Long-tail Session-based Recommendation

Siyi Liu
University of Electronic Science and Technology of China
Cheng Du, Si Chuan, China
ssuiliu.1022@gmail.com

Yujia Zheng*
University of Electronic Science and Technology of China
Cheng Du, Si Chuan, China
yjzheng19@gmail.com

ABSTRACT

Session-based recommendation focuses on the prediction of user actions based on anonymous sessions and is a necessary method in the lack of user historical data. However, none of the existing session-based recommendation methods explicitly takes the long-tail recommendation into consideration, which plays an important role in improving the diversity of recommendation and producing the serendipity. As the distribution of items with long-tail is prevalent in session-based recommendation scenarios (e.g., e-commerce, music, and TV program recommendations), more attention should be put on the long-tail session-based recommendation. In this paper, we propose a novel network architecture, namely TailNet, to improve long-tail recommendation performance, while maintaining competitive accuracy performance compared with other methods. We start by classifying items into short-head (popular) and long-tail (niche) items based on click frequency. Then a novel *preference mechanism* is proposed and applied in TailNet to determine user preference between two types of items, so as to softly adjust and personalize recommendations. Extensive experiments on two real-world datasets verify the superiority of our method compared with state-of-the-art works.

CCS CONCEPTS

• **Information systems → Recommender systems**.

KEYWORDS

Session-based recommendation, Long-tail recommendation, Neural network

ACM Reference Format:
Siyi Liu and Yujia Zheng. 2020. Long-tail Session-based Recommendation. In *Fourteenth ACM Conference on Recommender Systems (RecSys '20), September 22–26, 2020, Virtual Event, Brazil.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3383313.3412222

1 INTRODUCTION

Session-based Recommendation System (SRS) has attracted much attention because of numerous application areas in online services (e.g., e-commerce, music, and web page navigation). Such recommendation only relies on large, time-ordered action logs of anonymous users to predict the user's next action. [11–14, 25, 26].

Recent studies mainly focus on achieving state-of-the-art accuracy performance, without taking long-tail recommendation into consideration in SRS. However, previous studies provide a comprehensive explanation of the significance of long-tail based recommendation from two angles. For users, only recommending popular (short-head) items get them bored easily. Long-tail recommendation can increase the diversity and serendipity of a recommendation list, which surprises and satisfies them because long-tail items can still be very relevant to users [1, 8]. For business, long-tail items embrace relatively large marginal profit compared with short-head items, which means that long-tail recommendation can bring much more profit [2, 24]. Besides, long-tail recommendation can give users "one-stop shopping convenience", which can entice customers to consume both short-head items and long-tail items at one-step, and thereby creating more sales [5, 24].

Obstacles such as data sparsity stand in the way of applying long-tail recommendation, resulting in most existing recommendation systems, including SRS, being biased towards popular items [8, 19, 24]. In these methods, the preference to recommend the most popular items is a conservative but effective way to improve accuracy [1]. Previous works focusing on long-tail recommendations either sacrifice the accuracy of recommendations [1, 7, 19], or adopt side-information(e.g., user profiles, action types), which exceeds the data limit of SRS, to mitigate long-tail items data sparsity [3, 8].

The difficulty of long-tail recommendation is greater for session-based data. *Firstly,* due to the lack of side information and user profiles in sessions, it is hard for traditional long-tail recommendation methods to be applied in the session-based recommendation. *Secondly,* most of the traditional long-tail recommendation methods have difficulty in taking the sequential order of data into consideration. Thus, the recommendation accuracy of those methods severely drops when it comes to session-based recommendation. *Thirdly,* numerous session data is particularly sparse because each user could have several sessions and each session should be handled independently [6], which deteriorates the model performance on the long-tail recommendation. Therefore, the development of a new method that can solve the above difficulties is of great significance.

In this paper, to address long-tail phenomenon in SRS, we propose a novel network structure–TailNet. Unlike existing neural networks based recommendation algorithms, our method can significantly improve long-tail recommendation performance while maintaining competitive results with state-of-the-art method simultaneously. In particular, we first encode each session into latent representation. Then we introduce a novel *preference mechanism* into our model, which determines user preference between clicking

*Corresponding author. Email: yjzheng19@gmail.com

RecSys '20, September 22–26, 2020, Virtual Event, Brazil

ACM ISBN 978-1-4503-7583-2/20/09…$15.00
https://doi.org/10.1145/3383313.3412222

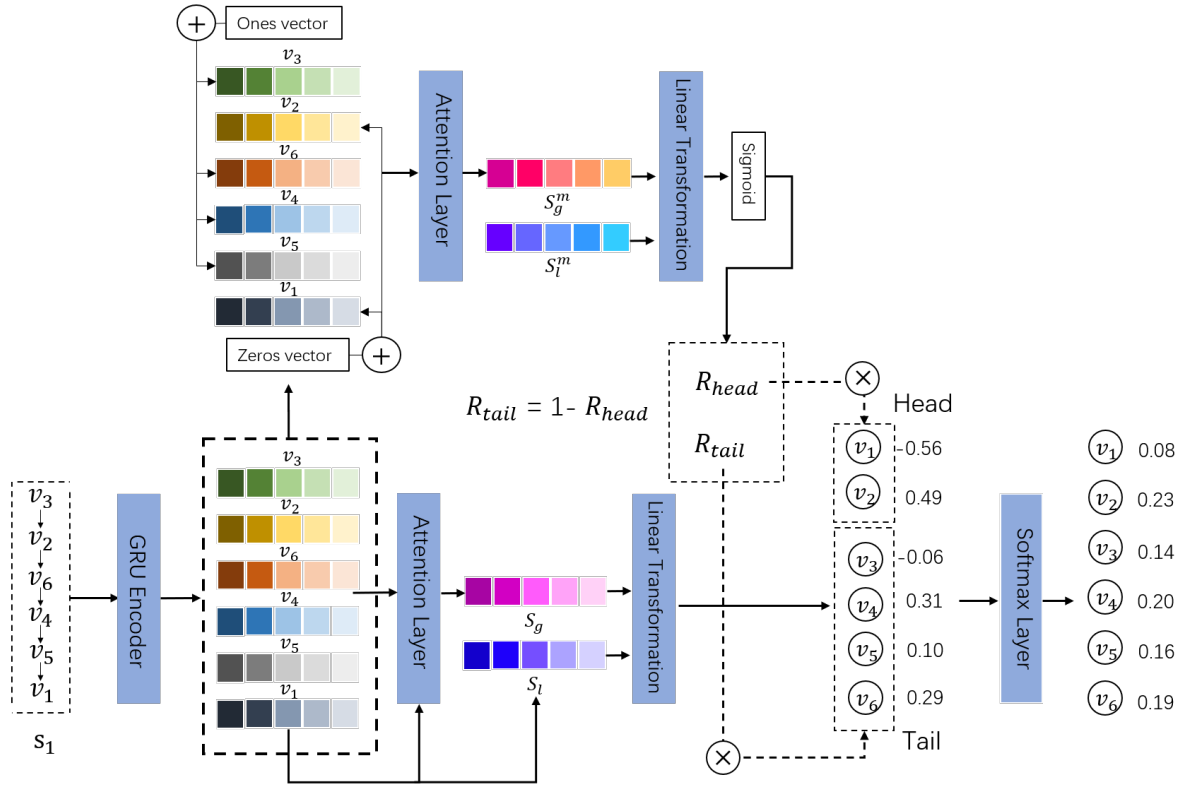


Figure 1: The framework of TailNet. We first encode the session by a GRU layer and get the session's latent representation V. Then it is sent to the *preference mechanism* and the attention layer. *Preference mechanism* uses V to generate *rectification factors* R_{head} and R_{tail}, which softly adjust the mode of TailNet. Next, the global session embedding S_g is generated by the attention layer. After that, the global embedding S_g and local embedding S_l (the last clicked item's representation) are concatenated and sent to a linear transformation. Finally, we predict the probability of each item with the adjusting of R_{head} and R_{tail}.

short-head items and long-tail items by learning from session representation and generate rectification factors. TailNet can utilize the rectification factors given by *preference mechanism* to softly adjust the recommendation mode of the model: recommend more short-head items or long-tail items. The *preference mechanism* is jointly learned with TailNet in an end-to-end back-propagation training paradigm. The main contributions of our work are two fold. First, We propose a novel neural network based model named TailNet which takes into consideration of long-tail phenomenon. Second, we conduct extensive experiments on real-world datasets. Experimental results show that our method achieves state-of-the-art performance on long-tail recommendation as well as relatively high accuracy.

2 MODEL

2.1 Notations

The aim of SRS can be defined as using the user's current session data to predict the user's next click item. $I = \{i_1, i_2, ..., i_{|I|}\}$ represents the set of unique items in all sessions. In this paper, we split items into I^H and I^T according to the **Pareto Principle**, which represents the set of short-head items and long-tail items respectively. $s = \{i_1, i_2, ...i_t\}$ represents the session which consists of items ordered by timestamps. $V = \{v_1, v_2, ..., v_t\}$ represents the latent representation for the session encoded by session encoder layer, where $v_t \in V$ represents a clicked item representation within the V at timestamp t. In TailNet, we first get vector $\widehat{c}$, where each element value is the score of the item before softmax. Then we use *preference mechanism* to get *rectification factors* marked as R_{head} and R_{tail} (corresponding to short-head items and long-tail items respectively) for soft adjustment. Finally, the TailNet outputs probabilities $\widehat{y}$ for possible items whereby each element value of vector $\widehat{y}$ is the recommendation score of the corresponding item.

embedding S_l^m (the last clicked item's representation), then compresses it into a representation S_p. After sending S_p to sigmoid function, we can get R_{head} and R_{tail}. We use Gated Recurrent Unit

(GRU) [4] to encode session s, where the GRU is defined as:

$$\begin{aligned} r_t &= \sigma(W_r \cdot [v_{t-1}, emb(i_t)]), \\ z_t &= \sigma(W_z \cdot [v_{t-1}, emb(i_t)]), \\ \widetilde{v}_t &= tanh(W_h \cdot [r_t \odot v_{t-1}, emb(i_t)]), \\ v_t &= (1 - z_t) \odot v_{t-1} + z_t \odot \widetilde{v}_t, \end{aligned} \tag{1}$$

where W_r, W_z and W_h denote the weights of the corresponding gates; $emb(i_t)$ denotes the embedding of item i_t; σ denotes the sigmoid function. We initialize $v_0 = 0$. And each session s is encoded into $V = \{v_1, v_2, ..., v_t\}$.

2.2 Preference Mechanism

We introduce a *preference mechanism* to softly adjust the mode of TailNet. As illustrated in Figure 1, The *preference mechanism* contains an encoder, a linear transformation layer with attention mechanism and a sigmoid function. The mechanism takes session's latent representations $V = \{v_1, v_2, ..., v_t\}$ as an input.

First, we need to embed the item type into the item representation. For inputs of long-tail and short-head items (splitted by **Pareto Principle** [2]), we add a vector of ones and a vector of zeros, respectively. However, making these vectors adjustable should distinguish the popularity bias more flexibility. To most directly illustrate the effectiveness of the proposed mechanism, we use vectors of ones and zeros here:

$$TE(v_i) = \begin{cases} v_i + \{1, 1, ..., 1\} & \text{if } \tau(v_i) \in I^T \\ v_i + \{0, 0, ..., 0\} & \text{if } \tau(v_i) \in I^H, \end{cases} \tag{2}$$

where $\tau(v_i)$ denotes a mapping function, $\tau(v_i)$ maps v_i to corresponding item i. The *ones* and *zeros* vector have the same size as the input, i.e.*ones*, *zeros* $\in \mathbb{R}^d$. d is the dimension of the inputs.

Then we adopt the attention mechanism to represent the session preference S_p:

$$\begin{aligned} \alpha_i^m &= W_0^m tanh(W_1^m TE(v_t) + W_2^m TE(v_i) + b^m), \\ S_l^m &= TE(v_t), \\ S_g^m &= \sum_{i=1}^{t} \alpha_i^m TE(v_i), \\ S_p &= [S_l^m; S_g^m] W_3, \end{aligned} \tag{3}$$

where $TE(v_i), TE(v_t) \in \mathbb{R}^d$ denote the representation of item i and the last-cliked item after Tail Encoder respectively; $W_0^m \in \mathbb{R}^{1 \times d}$ and $W_1^m, W_2^m \in \mathbb{R}^{d \times d}$ are weights in the attention mechanism of *preference mechanism*; matrix $W_3 \in \mathbb{R}^{2d \times 1}$ compresses two combined embedding vectors into the latent space $\mathbb{R}^1$; $b^m \in \mathbb{R}^d$ is a bias vector; α_i^m represents each item i's attention weight coefficient.

After that, a sigmoid function is ued to S_p to get *rectification factors*:

$$\begin{aligned} R_{head} &= \frac{1}{1 + e^{-S_p}}, \\ R_{tail} &= 1 - R_{head}, \end{aligned} \tag{4}$$

where R_{head} and R_{tail} denote the factors corresponding to short-head items and long-tail items respectively.

2.3 Session Pooling and Soft Adjustment

Same as the *preference mechanism* above, an attention layer and a linear transformation are used to process latent representations V:

$$\begin{aligned} \alpha_i &= W_0 tanh(W_1 v_t + W_2 v_i + b), \\ S_l &= v_t, \\ S_g &= \sum_{i=1}^{t} \alpha_i v_i, \\ \widehat{c} &= [S_l; S_g] W_4, \end{aligned} \tag{5}$$

where W_0, W_1, W_2, W_4 are weight matrices and b is the bias vector. S_g denotes global embedding and v_t denotes local embedding. Different from S_g^m and S_l^m in *preference mechanism*, S_g and S_l represent user's general and current interests in next-click item. Each element's value of $\widehat{c} = \{\widehat{c}_1, \widehat{c}_2, ..., \widehat{c}_{|I|}\}$ is the score of the corresponding item before softmax.

Finally, to adjust recommendation mode, we multiply each $\widehat{c}_i$ with corresponding *rectification factor* (R_{head} *or* R_{tail}), then apply softmax function on it to get probabilities for recommendation:

$$\widehat{y} = softmax(\widehat{c} \odot R), \tag{6}$$

where $R \in \mathbb{R}^{|I|}$ is a vector, and each element's value of it is the R_{head} or R_{tail} corresponding to the item in head or in tail, e.g., $R = \{R_{head}, R_{tail}, ..., R_{tail}\}$. $\odot$ denotes the element wise product. Each element's value of vector $\widehat{y} \in \mathbb{R}^{|I|}$ is the recommendation score of the corresponding item. The items with top-K values in vector value will be recommended as the final output. In addition, TailNet is trained jointly with *preference mechanism* by Back-Propagation Through Time (BPTT) [22]. We use the sum of the cross-entropy of each item's prediction and the ground truth as the loss:

$$\mathcal{L}(\hat{\mathbf{y}}) = -\sum_{i=1}^{|I|} \mathbf{y}_i \log(\hat{\mathbf{y}}_i) + (1 - \mathbf{y}_i) \log(1 - \hat{\mathbf{y}}_i), \tag{7}$$

where $\mathbf{y}$ denotes the one-hot encoding vector of the ground truth item.

3 EXPERIMENTS

We evaluate our proposed method on two real-world datasets: YOOCHOOSE[1] and 30MUSIC[21]. For a fair comparison, we use the same data preprocessing approach as previous studies [10, 23]. We use the recent fractions 1/4 of the YOOCHOOSE datasets as [20]. For both datasets, we use the last day to generate test data. Because of collaborative filtering methods cannot recommend an item which has not appeared before [6], we filter out that kind of items in test data. According to **Pareto Principle** [2], we split the itemset into short-head items and long-tail items.

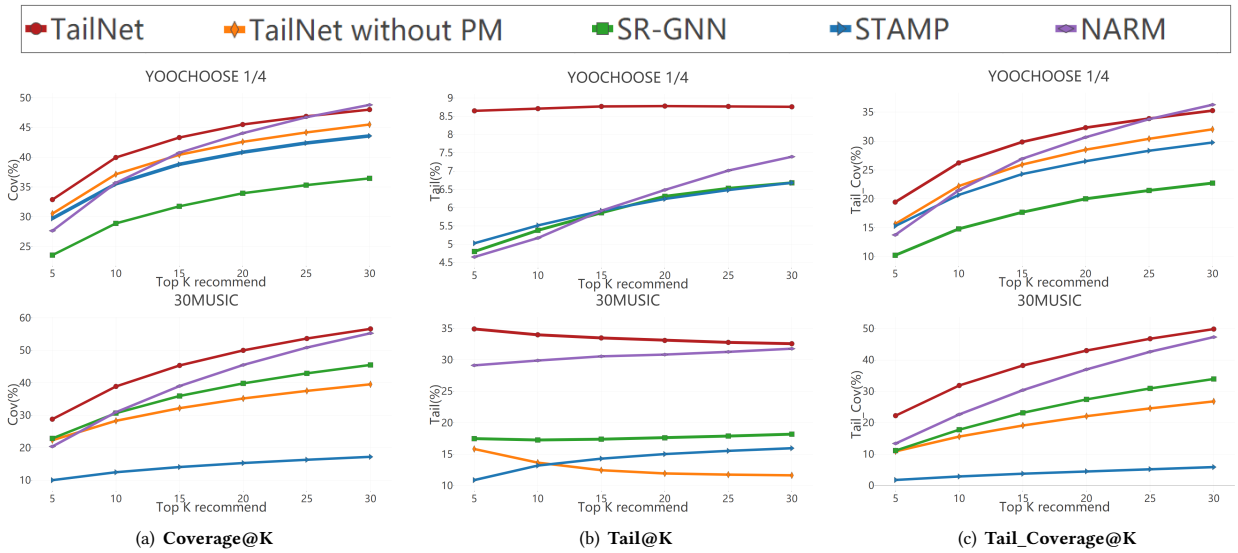
Baseline. We compare TailNet with frequency based methods (POP and S-POP), two RNN-based methods (GRU4REC [6] and NARM [9]), neighborhood-based method (Item-KNN [18]), repeat-explore method (RepeatNet [15]), attention-based method (STAMP [10]), two traditional Matrix Factorization or Markov Chain approaches (FPMC [17] and BPR-MF [16]), and GNN-based method (SR-GNN [23]).

[1] http://2015.recsyschallenge.com/challenge.html

Table 1: The performance of TailNet with other baseline methods on two datasets. (PM: Preference Mechanism)

Methods	30MUSIC					YOOCHOOSE 1/4				
	MRR@20	Recall@20	Coverage@20	Tail_Coverage@20	Tail@20	MRR@20	Recall@20	Coverage@20	Tail_Coverage@20	Tail@20
POP	0.18	0.69	0.01	0	0	0.3	1.36	0.06	0	0
S-POP	8.20	18.98	15.52	10.55	20.16	17.85	27.18	19.59	9.29	1.82
Item-KNN	15.71	37.68	75.14	79.04	54.60	21.68	53.01	65.93	60.72	14.65†
FPMC	9.17	14.47	84.37†	97.45†	60.00†	19.17	46.69	70.97†	84.74†	6.83
BPR-MF	6.97	12.25	49.62	86.95	19.94	16.37	23.77	44.82	68.32	2.96
GRU4REC	20.45	**39.12**†	36.60	26.18	18.64	22.41	59.58	25.79	11.71	1.87
NARM	20.19	36.68	45.51	36.99	30.84	28.88	69.29	44.05	30.65	6.49
STAMP	13.12	23.34	14.10	3.81	12.95	30.33	70.55	41.11	26.89	6.34
RepeatNet	18.01	33.01	32.24	24.49	23.06	**31.01**†	70.69	33.82	19.00	4.62
SR-GNN	27.28	37.86	40.59	28.71	20.48	30.64	**71.39**†	33.90	19.71	6.43
TailNet without PM	28.62	39.00	34.53	21.33	11.56	30.64	69.29	42.75	28.70	6.92
TailNet-propotion	27.91	37.29	38.55	25.37	19.85	28.99	67.04	43.86	30.32	8.43
TailNet	**28.70**†	38.34	**47.86**	**40.10**	**32.13**	30.97	69.41	**45.51**	**32.31**	**8.88**

Boldface indicates the best result in neural network based methods. † indicates the best results in all methods. The scores reported in [23] on the YOOCHOOSE dataset differ because of the discrepancy in test data. For neural network based methods, we choose the values of Coverage, Tail_Coverage and Tail when they are most accurate.



(a) **Coverage@K** (b) **Tail@K** (c) **Tail_Coverage@K**

Figure 2: Long-tail recommendation performance comparison between TailNet and state of the art baseline methods on two datasets. (PM: Preference Mechanism)

Evaluation Metrics. We use Recall@K and MRR@K to evaluate the performance of all algorithms in terms of accuracy. And for the evaluation of long-tail recommendation , we employ three diversity metrics into the evaluation.

- **Coverage@K and Tail-Coverage@K**: Coverage@K and Tail-Coverage@K measure how many different items and long-tail items ever appear in the top-K recommendations respectively. The Coverage@K and Tail-Coverage@K are defined as:

$$Coverage@K = \frac{|\cup_{s\subset S} L_K(s)|}{|I|}, \quad Tail\_Coverage@K = \frac{|\cup_{s\subset S} L_K^T(s)|}{|I^T|}, \tag{8}$$

where $L_K(s) = [i_1, i_2, ..., i_k]$ represents the list of top-K recommended items for session s and $L_K^T(s)$ represents a subset of $L_K(s)$ which contains items that belong to $|I^T|$

- **Tail@K**: Tail@K measure how many long-tail items in top-K for each recommendation list. The overall Tail@K is defined by averaging all test cases:

$$Tail@K = \frac{1}{|S|}\sum_{s\in S}\frac{|L_K^T(s)|}{K}. \tag{9}$$

4 RESULTS AND ANALYSIS

4.1 Comparison with Baseline Methods

The results of TailNet and all baseline methods over two datasets are presented in Table 1, and a more specific comparison between

TailNet and other neural network based methods in top-K recommendation is shown in Figure 2. The accuracy of traditional methods is relatively lower than that of neural network based methods. However, most of the traditional methods achieve better performance on long-tail recommendation than neural network based methods. This result indicates that only using accuracy to evaluate the quality of a recommendation algorithm may not be sufficient. Moreover, the trade-off between accuracy and long-tail recommendation highlights the challenge of balancing both metrics. In both datasets, TailNet achieves the best long-tail recommendation performance among state-of-the-art deep learning methods and the best accuracy among traditional methods, which confirms that our proposed method makes more comprehensive recommendations. Obviously, none of the baseline methods can balance long-tail recommendations and accurate recommendations as well as TailNet. Although our work mainly focuses on tackling long-tail recommendation, TailNet still achieves comparable results in terms of MRR@20, Recall@20. This result demonstrates that the application of *preference mechanism* only makes a bit of fluctuation in accuracy. *Preference mechanism* can accurately determine user preference between long-tail items and short-head items and adjust the recommendation result properly, not just rigidly recommend more niche items regardless of user preference.

4.2 The Effect of Soft Adjustment

TailNet can softly adjust the recommendation scores with the *rectification factors* generated by *preference mechanism*. However, because we divide short-head items and long-tail items according to **Pareto Principle**, we cannot exclude the possibility that the competitive performance of TailNet is largely due to this prior knowledge. Thus, we design a hard adjustment version of TailNet, namely TailNet-proportion, to study the effect of model based only on **Pareto Principle**. Specifically, for top-k recommendations, TailNet-proportion generates the recommendation list by combining the top $[k * p]$ short-head items and the top $k - [k * p]$ long-tail items, then sort them based on the scores. p represents the proportion of short-head items in a session and $[\cdot]$ indicates the rounding function. From the Table 1 we can observe that TailNet outperforms TailNet-proportion in all metrics, which verifies that soft adjustment rather than **Pareto principle** plays a major role in TailNet.

4.3 Application of Preference Mechanism in Other Models

Our proposed *preference mechanism* can be easily integrated with other neural network based model to improve the performance of long-tail recommendation (The mechanism only needs to take session's latent representation generated by the model as an input, and then outputs *rectification factors* R_{head} and R_{tail}). To verify that, we integrate the *preference mechanism* into two states of the art baseline models: STAMP and SR-GNN, and make comparisons between those models with and without *preference mechanism*. As shown in Table 2, STAMP and SR-GNN obtain significant improvement after applying the *preference mechanism*, which verifies the portability of the *preference mechanism*. By adopting the *preference mechanism*, neural network based methods can overcome information insufficiency obstacles in long-tail recommendations. Similar to TailNet's performance, the *preference mechanism* improves the longtail recommendation of STAMP and SR-GNN with a small range of accuracy fluctuation, which further validates the effectiveness of softly adjusting recommendation results according to user preferences.

Table 2: The performance of STAMP and SR-GNN, both with and without preference mechanism, on YOOCHOOSE 1/4 dataset. (PM: Preference Mechanism)

Methods	@20(%)				
	MRR	Recall	Coverage	Tail_Coverage	Tail
STAMP	30.33	**70.55**	41.11	26.89	6.34
STAMP with PM	**30.83**	70.46	**43.80**	**30.33**	**7.08**
SR-GNN	**30.64**	**71.39**	33.90	19.71	6.43
SR-GNN with PM	30.62	71.37	**35.46**	**21.99**	**6.57**

REFERENCES

[1] Gediminas Adomavicius and YoungOk Kwon. 2012. Improving aggregate recommendation diversity using ranking-based techniques. *IEEE Transactions on Knowledge and Data Engineering* 24, 5 (2012), 896–911.

[2] Chris Anderson. 2006. *The long tail: Why the future of business is selling less of more.* Hachette Books.

[3] Bing Bai, Yushun Fan, Wei Tan, and Jia Zhang. 2017. DLTSR: A deep learning framework for recommendation of long-tail web services. *IEEE Transactions on Services Computing* (2017).

[4] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* (2014).

[5] Sharad Goel, Andrei Broder, Evgeniy Gabrilovich, and Bo Pang. 2010. Anatomy of the long tail: ordinary people with extraordinary tastes. In *WSDM*.

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

[7] Joseph Johnson and Yiu-Kai Ng. 2017. Enhancing long tail item recommendations using tripartite graphs and Markov process. In *WI*.

[8] Jingjing Li, Ke Lu, Zi Huang, and Heng Tao Shen. 2017. Two birds one stone: on both cold-start and long-tail recommendation. In *MM*.

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

[10] Qiao Liu, Yifu Zeng, Refuoe Mokhosi, and Haibin Zhang. 2018. STAMP: Short-term attention/memory priority model for session-based recommendation. In *KDD*.

[11] Malte Ludewig and Dietmar Jannach. 2018. Evaluation of session-based recommendation algorithms. *User Modeling and User-Adapted Interaction* 28, 4-5 (2018), 331–390.

[12] Fei Mi and Boi Faltings. 2020. Memory Augmented Neural Model for Incremental Session-based Recommendation. *IJCAI* (2020).

[13] Fei Mi, Xiaoyu Lin, and Boi Faltings. 2020. ADER: Adaptively Distilled Exemplar Replay Towards Continual Learning for Session-based Recommendation. *RecSys* (2020).

[14] Massimo Quadrana, Paolo Cremonesi, and Dietmar Jannach. 2018. Sequence-aware recommender systems. *ACM Computing Surveys (CSUR)* 51, 4 (2018), 66.

[15] Pengjie Ren, Zhumin Chen, Jing Li, Zhaochun Ren, Jun Ma, and Maarten de Rijke. 2018. RepeatNet: A Repeat Aware Neural Recommendation Machine for Session-based Recommendation. *arXiv preprint arXiv:1812.02646* (2018).

[16] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *UAI*.

[17] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2010. Factorizing personalized markov chains for next-basket recommendation. In *WWW*.

[18] Badrul Munir Sarwar, George Karypis, Joseph A Konstan, John Riedl, et al. 2001. Item-based collaborative filtering recommendation algorithms. *WWW* (2001).

[19] Lei Shi. 2013. Trading-off among accuracy, similarity, diversity, and long-tail: a graph-based recommendation approach. In *RecSys*.

[20] Yong Kiam Tan, Xinxing Xu, and Yong Liu. 2016. Improved recurrent neural networks for session-based recommendations. In *DLRS*.

[21] Roberto Turrin, Massimo Quadrana, Andrea Condorelli, Roberto Pagano, and Paolo Cremonesi. 2015. 30Music Listening and Playlists Dataset.. In *RecSys*.
[22] Paul J Werbos et al. 1990. Backpropagation through time: what it does and how to do it. *Proc. IEEE* 78, 10 (1990), 1550–1560.
[23] Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan. 2019. Session-based Recommendation with Graph Neural Networks. *AAAI* (2019).
[24] Hongzhi Yin, Bin Cui, Jing Li, Junjie Yao, and Chen Chen. 2012. Challenging the long tail recommendation. *Proceedings of the VLDB Endowment* 5, 9 (2012), 896–907.
[25] Feng Yu, Yanqiao Zhu, Qiang Liu, Shu Wu, Liang Wang, and Tieniu Tan. 2020. TAGNN: Target Attentive Graph Neural Networks for Session-based Recommendation. In *SIGIR*.
[26] Yujia Zheng, Siyi Liu, and Zailei Zhou. 2019. Balancing Multi-level Interactions for Session-based Recommendation. *arXiv preprint arXiv:1910.13527* (2019).