Memory Bank Augmented Long-tail Sequential Recommendation

Yidan Hu
yidan001@e.ntu.edu.sg
School of Computer Science and Engineering & Joint NTU-UBC Research Centre of Excellence in Active Living for the Elderly
Nanyang Technological University
Singapore

Yong Liu
stephenliu@ntu.edu.sg
Joint NTU-UBC Research Centre of Excellence in Active Living for the Elderly
Nanyang Technological University
Singapore

Chunyan Miao*
ascymiao@ntu.edu.sg
School of Computer Science and Engineering & Joint NTU-UBC Research Centre of Excellence in Active Living for the Elderly
Nanyang Technological University
Singapore

Yuan Miao
Yuan.Miao@vu.edu.au
Institute for Sustainable Industries & Liveable Cities (ISILC) & College of Engineering and Science
Victoria University
Melbourne, Australia

ABSTRACT

The goal of sequential recommendation is to predict the next item that a user would like to interact with, by capturing her dynamic historical behaviors. However, most existing sequential recommendation methods do not focus on solving the long-tail item recommendation problem that is caused by the imbalanced distribution of item data. To solve this problem, we propose a novel sequential recommendation framework, named MASR (*i.e.*, Memory Bank Augmented Long-tail Sequential Recommendation). MASR is an "Open-book" model that combines novel types of memory banks and a retriever-copy network to alleviate the long-tail problem. During inference, the designed retriever-copy network retrieves related sequences from the training samples and copies the useful information as a cue to improve the recommendation performance on tail items. Two designed memory banks provide reference samples to the retriever-copy network by memorizing the historical samples appearing in the training phase. Extensive experiments have been performed on five real-world datasets to demonstrate the effectiveness of the proposed MASR model. The experimental results indicate that MASR consistently outperforms baseline methods in terms of recommendation performance on tail items.

CCS CONCEPTS

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

*Corresponding author.

KEYWORDS

Sequential recommendation; Long-tail; Memory bank

ACM Reference Format:
Yidan Hu, Yong Liu, Chunyan Miao, and Yuan Miao. 2022. Memory Bank Augmented Long-tail Sequential Recommendation. In *Proceedings of the 31st ACM International Conference on Information and Knowledge Management (CIKM '22), October 17–21, 2022, Atlanta, GA, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3511808.3557391

1 INTRODUCTION

Sequential recommendation systems exploit users' sequential behavior patterns from their historical behavior data to predict the next items they would like to interact with [27, 45, 47]. In practice, different deep learning techniques, *e.g.*, Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), and Graph Neural Networks (GNN), have been applied to improve the performance of sequential recommendation systems [19, 27, 30, 50, 53]. Although these methods usually achieve state-of-the-art performance, the sequential recommendation is still a very challenging task, due to the item frequency following the long-tail distribution. Specifically, one important statistical characteristic of users' behavior data is that most of the tail items only receive a little user feedback, while users usually give lots of feedback on popular items (*i.e.*, head items). The long-tail training data will cause conventional models to prefer recommending popular items and ignore items with little user feedback. Thus, conventional sequential recommendation models trained on such imbalanced user behavior data usually perform poorly on tail items [5, 24, 41].

In the literature, various solutions have been proposed to mitigate the long-tail problems in machine learning tasks. For example, some works [3, 4, 17, 25, 49] use re-sampling strategies, *i.e.*, undersampling and over-sampling, to modify the data distribution. Some other studies [6, 9, 12, 23] employ re-weighting strategies to assign higher weights to the loss values of tail samples. In recommendation research, the long-tail recommendation problems can also be handled by information augmentation methods, *e.g.*, MIRec [61], which transfers the learned knowledge from head items to improve the representations of tail items in model training. Moreover, there also exist other module improvement methods. For example, [31] incorporates the bi-lateral branch network to learn representations and the classifier, and [5] enhances the tail item representations with the representations of co-occurrence items.

These existing methods follow the common training and testing strategy, called "Closed-book". In this strategy, the training data is used solely to learn model parameters to maximize probabilities of ground-truth. Then, the recommendation performance is evaluated based on the testing dataset without extra information. However, due to the nature of learning objective functions, little knowledge about rare samples (*e.g.*, tail items) can be learned, and thus lead to bad generalization performance on them.

In Natural Language Processing (NLP) research, several recent studies [39, 63] use "Open-book" strategies to solve NLP problems. They utilize related training samples as hints in the inference stage to improve the model performance. Inspired by the success of these methods, we propose a novel "Open-book" method, named MASR (*i.e.*, Memory Bank Augmented Long-tail Sequential Recommendation) to solve the long-tail problem in sequential recommendation. MASR aggregates long-tail distribution knowledge in the training phase via memory bank components that are then referenced. Thus, the recommendation performance on head and tail items can both be significantly improved.

More specifically, MASR uses a retriever-copy network to search the top-K most relevant sequences for the given user. The retrieved representations of sequences and their labels are used as extra hints to enhance the recommendation performance. Compared to the "Closed book" training strategy, the proposed MASR model can directly refer to relevant sequences and corresponding recommendation results in the whole training set. Moreover, the memory bank provides a viable way to enable a retriever-copy network. Vanilla memory bank [57] is a Least Recently Used (LRU) cache that consists of pairs. Each pair contains a representation of the item sequence and its ground-truth label. As vanilla memory banks store recently sampled items, head items will predominate, and the effects of tail items will be overwhelmed in the case of a long-tail distribution. To address this issue, we design two balanced memory banks, *i.e.*, cluster-wise memory bank and centroid-wise memory bank, to ensure that all items appear in the memory bank with a balanced number of samples. With the balanced memory banks, rare items can be memorized and retrieved effectively. In addition, a dual channel consisting of the retriever-copy network and memory bank is utilized to deal with the long-tail problem and ensure that the performance of the head item is not degraded.

In this work, we have made the following contributions.

- We propose a novel recommendation framework, namely MASR, to solve the long-tail problem in sequential recommendation. Specifically, MASR includes a head channel and a tail channel to improve the recommendation performance on head and tail items, respectively.
- We design the cluster-wise and centroid-wise memory banks to store historical training samples and ensure a balanced number of samples among all items. The retriever-copy network is also developed to ensure the information in historical data samples can be effectively exploited in both tail and head channels.
- We propose a Randomly sampled Memory Contrastive Loss (RanMCL) to improve the quality of tail item representations effectively. RanMCL tries to reduce the variance of feature distribution of each tail item by making features that have the same labels to be close, while features that have different labels be far.
- To demonstrate the effectiveness of the proposed MASR model, we perform extensive experiments on five real-world datasets. The experimental results demonstrate that MASR outperforms state-of-the-art baseline methods in terms of recommendation accuracy on both head and tail items.

2 RELATED WORK

2.1 Sequential Recommendation

With the quick development of deep learning, RNN [15, 33] and its variants, *e.g.*, Long short-Term Memory (LSTM) [11, 21] and Gated Recurrent Unit (GRU) [8, 10] were applied to encode the item sequence into a vector and then predict the next item. For example, [20] introduced a parallel RNN architecture to model sessions based on items and their features which are extracted from visual information. [19] proposed a GRU-based method that introduced session-parallel mini-batches for the session-based recommendation. Besides the RNN-based methods, CNN can also capture the sequential dependency by treating the interaction matrix as an "image"[53, 59]. For example, Caser [53] was proposed to capture general preference and sequential patterns over the item sequence by convolutional filters. Motivated by the wide application of attention mechanism [55, 60] in NLP, BERT4Rec [50] was proposed to model user sequential behaviors by deep bidirectional self-attention. [64] proposed a transformer-based model to reduce the noise information in the frequency domain and improve the sequential recommendation performance. In addition, some recent works introduced GNN [7, 16, 62, 66] for sequential recommendation. However, these models do not consider the long-tail item distribution, which commonly exists in real-world datasets [61].

2.2 Long-tail Recommendation

In recommendation system research, some studies improve long-tail recommendation with the re-balancing solution. For example, [22] proposed a re-balancing solution to solve the long-tail problem by producing re-sampling weights directly to select samples. [42] proposed a Clustered Tail (CT) method, which groups tail items using clustering methods and then recommends tail items based on the ratings in the clusters. [48, 61] enhanced the tail item representations by transferring knowledge from many-shot items to few-shot items with meta-leaning. [31] adopted the bilateral branch network to improve the diversity of recommendation. [5] alleviated the long-tail problem by enhancing the item representation with co-occurrence items. Moreover, some other works [29, 56, 58] proposed multi-objective optimization and adversarial approach to improving the long-tail recommendation performance. However, none of these models are designed for sequential recommendation tasks. [37] proposed a soft logit adjustment method, named TailNet, based on the preference mechanism. However, the performance improvements on tail items will hurt the performance on the head items. Recently, [24] proposed the CITIES model that addresses the long-tail problem by enhancing the embedding of tail items with

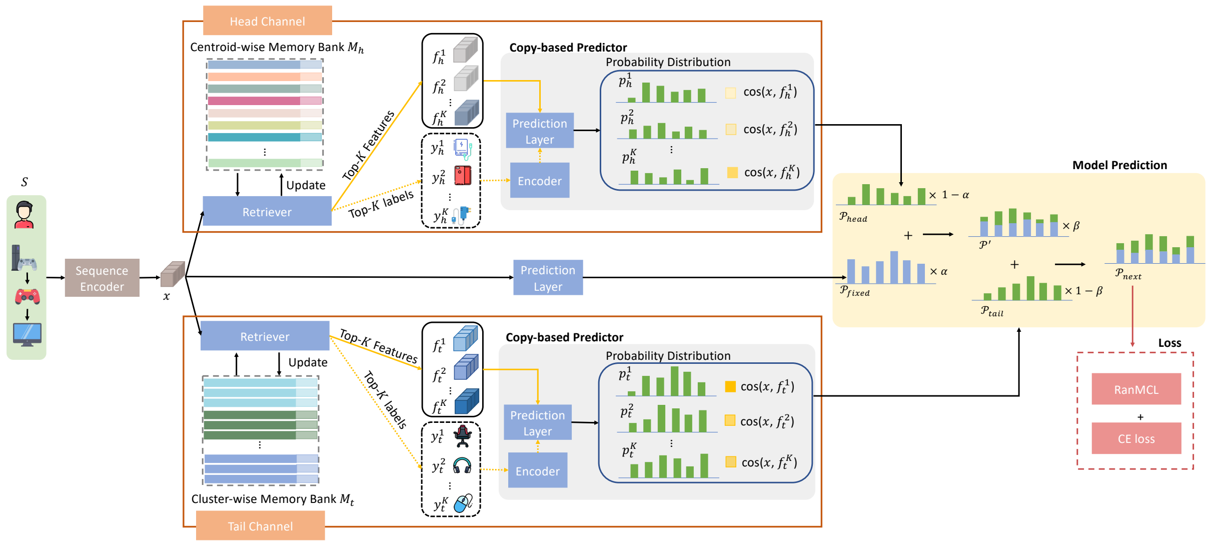


Figure 1: The framework of the proposed MASR model. The yellow dash line represents the workflow of the label-based copy mechanism, and the yellow line represents that of the feature-based copy mechanism. The subscript "h" and "t" denote the head channel and the tail channel respectively.

context information. However, CITIES requires multi-stage training, which introduces difficulties in deployment on real-world recommender systems. [28] proposed a solution that performs K-means clustering and relocates consumed tail items to provide pseudo labels, which will be used to train the model and perform recommendations. However, it can only recommend a cluster of target tail items instead of an individual item. Besides, the content information (*i.e.*, blog article and its title) is required in [28], while our model does not assume the existence of such auxiliary information.

2.3 Memory Bank

Memory banks can memorize the data samples that occurred in the recent iterations and have shown power in various computer vision tasks [14, 34, 35]. Moreover, it has also been widely used in recommendation systems for different purposes [1, 13, 26, 36, 44]. For example, several previous works [1, 13, 36, 54] use the memory bank to store historical data samples for long-term knowledge and utilize the attention mechanism in the retrieval stage. MMCF [26] proposed a deep multiplex memory network to jointly capture fine-grained preferences and context information. MMInfoRec [44] provided a sequential recommendation model which uses memory to solve the sparse training signal issue. DMAN [51] employed an external memory network to capture the long-term interests of users. However, these models can not solve the long-tail problem.

3 THE PROPOSED MASR FRAMEWORK

Let $\mathcal{U}$ denote the set of users, $\mathcal{I}$ denote the set of items, and $S = \{s_1, s_2, \cdots, s_m\}$ denote the list of m items consumed by a user, where the $s_m \in \mathcal{I}$. In general, the objective of a sequential recommendation model is to predict the next item s_{m+1} that will be interacted by the user at the next timestamp, given her historical interaction sequence S. It can be formulated as $\mathcal{P}_s = (y = s_{m+1}|S)$, where y denote the label of sequence S. In practice, users' historical interaction data in real-world recommendation scenarios often follow a long-tail distribution, where popular items often attract much more user feedback than tail items. This usually leads to weak representations of tail items and decreases the recommendation performance on tail items. In this work, we propose the MASR model to enhance the sequential recommendation performance on tail items without hurting the performance on head items.

Figure 1 shows the overall structure of MASR. In summary, MASR has the following main components.

- **Sequence Encoder**: The goal of the sequence encoder $f_e(\cdot)$ is to obtain the feature representation $x = f_e(S)$ of a given input sequence S. In this work, we choose BERT4Rec [50] as the sequence encoder, which usually achieves state-of-the-art performance in sequential recommendation tasks. Note that other sequential recommendation models, *e.g.*, GRU4Rec [19], SASRec [27], and HGN [38] can also be used as the sequence encoder.
- **Head and Tail Channels**: Both channels consist of a memory bank and a retriever-copy network. The only difference is the type of memory bank involved. There are two types of balanced memory banks, *i.e.*, centroid-wise and cluster-wise memory banks. They are proposed to memorize the historical data in the training dataset. During the inference, these two memory banks make it efficient to use the relevant training data as reference explicitly. The retriever-copy network includes a retriever and a copy-based predictor. The retriever module retrieves the representations of K most similar historical sequences along with their ground-truth labels from a memory bank. It aims to provide external hints by searching for similar user behavior sequences. Moreover, the copy-based predictor processes the information retrieved from the memory bank, to predict the probabilities of target items that are further used to enhance the model prediction. In this work, we propose two different copy mechanisms: 1) *feature-based copy mechanism* that employs the retrieved features to enhance the output, and 2) *label-based copy mechanism* that

uses the corresponding labels of the K most similar historical features for prediction.
- **Model Prediction**: It fuses the outputs of the sequence encoder, head channel, and tail channel to produce the final prediction $\mathcal{P}_{next}$ of interaction probabilities of candidate items.

Next, we introduce the details of the memory bank, retriever-copy network, and model prediction.

3.1 Memory Bank

To efficiently summarize historical user behavior sequence data, we design two balanced memory banks, *i.e.*, centroid-wise and cluster-wise memory banks, which are used to memorize head items and tail items, respectively. The memory banks allow the proposed model to search representations of related historical sequences and their ground-truth labels from the past training iterations, even from the entire training stage.

3.1.1 Centroid-wise Memory Bank. The main idea is to generate a representer for each item. The building and updating operations are based on an extra vanilla memory bank. Specifically, let $X = \{(b_0, y_0), (b_1, y_1), \cdots, (b_i, y_i), \cdots, (b_{|X|}, y_{|X|})\}$ denote a vanilla memory bank, where b_i is the feature from the sequence encoder learned in the past iteration and y_i is the corresponding label. X is updated by inserting pair (b_i, y_i) during training. The size of a vanilla memory bank $|X|$ is predefined. Then, for each item, we maintain a running centroid (m_i, y_i) of the subset $B = \{(b_j, y_j)\}_{y_j = y_i}$ as follows,

$$m_i = \frac{1}{|B|} \sum_{(b_j, y_j) \in B} \frac{b_j}{\|b_j\|}, \tag{1}$$

where $\| \cdot \|$ means L2 normalization. However, the centroid construction is sensitive to noise when B is a small set, which is exactly the case of tail items. To solve this problem, we also propose a cluster-wise memory bank.

3.1.2 Cluster-wise Memory Bank. A cluster-wise memory bank is a key-value dictionary where keys are item IDs and values are vanilla memory banks with predetermined capacity N. As mentioned in Section 1, the vanilla memory bank is an LRU cache. Thus, the data balance is assured by keeping the most recent N records for each item. When adding a feature and its label to the cluster-wise memory bank, we first use the label as the key to find the corresponding vanilla memory bank, and the feature is then inserted into the vanilla memory bank.

3.2 Retriever-copy Network

3.2.1 Retriever. The retriever operates by fetching K most similar pairs with the feature x from a memory bank. Let $M = \{(m_i, y_i)\}_{i=1}^{|M|}$ denote the set of pairs in the memory bank. The memory bank can be centroid-wise or cluster-wise memory bank. We can calculate the similarity scores as follows,

$$C = \{\cos(x, m_i)\}_{i=1}^{|M|}, \tag{2}$$

where $\cos(\cdot)$ denotes the cosine similarity between two feature vectors. Then, we can obtain the K highest similarity scores $A = \{a_i\}_{i=1}^{K}$ and the corresponding pairs $Z = \{f_i, y_i\}_{i=1}^{K}$ which are selected from the M. Here, a_i is the cosine similarity score between x and f_i.

3.2.2 Copy-based Predictor. Copy mechanism shows effectiveness for NLP area to solve the out-of-vocabulary(OOV) issue in text generation tasks [65]. It can directly copy the existing sub-span to the target output from the input text. In this work, we extend copy mechanism to the long-tail sequential recommendation by copying the related features or labels from the retrieved candidates to improve the recommendation accuracy, especially for tail items. In this work, we propose the following two copy mechanisms.

- **Feature-based Copy Mechanism**: With a similar pair $(f_i, y_i) \in Z$, the interaction scores of candidate items P_i can be computed by a predictor layer as follows,

$$P_i = \text{softmax}(\mathbf{W}_f f_i + \mathbf{b}_f), \tag{3}$$

where $\mathbf{W}_f \in \mathbb{R}^{d \times |\mathcal{I}|}$ and $\mathbf{b}_f$ are parameters, and d is the dimension of the hidden state. The interaction scores $\mathcal{P}$ predicted by the Copy-based predictor is calculated by accumulating P_i as,

$$\mathcal{P} = \sum_{a_i \in A} a_i P_i. \tag{4}$$

- **Label-based Copy Mechanism**: In this mechanism, the interaction scores of candidate items are predicted based on the labels in the similar pairs Z. For a similar pair $(f_i, y_i) \in Z$, we denote the embedding of y_i by l_i. It is obtained by feeding y_i into the item embedding layer in the sequence encoder. Then, its probability over all possible items P_i is computed by a predictor layer,

$$P_i = \text{softmax}(\mathbf{W}_l l_i + \mathbf{b}_l), \tag{5}$$

where $\mathbf{W}_l \in \mathbb{R}^{d \times |\mathcal{I}|}$ and $\mathbf{b}_l$ are parameters. Compared to the one-hot embedding, P_i contains more information about the relevant items. The probability of Copy-based predictor $\mathcal{P}$ is also calculated via Eq. 4.

3.3 Model Prediction

The output interaction scores from the tail and head channels are denoted by $\mathcal{P}_{tail}$ and $\mathcal{P}_{head}$, respectively. Moreover, the output interaction score $\mathcal{P}_{fixed}$ from the backbone sequence encoder is obtained by feeding feature x into a linear layer with softmax,

$$\mathcal{P}_{fixed} = \text{softmax}(\mathbf{W}_p x + \mathbf{b}_p), \tag{6}$$

where $\mathbf{W}_p \in \mathbb{R}^{d \times |\mathcal{I}|}$ and $\mathbf{b}_p$ are parameters. The interaction score of the next item $\mathcal{P}_{next}$ is a combination of $\mathcal{P}_{fixed}$, $\mathcal{P}_{head}$, and $\mathcal{P}_{tail}$. Firstly, we obtain the weight interaction score $\mathcal{P}'$ by weight summing $\mathcal{P}_{fixed}$ and $\mathcal{P}_{head}$ based on coefficient α as,

$$\mathcal{P}' = \alpha \mathcal{P}_{fixed} + (1 - \alpha) \mathcal{P}_{head}, \tag{7}$$

$$\alpha = \text{softmax}(\mathbf{W}_a [x; f] + \mathbf{b}_a), \tag{8}$$

$$f = \sum_{a_i \in A} a_i f_i, \tag{9}$$

where $\mathbf{W}_a \in \mathbb{R}^{2d \times 1}$ and $\mathbf{b}_a$ are parameters, and $[\cdot ; \cdot]$ refers to concatenation operation. Next, we consider $\mathcal{P}_{tail}$ and obtain the final interaction score of the next item as follows,

$$\mathcal{P}_{next} = \beta \mathcal{P}' + (1 - \beta) \mathcal{P}_{tail}, \tag{10}$$

$$\beta = \text{softmax}(\mathbf{W}_b [\mathcal{P}'; \mathcal{P}_{tail}] + \mathbf{b}_b), \tag{11}$$

where $\mathbf{W}_b \in \mathbb{R}^{2|\mathcal{I}| \times 1}$ and $\mathbf{b}_b$ are parameters . Here, we compute β latter to enable the interaction score prediction to focus more on the output from the tail channel $\mathcal{P}_{tail}$.

Algorithm 1: The Training Algorithm of MASR

```
1  Inputs: user set $\mathcal{U}$, batch size $B$, interaction sequence $S_u$ of the
     user $u \in \mathcal{U}$, head channel memory bank $M_h$, tail channel memory
     bank $M_t$, sequence encoder $f_e(\cdot)$;
2  Sample a minibatch $\mathcal{B} = \{(S_0, y_0), ..., (S_B, y_B)\}$;
3  for each $(S, y) \in \mathcal{B}$ do
4      Compute $x$ through sequence encoder according to $x = f_e(S)$;
5      Retrieve top-$K$ similar pairs $Z_h$, $Z_t$ from the $M_h$, $M_t$ with
         input $x$;
6      if using label-based copy mechanism then
7          Calculate $P_i$ for head and tail channels respectively
             according to Eq. 5;
8      else
9          Calculate $P_i$ for head and tail channels respectively
             according to Eq. 3;
10     end
11     Calculate $\mathcal{P}$ for head and tail channels respectively according
         to Eq. 4;
12     Calculate $\mathcal{P}_{fixed}$, $\mathcal{P}_{next}$ according to Eq. 6 and Eq. 10;
13     if $y$ is a head item then
14         Update $M_h$ with $(x, y)$;
15     else
16         Update $M_t$ with $(x, y)$;
17     end
18 end
19 Calculate $\mathcal{L}_{\text{RanMCL}}$ and $\mathcal{L}_{\text{CE}}$ according to Eq. 12 and Eq. 13;
20 Use $\mathcal{L}_{\text{CE}}+\mathcal{L}_{\text{RanMCL}}$ to update model parameters;
```

3.4 Model Training

The insufficient user feedback on tail items may cause weak representations of tail items and degrade the retrieval accuracy. To mitigate this issue, we propose to improve item representations by Memory Contrastive Loss (MCL) [57] making the features for the same items close and features from different items far away. However, MCL is not computationally efficient, due to a large number of negative pairs. Thus, we propose a randomly sampled memory contrastive loss, *i.e.*, $\mathcal{L}_{\text{RanMCL}}$, to address this issue. In the $\mathcal{L}_{\text{RanMCL}}$, each mini-batch data share the same negative sampling to decrease the computation. The definition of $\mathcal{L}_{\text{RanMCL}}$ is as follows,

$$\mathcal{L}_{\text{RanMCL}} = \frac{1}{|\mathcal{B}|} \sum_{(x,y)\in\mathcal{B}} \Bigg[\sum_{(m_i,y_i)\in M'} \max(cos(x, m_i) - \lambda, 0) - \sum_{(m_j,y_j)\in M, y_j=y} cos(x, m_j) \Bigg], \tag{12}$$

where $\mathcal{B}$ is the minibatch data, M is the set of all pairs in the memory bank, λ is a hyperparameter. M' is a randomly sampled subset of M excluding items belonging to y, and the number of samples is a hyper-parameter. $\mathcal{L}_{\text{RanMCL}}$ is used on the tail channel. The features in the memory bank are not updated during stochastic gradient descent, only the feature x benefits from the $\mathcal{L}_{\text{RanMCL}}$. We also use the following cross-entropy loss to compare the predicted interaction scores and ground-truth labels,

$$\mathcal{L}_{\text{CE}} = -\sum_{u\in\mathcal{U}} \sum_{i\in O_i^u} \log(\mathcal{P}_{next}), \tag{13}$$

where O_i^u is the masked item set. During model training, we randomly mask some items to make the training process more effective. For example, assuming the original input sequence is $\{s_1, s_2, s_3, s_4\}$, and the input sequence becomes $\{s_1, [mask], s_3, [mask]\}$ after the masking operation. In this case, O_i^u is $\{s_2, s_4\}$. The entire framework can be effectively trained by minimizing the sum of $\mathcal{L}_{\text{CE}}$ and $\mathcal{L}_{\text{RanMCL}}$ in an end-to-end manner. The details of the training algorithm for MASR are shown in Algorithm 1.

3.5 Discussions

For the centroid-wise memory bank, the updating operations is based on an extra vanilla memory bank X. The time complexity of updating vanilla memory bank X and obtaining centroids based on the updated feature $x \in \mathbb{R}^d$ is $O(|X|d)$. Because the size of the centroid-wise memory bank is $|\mathcal{I}_h|d$, so the time complexity for retrieving top-K features is $O(|\mathcal{I}_h|d + |\mathcal{I}_h|logK)$. The space complexity is $O(|\mathcal{I}_h|d + |X|d)$. For the cluster-wise memory bank, the size of the memory bank is $(|\mathcal{I}_t|N)$. The time complexity of memory bank updating a feature $x \in \mathbb{R}^d$ is $O(d)$. The time complexity for retrieving top-K features is $O(|\mathcal{I}_t|Nd + |\mathcal{I}_t|NlogK)$. The space complexity is $O(|\mathcal{I}_t|Nd)$.

4 EXPERIMENTS

4.1 Experimental Datasets

Table 1: Statistics of the experimental datasets.

Dataset	# Users	# Items	# Feedback	Sparsity
ML-1M	6,040	3,706	1,000,209	95.53%
Musical	10,073	41,140	168,983	99.96%
Video	15,517	37,077	284,867	99.95%
Diginetica	26,018	78,586	361,150	99.98%
Yoochoose	115,639	24,105	1,812,801	99.93%

To evaluate the effectiveness of the proposed model, we conduct experiments on the following datasets: MovieLens-1M[1] (denoted by ML-1M), Amazon [18], Yoochoose[2], and Diginetica[3]. They are popular benchmark datasets used in sequential recommendation tasks. For the Amazon dataset, the "Musical Instruments" and "Video Games" subsets are selected for evaluation (denoted as Musical and Video respectively). We use the recent fractions 1/4 of the Yoochoose datasets as [37, 52]. These datasets suffer from long-tail and data imbalance problems, especially for the Amazon and Diginetica datasets where most items have less than 10 interaction records. We follow the prior works [27, 47, 50, 53] to perform the data pre-processing. For ML-1M and Amazon datasets, we treat all observed reviews or ratings as implicit feedback. For all datasets, we build an item sequence for each user by sorting all feedback based on the interaction timestamps. We remove users with fewer than 5 interactions. We split the data into training, validation, and test data using the leave-one-out method. For each user's historical sequence, the most recent interaction item is used for model testing, the second recent interaction item is used for model validation, and

[1]https://grouplens.org/datasets/movielens/1m/
[2]https://www.kaggle.com/datasets/chadgostopp/recsys-challenge-2015
[3]http://cikm2016.cs.iupui.edu/cikm-cup

Table 2: Recommendation performance achieved by different methods. The best results are in boldfaces.

		All Items				Head Items				Tail Items			
Dataset	Method	HR@20	N@20	HR@10	N@10	HR@20	N@20	HR@10	N@10	HR@20	N@20	HR@10	N@10
ML-1M	DMAN	0.2474	0.0995	0.1552	0.0763	0.3144	0.1263	0.1950	0.0964	0.0370	0.0126	0.0158	0.0075
	BERT4Rec	0.4753	0.2260	0.3605	0.1970	0.5362	0.2584	0.4093	0.2264	0.2816	0.1182	0.2062	0.0994
	BERT4Rec+FL	0.4549	0.2164	0.3415	0.1877	0.5048	0.2419	0.3823	0.2107	0.2861	0.1314	0.1996	0.1096
	BERT4Rec+AFL	0.4627	0.2190	0.3480	0.1899	0.5130	0.2469	0.3902	0.2158	0.2932	0.1280	0.2022	0.1049
	BERT4Rec+LG	0.4740	0.2256	0.3608	0.1970	0.5342	0.2581	0.4105	0.2269	0.2813	0.1179	0.2056	0.0989
	TailNet	0.4646	0.2023	0.3300	0.1683	0.5241	0.2313	0.3758	0.1938	0.2890	0.1199	0.1949	0.0962
	CITIES	0.4752	0.2258	0.3602	0.1968	**0.5365**	0.2583	0.4089	0.2261	0.2816	0.1183	0.2062	0.0995
	MASR	**0.4828**	**0.2381**	**0.3802**	**0.2121**	**0.5365**	**0.2676**	**0.4215**	**0.2385**	**0.3056**	**0.1422**	**0.2331**	**0.1239**
Musical	DMAN	0.1988	0.0948	0.1413	0.0804	**0.3303**	0.1579	**0.2350**	0.1339	0.0018	0.0005	0.0007	0.0002
	BERT4Rec	0.2301	0.1113	0.1627	0.0943	0.2978	0.1458	0.2142	0.1247	0.0195	0.0075	0.0106	0.0052
	BERT4Rec+FL	0.2426	0.1184	0.1717	0.1005	0.3166	0.1555	0.2260	0.1327	0.0182	0.0059	0.0096	0.0038
	BERT4Rec+AFL	0.2431	0.1187	0.1750	0.1016	0.3147	0.1551	0.2293	0.1337	0.0191	0.0068	0.0078	0.0039
	BERT4Rec+LG	0.2408	0.1182	0.1733	0.1013	0.3140	0.1552	0.2267	0.1332	0.0210	0.0086	0.0110	**0.0061**
	TailNet	0.2434	0.1202	0.1847	0.1053	0.3125	0.1543	0.2379	0.1354	0.0053	0.0018	0.0021	0.0010
	CITIES	0.1577	0.0677	0.1044	0.0543	0.2042	0.0878	0.1381	0.0712	0.0072	0.0020	0.0021	0.0008
	MASR	**0.2518**	**0.1234**	**0.1794**	**0.1052**	0.3241	**0.1594**	0.2323	**0.1363**	**0.0259**	**0.0092**	**0.0120**	0.0057
Video	DMAN	0.3160	0.1516	0.2308	0.1302	0.4365	0.2114	0.3221	0.1825	0.0620	0.0231	0.0351	0.0163
	BERT4Rec	0.3749	0.1862	0.2765	0.1613	0.4722	0.2367	0.3501	0.2059	0.0871	0.0347	0.0527	0.0261
	BERT4Rec+FL	0.3626	0.1830	0.2740	0.1607	0.4594	0.2337	0.3511	0.2065	0.0834	0.0361	0.0531	0.0284
	BERT4Rec+AFL	0.3757	0.1880	0.2835	0.1648	0.4763	0.2402	0.3619	0.2113	0.0753	0.0308	0.0462	0.0236
	BERT4Rec+LG	0.3743	0.1885	0.2794	0.1645	0.4721	0.2403	0.3565	0.2110	0.0844	0.0347	0.0505	0.0262
	TailNet	0.3797	0.1870	0.2835	0.1628	0.4913	0.2461	0.3726	0.2162	0.0541	0.0209	0.0277	0.0143
	CITIES	0.1431	0.0622	0.0940	0.0498	0.1769	0.0769	0.1177	0.0621	0.0335	0.0130	0.0165	0.0088
	MASR	**0.4060**	**0.2010**	**0.3022**	**0.1748**	**0.5090**	**0.2548**	**0.3841**	**0.2233**	**0.1076**	**0.0424**	**0.0654**	**0.0319**
Diginetica	DMAN	0.5968	0.3885	0.5364	0.3731	0.7637	0.5193	0.7031	0.5039	0.2424	0.1159	0.1842	0.1011
	BERT4Rec	0.6977	0.5494	0.6496	0.5373	0.7750	0.6200	0.7307	0.6088	0.4168	0.2876	0.3570	0.2725
	BERT4Rec+FL	0.7048	0.5531	0.6570	0.5410	0.7802	0.6225	0.7362	0.6114	0.4288	0.2958	0.3631	0.2792
	BERT4Rec+AFL	0.6999	0.5498	0.6502	0.5373	0.7773	0.6211	0.7318	0.6096	0.4168	0.2866	0.3543	0.2709
	BERT4Rec+LG	0.7011	0.5455	0.6497	0.5325	0.7788	0.6155	0.7298	0.6031	0.4151	0.2877	0.3558	0.2728
	TailNet	0.6941	0.5342	0.6477	0.5224	0.7818	0.6141	0.7405	0.6037	0.3789	0.2485	0.3198	0.2335
	CITIES	0.7139	0.5649	0.6693	0.5536	0.7880	0.6347	0.7458	0.6239	0.4427	0.3079	0.3856	0.2935
	MASR	**0.7491**	**0.5911**	**0.7007**	**0.5788**	**0.8215**	**0.6617**	**0.7793**	**0.6510**	**0.4847**	**0.3296**	**0.4111**	**0.3110**
Yoochoose	DMAN	0.7940	0.5496	0.7185	0.5304	0.8641	0.6143	0.7906	0.5956	0.5612	0.3349	0.4784	0.3139
	BERT4Rec	0.8009	0.5876	0.7378	0.5716	0.8804	0.6504	0.8162	0.6341	0.5288	0.3728	0.4693	0.3577
	BERT4Rec+FL	0.8008	0.5889	0.7372	0.5728	0.8800	0.6528	0.8163	0.6366	0.5306	0.3697	0.4668	0.3536
	BERT4Rec+AFL	0.8005	0.5873	0.7369	0.5711	0.8814	0.6518	0.8178	0.6357	0.5232	0.3662	0.4609	0.3505
	BERT4Rec+LG	0.8008	0.5873	0.7373	0.5712	0.8814	0.6516	0.8176	0.6353	0.5247	0.3667	0.4622	0.3508
	TailNet	0.7993	0.5914	0.7373	0.5756	0.8734	0.6494	0.8100	0.6333	0.5484	0.3910	0.4893	0.3760
	CITIES	0.8012	0.5880	0.7383	0.5720	0.8803	0.6505	0.8162	0.6342	0.5308	0.3743	0.4713	0.3593
	MASR	**0.8106**	**0.6098**	**0.7497**	**0.5943**	**0.8818**	**0.6690**	**0.8188**	**0.6530**	**0.5677**	**0.4069**	**0.5122**	**0.3929**

the remaining items in the sequence are used for model training. Table 1 summarizes the statistics of these experimental datasets.

4.2 Implementation Details

In this work, we use PyTorch [43] to implement all evaluated methods. The learning rate is chosen from $\{0.007, 0.002, 0.0007, 0.0002\}$ based on the performance on the validation set. The batch size is 128. The maximum item sequence length is 50. The dimension of the hidden state d is 256. Moreover, the number of transformer layers is 2. All the other hyper-parameters are set following prior works [24, 32, 40, 46, 50, 51] and tuned based on the performance on validation data. For our model, we choose the size $|X|$ of vanilla memory bank used in centroid-wise memory bank from $\{10000, 20000, 30000, 40000, 50000\}$. For the cluster-wise memory bank, the size of vanilla memory bank N for each item is selected from $\{3, 5, 7, 9, 11\}$. In the retriever module, K is chosen from $\{2, 4, 6, 8, 10\}$. In the $\mathcal{L}_{\text{RanMCL}}$, the number of negative sample is 100 and the margin λ is 0.5. Besides, inspired by slow drift phenomena [57], we do not utilize the memory bank before 25 epochs.

4.3 Baseline Methods

We compare MASR with existing sequential recommendation methods and various baseline methods dealing with the long-tail problem, including the re-weighting methods and multi-stage training

Table 3: Recommendation performance achieved by MASR with different combinations of copy mechanisms.

Dataset	Method	All Items				Head Items				Tail Items			
		HR@20	N@20	HR@10	N@10	HR@20	N@20	HR@10	N@10	HR@20	N@20	HR@10	N@10
ML-1M	BERT4Rec+FL	0.4549	0.2164	0.3415	0.1877	0.5048	0.2419	0.3823	0.2107	0.2861	0.1314	0.1996	0.1096
	MASR$_{ff}$	**0.4828**	**0.2381**	**0.3802**	**0.2121**	**0.5365**	**0.2676**	**0.4215**	**0.2385**	**0.3056**	**0.1422**	**0.2331**	**0.1239**
	MASR$_{lf}$	0.4780	0.2322	0.3641	0.2035	0.5329	0.2627	0.4067	0.2310	0.2952	0.1292	0.2133	0.1085
	MASR$_{fl}$	0.4738	0.2294	0.3668	0.2023	0.5244	0.2579	0.4127	0.2297	0.2982	0.1333	0.2123	0.1114
	MASR$_{ll}$	0.4689	0.2257	0.3582	0.1978	0.5247	0.2541	0.4021	0.2231	0.2822	0.1274	0.2042	0.1078
Musical	BERT4Rec+FL	0.2426	0.1184	0.1717	0.1005	0.3166	0.1555	0.2260	0.1327	0.0182	0.0059	0.0096	0.0038
	MASR$_{ff}$	0.2477	0.1216	0.1786	0.1042	0.3172	0.1580	0.2304	0.1362	**0.0271**	0.0089	**0.0139**	0.0055
	MASR$_{lf}$	0.2377	0.1167	0.1674	0.0990	0.3110	0.1529	0.2190	0.1296	0.0219	0.0075	0.0113	0.0049
	MASR$_{fl}$	**0.2518**	**0.1234**	**0.1794**	**0.1052**	**0.3241**	**0.1594**	**0.2323**	**0.1363**	0.0259	**0.0092**	0.0120	**0.0057**
	MASR$_{ll}$	0.2431	0.1168	0.1722	0.0990	0.3174	0.1537	0.2281	0.1313	0.0191	0.0067	0.0085	0.0040
Video	BERT4Rec+FL	0.3626	0.1830	0.2740	0.1607	0.4594	0.2337	0.3511	0.2065	0.0834	0.0361	0.0531	0.0284
	MASR$_{ff}$	0.4047	**0.2025**	**0.3055**	**0.1775**	0.5118	**0.2580**	**0.3899**	**0.2272**	0.0959	0.0411	0.0578	0.0316
	MASR$_{lf}$	0.3940	0.1978	0.2993	0.1739	0.4957	0.2507	0.3830	0.2223	0.0973	0.0388	0.0560	0.0284
	MASR$_{fl}$	0.4060	0.2010	0.3022	0.1748	0.5090	0.2548	0.3841	0.2233	**0.1076**	**0.0424**	**0.0654**	**0.0319**
	MASR$_{ll}$	**0.4067**	0.2005	0.3030	0.1744	**0.5129**	0.2554	0.3871	0.2237	0.0982	0.0382	0.0542	0.0272
Diginetica	BERT4Rec+FL	0.7048	0.5531	0.6570	0.5410	0.7802	0.6225	0.7362	0.6114	0.4288	0.2958	0.3631	0.2792
	MASR$_{ff}$	0.7445	0.5859	0.6955	0.5734	0.8175	0.6581	0.7755	0.6474	0.4778	0.3204	0.4030	0.3013
	MASR$_{lf}$	0.7406	0.5836	0.6918	0.5713	0.8154	0.6557	0.7701	0.6443	0.4638	0.3134	0.3946	0.2960
	MASR$_{fl}$	**0.7491**	**0.5911**	**0.7007**	**0.5788**	**0.8215**	**0.6617**	**0.7793**	**0.6510**	**0.4847**	**0.3296**	**0.4111**	**0.3110**
	MASR$_{ll}$	0.7402	0.5811	0.6929	0.5691	0.8125	0.6514	0.7701	0.6406	0.4711	0.3209	0.4056	0.3043
Yoochoose	BERT4Rec+FL	0.8008	0.5889	0.7372	0.5728	0.8800	0.6528	0.8163	0.6366	0.5306	0.3697	0.4668	0.3536
	MASR$_{ff}$	**0.8106**	**0.6100**	0.7495	**0.5944**	0.8817	0.6689	0.8179	0.6527	**0.5679**	**0.4081**	**0.5128**	**0.3941**
	MASR$_{fl}$	**0.8106**	0.6098	**0.7497**	0.5943	**0.8818**	**0.6690**	**0.8188**	**0.6530**	0.5677	0.4069	0.5122	0.3929
	MASR$_{lf}$	0.8073	0.6049	0.7451	0.5891	0.8792	0.6647	0.8148	0.6484	0.5598	0.4001	0.5043	0.3860
	MASR$_{ll}$	0.8051	0.6015	0.7418	0.5854	0.8773	0.6605	0.8115	0.6438	0.5563	0.3985	0.5000	0.3843

solutions. For a fair comparison, BERT4Rec is used for both MASR and other baselines, except for DMAN and TailNet. We list all the baseline models here.

- **Sequential Recommendation Models:** 1) BERT4Rec [50], which employs the transformer layer to model users' historical behaviors; 2) DMAN [51], which uses a dynamic memory-based attention model to capture users' long-term behaviors.
- **Re-weighting Solutions:** 1) Focal Loss (FL) [32], which mitigates the long-tail problem by dynamically scaling loss. The scaling factor focuses on hard examples during training and decreases the weight of easy examples; 2) Anti-Focal Loss (AFL) [46], which incorporates the inductive biases of the beam search to solve the long-tail problem in the text generation task; 3) Logit Adjustment (LG) [40], which modifies the softmax cross-entropy training by unifying several existing methods. 4) TailNet [37], which designs a preference mechanism to softly adjust the recommendation scores.
- **Multi-stage Training Solution:** 1) CITIES [24], which uses an embedding-inference function to replace the tail item embedding with the multiple contextual head item embeddings.

4.4 Evaluation Metrics

We adopt the Hit Ratio@H and NDCG@H (respectively denoted by HR@H and N@H) as the evaluation metrics, which are widely used in the recommendation tasks. In this work, we select the H from $\{10, 20\}$. Following [24, 50], the test process is accelerated by pairing each ground-truth item with Q randomly-chosen negative items that the user has not interacted with. In [24, 50], Q is set to 100. To make the experimental results more confident, we set $Q = 1,000$. We use Pareto Principle[2] as the criteria to split the head and tail items, in which 80% of user feedback is for head items and the rest feedback is for tail items. Note that all items are used to train the model. The reported performance for different data splits (*i.e.*, all items, head item, and tail item split) is obtained on the corresponding splits of the test dataset.

Table 4: Performance of MASR using different combinations of memory banks on Musical dataset. Head means head channel, and tail means tail channel. The Centroid and Cluster refer to centroid-wise and cluster-wise memory banks.

Head	Tail	All items		Head items		Tail items	
		N@20	N@10	N@20	N@10	N@20	N@10
Centroid	Centroid	0.1133	0.0951	0.1474	0.1245	0.0079	0.0050
Centroid	Cluster	**0.1234**	**0.1052**	**0.1594**	0.1363	**0.0092**	**0.0057**
Cluster	Centroid	0.1135	0.0973	0.1482	0.1279	0.0079	0.0044
Cluster	Cluster	0.1217	0.1048	0.1584	**0.1374**	0.0087	0.0049

4.5 Performance Comparison

Table 2 summarizes the recommendation performance of all items, head items, and tail items on five datasets. The proposed MASR model consistently leads to significant gains on all three splits (*i.e.*, all items, head items, and tail items). Compared with the sequential recommendation methods (*i.e.*, BERT4Rec and DMAN), MASR achieves better performance on all three splits on five datasets. Note

Table 5: Ablation study of MASR on the Musical dataset.

	All items		Head items		Tail items	
	N@20	N@10	N@20	N@10	N@20	N@10
MASR	**0.1234**	**0.1052**	**0.1594**	**0.1363**	**0.0092**	**0.0057**
w/o tail channel	0.1215	0.1034	0.1580	0.1353	0.0067	0.0041
w/o head channel	0.1219	0.1033	0.1578	0.1346	0.0091	0.0055
w/o$\mathcal{L}_{\text{RanMCL}}$	0.1183	0.1005	0.1539	0.1319	0.0070	0.0043
w/ vanilla memory bank	0.1119	0.0947	0.1441	0.1230	0.0079	0.0045

Table 6: Performance for different size $|X|$ of vanilla memory bank in head channel on Musical dataset.

	Metric	10000	20000	30000	40000	50000
All Items	N@20	**0.1234**	0.1227	0.1221	0.1229	0.1096
	N@10	**0.1052**	0.1050	0.1049	0.1047	0.0926
Head Items	N@20	**0.1594**	0.1591	0.1586	0.1588	0.1434
	N@10	0.1363	0.1366	**0.1370**	0.1357	0.1216
Tail Items	N@20	0.0092	0.0081	**0.0094**	0.0090	0.0083
	N@10	**0.0057**	0.0052	0.0055	**0.0057**	0.0054

that DMAN also utilizes the external memory bank but MASR outperforms it. This indicates that the proposed cluster-wise memory bank and retriever-copy network can benefit long-tail distribution data. Moreover, MASR outperforms the re-weighting methods (*i.e.*, FL, AFL, LG and TailNet), on all three splits. Among these methods, AFL is worse than FL and LG on the tail items splits but achieves comparable performance with FL. In addition, compared with the multi-stage solution (*i.e.*, CITIES), MASR achieves better performance, which may indicate the benefits of one-stage end-to-end training. Surprisingly, CITIES does not perform well on the Musical and Video dataset. One possible reason is that the training data size of Musical and Video is far less than the other three datasets, which is not enough for CITIES to learn a strong representation for each tail item. Our method utilizes a cluster-wise memory bank to mitigate this issue and outperforms the previous solution.

4.6 Analysis of Copy Mechanisms

We investigate the performance of different combinations of copy mechanisms. Specifically, four variants of MASR(*i.e.*, MASR$_{ff}$, MASR$_{fl}$, MASR$_{lf}$, MASR$_{ll}$) are compared. For each variant, the first subscript represents the type of copy mechanism in the head channel and the second subscript represent that in the tail channel. l refers to the label-based copy mechanism while f is the feature-based copy mechanism. Table 3 summarizes the performance of MASR with different combinations of copy mechanisms. Compared to the state-of-the-art baseline model, MASR achieves the best results. Even the worst-performing variant MASR$_{ll}$ can outperform the baseline in four datasets. For the other three variants, the performance of MASR$_{ff}$ and MASR$_{fl}$ are better than MASR$_{lf}$ on three splits. On the other hand, MASR$_{fl}$ achieves comparable performance with that of MASR$_{ff}$. One likely reason is that the model can retrieve the correct ground-truth item with a high probability by comparing the feature x to the historical features in the memory bank. In this case, the ground-truth item label is useful to improve the recommendation accuracy of tail items.

Table 7: Average training time per epoch with respect to different settings of $|X|$ and N.

$\|X\|$	10000	20000	30000	40000	50000
Time(s)	29.6039	29.8577	30.1803	30.0401	29.8099
N	3	5	7	9	11
Time(s)	27.3025	29.4856	32.4622	34.3645	36.9988

4.7 Analysis of Memory Banks

Table 4 shows the performance of MASR, with respect to different combinations of memory banks. As shown in Table 4, when the tail channel utilizes the cluster-wise memory bank and the head channel uses the centroid-wise memory bank, the overall performance is better than other variants. This aligns with our previous assumptions: more samples need to be retrieved from the memory bank for a better representation of tail items.

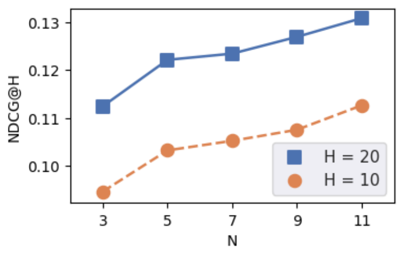
4.8 Ablation Study

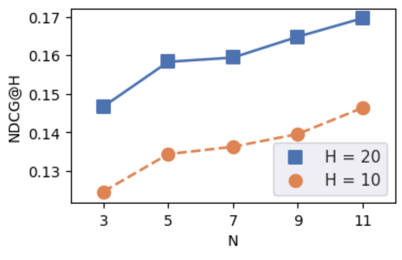
We perform ablation studies to analyze different components of our model. The results are shown in Table 5. We can see that the full model outperforms all variants on three splits, which indicates that all main components contribute to the performance improvement. The further analysis of each component is as follows,

- **w/o Tail channel**: In this variant, we only employ the head channel to improve recommendation performance for head items. The performance of the model without the tail channel is poor on tail items, which indicates that retrieving information from the memory bank M_t and copying relevant patterns benefits the tail item recommendation performance.
- **w/o Head channel**: In this variant, the head channel is removed and the tail channel remains. In this case, the model only retrieves similar historical features or their labels from memory bank M_t. Without the head channel, the head items have more significant performance degradation than tail items. It shows that the head channel brings improvements for head items.
- **w/o $\mathcal{L}_{\text{RanMCL}}$**: In this variant, the randomly sampled memory contrastive loss $\mathcal{L}_{\text{RanMCL}}$ is removed and we only utilize the cross-entropy loss function $\mathcal{L}_{\text{CE}}$ to train the model. It is clear to see that MASR w/o $\mathcal{L}_{\text{RanMCL}}$ performs poorly. It indicates that the $\mathcal{L}_{\text{RanMCL}}$ can effectively enhance the item representation and improve the accuracy of retrieval to improve the recommendation performance.
- **w/ vanilla memory bank**: In this variant, the cluster-wise memory bank in the tail channel is replaced with the vanilla memory bank. Besides, the $\mathcal{L}_{\text{RanMCL}}$ is also removed because it is not designed for vanilla memory banks. We can observe that the model performs poorly. One possible reason is that the vanilla memory bank is hard to cover all unique tail items due to the huge amount of it. The proposed cluster-wise memory bank can mitigate this problem well.

4.9 Parameter Sensitivity Study

In this section, we study the influence of the parameters from three aspects: memory bank in head and tail channels, and retriever. For the memory bank part, we study how many recent historical


(a) All Items


(b) Head Items

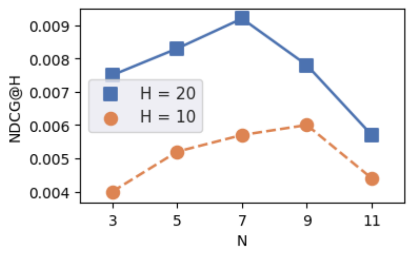
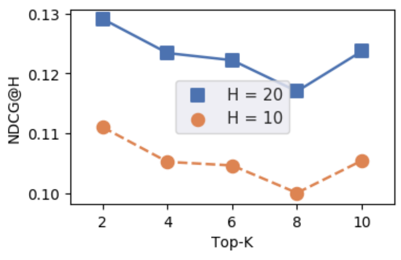
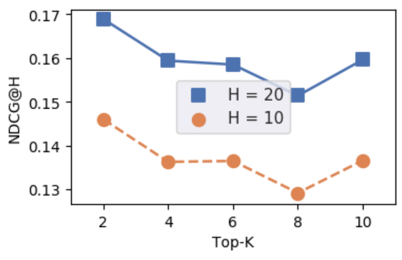

(c) Tail Items

Figure 2: Recommendation performance with respect to different size N in tail channel memory bank on Musical dataset.


(a) All Items


(b) Head Items

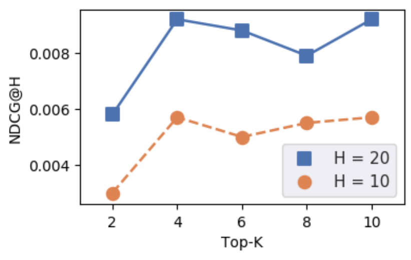

(c) Tail Items

Figure 3: Recommendation performance for different number top-K of retrieved historical samples on Musical dataset.

samples should be memorized by changing the memory size. For the retriever module, we analyze the performance when changing the number of retrieved historical features or labels K.

- **Impacts of the vanilla memory size $|X|$ in centroid-wise memory bank**: Table 6 shows the performance change on all three splits according to vanilla memory size $|X|$. We observe that the performance is generally decreasing on all items and head items when the memory size increases. One possible reason is that more noise data are included with the memory size increasing, which leads to performance degradation. On tail item split, the performance changes little when $|X|$ changes from 10,000 to 40,000, suggesting that the head memory has little influence on the tail item representation learning. In the experiments, we empirically set the memory size of head items $|X|$ to 10,000 due to its best overall performance. Besides, we also analyzed the effect of changing the value of $|X|$ on the training time. As shown in Table 7, with the change of $|X|$, the running time is almost unchanged. The possible reason is the GPU parallel computing.
- **Impacts of the vanilla memory size N in cluster-wise memory bank**: Figure 2 demonstrates the performance of models with various capacity N. Among the various range of N, although the $N = 11$ achieves the best performance on all items and head items, it almost performs poorest on the tail items. This indicates that there is a trade-off between the performance on head items and that on tail items. When $N = 7$, the performance for tail items outperforms all other N values. In this work, we set the capacity $N = 7$ due to the best performance on tail items. Moreover, we also analyzed the influence of changing the value of N on the training time. As shown in Table 7, with N increasing, the increase in training time is reasonable.
- **Impacts of the number of retrieved historical features/labels**: As shown in Fig. 3, the performance on all item split and head item split are decreasing as K increases. When K is selected from $\{4, 6, 10\}$, the model achieves comparable performance for tail items. In the experiments, we empirically set K to 4.

5 CONCLUSION

In this paper, we propose a novel sequential recommendation model, namely MASR (*i.e.*, Memory Bank Augmented Long-tail Sequential Recommendation), to improve sequential recommendation performance on tail items. Specifically, we propose a centroid-wise memory bank and a cluster-wise memory bank to store historical data samples. Moreover, the retriever-copy network is designed to search similar sequences and copy relevant information to enhance the recommendation performance for head items and tail items. In addition, we propose randomly sampled memory contrastive loss to enhance the representation quality of the tail items. We validate our method on five real-world datasets and the experimental results demonstrate that MASR outperforms state-of-the-art baseline methods.

ACKNOWLEDGMENTS

This research is supported, in part, by the National Research Foundation (NRF), Singapore under its AI Singapore Programme (AISG Award No: AISG-GC-2019-003), and by the Development of Cryptographic Library and Support Systems (LP180101062), Australian Research Council. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not reflect the views of National Research Foundation, Singapore and Australian Research Council.

REFERENCES

[1] Linas Baltrunas, Bernd Ludwig, and Francesco Ricci. 2011. Matrix factorization techniques for context aware recommendation. In *Proceedings of the fifth ACM conference on Recommender systems*. 301–304.

[2] George EP Box and R Daniel Meyer. 1986. An analysis for unreplicated fractional factorials. *Technometrics* 28, 1 (1986), 11–18.

[3] Mateusz Buda, Atsuto Maki, and Maciej A Mazurowski. 2018. A systematic study of the class imbalance problem in convolutional neural networks. *Neural Networks* 106 (2018), 249–259.

[4] Jonathon Byrd and Zachary Lipton. 2019. What is the effect of importance weighting in deep learning?. In *International Conference on Machine Learning*. PMLR, 872–881.

[5] Yinjiang Cai, Zeyu Cui, Shu Wu, Zhen Lei, and Xibo Ma. 2021. Represent Items by Items: An Enhanced Representation of the Target Item for Recommendation. *arXiv preprint arXiv:2104.12483* (2021).

[6] Kaidi Cao, Colin Wei, Adrien Gaidon, Nikos Arechiga, and Tengyu Ma. 2019. Learning imbalanced datasets with label-distribution-aware margin loss. *Advances in neural information processing systems* 32 (2019).

[7] Jianxin Chang, Chen Gao, Yu Zheng, Yiqun Hui, Yanan Niu, Yang Song, Depeng Jin, and Yong Li. 2021. Sequential recommendation with graph neural networks. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 378–387.

[8] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. In *EMNLP*.

[9] Hsin-Ping Chou, Shih-Chieh Chang, Jia-Yu Pan, Wei Wei, and Da-Cheng Juan. 2020. Remix: Rebalanced mixup. In *European Conference on Computer Vision*. Springer, 95–110.

[10] Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NIPS 2014 Workshop on Deep Learning, December 2014*.

[11] Felipe Costa, Sixun Ouyang, Peter Dolog, and Aonghus Lawlor. 2018. Automatic generation of natural language explanations. In *Proceedings of the 23rd international conference on intelligent user interfaces companion*. 1–2.

[12] Yin Cui, Menglin Jia, Tsung-Yi Lin, Yang Song, and Serge Belongie. 2019. Class-balanced loss based on effective number of samples. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 9268–9277.

[13] Travis Ebesu, Bin Shen, and Yi Fang. 2018. Collaborative memory network for recommendation systems. In *The 41st international ACM SIGIR conference on research & development in information retrieval*. 515–524.

[14] Chengjian Feng, Yujie Zhong, and Weilin Huang. 2021. Exploring classification equilibrium in long-tailed object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 3417–3426.

[15] Fuli Feng, Xiangnan He, Xiang Wang, Cheng Luo, Yiqun Liu, and Tat-Seng Chua. 2019. Temporal relational ranking for stock prediction. *ACM Transactions on Information Systems (TOIS)* 37, 2 (2019), 1–30.

[16] Lei Guo, Li Tang, Tong Chen, Lei Zhu, Quoc Viet Hung Nguyen, and Hongzhi Yin. 2021. DA-GCN: A domain-aware attentive graph convolution network for shared-account cross-domain sequential recommendation. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI*. 2483–2489.

[17] Haibo He and Edwardo A Garcia. 2009. Learning from imbalanced data. *IEEE Transactions on knowledge and data engineering* 21, 9 (2009), 1263–1284.

[18] Ruining He and Julian McAuley. 2016. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *Proceedings of the 25th international conference on World wide web*. 507–517.

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

[20] 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.

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

[22] Jiayuan Huang, Arthur Gretton, Karsten Borgwardt, Bernhard Schölkopf, and Alex Smola. 2006. Correcting sample selection bias by unlabeled data. *Advances in neural information processing systems* 19 (2006), 601–608.

[23] Muhammad Abdullah Jamal, Matthew Brown, Ming-Hsuan Yang, Liqiang Wang, and Boqing Gong. 2020. Rethinking class-balanced methods for long-tailed visual recognition from a domain adaptation perspective. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 7610–7619.

[24] Seongwon Jang, Hoyeop Lee, Hyunsouk Cho, and Sehee Chung. 2020. CITIES: Contextual Inference of Tail-item Embeddings for Sequential Recommendation. In *2020 IEEE International Conference on Data Mining (ICDM)*. IEEE, 202–211.

[25] Nathalie Japkowicz and Shaju Stephen. 2002. The class imbalance problem: A systematic study. *Intelligent data analysis* 6, 5 (2002), 429–449.

[26] Xunqiang Jiang, Binbin Hu, Yuan Fang, and Chuan Shi. 2020. Multiplex memory network for collaborative filtering. In *Proceedings of the 2020 SIAM International Conference on Data Mining*. SIAM, 91–99.

[27] Wang-Cheng Kang and Julian McAuley. 2018. Self-attentive sequential recommendation. In *2018 IEEE International Conference on Data Mining (ICDM)*. IEEE, 197–206.

[28] Yejin Kim, Kwangseob Kim, Chanyoung Park, and Hwanjo Yu. 2019. Sequential and Diverse Recommendation with Long Tail.. In *IJCAI*, Vol. 19. 2740–2746.

[29] Adit Krishnan, Ashish Sharma, Aravind Sankar, and Hari Sundaram. 2018. An adversarial approach to improve long-tail performance in neural collaborative filtering. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. 1491–1494.

[30] Chenyi Lei, Yong Liu, Lingzi Zhang, Guoxin Wang, Haihong Tang, Houqiang Li, and Chunyan Miao. 2021. Semi: a sequential multi-modal information transfer network for e-commerce micro-video recommendations. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 3161–3171.

[31] Yile Liang and Tieyun Qian. 2021. Recommending accurate and diverse items using bilateral branch network. *arXiv preprint arXiv:2101.00781* (2021).

[32] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. 2017. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*. 2980–2988.

[33] Zachary C Lipton, John Berkowitz, and Charles Elkan. 2015. A critical review of recurrent neural networks for sequence learning. *arXiv preprint arXiv:1506.00019* (2015).

[34] Chang Liu, Han Yu, Boyang Li, Zhiqi Shen, Zhanning Gao, Peiran Ren, Xuansong Xie, Lizhen Cui, and Chunyan Miao. 2021. Noise-resistant deep metric learning with ranking-based instance selection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 6811–6820.

[35] Jialun Liu, Wenhui Li, and Yifan Sun. 2022. Memory-based jitter: Improving visual recognition on long-tailed data with diversity in memory. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 36. 1720–1728.

[36] 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*. 1831–1839.

[37] Siyi Liu and Yujia Zheng. 2020. Long-tail session-based recommendation. In *Fourteenth ACM conference on recommender systems*. 509–514.

[38] Chen Ma, Peng Kang, and Xue Liu. 2019. Hierarchical gating networks for sequential recommendation. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. 825–833.

[39] Yuxian Meng, Shi Zong, Xiaoya Li, Xiaofei Sun, Tianwei Zhang, Fei Wu, and Jiwei Li. 2022. GNN-LM: Language Modeling Based on Global Contexts via GNN. In *ICLR 2022 Workshop on Deep Learning on Graphs for Natural Language Processing*.

[40] Aditya Krishna Menon, Sadeep Jayasumana, Ankit Singh Rawat, Himanshu Jain, Andreas Veit, and Sanjiv Kumar. 2020. Long-tail learning via logit adjustment. In *International Conference on Learning Representations*.

[41] Wanli Ouyang, Xiaogang Wang, Cong Zhang, and Xiaokang Yang. 2016. Factors in finetuning deep model for object detection with long-tail distribution. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 864–873.

[42] Yoon-Joo Park and Alexander Tuzhilin. 2008. The long tail of recommender systems and how to leverage it. In *Proceedings of the 2008 ACM conference on Recommender systems*. 11–18.

[43] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*. 8024–8035.

[44] Ruihong Qiu, Zi Huang, and Hongzhi Yin. 2021. Memory Augmented Multi-Instance Contrastive Predictive Coding for Sequential Recommendation. In *2021 IEEE International Conference on Data Mining (ICDM)*. IEEE, 519–528.

[45] Xuan Rao, Lisi Chen, Yong Liu, Shuo Shang, Bin Yao, and Peng Han. 2022. Graph-Flashback Network for Next Location Recommendation. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 1463–1471.

[46] Vikas Raunak, Siddharth Dalmia, Vivek Gupta, and Florian Metze. 2020. On Long-Tailed Phenomena in Neural Machine Translation. In *Findings of the Association for Computational Linguistics: EMNLP 2020*. 3088–3095.

[47] 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.

[48] Aravind Sankar, Junting Wang, Adit Krishnan, and Hari Sundaram. 2021. Protocf: Prototypical collaborative filtering for few-shot recommendation. In *Fifteenth ACM Conference on Recommender Systems*. 166–175.

[49] Li Shen, Zhouchen Lin, and Qingming Huang. 2016. Relay backpropagation for effective learning of deep convolutional neural networks. In *European conference on computer vision*. Springer, 467–482.

[50] 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.

[51] Qiaoyu Tan, Jianwei Zhang, Ninghao Liu, Xiao Huang, Hongxia Yang, Jignren Zhou, Xia Hu, et al. 2021. Dynamic memory based attention network for sequential recommendation. *AAAI2021* (2021).

[52] 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.

[53] 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.

[54] Thanh Tran, Xinyue Liu, Kyumin Lee, and Xiangnan Kong. 2019. Signed distance-based deep memory recommender. In *The World Wide Web Conference*. 1841–1852.

[55] 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).

[56] Shanfeng Wang, Maoguo Gong, Haoliang Li, and Junwei Yang. 2016. Multi-objective optimization for long tail recommendation. *Knowledge-Based Systems* 104 (2016), 145–155.

[57] Xun Wang, Haozhi Zhang, Weilin Huang, and Matthew R Scott. 2020. Cross-batch memory for embedding learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 6388–6397.

[58] Yule Wang, Xin Xin, Yue Ding, and Dong Wang. 2021. ICMT: Item Cluster-Wise Multi-Objective Training for Long-Tail Recommendation. *arXiv preprint arXiv:2109.12887* (2021).

[59] Fajie Yuan, Alexandros Karatzoglou, Ioannis Arapakis, Joemon M Jose, and Xiangnan He. 2019. A simple convolutional generative network for next item recommendation. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*. 582–590.

[60] Shengyu Zhang, Ziqi Tan, Zhou Zhao, Jin Yu, Kun Kuang, Tan Jiang, Jingren Zhou, Hongxia Yang, and Fei Wu. 2020. Comprehensive information integration modeling framework for video titling. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2744–2754.

[61] Yin Zhang, Derek Zhiyuan Cheng, Tiansheng Yao, Xinyang Yi, Lichan Hong, and Ed H Chi. 2021. A Model of Two Tales: Dual Transfer Learning Framework for Improved Long-tail Item Recommendation. In *Proceedings of the Web Conference 2021*. 2220–2231.

[62] Yixin Zhang, Yong Liu, Yonghui Xu, Hao Xiong, Chenyi Lei, Wei He, Lizhen Cui, and Chunyan Miao. 2022. Enhancing Sequential Recommendation with Graph Contrastive Learning. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence*. International Joint Conferences on Artificial Intelligence Organization, 2398–2405.

[63] Ziqi Zhang, Zhongang Qi, Chunfeng Yuan, Ying Shan, Bing Li, Ying Deng, and Weiming Hu. 2021. Open-book video captioning with retrieve-copy-generate network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 9837–9846.

[64] Kun Zhou, Hui Yu, Wayne Xin Zhao, and Ji-Rong Wen. 2022. Filter-enhanced MLP is All You Need for Sequential Recommendation. In *Proceedings of the ACM Web Conference 2022*. 2388–2399.

[65] Qingyu Zhou, Nan Yang, Furu Wei, and Ming Zhou. 2018. Sequential copying networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 32.

[66] Tianyu Zhu, Leilei Sun, and Guoqing Chen. 2021. Graph-based Embedding Smoothing for Sequential Recommendation. *IEEE Transactions on Knowledge and Data Engineering* (2021).