Co-occurrence Embedding Enhancement for Long-tail Problem in Multi-Interest Recommendation

Yaokun Liu
Tianjin University
Tianjin, China
yaokunl@tju.edu.cn

Xiaowang Zhang
Tianjin University
Tianjin, China
xiaowangzhang@tju.edu.cn

Minghui Zou
Tianjin University
Tianjin, China
minghuizou@tju.edu.cn

Zhiyong Feng
Tianjin University
Tianjin, China
zyfeng@tju.edu.cn

ABSTRACT

Multi-interest recommendation methods extract multiple interest vectors to represent the user comprehensively. Despite their success in the matching stage, previous works overlook the long-tail problem. This results in the model excelling at suggesting head items, while the performance for tail items, which make up more than 70% of all items, remains suboptimal. Hence, enhancing the tail item recommendation capability holds great potential for improving the performance of the multi-interest model.

Through experimental analysis, we reveal that the insufficient context for embedding learning is the reason behind the underperformance of tail items. Meanwhile, we face two challenges in addressing this issue: the absence of supplementary item features and the need to maintain head item performance. To tackle these challenges, we propose a CoLT module (**Co**-occurrence embedding enhancement for **L**ong-**T**ail problem) that replaces the embedding layer of existing multi-interest frameworks. By linking co-occurring items to establish "assistance relationships", CoLT aggregates information from relevant head items into tail item embeddings and enables joint gradient updates. Experiments on three datasets show our method outperforms SOTA models by 21.86% Recall@50 and improves the Recall@50 of tail items by 14.62% on average.

CCS CONCEPTS

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

KEYWORDS

Recommender Systems, Multi-interest Learning, Long-tail problem

ACM Reference Format:
Yaokun Liu, Xiaowang Zhang, Minghui Zou, and Zhiyong Feng. 2023. Co-occurrence Embedding Enhancement for Long-tail Problem in Multi-Interest Recommendation. In *Seventeenth ACM Conference on Recommender Systems (RecSys '23), September 18–22, 2023, Singapore, Singapore.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3604915.3608835

RecSys '23, September 18–22, 2023, Singapore, Singapore

ACM ISBN 979-8-4007-0241-9/23/09…$15.00
https://doi.org/10.1145/3604915.3608835

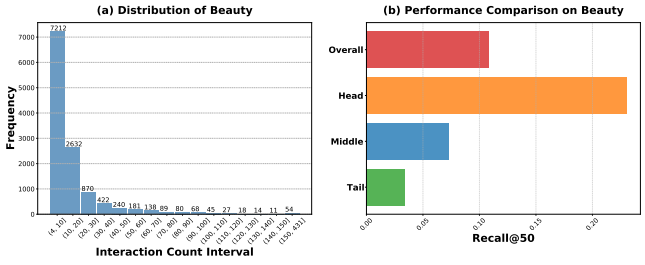

Figure 1: (a): The Beauty dataset exhibits a long-tail distribution in item interactions. (b): The ComiRec-SA model underperforms on tail items in the Beauty dataset.

1 INTRODUCTION

The goal of the matching stage in large-scale recommender systems is to retrieve a subset of items that may interest users from a corpus of billions, while meeting the low computational cost [6, 15, 16, 20]. Previous methods [5, 10, 20, 21] extract a unified interest vector for each user based on behavior sequence, which struggles to represent the multiple interests of the user.In response, multi-interest recommendation methods [2–4, 7, 14, 23] are proposed in recent years, employing multiple vectors to represent user interests and exhibit remarkable performance.

However, prior works overlook the long-tail problem which arises from the vast item set and highly skewed sparse user-item interactions (Figure 1) [1, 18, 22]. We find that the strength of these models lies in its ability to accurately recommend head items, while the performance on tail items is inferior. Despite infrequently interacted with, tail items account for over 70% of the total number of items, which substantially decreases the model's overall performance, as shown in Figure 1.

To investigate why multi-interest models fail on recommending tail items, we conduct a motivating experiment in Section 2.3. The results reveal the main reason: due to the highly skewed user-item interaction in a long-tail distribution, the embedding learning context of tail items is insufficient, causing model overfitting. However, solving the above problems is not trivial. We face the following

challenges: (1) How to enhance the embedding representation without introducing additional item features? (2) How to improve the performance of tail items without compromising the performance of head items?

Notably, we observe that the long-tail problem has two sides: insufficient training for tail items but sufficient training for head items. Therefore, we can establish an "assistance relationship" between relevant head and tail items to tackle the challenges above. That is, aggregate the information of co-occurring head items into tail items to enhance the embedding, and update the embedding of tail items along with the head items during gradient descent to enrich the context while maintaining the head item's performance.

For this purpose, we propose a module called **Co**-occurrence embedding enhancement for **L**ong-**T**ail problem (**CoLT**), to replace the existing embedding layer in multi-interest recommendation framework. Before training begins, we create an adjacency matrix containing the co-occurrence relationship between items in all user sequences. The value of each element in the matrix is related to the popularity of the corresponding adjacent items. Then, we weighted sum the embeddings of co-occurring head and tail items by multiplying the embedding matrix with the normalized adjacency matrix, which enhances the item embedding representation, particularly for tail items.

The main contributions can be summarized as follows:

- We recognize the long-tail problem in multi-interest recommendation and identify the reason contributing to the inferior performance of tail items through experiments and theoretical analysis.
- We propose the CoLT module, which enhances embeddings and enriches training contexts of tail items by associating with the co-occurring head items.
- Our method significantly improves the recommendation performance of various multi-interest models on tail items, outperforming state-of-the-art methods in extensive experiments on three datasets.

2 PRELIMINARIES

2.1 Problem Formulation

Let $\mathcal{U}$ be the set of users and $\mathcal{I}$ be the corpus of items. Each user $u \in \mathcal{U}$ has a historical interaction sequence $S^u = \{i_1, i_2, \ldots, i_L\}$ ordered chronologically, where L denotes the maximum length of the user behavior sequence and i_l denotes the l-th item the user interacted with. The goal of multi-interest recommendation is to predict a subset of item that the user is likely to interact with.

2.2 Multi-Interest Framework

In this section, we provide a summary of the multi-interest framework, shown in Figure 2.

2.2.1 Embedding Layer. The embedding matrix $\mathbf{E} \in \mathbb{R}^{|\mathcal{I}| \times d}$ represents all items densely, where the dimension of the embedding vectors is d. Then the user sequence S^u are mapped into a latent embedding space by performing the embedding lookup operation:

$$\mathbf{H} = \text{Embedding}(S^u; \mathbf{E}) = [\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_L] \in \mathbb{R}^{L \times d} \tag{1}$$

where $\mathbf{e}_n \in \mathbb{R}^d$ is the embedding of the n-th item.

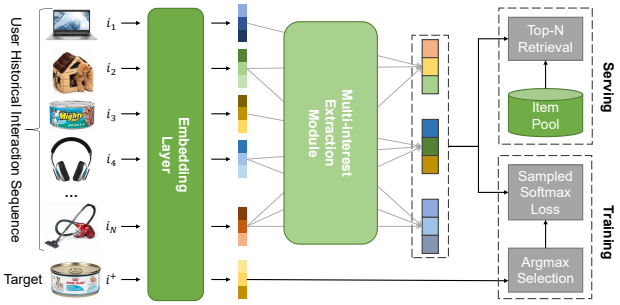

Figure 2: Framework of Multi-Interest Recommendation.

2.2.2 Multi-interest Extraction Module. The main objectives of multi-interest extraction module can be formulated as calculating the interest weight matrix $\mathbf{W}$ and subsequently compute the multi-interest matrix $\mathbf{V}_u$ of the user u:

$$\mathbf{W} = \mathcal{G}(\mathbf{H}) \in \mathbb{R}^{K \times L} \tag{2}$$

$$\mathbf{V}_u = \mathbf{W}\mathbf{H} \in \mathbb{R}^{K \times d} \tag{3}$$

where $\mathcal{G}$ is the function to obtain the interest weight matrix, and K is the number of interest vectors.

2.2.3 Training and Serving. In the training stage, we first use the argmax operation to select the interest vector most relevant to the target item i^+, and the relevance is calculated by the dot product:

$$\mathbf{v}_u = \mathbf{V}_u\,[:, \text{argmax}(\mathbf{V}_u \mathbf{e}_{i^+})] \in \mathbb{R}^d \tag{4}$$

where $\mathbf{e}_{i^+}$ is embedding of the target item i^+. Our goal is to maximize $P(i^+ \mid u)$, which indicates the likelihood of user u with interest $\mathbf{v}_u$ to interact with the target item i^+. To mitigate the computational complexity arising from the large number of items, the sampled softmax technique [5, 11] is employed to compute $P(i^+ \mid u)$ and minimize the objective function below to optimize model parameters:

$$\begin{aligned} \mathcal{L}(\theta) &= \sum_{u \in \mathcal{U}} -\log P(i^+ \mid u) \\ &= \sum_{u \in \mathcal{U}} -\log \frac{\exp(\mathbf{v}_u^\top \mathbf{e}_{i^+})}{\exp(\mathbf{v}_u^\top \mathbf{e}_{i^+}) + \sum_{i^- \in \mathcal{N}} \exp(\mathbf{v}_u^\top \mathbf{e}_{i^-})} \end{aligned} \tag{5}$$

where $\mathcal{N}$ is the set of randomly sampled negative samples shared per batch.

In the online serving stage, we first use Faiss [12] to retrieve the top-N items from the item pool for each interest of the user. Then, calculate the matching degree $f(u, i)$ between the candidate item i and the user u:

$$f(u, i) = \max_{1 \le k \le K} (\mathbf{e}_i^\top \mathbf{v}_u^{(k)}) \tag{6}$$

where $\mathbf{v}_u^{(k)}$ is the k-th interest of user u. Based on the calculated matching score, we select the top-N items with the highest scores from $K \times N$ candidate items as the final recommendation results.

2.3 Investigation of Long-tail Problem

To identify the reasons for the failure of prior multi-interest models on tail items, we conduct the motivation experiment and result analysis.

Table 1: Dataset Statistics.

Datasets	#Users	#Items	#Interactions	Sparsity	#Head (prop.)	#Mid (prop.)	#Tail (prop.)
Beauty	22,363	12,101	198,502	99.927%	750 (6.20%)	2,700 (22.31%)	8,651 (71.49%)
RetailRocket	60,112	34,185	693,382	99.966%	2,236 (6.54%)	7,374 (21.57%)	24,575 (71.89%)
Books	603,668	367,982	8,898,041	99.996%	12,173 (3.31%)	62,498 (16.98%)	293,311 (79.71%)

2.3.1 Over-fitting. We first investigate the performance of head and tail items on the training and validation sets using a representative multi-interest model (ComiRec [2]).

Notably, to avoid bias in subsequent experiments, we divided the dataset into head, middle, and tail items, with each part accounting for one-third of the total interactions. The dataset partition details are presented in Table 1.

As depicted in Figure 3, while the recall rate of tail items is as high as 84.70% in the training set, even surpassing the head items, it sharply drops to 3.68% in the validation set. Results on RetailRocket and Books follow the same trend as Beauty, which are not shown due to space constraints. This implies that the model is overfitting to the tail items in the training set, leading to unqualified embedding representations of tail items.

2.3.2 Insufficient Context. Furthermore, we theoretically analyze when the embeddings of tail items can be updated and why it is overfitted.

Let $\mathcal{B}$ denotes a batch of training samples. In the backpropagation of batch $\mathcal{B}$ we can obtain the following formulas according to equation 5:

$$\mathcal{I}_{\mathcal{B}} = \mathcal{N} \cup \bigcup_{u \in \mathcal{B}} (S^u \cup i^+) \tag{7}$$

$$\nabla_{\mathbf{e}_i} \begin{cases} \neq 0, \text{ if } i \in \mathcal{I}_{\mathcal{B}} \\ = 0, \text{ if } i \in \mathcal{I} - \mathcal{I}_{\mathcal{B}} \end{cases} \tag{8}$$

where $\nabla_{\mathbf{e}_i}$ denotes the gradient of item i, and $\mathcal{I}_{\mathcal{B}}$ is the union of items in the user sequence (including target item) or negative samples set of $\mathcal{B}$.

The above formulas imply that tail items can only receive gradient updates when they appear in the user sequence or become a negative sample. However, since tail items are infrequently interacted by users, limited sequences can provide explicit contexts for embedding updates. Additionally, when tail items appear as negative samples, the context is implicit and ambiguous as they are randomly selected, providing little constraint on embedding updates.

We conclude that the overfitting of tail items is due to the lack of informative contexts for embedding learning.

3 METHOD

Through the analysis in Section 2.3, we confirm the reason for the poor performance on tail items is the lack of context for embedding learning. However, to address this problem, we face two constraints: (1) the absence of supplementary item features and (2) the need to maintain the performance of other items.

In this section, we propose the CoLT module to foster an "assistance relationship" between co-occurring head and tail items, replacing the multi-interest framework embedding layer. CoLT enriches the training context and enhances the embeddings for tail items while satisfying the given constraints.

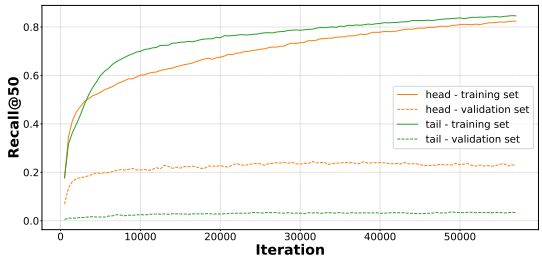

Figure 3: Training and validation curves of ComiRec on Beauty dataset, which are evaluated by training and validation sets per 500 iterations.

3.1 Construction of Item Adjacency Matrix

Before training, we first create an adjacency matrix $\mathbf{A} \in \mathbb{R}^{|\mathcal{I}| \times |\mathcal{I}|}$ representing the co-occurrence relationships among items in all sequences within the training set. $\mathbf{A}$ is initialized as a zero matrix. When the distance between item i and j in a user sequence is less than t (*i.e.*, item i and j co-occur within the step interval t), we consider them correlated and set the i-th row and j-th column element of $\mathbf{A}$ to a correlation weight a_{ij}, whose value is determined by the popularity of item i and j:

$$\mathrm{a_{ij}} = \begin{cases} 3, \text{ if } i \text{ is tail item or } j \text{ is tail item;} \\ 1, \text{ elif } i \text{ is head item and } j \text{ is head item;} \\ 2, \text{ else.} \end{cases} \tag{9}$$

Here, we assign higher values to the elements in the adjacency matrix that related to tail items, to help strengthen the training of tail items in subsequent steps.

3.2 CoLT Module

In training, we replace the embedding layer with the CoLT module to enhance the embedding matrix. Let $\mathbf{e}_i \in \mathbb{R}^d$ denote the embedding of item i obtained by querying the embedding matrix $\mathbf{E}$, and $\mathcal{I}_i$ is the set of items that have co-occurrence relationships with item i. In each batch iteration, we first operate embedding aggregation as follows:

$$\mathbf{e}'_i = \frac{1}{1 + \sum_{j \in \mathcal{I}_i} a_{ij}} \left(\mathbf{e}_i + \sum_{j \in \mathcal{I}_i} a_{ij} \mathbf{e}_j \right) \tag{10}$$

where $\mathbf{e}'_i \in \mathbb{R}^d$ is the updated embedding of item i. We adopt the weighted sum aggregator for embedding enhancement, which can

be seen as a special type of graph convolution [8, 19]. However, it is worth noting that our method does not introduce any parameter and only performs propagation between first-order co-occurring items.

In practical applications, we implement CoLT with the matrix form, which allows us to update the embedding matrix in a rather efficient way:

$$\widehat{\mathbf{A}} = \mathbf{A} + \mathbf{I} \tag{11}$$

$$\mathbf{E}' = \widehat{\mathbf{D}}^{-1}\widehat{\mathbf{A}}\mathbf{E} \tag{12}$$

where $\mathbf{I} \in \mathbb{R}^{|\mathcal{I}|\times|\mathcal{I}|}$ is the identity matrix, and $\widehat{\mathbf{D}}$ is the degree matrix of $\widehat{\mathbf{A}}$, which diagonal entry $\widehat{d}_{ii}$ is obtained by summing the rows of the adjacency matrix $\widehat{\mathbf{A}}$; $\mathbf{E}'$ is the enhanced embedding matrix.

Note that $\widehat{\mathbf{A}}$ and $\widehat{\mathbf{D}}$ only need to be computed once before training, since they can be reused in subsequent computations. $\mathbf{E}'$ is updated per batch iteration.

Afterwards, we embed the sequence S^u with the enhanced embedding matrix $\mathbf{E}'$:

$$\mathbf{H}' = \text{Embedding}\left(S^u; \mathbf{E}'\right) = \left[\mathbf{e}'_1, \mathbf{e}'_2, \ldots, \mathbf{e}'_L\right] \in \mathbb{R}^{L\times d} \tag{13}$$

where $\mathbf{H}'$ is the enhanced sequence embedding representation, which will be fed into the subsequent multi-interest extraction module.

In summary, the factors contributing to the effectiveness of CoLT are: (1) When updating the embeddings of head items, the embeddings of co-occurring tail items are also updated together, as they are aggregated in the embeddings of head items; thus, CoLT enriches the training context. (2) Tail item embeddings also incorporate embeddings of co-occurring head items, which are relevant and more accurate; thus, CoLT enhances the embedding representation.

3.3 Complexity

In this section, we examine the extra space and time complexity incurred by replacing the embedding layer with the CoLT module and draw the following conclusions: during training, the CoLT module increases the time complexity but also speeds up the model's convergence; during serving, the CoLT module does not introduce any additional time complexity.

The sparse adjacency matrix $\widehat{\mathbf{A}}$ is stored in sparse format with about tM non-zero elements, where M is the number of user-item interactions in the dataset and t is the co-occurring step interval. Therefore, the additional space complexity is $O(M)$. For the training phase, the extra training time complexity is $O(Md)$ using sparse matrix multiplication. For the serving phase, since the learned enhanced embedding matrix doesn't require update and can be directly queried, there is no additional time complexity.

In addition, Section 4.4 shows through experiments that, although the CoLT module introduces additional time complexity during training, it is acceptable as models using the CoLT module can converge quicker and achieve better results with fewer iterations, leading to a shorter overall training time.

4 EXPERIMENTS

4.1 Experimental Setting

4.1.1 Datasets. We use three public benchmarks(details in Table 1): **Amazon Beauty**[1] [17], **Amazon Books**[1] [17] and **RetailRocket**[2]. For each dataset, we partition users into training, validation, and test sets with an 8:1:1 ratio and training sample length is truncated to 20. During training, we use the entire user sequence to train. During evaluation, we extract 80% for learning user embeddings and predict the remaining 20% for metrics.

4.1.2 Baselines. We compare our model with seven different baselines. An item popularity-based method: **PopRec**; two traditional sequential recommendation models commonly used in the matching phase: **YouTube DNN** (Y-DNN) [5] and **GRU4Rec** [9]; four state-of-the-art multi-interest recommendation models: **MIND** [14], **ComiRec-SA** (ComiRec) [2], **PIMIRec** [4] and **Re4** [23].

4.1.3 Implementation and Evaluation. We implement the CoLT module on **ComiRec-SA** [2] by default for evaluation and named as **CoLTRec**, except for section 4.3. We use the Adam optimzer [13] for training. For Beauty, RetailRocket and Books datasets, we set the batch size to 256, 256 and 128, and the co-occurring interval to 2, 2 and 5, respectively. The embedding dimension is 64, learning rate is 0.001, and the number of interests is 4. We use a sample size of 10 for the softmax loss function. We evaluate matching quality using Recall (R), NDCG (ND), and Hit Rate (HR). All models are implemented and hyper-parameters are tuned based on the authors' recommendations.

4.2 Overall Performance

Table 2 shows the overall performance comparison of three datasets. We find that: (1) Traditional sequential recommendation models outperform PopRec as they meet personalized recommendation requirements by learning user representations from historical sequences. (2) Multi-interest models outperform traditional sequential recommendation models that rely on a single user representation, indicating the effectiveness of multi-interest frameworks in representing users with diverse interests due to item diversity. (3) Although CoLTRec is based on simple ComiRec, it performs significantly better than state-of-the-art models on three datasets in most cases. This reflects the severity of the long-tail problem in the matching stage, as well as the efficacy of our method in addressing this issue and improving the model's recommendation performance.

4.3 Enhancement Study

In this section, we experimentally answer two questions: (1) Can CoLT improve the performance of different multi-interest models? (2) Does CoLT improve tail performance while compromising head performance? We compare original and CoLT-enhanced versions of SOTA multi-interest models: ComiRec-SA [2], PIMIRec [4] and Re4 [23]. We deliberately choose to experiment on the RetailRocket

[1] https://nijianmo.github.io/amazon
[2] https://www.kaggle.com/datasets/retailrocket/ecommerce-dataset

Table 2: Model comparison. Bold: best performance. Underlined: suboptimal performance.

Datasets	Metric	PopRec	Y-DNN	GRU4Rec	MIND	ComiRec	PIMIRec	Re4	CoLTRec	Improv.
Beauty	R@20	0.0264	0.0647	0.0552	0.0784	0.0594	0.0665	0.0767	**0.1098**	+40.14%
	R@50	0.0457	0.1040	0.0900	0.1203	0.1101	0.1181	0.1213	**0.1797**	+48.13%
	ND@20	0.0170	0.0503	0.0457	0.0599	0.0412	0.0395	0.0526	**0.0756**	+26.36%
	ND@50	0.0243	0.0620	0.0555	0.0752	0.0562	0.0548	0.0653	**0.0934**	+24.14%
	HR@20	0.0460	0.1095	0.0992	0.1287	0.1085	0.1093	0.1261	**0.1837**	+42.71%
	HR@50	0.0823	0.1721	0.1524	0.1958	0.1824	0.1851	0.1891	**0.2745**	+40.18%
Books	R@20	0.0135	0.0483	0.0407	0.4862	0.0532	0.0649	0.0615	**0.0794**	+22.43%
	R@50	0.0236	0.0757	0.0649	0.7638	0.0812	0.1025	0.1008	**0.1280**	+24.93%
	ND@20	0.0123	0.0490	0.0404	0.0446	0.0403	0.0520	0.0570	**0.0620**	+8.81%
	ND@50	0.0164	0.0620	0.0498	0.0511	0.0514	0.0662	0.0715	**0.0787**	+10.09%
	HR@20	0.0299	0.0595	0.0880	0.1062	0.1093	0.1345	0.1315	**0.1588**	+18.07%
	HR@50	0.0509	0.1629	0.1374	0.1615	0.1644	0.2063	0.2064	**0.2427**	+17.58%
Retail Rocket	R@20	0.0110	0.3396	0.3395	0.3921	0.4408	0.4440	**0.5514**	0.4904	-11.07%
	R@50	0.0215	0.3942	0.3993	0.4503	0.4929	0.4962	**0.6100**	0.5643	-7.48%
	ND@20	0.0108	0.3487	0.3746	0.4227	0.4714	0.4796	0.4621	**0.5027**	+4.83%
	ND@50	0.0151	0.3589	0.3845	0.4263	0.4791	0.4866	0.4685	**0.5111**	+5.04%
	HR@20	0.0241	0.5211	0.5403	0.5935	0.6603	0.6679	0.6740	**0.7134**	+5.85%
	HR@50	0.0462	0.5885	0.6098	0.6512	0.7171	0.7196	0.7272	**0.7814**	+7.46%

Table 3: Enhancement experiments on RetailRocket dataset. Red: improvement percentages.

	Recall@50				Hit Rate@50			
	Tail	Mid	Head	Overall	Tail	Mid	Head	Overall
ComiRec	0.4128	0.5976	0.6967	0.4929	0.4658	0.6440	0.7415	0.7171
+ CoLT	**0.5126**$^{+24.18\%}$	**0.6757**$^{+13.07\%}$	**0.7455**$^{+7.00\%}$	**0.5643**$^{+14.49\%}$	**0.5571**$^{+19.60\%}$	**0.7142**$^{+10.90\%}$	**0.7891**$^{+6.42\%}$	**0.7814**$^{+8.97\%}$
PIMIRec	0.4142	0.6011	0.7054	0.4962	0.4656	0.6489	0.7464	0.7196
+ CoLT	**0.4633**$^{+11.85\%}$	**0.6176**$^{+2.74\%}$	**0.7152**$^{+1.39\%}$	**0.5183**$^{+4.45\%}$	**0.5218**$^{+12.07\%}$	**0.6664**$^{+2.70\%}$	**0.7568**$^{+1.39\%}$	**0.7458**$^{+3.64\%}$
Re4	0.3774	0.5021	0.5772	0.6100	0.4824	0.6408	0.7460	0.7272
+ CoLT	**0.4069**$^{+7.82\%}$	**0.5489**$^{+9.32\%}$	**0.6174**$^{+6.96\%}$	**0.6433**$^{+5.46\%}$	**0.5122**$^{+6.18\%}$	**0.6891**$^{+7.54\%}$	**0.7935**$^{+6.37\%}$	**0.7586**$^{+6.32\%}$

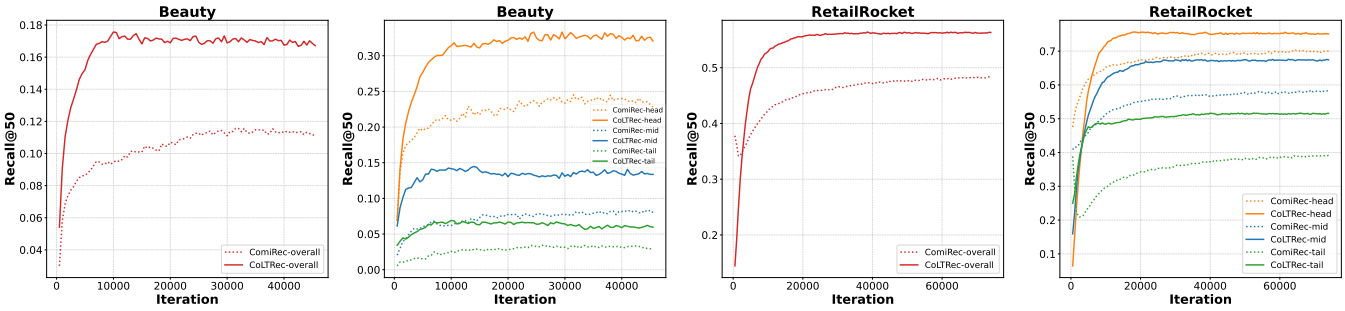

Figure 4: Training curves of ComiRec and CoLTRec, evaluated on the recall@50 per 500 iterations on validation set of Beauty and RetailRocket.

dataset, which performs averagely in overall performance comparison, to demonstrate the effectiveness of CoLT. From the results in Table 3, we find that:

- CoLT significantly improves the performance of all multi-interest recommendation models with an average increase of 8.13% and 6.31% in Recall@50 and Hit Rate@50 metrics, respectively, demonstrating good compatibility.
- CoLT achieves a good balance among head, mid, and tail items, improving the recommendation performance of all items without sacrificing head performance for tail performance.

- The main source of overall benefits is the improvement in tail item recommendations, with an average increase of 10.96% and 12.62% in Recall@50 and Hit Rate@50 metrics, respectively. This verifies the effectiveness of the CoLT module in solving the long-tail problem and demonstrates the potential of tail items in improving overall recommendation performance.

4.4 Convergence Speed Comparison

Figure 4 depicts that the overall recall of CoLTRec performs the best on Beauty and RetailRocket at the 10000th and 38500th iterations of testing, respectively. However, ComiRec peaks at the 31500th and 147000th iterations, respectively (only the first 74000 iterations of ComiRec on RetailRocket are displayed due to its long convergence iterations). CoLTRec only spends 1/3 of the iterations that ComiRec needs to peak, demonstrating the advantage of the CoLT module in accelerating model convergence.

The CoLT module is particularly effective in accelerating the convergence of tail items. The reason is that, by aggregating the embeddings of co-occurring items, CoLT enables the simultaneous update of tail items during the gradient descent of its co-occurring items, which leads to higher training frequency and more informative embedding learning context for tail items.

5 CONCLUSION

In this work, we pinpoint the long-tail problem in prior multi-interest works and suggest improving the recommendation for tail items, which has great potential for boosting the model's overall performance. We propose the CoLT module utilizing the co-occurrence relationship between items to enrich the training context and enhance the embeddings of tail items.

There are two limitations that will be addressed in future work: (1) the increased time complexity during training cannot be ignored when the number of interactions in the dataset surging; (2) further experiments comparing CoLT to common long-tail solutions (*e.g.*, re-weighting, margin loss) are needed to demonstrate its effectiveness. We believe our work offers a promising approach to tackle the long-tail problem, which has broader implications beyond multi-interest recommendation and can be applied to other types of recommendation tasks, thus stimulating future research.

REFERENCES

[1] Alex Beutel, Ed Huai-hsin Chi, Zhiyuan Cheng, Hubert Pham, and John R. Anderson. 2017. Beyond Globally Optimal: Focused Learning for Improved Recommendations. In *WWW*. 203–212.

[2] Yukuo Cen, Jianwei Zhang, Xu Zou, Chang Zhou, Hongxia Yang, and Jie Tang. 2020. Controllable Multi-Interest Framework for Recommendation. In *KDD*. 2942–2951.

[3] Zheng Chai, Zhihong Chen, Chenliang Li, Rong Xiao, Houyi Li, Jiawei Wu, Jingxu Chen, and Haihong Tang. 2022. User-Aware Multi-Interest Learning for Candidate Matching in Recommenders. In *SIGIR*. 1326–1335.

[4] Gaode Chen, Xinghua Zhang, Yanyan Zhao, Cong Xue, and Ji Xiang. 2021. Exploring Periodicity and Interactivity in Multi-Interest Framework for Sequential Recommendation. In *IJCAI*. 1426–1433.

[5] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep Neural Networks for YouTube Recommendations. In *RecSys*. 191–198.

[6] Carlos A Gomez-Uribe and Neil Hunt. 2015. The netflix recommender system: Algorithms, business value, and innovation. *ACM Transactions on Management Information Systems (TMIS)* 6, 4 (2015), 1–19.

[7] Long Guo, Fei Fang, Binqiang Zhao, and Bin Cui. 2022. UDM: A Unified Deep Matching Framework in Recommender Systems. In *CIKM*. 3122–3130.

[8] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yong-Dong Zhang, and Meng Wang. 2020. LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation. In *SIGIR*. 639–648.

[9] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2016. Session-based Recommendations with Recurrent Neural Networks. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.

[10] Jui-Ting Huang, Ashish Sharma, Shuying Sun, Li Xia, David Zhang, Philip Pronin, Janani Padmanabhan, Giuseppe Ottaviano, and Linjun Yang. 2020. Embedding-based Retrieval in Facebook Search. In *KDD*. 2553–2561.

[11] Sébastien Jean, KyungHyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. On Using Very Large Target Vocabulary for Neural Machine Translation. In *ACL*. 1–10.

[12] Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data* 7, 3 (2019), 535–547.

[13] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

[14] Chao Li, Zhiyuan Liu, Mengmeng Wu, Yuchi Xu, Huan Zhao, Pipei Huang, Guoliang Kang, Qiwei Chen, Wei Li, and Dik Lun Lee. 2019. Multi-Interest Network with Dynamic Routing for Recommendation at Tmall. In *CIKM*. 2615–2623.

[15] Houyi Li, Zhihong Chen, Chenliang Li, Rong Xiao, Hongbo Deng, Peng Zhang, Yongchao Liu, and Haihong Tang. 2021. Path-based Deep Network for Candidate Item Matching in Recommenders. In *SIGIR*. 1493–1502.

[16] Fuyu Lv, Taiwei Jin, Changlong Yu, Fei Sun, Quan Lin, Keping Yang, and Wilfred Ng. 2019. SDM: Sequential Deep Matching Model for Online Large-scale Recommender System. In *CIKM*. 2635–2643.

[17] Julian J. McAuley, Christopher Targett, Qinfeng Shi, and Anton van den Hengel. 2015. Image-Based Recommendations on Styles and Substitutes. In *SIGIR*. 43–52.

[18] Mohit Sharma and George Karypis. 2019. Adaptive matrix completion for the users and the items in tail. In *WWW*. 3223–3229.

[19] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural Graph Collaborative Filtering. In *SIGIR*. 165–174.

[20] Ji Yang, Xinyang Yi, Derek Zhiyuan Cheng, Lichan Hong, Yang Li, Simon Xiaoming Wang, Taibai Xu, and Ed H. Chi. 2020. Mixed Negative Sampling for Learning Two-tower Neural Networks in Recommendations. In *WWW (Companion Volume)*. 441–447.

[21] Xinyang Yi, Ji Yang, Lichan Hong, Derek Zhiyuan Cheng, Lukasz Heldt, Aditee Kumthekar, Zhe Zhao, Li Wei, and Ed H. Chi. 2019. Sampling-bias-corrected neural modeling for large corpus item recommendations. In *RecSys*. 269–277.

[22] Jianwen Yin, Chenghao Liu, Weiqing Wang, Jianling Sun, and Steven C. H. Hoi. 2020. Learning Transferrable Parameters for Long-tailed Sequential User Behavior Modeling. In *KDD*. 359–367.

[23] Shengyu Zhang, Lingxiao Yang, Dong Yao, Yujie Lu, Fuli Feng, Zhou Zhao, Tat-Seng Chua, and Fei Wu. 2022. Re4: Learning to Re-contrast, Re-attend, Re-construct for Multi-interest Recommendation. In *WWW*. 2216–2226.