{i}_2, \ldots, \mathbf{i}_m]^T \in \mathbb{R}^{m \times d}$ such that each row $\mathbf{i}_j \in \mathbb{R}^d$ is the d-dimensional representation or embedding[1] vector corresponding to item $i_j \in \mathcal{I}$ ($j = 1 \ldots m$). Consider any function f (e.g. a neural network as in [8, 14])—parameterized by $\boldsymbol{\theta}$—that maps the items in a session s to session embedding $\mathbf{s} = f(\mathbf{I}_s; \boldsymbol{\theta})$, where[2] $\mathbf{I}_s = [\mathbf{i}_{s,1}, \mathbf{i}_{s,2}, \ldots, \mathbf{i}_{s,l}]^T \in \mathbb{R}^{l \times d}$. Along with $\mathbf{I}_s$ as an input which considers s as a sequence, we also introduce an adjacency matrix $\mathbf{A}_s$ to incorporate the graph structure. We discuss this in more detail later in Section 5.

The goal is to obtain $\mathbf{s}$ that is close to the embedding $\mathbf{i}_{s,l+1}$ of the target item $i_k = i_{s,l+1}$, such that the estimated index/class for the target item is $k = \arg\max_j \mathbf{i}_j^T \mathbf{s}$ with $j = 1 \ldots m$. In a DNN-based model f, this is approximated via a differentiable softmax function such that the probability of next item being i_k is given by:

$$p_k(\mathbf{s}) = \hat{y}_k = \frac{\exp(\mathbf{i}_k^T \mathbf{s})}{\sum_{j=1}^{m} \exp(\mathbf{i}_j^T \mathbf{s})}. \tag{1}$$

For this m-way classification task, softmax (cross-entropy) loss is used during training for estimating $\boldsymbol{\theta}$ by minimizing the sum of $\mathcal{L}(\hat{y}) = -\sum_{j=1}^{m} \mathbf{y}_j \log(\hat{y}_j)$ over all training samples, where $\mathbf{y} \in \{0, 1\}^m$ is a 1-hot vector with $\mathbf{y}_k = 1$ corresponding to the correct (target) class k.

We next introduce the radial property [12, 16] of softmax loss, and then use it to motivate the need for normalizing the item and session representations in order to reduce popularity bias.

4.1 Radial Property of Softmax Loss

It is well-known that optimizing for the softmax loss leads to a radial distribution of features for a target class [12, 16]: If $k = \arg\max_j \mathbf{i}_j^T \mathbf{s}$, then it is easy to show that

$$\frac{\exp(\sigma \mathbf{i}_k^T \mathbf{s})}{\sum_{j \neq k} \exp(\mathbf{i}_j^T \mathbf{s}) + \exp(\sigma \mathbf{i}_k^T \mathbf{s})} > \frac{\exp(\mathbf{i}_k^T \mathbf{s})}{\sum_{j=1}^{m} \exp(\mathbf{i}_j^T \mathbf{s})} \tag{2}$$

for any $\sigma > 1$. This, in turn, implies that softmax loss favors large norm of features for easily classifiable instances. We omit details for brevity and refer the reader to [12, 16] for a thorough analysis and proof. This means that a high value of $\hat{y}_k$ can be attained by multiplying vector $\mathbf{i}_k$ by a scalar value $\sigma > 1$; or simply by ensuring a large norm for the item embedding vector.

4.2 Normalizing the Representations

We note that the radial property has an interesting implication in the context of popularity bias: target items that are easier to predict are likely to have higher L_2 norm. We illustrate this with the help of an example: Items that are popular are likely to be clicked more often, and hence the trained parameters $\boldsymbol{\theta}$ and $\mathbf{I}$ should have values that ensure these items get recommended more often. The radial property implies that for a given input s and a popular target item i_k, a correct classification decision can be obtained as follows: learn the

---

[1]We use the terms representation and embedding interchangeably.
[2]To ensure same dimensions of $\mathbf{I}_s \in \mathbb{R}^{L \times d}$ across sessions, we can pad with a dummy vector $L - l$ times.
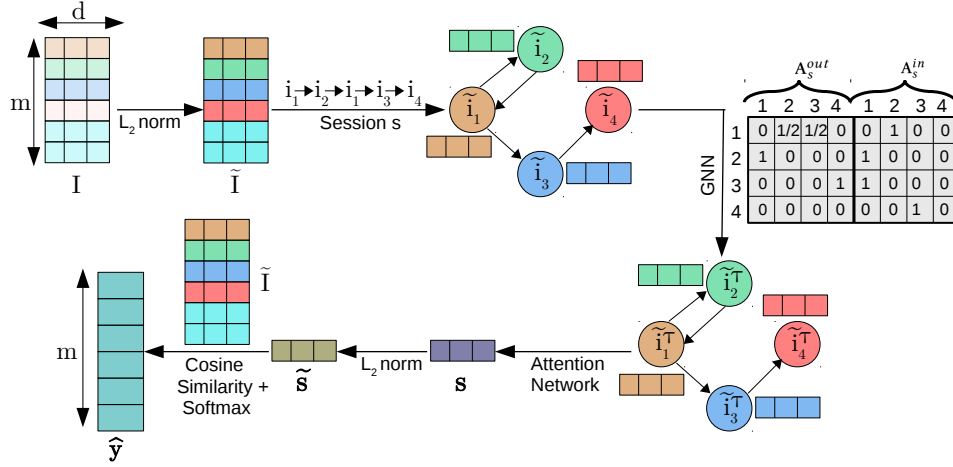
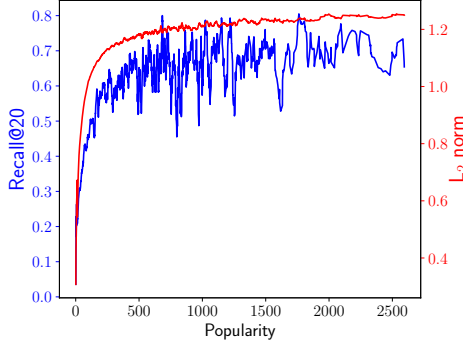Figure 2: Illustrative flow diagram for NISER.

Figure 3: Recall@20 and L_2 norm of learned item embeddings decreases with decreasing popularity in GNN [14].

embedding vector $\mathbf{i}_k$ with high $||\mathbf{i}_k||_2$ such that the inner product $\mathbf{i}_k^T\mathbf{s} = ||\mathbf{i}_k||_2||\mathbf{s}||_2 cos\alpha$ (where α is the angle between the item and session embeddings) is likely to be high to ensure large value for $\hat{\mathbf{y}}_k$ (even when α is not small enough and $||s||_2$ is not large enough). When analyzing the item embeddings from a GNN model [14], we observe that this is indeed the case: As shown in Fig. 3, items with high popularity have high L_2 norm while less popular items have significantly lower L_2 norm. Further, performance of GNN (depicted in terms of Recall@20) degrades as the popularity of the target item decreases.

4.3 NISER

Based on the above observations, we consider to minimize the influence of embedding norms in the final classification and recommendation decision. We propose optimizing for cosine similarity as a measure of similarity between item and session embeddings instead of the above-stated inner product. Therefore, during training as well as inference, we normalize the item embeddings as $\tilde{\mathbf{i}}_k = \frac{\mathbf{i}_k}{||\mathbf{i}_k||_2}$, and use them to get $\tilde{\mathbf{I}}_s$. The session embedding is then obtained as $\mathbf{s} = f(\tilde{\mathbf{I}}_s; \boldsymbol{\theta})$, and is similarly normalized to $\tilde{\mathbf{s}}$ to enforce a unit norm. The normalized item and session embeddings are then

used to obtain the relevance score for next clicked item i_k computed as,

$$\hat{\mathbf{y}}_k = \frac{\exp(\sigma\tilde{\mathbf{i}}_k^T\tilde{\mathbf{s}})}{\sum_{j=1}^{m}\exp(\sigma\tilde{\mathbf{i}}_j^T\tilde{\mathbf{s}})}. \tag{3}$$

Note that the cosine similarity $\tilde{\mathbf{i}}_k^T\tilde{\mathbf{s}}$ is restricted to $[-1, 1]$. As shown in [12], this implies that the softmax loss is likely to get saturated at high values for the training set: a scaling factor $\sigma > 1$ is useful in practice to allow for better convergence.

5 LEVERAGING NISER WITH GNN

We consider learning the representations of items and session-graphs with GNNs where the session-graph is represented by $\mathcal{G}_s$ as introduced in Section 3. Consider two normalized adjacency matrices $\mathbf{A}_s^{in} \in \mathbb{R}^{l,l}$ and $\mathbf{A}_s^{out} \in \mathbb{R}^{l,l}$ corresponding to the incoming and outgoing edges in graph $\mathcal{G}_s$ as illustrated in Fig. 2. GNN takes adjacency matrices $\mathbf{A}_s^{in}$ and $\mathbf{A}_s^{out}$, and the normalized item embeddings $\tilde{\mathbf{I}}_s$ as input, and returns an updated set of embeddings after τ iterations of message propagation across vertices in the graph using gated recurrent units [7]: $[\tilde{\mathbf{i}}_{s,1}^\tau, \tilde{\mathbf{i}}_{s,2}^\tau, \ldots, \tilde{\mathbf{i}}_{s,l}^\tau] = G(\mathbf{A}_s^{in}, \mathbf{A}_s^{out}, \tilde{\mathbf{I}}_s; \boldsymbol{\theta}_g)$, where $\boldsymbol{\theta}_g$ represents the parameters of the GNN function G. For any node in the graph, the current representation of the node and the representations of its neighboring nodes are used to iteratively update the representation of the node τ times. More specifically, the representation of node $i_{s,j}$ in the t-th message propagation step is updated as follows:

$$\mathbf{a}_{s,j}^t = [\mathbf{A}_{s,j:}^{in}\tilde{\mathbf{I}}_s^{t-1}\mathbf{H}_1, \mathbf{A}_{s,j:}^{out}\tilde{\mathbf{I}}_s^{t-1}\mathbf{H}_2]^T + \mathbf{b}, \tag{4}$$

$$\mathbf{z}_{s,j}^t = \sigma(\mathbf{W}_z\mathbf{a}_{s,j}^t + \mathbf{U}_z\tilde{\mathbf{i}}_{s,j}^{t-1}), \tag{5}$$

$$\mathbf{r}_{s,j}^t = \sigma(\mathbf{W}_r\mathbf{a}_{s,j}^t + \mathbf{U}_r\tilde{\mathbf{i}}_{s,j}^{t-1}), \tag{6}$$

$$\hat{\mathbf{i}}_{s,j}^t = \tanh(\mathbf{W}_o\mathbf{a}_{s,j}^t + \mathbf{U}_o(\mathbf{r}_{s,j}^t \odot \tilde{\mathbf{i}}_{s,j}^{t-1})), \tag{7}$$

$$\tilde{\mathbf{i}}_{s,j}^t = (1 - \mathbf{z}_{s,j}^t) \odot \tilde{\mathbf{i}}_{s,j}^{t-1} + \mathbf{z}_{s,j}^t \odot \hat{\mathbf{i}}_{s,j}^t, \tag{8}$$

where $\mathbf{A}_{s,j:}^{in}, \mathbf{A}_{s,j:}^{out} \in \mathbb{R}^{1\times l}$ depicts the j-th row of $\mathbf{A}_s^{in}$ and $\mathbf{A}_s^{out}$ respectively, $\mathbf{H}_1, \mathbf{H}_2 \in \mathbb{R}^{d\times d}$, $\mathbf{W}_{(.)}$ and $\mathbf{U}_{(.)}$ are trainable parameters,

$\sigma(.)$ is the sigmoid function, and $\odot$ is the element-wise multiplication operator.

To incorporate sequential information of item interactions, we optionally learn *position embeddings* and add them to item embeddings to effectively obtain position-aware item (and subsequently session) embeddings. The final embeddings for items in a session are computed as $\tilde{\mathbf{i}}_{s,j}^{\tau,p} = \tilde{\mathbf{i}}_{s,j}^{\tau} + \mathbf{p}_j$, where $\mathbf{p}_j \in \mathbb{R}^d$ is embedding vector for position j obtained via a lookup over the position embeddings matrix $\mathbf{P} = [\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_L]^T \in \mathbb{R}^{L \times d}$, where L denotes the maximum length of any input session such that position $l \leq L$.

The soft-attention weight of the j-th item in session s is computed as $\alpha_j = \mathbf{q}^T \text{sigmoid}(\mathbf{W}_1 \tilde{\mathbf{i}}_{s,l}^{\tau,p} + \mathbf{W}_2 \tilde{\mathbf{i}}_{s,j}^{\tau,p} + \mathbf{c})$, where $\mathbf{q}, \mathbf{c} \in \mathbb{R}^d$, $\mathbf{W}_1, \mathbf{W}_2 \in \mathbb{R}^{d \times d}$. The α_js are further normalized using softmax. An intermediate session embedding $\mathbf{s}'$ is computed as: $\mathbf{s}' = \sum_{j=1}^{t} \alpha_j \tilde{\mathbf{i}}_{s,j}^{\tau,p}$. The session embedding $\mathbf{s}$ is a linear transformation over the concatenation of intermediate session embedding $\mathbf{s}'$ and the embedding of most recent item $\tilde{\mathbf{i}}_{s,l}^{\tau,p}$, s.t. $\mathbf{s} = \mathbf{W}_3[\mathbf{s}'; \tilde{\mathbf{i}}_{s,l}^{\tau,p}]$, where $\mathbf{W}_3 \in \mathbb{R}^{d \times 2d}$.

The final recommendation scores for the m items are computed as per Eq. 3. Note that while the session-graph embedding is obtained using item embeddings $\tilde{\mathbf{i}}_{s,t}^{\tau,p}$ that are aware of the session-graph and sequence, the normalized item embeddings $\tilde{\mathbf{i}}_j$ ($j = 1 \ldots m$) independent of a particular session are used to compute the recommendation scores.

6 EXPERIMENTAL EVALUATION

Dataset Details: We evaluate NISER on publicly available benchmark datasets: i) Yoochoose (YC), ii) Diginetica (DN), and iii) RetailRocket (RR). The YC[3] dataset is from RecSys Challenge 2015, which contains a stream of user clicks on an e-commerce website within six months. Given the large number of sessions in YC, the recent 1/4 and 1/64 fractions of the training set are used to form two datasets: YC-1/4 and YC-1/64, respectively, as done in [14]. The DN[4] dataset is transactional data from CIKM Cup 2016 challenge. The RR[5] dataset is from an e-commerce personalization company retailrocket, which published dataset with six month of user browsing activities.

Offline and Online setting: We consider two evaluation settings: i. offline and ii. online. For evaluation in offline setting, we consider static splits of train and test as used in [14] for YC and DN. For RR, we consider sessions from last 14 days for testing and remaining 166 days for training. The statistics of datasets are summarized in Table 1. For evaluation in online setting, we re-train the models every day for 2 weeks (number of sessions per day for YC is much larger, so we evaluate for 1 week due to computational constraints) by appending the sessions from that day to the previous train set, and report the test results of the trained model on sessions from the subsequent day.

NISER and its variants: We apply our approach over GNN and adapt code[6] from [14] with suitable modification described later. We found that for sessions with length $l > 10$, considering only the most recently clicked 10 items to make recommendations worked

[3] http://2015.recsyschallenge.com/challge.html
[4] http://cikm2016.cs.iupui.edu/cikm-cup
[5] https://www.dropbox.com/sh/dbzmtq4zhzbj5o9/AACldzQWbw-igKjcPTBI6ZPAa?dl=0
[6] https://github.com/CRIPAC-DIG/SR-GNN

Table 1: Statistics of the datasets used for offline experiments.

Statistics	DN	RR	YC-1/64	YC-1/4
#train sessions	0.7 M	0.7 M	0.4 M	0.6 M
#test sessions	60,858	60,594	55,898	55,898
#items	43,097	48,759	16,766	29,618
Average length	5.12	3.55	6.16	5.17

consistently better across datasets. We refer to this variant as GNN+ and use this additional pre-processing step in all our experiments. We consider enhancement over GNN+, and proposed following variants of the embedding normalization approach:

- *Normalized Item Representations (NIR)*: only item embeddings are normalized and scale factor σ is not used[7],
- *Normalized Item and Session Representations (NISER)*: both item and session embeddings are normalized,
- *NISER+*: NISER with position embeddings and dropout applied to input item embeddings.

Hyperparameter Setup: Following [14], we use $d = 100$ and learning rate of 0.001 with Adam optimizer. We use 10% of train set which is closer to test test in time as hold-out validation set for hyperparameter tuning including scale factor σ. We found $\sigma = 16.0$ to work best across most models trained on respective hold-out validation set chosen from $\{4.0, 9.0, 16.0, 25.0\}$, and hence, we use the same value across datasets for consistency. We use dropout probability of 0.1 on dimensions of item embeddings in NISER+ across all models. Since the validation set is closer to the test set in time, therefore, it is desirable to use it for training the models. After finding the best epoch via early stopping based on performance on validation set, we re-train the model for same number of epochs on combined train and validation set. We train five models for the best hyperparameters with random initialization, and report average and standard deviation of various metrics for all datasets except for YC-1/4 where we train three models (as it is a large dataset and takes around 15 hours for training one model).

Evaluation Metrics: We use same evaluation metrics Recall@K and Mean Reciprocal Rank (MRR@K) as in [14] with $K = 20$. **Recall@K** represents the proportion of test instances which has desired item in the top-K items. **MRR@K** (Mean Reciprocal Rank) is the average of reciprocal ranks of desired item in recommendation list. Large value of MRR indicates that desired item is in the top of the recommendation list. For evaluating popularity bias, we consider the following metrics as used in [2]:

Average Recommendation Popularity (ARP): This measure calculates the average popularity of the recommended items in each list given by:

$$ARP = \frac{1}{|S|} \sum_{s \in S} \frac{\sum_{i \in L_s} \phi(i)}{K}, \tag{9}$$

where $\phi(i)$ is popularity of item i, i.e. the number of times item i appears in the training set, L_s is the recommended list of items for session s, and $|S|$ is the number of sessions in the test set. An item i belongs to the set Γ_{ϕ^*} of long-tail items or less popular items if $\frac{\phi(i)}{max_i \phi(i)} \leq \phi^*$. We evaluate the performance in terms of Recall@20

[7] $\tilde{\mathbf{i}}_k^T \mathbf{s}$ is not restricted to [-1,1], in general $||\mathbf{s}||_2 \neq 1$.
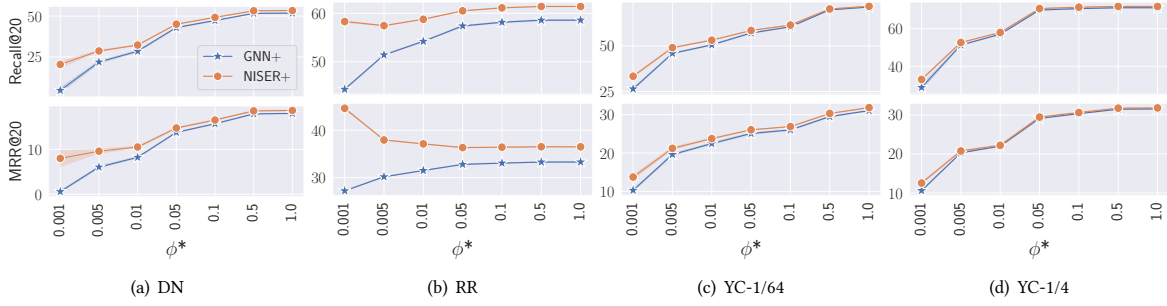
Figure 4: Offline setting evaluation: Recall@20 and MRR@20 with varying ϕ^* indicating larger gains by using NISER+ over GNN+ for less popular items.
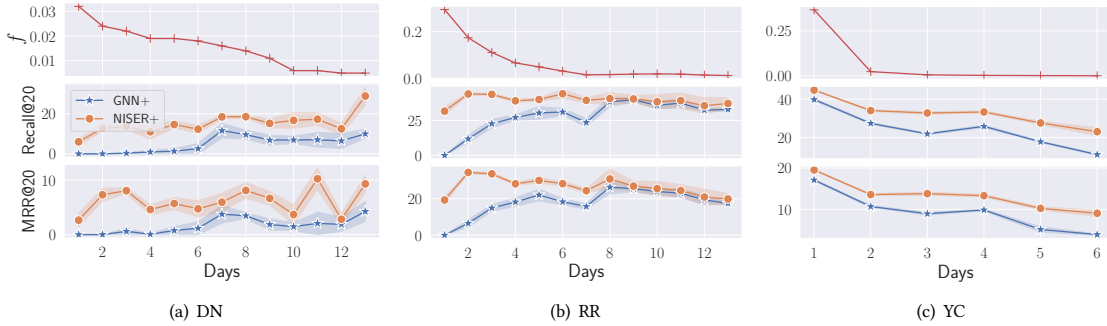

Figure 5: Online setting evaluation: Recall@20 and MRR@20 for sessions where target item is one of the less popular newly introduced items from the previous day. f denotes the fraction of such sessions in the training set for $\phi^* = 0.01$. Standard deviations over five models are shown in lighter-shaded region around the solid lines.

Table 2: Offline setting evaluation: NISER+ versus GNN+ in terms of Average Recommendation Popularity (ARP). Lower values of ARP indicate lower popularity bias.

Method	DN	RR	YC-1/64	YC-1/4
GNN+	495.25±2.52	453.39±8.97	4128.54±27.80	17898.10±126.93
NISER+	**487.31±0.30**	**398.53±3.09**	**3972.40±41.04**	**16683.52±120.74**

and MRR@20 for the sessions with target item in the set Γ_{ϕ^*} by varying ϕ^*.

6.1 Results and Observations

(1) NISER+ reduces popularity bias in GNN+ in : Table 2 shows that ARP for NISER+ is significantly lower than GNN+ indicating that NISER+ is able to recommend less popular items more often than GNN+, thus reducing popularity bias. Furthermore, from Fig. 4, we observe that NISER+ outperforms GNN+ for sessions with less popular items as targets (i.e. when ϕ^* is small), with gains 13%, 8%, 5%, and 2% for DN, RR, YC-1/64, and YC-1/4 respectively for $\phi^* = 0.01$ in terms of Recall@20. Similarly, gains are 28%, 18%, 6%, and 2% in terms of MRR@20. Gains for DN and RR are high as compared to YC. This is due to the high value of $max_i\phi(i)$. If instead we consider $\phi^* = 0.001$, gains are as high as 26% and 9% for YC-1/64 and YC-1/4 respectively in terms of Recall@20. Similarly, gains are as high as 34% and 19% in terms of MRR@20. We also

note that NISER+ is at least as good as GNN+ even for the sessions with more popular items as targets (i.e. when ϕ^* is large).

(2) NISER+ improves upon GNN+ in online setting for newly introduced items in the set of long-tail items Γ_{ϕ^*}. These items have small number of sessions available for training at the end of the day they are launched. From Fig. 5, we observe that for the less popular newly introduced items, NISER+ outperforms GNN+ for sessions where these items are target items on the subsequent day. This proves the ability of NISER+ to recommend new items on very next day, due to its ability to reduce popularity bias. Furthermore, for DN and RR, we observe that during initial days, when training data is less, GNN+ performs poorly while performance of NISER+ is relatively consistent across days indicating potential regularization effect of NISER on GNN models in less data scenarios. Also, as days pass by and more data is available for training, performance of GNN+ improves with time but still stays significantly below NISER+. For YC, as days pass by, the popularity bias becomes more and more severe (as depicted by very small value for f, i.e. the fraction of sessions with less popular newly introduced items) such that the performance of both GNN+ and NISER+ degrades with time. However, importantly, NISER+ still performs consistently better than GNN+ on any given day as it better handles the increasing popularity bias.

Table 3: NISER+ versus other benchmark methods in offline setting. Numbers after ± are standard deviation values over five models.

Method	DN	RR	YC-1/64	YC-1/4
Recall@20				
SKNN [5]	48.06	56.42	63.77	62.13
STAN [3]	50.97	59.80	69.45	70.07
GRU4REC [4]	29.45	-	60.64	59.53
NARM [6]	49.70	-	68.32	69.73
STAMP [8]	45.64	53.94	68.74	70.44
GNN [14]	51.39±0.38	57.63±0.15	70.54±0.14	70.95±0.04
GNN+	51.81±0.11	58.59±0.10	70.85±0.08	71.10±0.07
NIR	52.40±0.06	60.67±0.08	71.12±0.05	71.32±0.11
NISER	52.63±0.09	60.85±0.06	70.86±0.15	71.69±0.03
NISER+	**53.39±0.06**	**61.41±0.09**	**71.27±0.05**	**71.80±0.09**
MRR@20				
SKNN [5]	16.95	33.16	25.22	24.82
STAN [3]	18.48	35.32	28.74	28.89
GRU4REC [4]	8.33	-	22.89	22.60
NARM [6]	16.17	-	28.63	29.23
STAMP [8]	14.32	28.49	29.67	30.00
GNN [14]	17.79±0.16	32.74±0.09	30.80±0.09	31.37±0.13
GNN+	18.03±0.05	33.29±0.03	30.84±0.10	31.51±0.05
NIR	18.52±0.06	35.57±0.05	30.99±0.10	31.73±0.11
NISER	18.27±0.10	36.09±0.03	31.50±0.11	**31.80±0.12**
NISER+	**18.72±0.06**	**36.50±0.05**	**31.61±0.02**	31.77±0.10

Table 4: Ablation results for NISER+ indicating that normalization of embeddings (L_2 norm) contributes the most to performance improvement. Here PE: Position Embeddings.

Method	DN	RR	YC-1/64	YC-1/4
Recall@20				
NISER+	**53.39±0.06**	**61.41±0.09**	**71.27±0.05**	71.80±0.09
-L_2 norm	52.23±0.10	59.16±0.10	71.10±0.09	71.46±0.19
-Dropout	52.81±0.12	60.99±0.09	71.07±0.13	**71.90±0.03**
-PE	53.11±0.12	61.22±0.03	71.13±0.04	71.70±0.11
MRR@20				
NISER+	**18.72±0.06**	**36.50±0.05**	31.61±0.02	31.77±0.10
-L_2 norm	18.11±0.05	33.78±0.04	30.90±0.07	31.49±0.07
-Dropout	18.43±0.11	35.99±0.02	31.56±0.06	**31.93±0.17**
-PE	18.60±0.09	36.32±0.03	**31.68±0.05**	31.71±0.06

(3) NISER and NISER+ outperform GNN and GNN+ in offline setting: From Table 3, we observe that NISER+ shows consistent and significant improvement over GNN in terms of Recall and MRR, establishing a *new state-of-the-art* in SR.

We also conduct an **ablation study** (removing one feature of NISER+ at a time) to understand the effect of each of the following features of NISER+: i. L_2 norm of embeddings, ii. including position embeddings, and iii. applying dropout on item embeddings. As shown in Table 4, we observe that L_2 norm is the most important factor across datasets while dropout and position embeddings contribute in varying degrees to the overall performance of NISER+.

7 DISCUSSION

In this work, we highlighted that the typical item-frequency distribution with long tail leads to popularity bias in state-of-the-art deep learning models such as GNNs [14] for session-based recommendation. We then argued that this is partially related to the 'radial'

property of the softmax loss that, in our setting, implies that the norm for popular items will likely be larger than the norm of less popular items. We showed that learning the representations for items and session-graphs by optimizing for cosine similarity instead of inner product can help mitigate this issue to a large extent. Importantly, this ability to reduce popularity bias is found to be useful in the online setting where the newly introduced items tend to be less popular and are poorly modeled by existing approaches. We observed significant improvements in overall recommendation performance by normalizing the item and session-graph representations and improve upon the existing state-of-the-art results. In future, it would be worth exploring NISER to improve other algorithms like STAMP [8] that rely on similarity between embeddings for items, sessions, users, etc.

REFERENCES

[1] Himan Abdollahpouri, Robin Burke, and Bamshad Mobasher. 2017. Controlling popularity bias in learning-to-rank recommendation. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*. ACM, 42–46.

[2] Himan Abdollahpouri, Robin Burke, and Bamshad Mobasher. 2019. Managing Popularity Bias in Recommender Systems with Personalized Re-ranking. *arXiv preprint arXiv:1901.07555* (2019).

[3] Diksha Garg, Priyanka Gupta, Pankaj Malhotra, Lovekesh Vig, and Gautam Shroff. 2019. Sequence and Time Aware Neighborhood for Session-based Recommendations: STAN. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'19)*. ACM, 1069–1072.

[4] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2016. Session-based recommendations with recurrent neural networks. In *Proceedings of the 4th International Conference on Learning Representations (ICLR-2016)*.

[5] Dietmar Jannach and Malte Ludewig. 2017. When recurrent neural networks meet the neighborhood for session-based recommendation. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*. ACM, 306–310.

[6] Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Tao Lian, and Jun Ma. 2017. Neural attentive session-based recommendation. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. ACM, 1419–1428.

[7] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. 2015. Gated graph sequence neural networks. *arXiv preprint arXiv:1511.05493* (2015).

[8] Qiao Liu, Yifu Zeng, Refuoe Mokhosi, and Haibin Zhang. 2018. STAMP: short-term attention/memory priority model for session-based recommendation. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 1831–1839.

[9] Hao Peng, Lili Mou, Ge Li, Yunchuan Chen, Yangyang Lu, and Zhi Jin. 2015. A comparative study on regularization strategies for embedding-based neural networks. *arXiv preprint arXiv:1508.03721* (2015).

[10] Harald Steck. 2011. Item popularity and recommendation accuracy. In *Proceedings of the fifth ACM conference on Recommender systems*. ACM, 125–132.

[11] 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 *Advances in neural information processing systems*. 5998–6008.

[12] Feng Wang, Xiang Xiang, Jian Cheng, and Alan Loddon Yuille. 2017. Normface: L2 hypersphere embedding for face verification. In *Proceedings of the 25th ACM international conference on Multimedia*. ACM, 1041–1049.

[13] Meirui Wang, Pengjie Ren, Lei Mei, Zhumin Chen, Jun Ma, and Maarten de Rijke. 2019. A Collaborative Session-based Recommendation Approach with Parallel Memory Modules. In *Proceedings of the 42Nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'19)*. ACM, 345–354.

[14] Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan. 2019. Session-based Recommendation with Graph Neural Networks. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence*.

[15] Longqi Yang, Yin Cui, Yuan Xuan, Chenyang Wang, Serge Belongie, and Deborah Estrin. 2018. Unbiased offline recommender evaluation for missing-not-at-random implicit feedback. In *Proceedings of the 12th ACM Conference on Recommender Systems*. ACM, 279–287.

[16] Yutong Zheng, Dipan K Pal, and Marios Savvides. 2018. Ring loss: Convex feature normalization for face recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 5089–5097.