applied sciences

MDPI

Article

Logit Averaging: Capturing Global Relation for Session-Based Recommendation

Heeyoon Yang †, Gahyung Kim † and Jee-Hyoung Lee *

Department of Artificial Intelligent, SungKyunKwan University, Suwon 16419, Korea; chri0220@skku.edu (H.Y.); ghkim10202@skku.edu (G.K.)
* Correspondence: john@skku.edu
† These authors contributed equally to this work.

Abstract: Session-based recommendation predicts an anonymous user's next action, whether she or he is likely to purchase based on the user's behavior in the current session as sequences. Most recent research on session-based recommendations makes predictions based on a single-session without incorporating global relationships between sessions. It does not guarantee a better performance because item embeddings learned by solely utilizing a single session (inter-session) have less item transition information than utilizing both intra- and inter-session ones. Some existing methods tried to enhance recommendation performance by adopting memory modules and global transition graphs; however, those need more computation cost and time. We propose a novel algorithm called Logit Averaging (LA), utilizing both (i) local-level logits, which come from intra-sessions item transitions and (ii) global-level logits, which come from gathered logits of related sessions. The proposed method shows an improvement in recommendation performance in respect of accuracy and diversity through extensive experiments.

Keywords: session-based recommendation; global relation; long-tail distribution; Logit Averaging

check for updates

Citation: Yang, H.; Kim, G.; Lee, J.-H. Logit Averaging: Capturing Global Relation for Session-Based Recommendation. *Appl. Sci.* **2022**, *12*, 4256. https://doi.org/10.3390/app12094256

Academic Editor: Yosoon Choi

Received: 18 March 2022
Accepted: 20 April 2022
Published: 22 April 2022

1. Introduction

With the rapid growth of Internet commerce platforms, recommender systems now play an imperative role in improving user experience and satisfaction on the platforms [1]. Predicting users' next behavior from current and previous actions not only alleviates information overload problems but can also offer a personalized service to targeted users in various applications [2].

A conventional recommender system [3–8] utilizes a history of user-item interactions and side information to learn each user's long-term and static preferences on items. Interest-aware Message-Passing GCN (IMP-GCN) [8] was a model that plays to the strength of the Graph Convolution Network in recommender systems by alleviating the over-smoothing problem. Liu et al. pinpointed the risk of involving higher-order neighboring users with no common interests with a user and learning their embeddings in the graph convolution operation. To overcome the addressed problem, IMP-GCN performed high-order graph convolution inside subgraphs that consist of users with similar features. We found this model very effective in its ability to avoid the propagation of non-related information from high-order neighbors into embedding learning. IMP-GCN was designed based on the user-item bipartite graph datasets, which consist of user and item features. However, this form of dataset is often invalid in many cases such as numerous anonymous users interacting with items in a very short period of time or the same account can be shared with multiple people [9]. Thus, a new system to deal with the problem has emerged, which is a session-based recommendation [10]. Unlike conventional ones, session-based recommender systems only utilize the ongoing movement of an anonymous user in one session to predict the next item, where a session is composed of items that users interacted with within a continuous period of time. Owing to its practicality in various e-commerce platform scenarios, the session-based recommender system has gained much attention.

To predict the users' next click items, the session-based recommender system utilizes a deep neural network and has achieved outstanding performances. Recurrent Neural Networks (RNNs) or attention mechanism models were first applied to a session-based recommendation task for extracting sequential patterns, which is a straightforward approach to handling a session as it is the sequence of items user clicked sorted by time [11,12]. NARM [11] utilized gated recurrent unit (GRU) [13], which is the variation of RNN to extract sequential patterns in sessions. SRSAN [12] took dependencies among items in sessions by using self-attention networks.

Furthermore, recent studies incorporate Graph Neural Networks (GNNs) [14] to construct a session-based recommender system due to its effectiveness in representing session consistency along with item dependencies and shows a competitive performance comparable to RNNs-based approaches [15–19]. SRGNN [15] was the first GNN-based framework and has been used as a basic structure in many studies. A session graph was sent into gated-GNN [20] to obtain the latent representation of nodes, and the session representation was generated by a soft-attention mechanism based on the node representations. Gupta et al. [16] normalized representations of items and sessions generated from GNNs to increase the overall recommendation performances. Chen et al. [17] introduced a novel GNN model that focuses on aggregating the information passed from neighboring nodes using a GRU while preserving the item orders in sessions. This layer is called the Edge-Order Preserving Aggregation (EOPA) layer and shows a promising performance while minimizing the information loss problems. TAGNN++ [18] adaptively activated the attention model with respect to varied target items and improved the expressiveness of the model. Many researchers have tried to capture the accurate preference of an anonymous user from a short session, but it is still hard. To overcome this, Li et al. [19] applied disentangle representation learning to make better item representation. They assumed that the chunks of item embeddings represent factors such as categories and colors, and learned item embeddings with a distance correlation function aiming for the factors to disentangle each other. It is useful for the session-based recommendation task because of the absence of additional information in a session dataset.

However, existing GNN-based models only make predictions with items in a single session and do not consider the global-level contexts that exist between inter-sessions. They ignore the useful item transition information from other sessions. In [9], the session-based recommendation has several unique characteristics: Session consistency, Repeated item consumption, Timeliness of sessions, and Sequential dependency. Here, we mainly take sequential dependency into consideration. Some items can be explored in a certain order from the majority of users. For example, we can easily find the item-transition information, ['phone', 'phone case'] or ['phone', 'wireless earphone']. This pattern will appear in multiple sessions. Thus, taking the global-level context of items into consideration is necessary to improve the prediction accuracy and diversify the recommendation lists. In addition, the number of items in a session is lower than 10 generally. The short length of sessions may lead to incorrect predictions when we make a prediction based on a single session. Therefore, it will be helpful to recommend items with global relations from multiple sessions which share the same items.

Some studies are trying to model the global relations of sessions. Wang et al. [21] applied collaborative neighborhood information to recommend items. It consists of an inner memory encoder and an outer memory encoder that extract the current session's context and the collaborative information from other sessions which are similar to the current session, respectively. Wang et al. [22] conducted a similar attempt to infer global context through the GCE-GNN (Global Context Enhanced Graph Neural Networks) model. It constructed a global graph using all the unique items that exist in inter-sessions and extracted the global-level item embedding. Fei et al. [23] proposed MAN (Memory Augmented Model) for a session-based recommendation, which augments a base recommender model using updated memory, and recommends items with memory components. Zhou et al. [24] utilized a graph including item relations of each day to capture timely user interests. By doing this, an item can have different neighborhoods on different days. Huang et al. [25] tried to model inter- and intra- item transition through multi-task learning. Each session went into the self-attention layer to capture intra-sessions item transition, and item representations which are constructed at this step were set as initialized values of item representations for learning global item relations. They could improve the recommendation accuracy by designing different layer architectures depending on whether the item is from global or local relations. Xia et al. [26] also tried to reflect the global relation among sessions. They proposed two different graph view augmentations with respect to a session dataset to adopt the self-supervised co-training method on a session-based recommendation. One is the item view graph, generally utilized in session-based recommendation, and the other is the session view graph which represents the inter-session relationship with Jaccard similarity. By learning from these two views, items and sessions, session representations became richer. The Heterogeneous Global Graph Neural Network (HG-GNN) [27] was another model that attempts to extract global item co-occurrence by constructing user-item interactions and global co-occurrence item graphs. They used heterogeneous global graphs to learn long-term user preferences and item representations. Even though their work was to implement the truly personalized session-based recommendation, we found this work has a limitation on data availability. Most of the existing session datasets do not contain user information including their user IDs. Since session-based recommender systems are proposed to make recommendations solely based on the sequence of items anonymous users interacted with within a limited period of time, HG-GNN can be utilized in very few circumstances where the user's ID and historical sessions are available.

However, those attempts rather increase the computation costs in storing memory and extracting global-level item representation with over a million item nodes and fails to show drastic performance improvement. We propose a simple and novel algorithm called Logit Averaging (LA) that reflects global relations among sessions by averaging short-head item's logits which are the last representation of the model. There were several preceding studies using logit to improve representation learning in classification tasks such as logit adjustment. Adjusting logits encourages a large relative margin between logits of rare labels versus dominant ones. Menon et al. [28] have conducted extensive studies on classical logit adjustment techniques and proposed a novel logit adjusted loss function. Even though the effectiveness of the logit adjustment loss function in balancing the class label during training, we found this technique hard to apply in session-based recommendation tasks. Menon et al. utilized image datasets, such as CIFAR-10 and iNaturalist, to examine their proposed methods, which have only nine to ten classes. However, in session datasets, the class of each session is thousands of different last-clicked items. If we apply the logit adjustment loss technique to a session-based recommender system, the training of the model will not be performed properly.

Thus, we propose a suitable logit training technique for a session-based recommender system called Logit Averaging. We perform experiments on multiple session-based recommendation models and verify that our algorithm outperforms in prediction accuracy and diversity. Oour accomplishment in increasing the item diversity is especially important in recommender systems because increasing the probability of retrieving unusual or novel items relevant to users highly improves the users' satisfaction with the system [29–32]. We summarize our contributions as follows:

- By interpolating logits of the sessions with the same target value, we propose a simple yet effective method to incorporate global contexts for session-based recommendation and achieve high accuracy and diversity;
- We conduct extensive experiments on four real-world datasets in six different deep-learning-based session recommendation models and successfully demonstrate the effectiveness of our method;

- We can easily "plug and play" Logit Averaging (LA) in various neural network-based session recommendation models.

The structure of this paper is as follows. In Section 2, we define the problem and the corresponding assumption we made. Furthermore, an in-detail description of our proposed algorithm, Logit Averaging (LA), is mentioned. In Section 3, we describe experimental settings, including the datasets, baseline models, and evaluation metrics. Then, we make a detailed analysis of the experimental results. Finally, in Section 4, we summarize our proposed method and discuss the future works.

2. Proposed Method

2.1. Problem Definition and Notation

The final goal of the session-based recommendations task is the prediction of a last clicked item based on an anonymous user's past clicked items. A set $I = \{i_1, \ldots, i_N\}$ denotes the set of all unique items in all sessions. An anonymous user's session can be represented by $s = [i_1, i_2, i_3, \ldots, i_t]$, ordered by time. The recommendation model predicts the next clicked item, the $(t+1)$th item given a session s, then we get probabilities $\hat{y}$ for all candidate items in I. We split items into short-head and long-tail according to the Pareto Principle [33]. The items which account for 75% of all item appearances are short-head items, and the rest are long-tail items, respectively—I^H and I^T.

2.2. Logit Averaging

We propose a simple module that averages logits from a prediction layer for the better recommendation, so the other parts of a recommender system, such as an item encoder, a session encoder and a prediction layer, can be any models. In other words, our module can easily be plug-and-play for any other existing neural network-based session recommendation models. As shown in Figure 1, our proposed technique "Logit Averaging (LA)" is located between the session encoder and the final prediction layer.

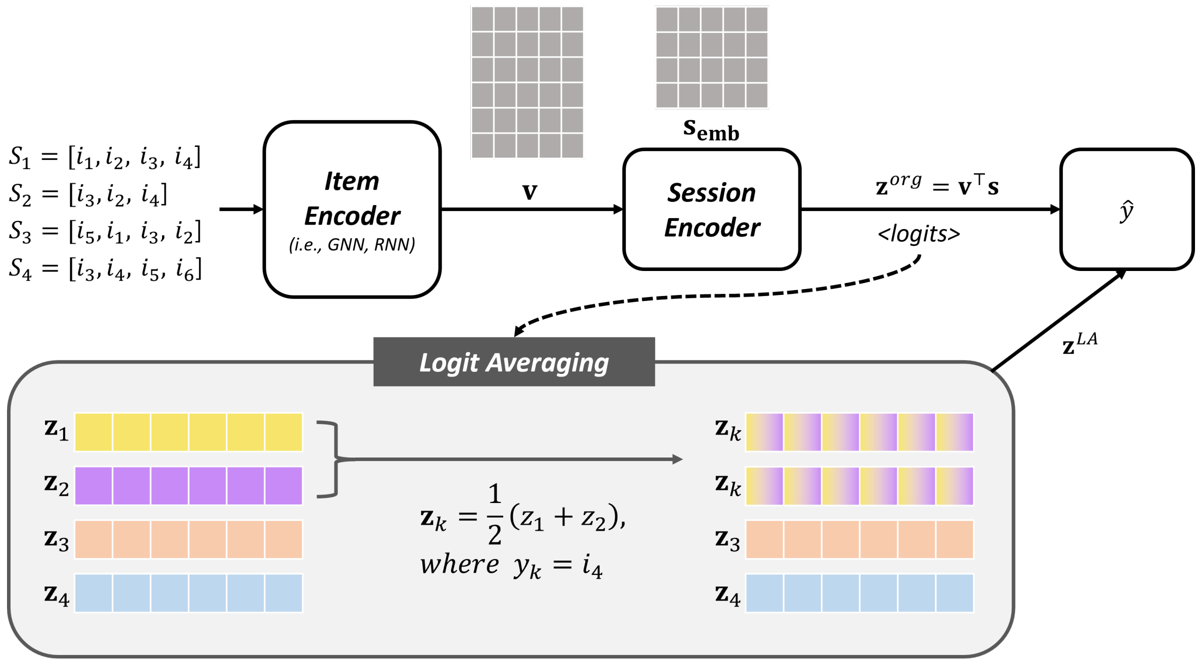

Figure 1. General deep-learning-based session-based recommendation models framework with the proposed methods.

Most neural network-based recommendation models are structured as the white box in Figure 1. Sessions in a batch go into the model as inputs and an item encoder learns item embeddings. Session embeddings are generated from a session encoder using item embeddings. The final prediction is commonly calculated by the cosine similarity between all unique item embeddings and session embeddings of the batch.

Logit averaging is a module that performs as augmentation, which can inject global context at the same time. As mentioned above, recent deep learning-based session recommendation models recommend items in respect of a single session. It is not enough to make better embeddings when solely using intra-session transition and not considering inter-sessions. Sessions have item consequences where several sessions have common items. This means that the sessions that share the same items have similar item transitions and can help to supplement the information with each other. Then, it is easier to predict items than use a single session alone.

GCE-GNN [22] tried to solve the session recommendation task with a global transition using a global item graph. The graph merges all the items in the entire sessions into a single global graph. This shadows the hidden global contexts in sessions and cannot distinguish the predominant or minor trends.

To overcome the problem, we propose a method to average logits of sessions that share the same target items instead of constructing a large graph. In detail, logits of each sample are obtained by calculating cosine similarity between session embeddings and item embeddings, and the logits are replaced by the mean value of logits that have the same target items. The Logit Averaging and replacing are only performed by sessions that have short-head items as target items. As shown in Figure 2, all the four public session datasets have long-tail item distribution, meaning that the majority of sessions share a few numbers of target items [34]. We assume that sessions with the same target items will have shared a global context. The short-head items take up seventy-five percent of sessions, so it can inject global relations into the majority of sessions without adopting all items. In other words, averaging logits of short-head items easily supplements the global relations information and increases the recommendation performance.

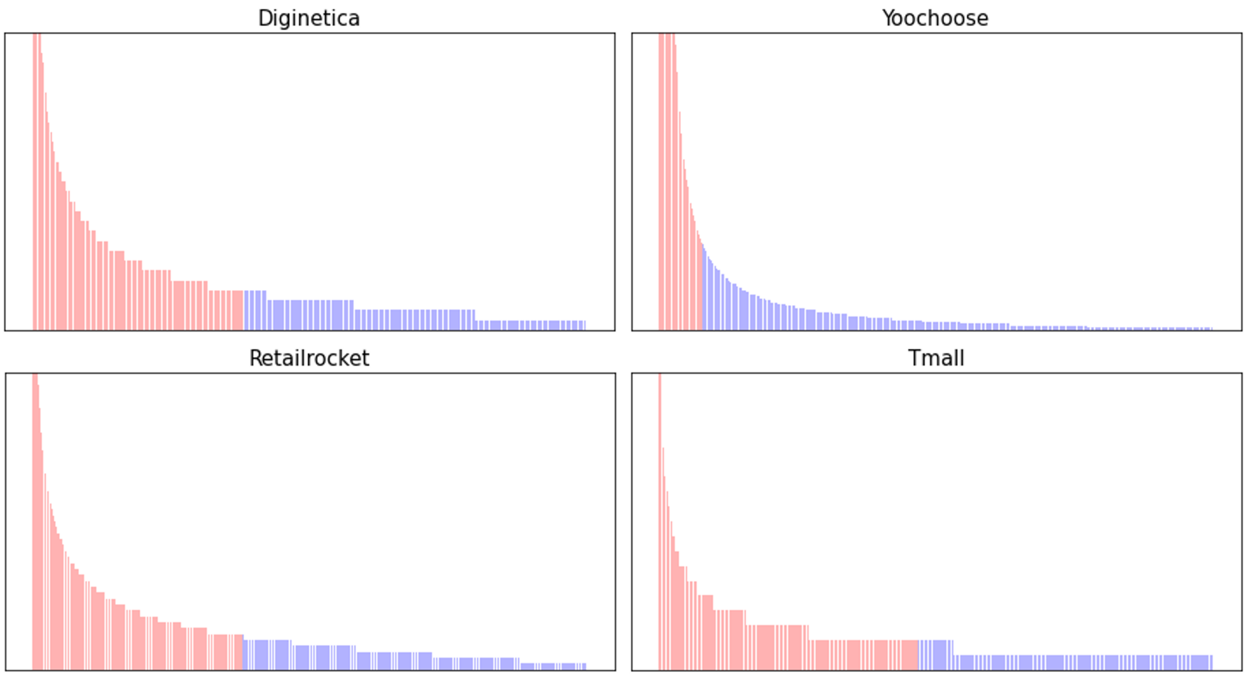

Figure 2. Long-tail distribution of real-world session datasets. From the left, the datasets are Diginetica, Yoochoose, Retailrocket, Tmall. The red part represents short-head and the blue part represents long-tail.

It is not enough to utilize averaged logits only for making better predictions because averaging can ignore the original item transition characteristics of sessions. Therefore, we adopt two loss functions, one is for the original logits and the other for the averaged logits. We get the final loss by the sum of two losses with λ.

$$L_{org}(y, Z^{org}) = -\sum_{i=1}^{m} y_i \log(Z_i^{org}) \tag{1}$$

$$L_{org}(y, Z^{LA}) = -\sum_{i=1}^{m} y_i \log(Z_i^{LA}) \tag{2}$$

$$L = L^{org} + \lambda \cdot L^{LA}. \tag{3}$$

The detailed process is shown in Algorithm 1.

Algorithm 1 Logit Averaging.

```
Input: sessions $S$, Short-head Items $I^H$, Item Encoder $IE()$, Session Encoder $SE()$
  Initialize IE and SE
  for each iteration do
      $V \leftarrow IE(S)$
      $S_{emb} \leftarrow SE(V, S)$
      $Z^{org} = S_{emb}^{\top} \cdot V$
      for all $I^H$ do
          if $I_i^H$ in target items of sessions then
              $Z_k^{LA} = \frac{1}{|n|} \sum_{k=0}^{n} Z_k$  (where the target item of $Z_k$ are same as $I_i^H$)
          end if
      end for
      $L$ is calculated by using Equation (3)
  end for
```

3. Results

We conducted extensive experiments to evaluate the performance of our proposed method. This section discusses the performance comparison between the proposed method and baseline methods. Before that, we describe the datasets, the baselines, and the evaluation metrics utilized in our experiments.

3.1. Setup

3.1.1. Datasets

We employed four real-world datasets: Yoochoose (https://www.kaggle.com/chadgostopp/recsys-challenge-2015, accessed on 22 April 2022), Diginetica (https://competitions.codalab.org/competitions/11161#learn_the_details-data2, accessed on 22 April 2022), RetailRocket (https://www.kaggle.com/retailrocket/ecommerce-dataset, accessed on 22 April 2022) and Tmall (https://ijcai-15.org/repeat-buyers-prediction-competition/, accessed on 22 April 2022), which are commonly used in session-based recommendation methods and the statistics of each dataset are shown in Table 1. Due to the size of the dataset, we used the recent fraction of 1/64 for the Yoochoose datasets. We keep only items appearing over five times and sessions whose length is more than one. The latest data are set to be a test dataset and the previous are set to be a training dataset. The label of each session is the last clicked item of it. The proposed method, Logit Averaging, works when the target of the sample is in the short-head items set and we split the items into short-head items and long-tail items. We sorted items by the number of item appearances in descending order. Then we chose the items as short-head, which are in the top 75% cumulative sum.

Table 1. Data Statistics.

Statistics	Diginetica	Retailrocket	Yoochoose1/64	Tmall
# Train Sessions	186,670	113,287	124,472	65,286
# Test Sessions	15,936	71,235	15,324	1027
# Items	43,098	27,413	37,484	40,728
Avg. Length	4.85	3.66	3.97	6.68

3.1.2. Baseline Methods

To compare the recommendation performance of the baseline methods and the proposed method, we employ six different baselines. Those can be categorized into two groups—attention-based methods and GNN-based methods. We utilize NARM [11] and SR-SAN [12] for the attention-based methods and SR-GNN [15], NISER [16], EOPA [17] and TAGNN++ [18] for the GNN-based methods. We set the hyperparameters of each baseline method as the baseline did. The specific hyperparameters used in the experiments are described in Table A2. The experimental results are shown in Table 2.

Table 2. The performance result with and without LA (Logit Averaging) method. In column LA, we mark 'X' to indicate the baseline without using LA and mark 'O' to indicate the baseline with using LA. Gain (%) is the average gain of the baselines with LA on the original baselines.

Dataset	Model	LA	HR@10	MRR@10	Coverage@10	HR@20	MRR@20	Coverage@20
RetailRocket	NARM	X	39.24	20.76	95.52	45.92	21.24	97.63
	EOPA		33.74	17.13	81.07	42.51	17.72	86.91
	NISER		33.24	22.24	88.66	37.54	22.53	93.42
	SR-GNN		31.98	18.81	76.09	37.24	19.17	80.40
	SR-SAN		37.96	17.25	93.97	44.91	17.74	96.53
	TAGNN++		34.30	21.54	81.59	39.83	21.92	85.47
	NARM	O	39.02	21.07	95.86	45.64	21.53	97.80
	EOPA		33.63	17.33	81.28	42.28	17.91	86.98
	NISER		33.33	22.09	88.59	37.66	22.39	93.39
	SR-GNN		34.40	21.92	85.34	39.45	22.27	89.29
	SR-SAN		37.50	17.26	92.44	44.33	17.75	95.45
	TAGNN++		34.10	21.71	82.23	39.61	22.09	86.21
	Gain (%)		0.86	3.23	1.97	0.54	3.14	1.84
Tmall	NARM	X	17.14	12.52	17.68	19.18	12.71	28.68
	EOPA		19.58	9.70	32.27	24.64	10.02	44.43
	NISER		18.89	14.04	18.93	21.03	14.18	31.83
	SR-GNN		16.55	11.00	13.79	18.79	11.17	21.52
	SR-SAN		18.99	12.74	12.67	21.62	12.92	19.65
	TAGNN++		33.11	16.17	15.88	39.82	16.64	25.01
	NARM	O	16.46	12.21	17.28	18.5	12.35	28.17
	EOPA		19.43	9.81	31.62	24.14	10.14	43.66
	NISER		18.11	13.52	19.21	20.35	13.60	32.43
	SR-GNN		16.85	11.92	15.47	19.67	12.11	24.81
	SR-SAN		20.55	14.25	12.63	23.17	14.43	19.54
	TAGNN++		30.77	14.55	16.36	36.42	14.96	25.42
	Gain (%)		−0.98	0.86	2.02	−0.92	0.71	2.46
Yoochoose1/64	NARM	X	59.95	29.46	29.24	70.67	30.20	35.73
	EOPA		51.83	24.91	28.81	62.85	25.68	31.74
	NISER		60.25	30.91	29.46	70.55	31.64	34.61
	SR-GNN		59.56	29.47	23.19	69.87	30.19	26.54
	SR-SAN		53.90	25.96	28.28	64.89	26.73	32.62
	TAGNN++		60.70	30.63	26.48	70.93	31.35	30.72
	NARM	O	59.96	29.65	28.88	70.61	30.40	35.37
	EOPA		51.54	24.90	29.02	62.64	25.68	31.73
	NISER		60.23	30.88	29.40	70.48	31.60	34.73
	SR-GNN		60.46	30.90	27.68	70.55	31.61	32.06
	SR-SAN		53.75	26.00	28.31	64.91	26.77	32.63
	TAGNN++		60.84	31.11	28.79	71.16	31.83	33.46
	Gain (%)		0.15	1.18	4.58	0.14	1.15	4.84

Table 2. *Cont.*

Dataset	Model	LA	HR@10	MRR@10	Coverage@10	HR@20	MRR@20	Coverage@20
Diginetica	NARM	X	35.86	18.60	68.78	44.83	19.22	82.60
	EOPA		34.25	14.52	62.09	47.09	15.37	75.91
	NISER		34.12	17.98	65.02	41.89	18.52	77.34
	SR-GNN		32.84	16.83	60.95	41.46	17.43	71.44
	SR-SAN		32.54	16.61	67.60	41.50	17.22	78.56
	TAGNN++		33.90	17.34	62.81	42.64	17.95	74.00
	NARM	O	35.76	18.51	69.05	44.49	19.12	83.07
	EOPA		34.54	14.60	62.33	47.03	15.47	75.81
	NISER		33.79	18.02	65.28	41.68	18.57	78.25
	SR-GNN		33.85	17.72	64.47	42.03	18.29	76.34
	SR-SAN		32.93	16.93	66.68	42.02	17.56	77.34
	TAGNN++		36.42	18.31	68.85	45.41	18.93	79.83
	Gain (%)		1.88	2.18	2.53	1.29	2.13	2.47
Total Gain (%)			0.48	1.86	2.78	0.26	1.78	2.90

3.1.3. Evaluation Metrics

To evaluate the performance of all algorithms, we use three metrics—Recall@K, MRR (Mean Reciprocal Rank)@K, and Coverage@K. We set K to 10 and 20 for all the metrics.

- **Recall** is the metric to measure recommendation accuracy. It is the proportion of correct prediction amongst top-k recommendation lists.

$$Recall = \frac{\#hits}{|S|} \times 100, \tag{4}$$

  where $\#hits$ is the number of hits that the generated the top-K recommend list containing the ground truth and $|S|$ is the number of test sessions.
- **Mean Reciprocal Rank (MRR)** also measures recommendation accuracy taking ranks into consideration. MRR is the mean of ground truths' reciprocal rank in the lists.

$$MRR = \frac{1}{|S|} \sum_{i=1}^{|S|} \frac{1}{rank_i} \times 100, \tag{5}$$

  where $rank_i$ is the reciprocal position of the ground truth in the top-K recommendation list.
- **Coverage** is the ratio of the unique items that appears in recommendation lists.

$$Coverage = \frac{|I^{rec}|}{|I|} \times 100, \tag{6}$$

  where $|I^{rec}|$ is the number of unique items included in the top-k recommendation list, and $|I|$ is the total number of unique items in the test dataset.

3.2. Experiment Result

Table 2 shows the performance of the baseline models with and without Logit Averaging (LA). The rows are the baselines and their combination with LA, and columns are the evaluation metrics. We evaluate the performance in both top-10 and top-20 recommendation lists. For each dataset, we calculate the average percentage of the performance gain in performance in all six baseline models.

At the bottom of Table 2, there is the Total Gain (%) that calculates the average percent of the gain in all four datasets. For all six metrics, HR@10, MRR@10, Coverage@10, HR@20, MRR@20, and Coverage@20, applying LA to the baselines improves the average performance by 0.48%, 1.86%, 2.78%, 0.26%, 1.78%, and 2.90%, respectively. This result proves the effectiveness of our proposed method in improving prediction accuracy and item diversity.

For RetailRocket dataset, HR@10, MRR@10, and Coverage@10 have increased by 0.86%, 3.23%, and 1.97%, respectively. HR@20, MRR@20, and Coverage@20 have increased by 0.54%, 3.14%, and 1.84%, respectively. Not only does prediction accuracy improve but also diversify the recommendation list.

In the case of the Tmall dataset, MRR@10 and Coverage@10 have increased by 0.86% and 2.02% while HR@10 decreases by −0.98%. Even though there has been a slight decrease in Recall, applying LA helps to make an accurate reciprocal ranking prediction and increases the item diversity.

For the Yoochoose1/64 dataset, the performance of all three metrics, HR@10, MRR@10, and Coverage@10, improves by 0.15%, 1.18%, and 4.58%, respectively. It particularly increases item coverage more than other metrics. As Isufi et al. [35] pointed out, usually, recommendation accuracy is tied with diversity in a delicate trade-off. Thus, a great increase in item coverage along with recommendation accuracy when LA is applied is a remarkable achievement.

Additionally, for the Diginetica dataset, HR@10, MRR@10, and Coverage@10 increase by 0.48%, 2.18%, and 2.53% respectively when LA is applied to the original baselines. This trend can also be observed when $K = 20$. The performance of HR@20, MRR@20, and Coverage@20 improves by 1.29%, 2.13%, and 2.47%.

The six baseline models have shown a different tendency to improve recommendation results depending on whether the model is based on an attention mechanism or GNNs. Firstly, the attention-based models, NARM and SR-SAN, show a large improvement in Hit Ratio and MRR after training with LA methods. SR-SAN gains 13.99% on MRR@10, but drops 3.2% on Coverage@10 on average. The GNN-based models, EOPA, NISER, SR-GNN, and TAGNN++, show the opposite tendency of attention-based models. The models applying LA show the steepest rise in coverage. SR-GNN and TAGNN++ gain 49.48% and 22.15% on Coverage@10, respectively.

We compare the performance of HR@20, MRR@20 and Coverage@20 in two different datasets in the same baseline model, TAGNN++ [18]. The result is shown in Figure 3. The salmon color bar is the performance of baseline models, and the yellow bar indicates the performance of baselines with our proposed method LA. Lastly, we applied a data augmentation technique, GraphMix [36], to further improve the performance, and the result is shown in the skyblue bar. For the two datasets, LA improves the result of HR@20, MRR@20, and Coverage@20. When GraphMix is used in addition to LA, it helps to increase prediction accuracy and item coverage in both datasets.

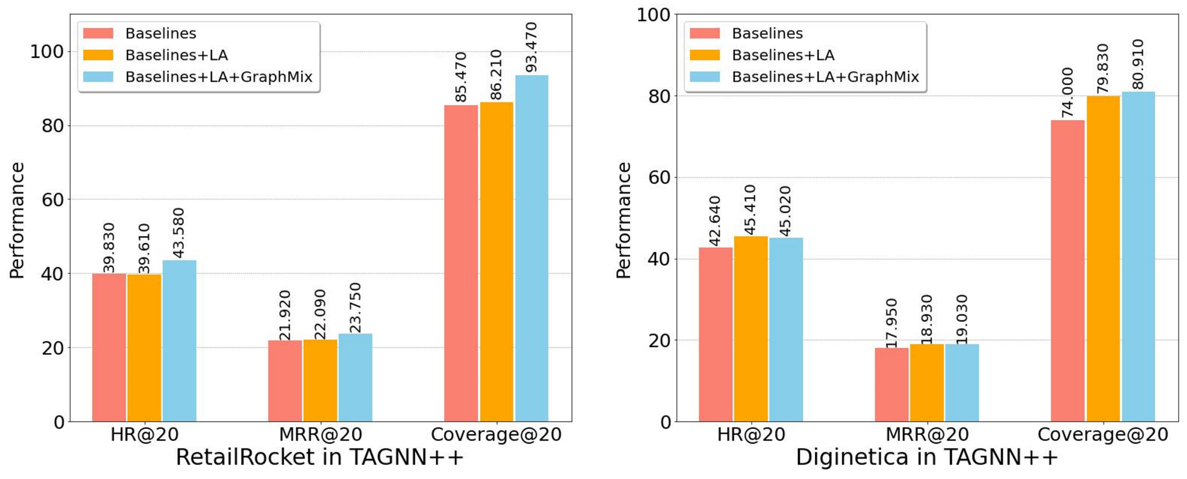

Figure 3. Comparing performance in two different datasets in the TAGNN++ model. Compare to baseline model performance, applying LA (Logit Averaging) improves performances in every metrics. In addition to LA, applying GraphMix, a data augmentation technique, improves performance of HR@20 and MRR@20.

3.3. Ablation Study

For the ablation study, we conducted two different experiments. First, we applied various data augmentation techniques to our proposed method. Second, we compared the performance of our method and GCE-GNN. The experimental results are shown in Tables 3 and 4, respectively.

Table 3. The average performance gains (%) of four baseline methods with LA and different data augmentation methods on the original baseline methods.

Dataset	Augmentation	LA	HR@10	MRR@10	Coverage@10	HR@20	MRR@20	Coverage@20
RetailRocket	X	O	−0.20	0.09	−0.15	−0.24	0.08	−0.05
	GraphMix	O	−0.44	0.88	−1.73	−0.81	0.85	−1.50
	FLAG	O	1.18	0.54	2.04	1.20	0.56	2.00
	Random deletion	O	0.82	−2.07	1.55	1.61	−2.02	1.09
	Random Insertion	O	−1.71	−3.31	−0.03	−1.12	−3.28	−0.05
Tmall	X	O	−0.56	−0.24	0.08	−0.80	−0.28	0.10
	GraphMix	O	1.70	0.95	1.46	2.15	0.97	2.72
	FLAG	O	0.46	−0.32	0.18	1.42	−0.27	0.25
	Random deletion	O	−0.44	−0.66	0.44	−0.73	−0.69	0.76
	Random Insertion	O	−3.31	−2.19	0.16	−3.82	−2.23	0.15
Yoochoose1/64	X	O	−0.01	0.17	0.48	0.03	0.17	0.63
	GraphMix	O	−0.73	−0.12	2.48	−0.64	−0.11	3.07
	FLAG	O	0.28	0.11	0.62	0.26	0.11	0.53
	Random deletion	O	−0.15	−0.82	−1.79	−0.02	−1.06	−1.92
	Random Insertion	O	−1.21	−1.72	−0.69	−0.55	−1.68	−0.54
Diginetica	X	O	0.62	0.31	1.41	0.69	0.32	1.50
	GraphMix	O	0.48	0.32	1.40	0.33	0.31	0.93
	FLAG	O	0.08	0.01	0.09	−0.05	0.01	0.22
	Random deletion	O	−0.17	−0.83	0.77	0.12	−0.81	1.49
	Random Insertion	O	−0.25	−0.37	−0.27	−0.44	−0.37	0.69

Table 4. The performance comparison on our proposed model with TAGNN++ and GCE-GNN using Tmall and Yoochoose1/64 datasets.

Dataset	Model	HR@10	MRR@10	Coverage@10	HR@20	MRR@20	Coverage@20
Tmall	GCE-GNN	23.76	15.20	16.18	26.29	15.35	25.71
	TAGNN++	30.77	14.55	16.36	36.42	14.96	25.42
	TAGNN++ w. GraphMix	**34.27**	**17.11**	**17.50**	**43.33**	**17.76**	**28.06**
Yoochoose1/64	GCE-GNN	60.32	29.66	**29.79**	71.02	30.40	**34.98**
	TAGNN++	**60.84**	**31.11**	28.79	**71.16**	**31.83**	33.46
	TAGNN++ w. GraphMix	60.75	30.86	27.64	71.15	31.59	32.27

3.3.1. Effectiveness of Augmentation

We utilized augmentation methods with four baselines in order to overcome the data sparsity problem in session datasets. We conduct four different data augmentation methods, which are GraphMix [36], FLAG [37], Random deletion, and Random insertion.

GraphMix [36] is proposed as a regularization method for GNN based semi-supervised object classification. It interpolates the hidden states and the corresponding labels to make a better feature representation. Even though it targets object classification tasks, we find this technique also applicable to session-based recommendation tasks as well. We interpolate the feature representations of two random sessions within a batch and let there be two target values, making it similar to a multi-label classification task. The equation is shown below.

$$F_{mix} = \lambda \cdot F_a + (1 - \lambda) \cdot F_b \tag{7}$$

$$L_m = \lambda \cdot CrossEntropyLoss(pred, y_a) + (1 - \lambda) \cdot CrossEntropyLoss(pred, y_b) \tag{8}$$

$$L = L_o + L_m, \tag{9}$$

First, we interpolate two random feature representations of sessions a and b, F_A and F_b with respect to λ value. We set λ value as 0.6 in all experiments. Then the *pred* is calculated using the session model, and the loss value is calculated using cross entropy as in Equation (8). The loss value calculated from mixed logits, L_m, is interpolated with respect to λ value as well. The final loss value is obtained in addition to the original loss L_o and the modified one, L_m.

FLAG [37] is introduced as a data augmentation technique, specifically for graph datasets to enhance the performance of the graph neural network through iteratively augmented node features with gradient-based adversarial perturbations during training.

Lastly, we execute a random deletion and random selection method, which is a straightforward data augmentation method widely used in the natural language processing (NLP) domain. For each session within a batch, we randomly delete or insert the items to sessions. For random insertion, we use a set of unique items within a batch to randomly select the one item to insert.

Table 3 shows the average percentage of the performance gain of different data augmentation methods in four baseline models, which are both attention-based (NARM, SRSAN) and GNN-based (NISER, TAGNN++). The raw result value of this experiment is shown in Table A1 in Appendix A.

Even though the results differ depending on the datasets, GraphMix and FLAG show an impressive performance, improving Recall, MRR and item coverage. Especially for the Tmall datset, GraphMix increases HR@20, MRR@20, and Coverage@20 by 2.15%, 0.97%, and 2.72%, respectively. However, applying random deletion and random insertion rather degrades the performance in almost all datasets. Thus, we could infer that GraphMix and FLAG are effective data augmentation techniques that could alleviate the data sparsity problem in session datasets, and further diversify the item recommendation lists.

3.3.2. Comparison on GCE-GNN

We compare the performance of our methods with GCE-GNN [22], which injects the global relation information by employing a global item transition graph. We conducted experiments using two datasets: Tmall and Yoochoose. GCE-GNN is the performance we obtain with the official code (https://github.com/CCIIPLab/GCE-GNN, accessed on 22 April 2022). As shown in Table 4, our method outperformed the existing method in overall performance. Although the coverage of our method was lower than that of GCE-GNN with Yoochoose datasets, TAGNN++ with LA and GraphMix augmentation improves all metrics by a large margin. Therefore, we demonstrate that our proposed methods, LA and the augmentation technique, are effective for improving recommendation performance.

4. Discussion

The recommender system is an essential part of an e-commerce platform to improve user experience and satisfaction. Most conventional recommendation models utilized user and item interaction information and side-information such as users' explicit feedback and user profiles [3–8]. However, in many cases, users look around internet shopping websites without logging in and some users share the same account even if they logged in. Thus, the session-based recommender system, which utilizes anonymous session sequences to recommend items, has gained much attention.

Session-based recommendation aims to predict an anonymous user's next action, whether she or he is likely to purchase based on the user's clicked-item in the current session, as sequences. Most recent research on session-based recommendations makes predictions solely based on a single session without incorporating global relationships between sessions [15–19,22]. We argue that this method does not guarantee better performance because item embeddings learned by utilizing a single session (inter-session) have less item transition information than utilizing both intra- and inter-session item transitions simultaneously. Some existing methods [21–27] tried to enhance recommendation performance by adopting memory modules and global transition graphs; however, they still carry limitations in terms of computation cost. We proposed a novel yet simple algorithm called Logit Averaging (LA), helping to learn both local-level logits which come from intra-sessions' item transitions and global-level logits, which come from gathered logits of related sessions. By considering the long-tail item distribution of session datasets, we stored and aggregated the logits of sessions with target items that the majority of other sessions shared. Then, we applied the mean value of logits as a global context that a single session might miss. Extensive experiments verified that the baseline models with our proposed method, LA (Logit Averaging), outperformed the baseline models. The total gains of LA with baselines on the baselines are all larger than zero, so we demonstrated that the LA module helps to increase both recommendation accuracy and diversity. Addtionally, we conducted additional experiments to confirm the improvement of the proposed methods with existing augmentation methods. The baseline methods with LA and GraphMix [36] showed the largest increment for all metrics.

Although we proposed the novel and simple method to overcome the challenges where session-based recommendation employs global item relations among the sessions, there are still a few limitations. In [9,24], sessions have temporal features, and user and items interactions and sessions represent consumption trends. Therefore, considering the temporal feature of sessions while collecting the global item transitions remains for our future work. Besides, the complex inter-relationships between sessions are still not entirely understood, and we think this is a fertile space for future exploration.

Author Contributions: Conceptualization, H.Y. and G.K.; methodology, H.Y. and G.K.; software, H.Y. and G.K.; formal analysis, H.Y. and G.K. and J.-H.L.; investigation, H.Y. and G.K.; data curation, H.Y. and G.K.; writing—original draft preparation, H.Y. and G.K.; writing—review and editing, H.Y. and G.K.; visualization, H.Y.; supervision, J.-H.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by MSIT (Ministry of Science and ICT), Korea, under the ICT Creative Consilience program IITP-2021-2020-0-01821 and Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. 2019-0-00421, Artificial Intelligence Graduate School Program (Sungkyunkwan University).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Publicly available datasets were analyzed in this study. RetailRocket dataset can be found here: [https://www.kaggle.com/retailrocket/ecommerce-dataset, accessed on 22 April 2022], Tmall dataset can be found here: [https://ijcai-15.org/repeat-buyers-prediction-competition/, accessed on 22 April 2022], Yoochoose dataset can be found here: [https://www.kaggle.com/chadgostopp/recsys-challenge-2015, accessed on 22 April 2022], Diginetica dataset can be found here: [https://competitions.codalab.org/competitions/11161#learn_the_details-data2, accessed on 22 April 2022].

Acknowledgments: This research was supported by the MSIT (Ministry of Science and ICT), Korea, under the ICT Creative Consilience program (IITP-2021-2020-0-01821) supervised by the IITP (Institute for Information & communications Technology Planning & Evaluation). This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. 2019-0-00421, Artificial Intelligence Graduate School Program (Sungkyunkwan University).

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A

Appendix A.1

Table A1 shows the raw result value of different data augmentation methods in four different datasets and four different baseline models. For data augmentation experiment, we use NARM [11], SRSAN [12], NISER [16], and TAGNN++ [18] to evaluate the performance.

Table A1. The result of different data augmentation methods in four different models. In column LA, we mark 'X' to indicate the baseline without using LA and mark 'O' to indicate the baseline with using LA.

Dataset	**Model**	**Aug**	**LA**	**HR@10**	**MRR@10**	**Coverage@10**	**HR@20**	**MRR@20**	**Coverage@20**
RetailRocket	NARM	X	X	39.24	20.76	95.52	45.92	21.24	97.63
		X	O	39.02	21.07	95.86	45.64	21.53	97.80
		GraphMix	O	37.89	19.46	95.99	44.76	19.94	98.14
		FLAG	O	39.94	21.16	95.54	46.21	21.64	97.75
		Random deletion	O	39.83	19.48	95.18	46.68	19.95	97.50
		Random Insertion	O	38.97	19.42	94.88	46.41	19.94	97.17
	NISER	X	X	33.24	22.24	88.66	37.54	22.53	93.42
		X	O	33.33	22.09	88.59	37.66	22.39	93.39
		GraphMix	O	33.19	22.37	85.65	37.49	22.66	90.79
		FLAG	O	33.40	22.45	89.56	37.76	22.75	94.40
		Random deletion	O	34.71	17.51	96.44	39.45	17.83	98.57
		Random Insertion	O	32.27	16.35	96.80	36.40	16.64	98.83
	SR-SAN	X	X	37.96	17.25	93.97	44.91	17.74	96.53
		X	O	37.50	17.26	92.44	44.33	17.75	95.45
		GraphMix	O	33.89	20.13	80.22	39.14	20.49	84.66
		FLAG	O	37.76	17.17	93.32	44.56	17.65	96.26
		Random deletion	O	37.42	17.01	91.90	45.57	17.58	94.86
		Random Insertion	O	35.06	15.97	90.53	42.61	16.48	94.10
	TAGNN++	X	X	34.30	21.54	81.59	39.83	21.92	85.47
		X	O	34.10	21.71	82.23	39.61	22.09	86.21
		GraphMix	O	38.00	23.36	90.96	43.58	23.75	93.47
		FLAG	O	38.35	23.18	89.47	44.47	23.61	92.62
		Random deletion	O	36.07	19.51	82.40	42.93	19.98	86.46
		Random Insertion	O	31.59	16.80	77.42	38.29	17.27	82.76
Tmall	NARM	X	X	17.14	12.52	17.68	19.18	12.71	28.68
		X	O	16.46	12.21	17.28	18.50	12.35	28.17
		GraphMix	O	19.18	13.74	18.56	21.62	13.88	30.85
		FLAG	O	16.65	12.41	17.81	18.70	12.51	28.84
		Random deletion	O	18.31	13.25	18.23	19.86	13.38	29.60
		Random Insertion	O	18.70	12.91	18.50	20.74	13.05	30.28
	NISER	X	X	18.89	14.04	18.93	21.03	14.18	31.83
		X	O	18.11	13.52	19.21	20.35	13.60	32.43
		GraphMix	O	19.08	13.76	18.65	20.45	13.85	31.08
		FLAG	O	19.18	13.80	19.03	21.13	13.93	32.01
		Random deletion	O	18.31	13.58	18.82	19.77	13.68	31.97
		Random Insertion	O	16.46	12.66	18.86	18.21	12.78	31.83
	SR-SAN	X	X	18.99	12.74	12.67	21.62	12.92	19.65
		X	O	20.55	14.25	12.63	23.17	14.43	19.54
		GraphMix	O	22.40	14.65	16.27	24.83	14.83	26.07
		FLAG	O	19.28	12.62	12.49	22.01	12.81	19.30
		Random deletion	O	19.28	11.84	13.12	22.59	12.03	20.45
		Random Insertion	O	17.92	10.55	13.00	20.84	10.76	20.12
	TAGNN++	X	X	33.11	16.17	15.88	39.82	16.64	25.01
		X	O	30.77	14.55	16.36	36.42	14.96	25.42
		GraphMix	O	34.27	17.11	17.50	43.33	17.76	28.06
		FLAG	O	34.86	15.38	16.56	45.47	16.14	26.01
		Random deletion	O	30.48	14.17	16.75	36.51	14.60	26.20
		Random Insertion	O	21.81	10.59	15.44	26.58	10.94	23.52
Yoochoose1/64	NARM	X	X	59.95	29.46	29.24	70.67	30.20	35.73
		X	O	59.96	29.65	28.88	70.61	30.40	35.37
		GraphMix	O	55.69	28.08	36.91	66.66	28.85	45.52
		FLAG	O	60.50	29.57	29.83	71.07	30.33	35.89
		Random deletion	O	59.99	29.38	29.17	70.54	30.13	35.75
		Random Insertion	O	60.49	29.68	30.50	71.00	30.42	37.09
	NISER	X	X	60.25	30.91	29.46	70.55	31.64	34.61
		X	O	60.23	30.88	29.40	70.48	31.60	34.73
		GraphMix	O	60.58	31.08	28.65	70.54	31.78	33.29
		FLAG	O	60.45	30.77	29.91	70.71	31.49	35.34
		Random deletion	O	60.36	29.04	31.82	70.41	29.74	38.69
		Random Insertion	O	59.88	29.18	34.17	69.88	29.88	42.22
Yoochoose 1/64	SR-SAN	X	X	53.90	25.96	28.28	64.89	26.73	32.62
		X	O	53.75	26.00	28.31	64.91	26.77	32.63
		GraphMix	O	54.87	26.48	30.19	66.14	27.27	34.88
		FLAG	O	54.31	26.10	28.40	65.24	26.86	32.86
		Random deletion	O	55.54	25.85	28.53	67.21	26.65	32.61
		Random Insertion	O	52.28	23.36	28.99	65.54	24.27	33.08
	TAGNN++	X	X	60.70	30.63	26.48	70.93	31.35	30.72
		X	O	60.84	31.11	28.79	71.16	31.83	33.46
		GraphMix	O	60.75	30.86	27.64	71.15	31.59	32.27
		FLAG	O	60.67	30.95	27.82	71.08	31.68	31.72
		Random deletion	O	58.32	29.43	16.79	68.79	29.17	18.96
		Random Insertion	O	57.32	27.88	17.03	68.44	28.65	19.12

Table A1. *Cont.*

Dataset	Model	Augmentation	LA	HR@10	MRR@10	Coverage@10	HR@20	MRR@20	Coverage@20
Diginetica	NARM	X	X	35.86	18.60	68.78	44.83	19.22	82.60
		X	O	35.76	18.51	69.05	44.49	19.12	83.07
		GraphMix	O	34.76	18.38	69.02	43.39	18.97	83.16
		FLAG	O	36.30	18.49	69.06	45.40	19.12	82.91
		Random deletion	O	35.96	18.61	68.95	45.32	19.25	82.66
		Random Insertion	O	36.59	18.71	68.31	45.59	19.36	82.13
	NISER	X	X	34.12	17.98	65.02	41.89	18.52	77.34
		X	O	33.79	18.02	65.28	41.68	18.57	78.25
		GraphMix	O	34.27	18.06	59.23	42.04	18.59	70.59
		FLAG	O	34.04	18.10	65.41	41.69	18.63	78.41
		Random deletion	O	33.55	18.37	64.18	39.95	18.81	77.65
		Random Insertion	O	33.11	18.28	64.04	39.68	18.74	77.69
	SR-SAN	X	X	32.54	16.61	67.60	41.50	17.22	78.56
		X	O	32.93	16.93	66.68	42.02	17.56	77.34
		GraphMix	O	32.94	16.95	70.66	41.72	17.56	81.55
		FLAG	O	32.62	16.62	67.57	41.28	17.22	78.39
		Random deletion	O	32.07	14.85	67.21	42.44	15.58	79.28
		Random Insertion	O	31.50	14.04	66.78	41.87	14.76	79.36
	TAGNN++	X	X	33.90	17.34	62.81	42.64	17.95	74.00
		X	O	36.42	18.31	68.85	45.41	18.93	79.83
		GraphMix	O	36.36	18.43	70.89	45.02	19.03	80.91
		FLAG	O	33.77	17.36	62.54	42.29	17.96	73.66
		Random deletion	O	34.17	15.37	66.93	43.61	16.03	78.87
		Random Insertion	O	34.20	18.02	63.99	41.97	18.56	76.07

Appendix A.2

Table A2 shows the hyperparameter settings that we executed. All the setting is same from the original implementation codes that authors of the paper published. For Yoochoose1/64 and Diginetica which have much larger train sessions, some modification on batch size in TAGNN++ were necessary. Otherwise, the memory error occurred. Especially when experimenting Random Deletion and Random Insertion augmentation techniques, batch size of Yoochoose1/64 was 8 and 16 for diginetica.

Table A2. The hyperparameter settings of six baseline models.

Hyperparameter	Model					
	NARM	EOPA	NISER	SRGNN	SRSAN	TAGNN++
Batch Size	128	128	128	128	128	64
Hidden Size	100	-	100	100	96	100
Epoch	100	30	30	30	30	30
L2 Penalty	-	0.0001	0.00001	0.00001	0.00001	0.00001
Learning rate	0.001	0.001	0.001	0.001	0.001	0.001
Learning rate decay rate	0.1	-	0.1	0.1	0.1	0.1
Number of steps after which the learning rate decay	80	-	3	3	3	3
The dropout ratio for features	-	0.2	-	-	-	-
GNN Propagation	-	-	1	1	1	1
Number of SAN layer	-	-	-	-	1	-
Number of heads of multi-head attention	-	-	-	-	2	-
Multipler of hidden size	-	-	-	-	1	-
Embedding Dimension	50	32	-	-		-
Number of layers	1	3	-	-	-	-

Appendix A.3

Figure A1 shows the performance differences depending on the various head-tail ratio setting. We examine the performance using SRGNN model with Diginetica dataset. The legend in each plot represents the recommendation performance of metrics. We found that the performance on both HR and MRR depending on the head-tail ratio do not differ very much. The performance differences were within ± 1, meaning that head-tail ratio are not significant hyperparmeter in Logit Averaging method.

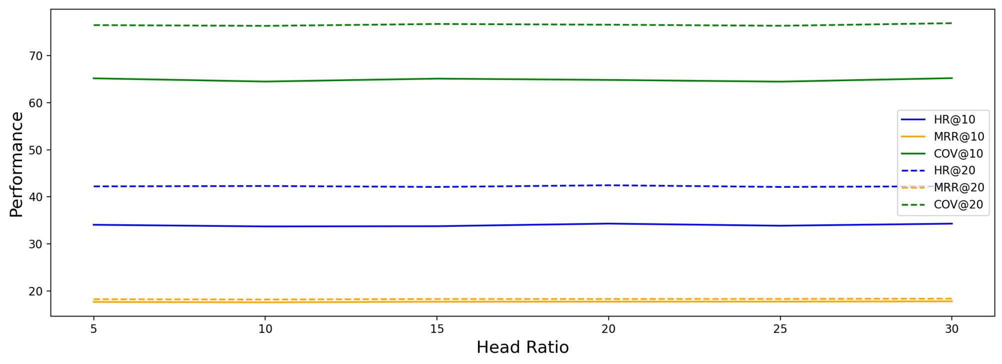

Figure A1. We compare the performance of LA in various head ratio from 5% to 30%. The legend shows the metrics. We examine the performance in SRGNN model using Digineitca dataset.

References

1. Zhang, S.; Yao, L.; Sun, A.; Tay, Y. Deep learning based recommender system: A survey and new perspectives. *arXiv* **2019**, arXiv:1707.07435.
2. Covington, P.; Adams, J.; Sargin, E. Deep neural networks for youtube recommendations. In Proceedings of the 10th ACM Conference on Recommender Systems, Boston, MA, USA, 15–19 September 2016; pp. 191–198.
3. Sarwar, B.; Karypis, G.; Konstan, J.; Riedl, J. Item-Based Collaborative Filtering Recommendation Algorithms. In Proceedings of the 10th International Conference on World Wide Web, Hong Kong, China, 1–5 May 2001; pp. 285–295.
4. Rendle, S.; Freudenthaler, C.; Schmidt-Thieme, L. Factorizing Personalized Markov Chains for Next-Basket Recommendation. In Proceedings of the 19th International Conference on World Wide Web, Raleigh, NC, USA, 26–30 April 2010; pp. 811–820.
5. He, X.; Liao, L.; Zhang, H.; Nie, L.; Hu, X.; Chua, T. Neural Collaborative Filtering. In Proceedings of the 26th International Conference on World Wide Web, Perth, Australia, 3–7 April 2017; pp. 173–182.
6. Chen, Y. ; de Rijke, M. A collective variational autoencoder for top-n recommendation with side information. In Proceedings of the 3rd Workshop on Deep Learning for Recommender Systems, Vancouver, BC, Canada, 6 October 2018; pp. 3–9.
7. Dong, X.; Yu, L.; Wu, Z.; Sun, Y.; Yuan, L.; Zhang, F. A hybrid collaborative filtering model with deep structure for recommender systems. In Proceedings of the AAAI Conference on Artificial Intelligence, San Francisco, CA, USA, 4–9 February 2017.
8. Liu, F.; Cheng, Z.; Zhu, L.; Gao, Z.; Nie, L. Interest-aware message-passing gcn for recommendation. In Proceedings of the Web Conference 2021, Ljubljana, Solvenia, 19–23 April 2021; pp. 1296–1305.
9. Choi, M.; Kim, J.; Lee, J.; Shim, H.; Lee, J. Session-Aware Linear Item-Item Models for Session-Based Recommendation. In Proceedings of the Web Conference 2021, Ljubljana, Slovenia, 19–23 April 2021; pp. 2186–2197.
10. Wang, S.; Cao, L.; Wang, Y.; Sheng, Q.Z.; Orgun, M.A.; Lian, D. A Survey on Session-Based Recommender Systems. *arXiv* **2021**, arXiv:1902.04864.
11. Li, J.; Ren, P.; Chen, Z.; Ren, Z.; Lian, T.; Ma, J. Neural attentive session-based recommendation. In Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, Singapore, 6–10 November 2017; pp. 1419–1428.
12. Fang, J. Session-based Recommendation with Self-Attention Networks. *arXiv* **2021**, arXiv:2102.01922.
13. Cho, K.; Merrienboer, V.B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; Bengio, Y. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, Doha, Qatar, 25–29 October 2014; pp. 1724–1734.
14. Kipf, T. N.; Welling, M. Semi-Supervised Classification with Graph Convolutional Networks. In Proceedings of the 5th International Conference on Learning Representations, Toulon, France, 24–26 April 2017.
15. Wu, S.; Tang, Y.; Zhu, Y.; Wang, L.; Xie, X.; Tan, T. Session-based recommendation with graph neural networks. In Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence, Honolulu, HI, USA, 27 January–1 February 2019.
16. Gupta, P.; Garg, D.; Malhotra, P.; Vig, L.; Shroff, G. NISER: Normalized item and session representations to handle popularity bias. In Proceedings of the 1st International Workshop on Graph Representation Learning on Its Applications (CIKM '19), Beijing, China, 3–7 November 2019.
17. Chen, T.; Wong, R.C.W. Handling information loss of graph neural networks for session-based recommendation. In Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Virtual Event, 23–27 August 2020; pp. 1172–1180.
18. Mitheran, S.; Java, A.; Sahu, S.K.; Shaikh, A. Introducing Self-Attention to Target Attentive Graph Neural Networks. *arXiv* **2022**, arXiv:2107.01516.
19. Li, A.; Cheng, Z.; Liu, F.; Gao, Z.; Guan, W.; Peng, Y. Disentangled Graph Neural Networks for Session-based Recommendation. *arXiv* **2022**, arXiv:2201.03482.
20. Li, Y.; Tarlow, D.; Brockschmidt, M.; Zemel, R.S. Gated Graph Sequence Neural Networks. In Proceedings of the 4th International Conference on Learning Representations, San Juan, Puerto Rico, 2–4 May 2016.

21. Wang, M.; Ren, P.; Mei, L.; Chen, Z; Ma, J.; de Rijke, M. 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, Paris, France, 21–25 July 2019; pp. 345–354.
22. Wang, Z.; Wei, W.; Cong, G.; Li, X.L.; Mao, X.L.; Qiu, M. Global context enhanced graph neural networks for session-based recommendation. In Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, Virtual Event, 25–30 July 2020; pp. 169–178.
23. Mi, F.; Faltings, B. Memory Augmented Neural Model for Incremental Session-Based Recommendation. In Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, Yokohama, Japan, 7–15 January 2020; pp. 2169–2176.
24. Zhou, H.; Tan, Q.; Huang, X.; Zhou, K.; Wang, X. Temporal Augmented Graph Neural Networks for Session-Based Recommendations. In Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, Virtual Event, 11–15 July 2021; pp. 1798–1802.
25. Huang, C.; Xia, X.; Dai, C.; Bo, Z. Graph-Enhanced Multi-Task Learning of Multi-Level Transition Dynamics for Session-based Recommendation. In Proceedings of the 35th AAAI Conference on Artificial Intelligence, Virtual Event, 2–9 February 2021; pp. 4123–4130.
26. Xia, X.; Yin, H.; Yu, J.; Shao, Y.; Cui, L. Self-Supervised Graph Co-Training for SEssion-based Recommendation. *Assoc. Comput. Mach.* **2021**, *11*, 2180–2190.
27. Pang, Y.; Wu, L.; Shen, Q.; Zhang, Y.; Wei, Z.; Xu, F.; Chang, E.; Long, B.; Pei, J.; Heterogeneous global graph neural networks for personalized session-based recommendation. In proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining, Virtual Event, 21–25 February 2022; pp. 775–783.
28. Menon, A.; Jayasumana, S.; Rawat, A.; Jain, H.; Veit, A.; Kumar, S. Long-tail learning via logit adjustment. In Proceedings of the 2021 International Conference on Learning Representation, Virtual Conference, 3–7 May 2021.
29. Hurley, N.; Zhang, M. Novelty and Diversity in Top-N Recommendation—Analysis and Evaluation. *ACM Trans. Internet Technol.* **2011**, *10*, 14. https://doi.org/10.1145/1944339.1944341
30. Bradley, K.; Smyth, B. Improving recommendation diversity. In Proceedings of the Twelfth Irish Conference on Artificial Intelligence and Cognitive Science, Maynooth, Ireland, 1 September 2001; pp. 141–152.
31. Miyamoto, S.; Zamami, T.; Yamana, H. Appearance frequency-based ranking method for improving recommendation diversity. In Proceedings of the 2019 IEEE 4th International Conference on Big Data Analytics (ICBDA), Suzhou, China, 15–18 March 2019; pp. 420–425
32. Wang, L.; Zhang, X.; Wang, T.; Wan, S.; Srivastava, G.; Pang, S.; Qi, L. Diversified and scalable service recommendation with accuracy guarantee. *IEEE Trans. Comput. Soc. Syst.* **2020**, *8*, 1182–1193. https://doi.org/10.1109/TCSS.2020.3007812.
33. Anderson, C. *The Long Tail: Why the Future of Business is Selling Less of More*; Hachette Books: London, UK, 2006.
34. Park, Y.J.; Tuzhilin, A. The long tail of recommender systems and how to leverage it. In Proceedings of the 2008 ACM Conference on Recommender Systems, Lausanne, Switzerland, 23–25 October 2008; pp. 11–18.
35. Isufi, E.; Pocchiari, M.; Hanjalic, A.T. Accuracy-diversity trade-off in recommender systems via graph convolutions. *Inf. Process. Manag.* **2021**, *58*, 102459. https://doi.org/10.1016/j.ipm.2020.102459.
36. Wang, Y.; Wang, W.; Liang, Y.; Cai, Y.; Hooi, B. Mixup for Node and Graph Classification. In Proceedings of the Web Conference 2021, Ljubljana, Slovenia, 12–16 April 2021; pp. 3663–3674.
37. Kong, K.; Li, G.; Ding, M.; Wu, Z.; Zhu, C.; Ghanem, B.; Taylor, G.; Goldstein, T. Flag: Adversarial data augmentation for graph neural networks. *arXiv* **2020**, arXiv:2010.09891.