Session-based Recommendation with Hierarchical Leaping Networks

Cheng Guo[1,2], Mengfei Zhang[1,2], Jinyun Fang[1], Jiaqi Jin[1,2], Mao Pan[1,2]
[1]Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China
[2]University of the Chinese Academy of Sciences, Beijing, China
{guocheng18s,zhangmengfei,fangjy,jinjiaqi19b,panmao17b}@ict.ac.cn

ABSTRACT

Session-based recommendation aims to predict the next item that users will interact based solely on anonymous sessions. In real-life scenarios, the user's preferences are usually various, and distinguishing different preferences in the session is important. However previous studies focus mostly on the transition modeling between items, ignoring the mining of various user preferences. In this paper, we propose a Hierarchical Leaping Network (HLN) to explicitly model the users' multiple preferences by grouping items that share some relationships. We first design a Leap Recurrent Unit (LRU) which is capable of skipping preference-unrelated items and accepting knowledge of previously learned preferences. Then we introduce a Preference Manager (PM) to manage those learned preferences and produce an aggregated preference representation each time LRU reruns. The final output of PM which contains multiple preferences of the user is used to make recommendations. Experiments on two benchmark datasets demonstrate the effectiveness of HLN. Furthermore, the visualization of explicitly learned subsequences also confirms our idea.

CCS CONCEPTS

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

KEYWORDS

Session-based Recommendation; Anonymous Sessions; Recommendation; Subsequence Learning; Leaping Network

ACM Reference Format:
Cheng Guo, Mengfei Zhang, Jinyun Fang, Jiaqi Jin, Mao Pan. 2020. Session-based Recommendation with Hierarchical Leaping Networks. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '20), July 25–30, 2020, Virtual Event, China.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3397271.3401217

1 INTRODUCTION

Session-based recommendation is the task of predicting future actions based solely on anonymous user sessions. It matches scenes like making recommendations for non-login users in online video, music and e-commerce services, which is crucial for attracting new users and increasing user stickiness.

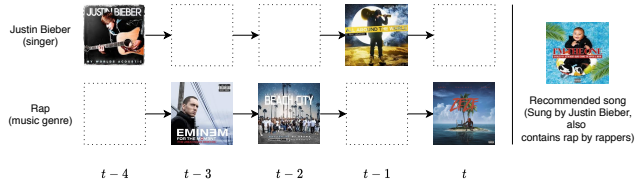

Figure 1: An example of music listening session which contains two preferences of the user. The recommended song matches both of his preferences.

Due to the limited information of session data, the time-ordered items have always been the main research objects. In recent works, Recurrent Neural Networks has been used to model the sequence behaviors and achieve promising results [1, 11]. Lately, attention mechanism is incorporated into the neural network to emphasize the user's main purpose [5, 6] and Memory Networks further improve the recommendation performance with collaborative information from neighborhood sessions [12]. More recently Graph Neural Networks are applied to capture complex item transitions by constructing sessions into graphs [13].

Despite the success of previous studies, however, the user's preferences are usually various and recommending items that match multiple preferences of the user is important. For example in Figure 1, the user's preferences at $t-4$ and $t-1$ are Justin Bieber's songs, while popular rap songs at $t-3$, $t-2$ and t. This session contains two preferences of the user. For the recommended song, existing approaches tend to only meet a single preference of the user, as they mainly model the transitions between items. However, it could potentially be more accurate if the recommended song caters to both preferences of the user, like the one in the example.

To address this issue, a comprehensive prior analysis is necessary. Firstly, different users' preferences for a same item are often diverse. For example, some users favor a song because of the singer, while others appeal to the genre. Ideally, the multiple preferences will be captured with greater granularity and clarity when items reflecting the same user preference are divided into a group. Secondly, an item could carry several preferences of a user. This indicates that an item may cross multiple preference groups. Lastly, the time order of items within the same preference group contains potential transfer relationships between items. Thus considering the item order is beneficial to modeling the preference in each group.

Based on the above insights, in this paper, we propose a Hierarchical Leaping Network (HLN), which explicitly identifies the user's various preference groups and aggregates them for accurate

SIGIR '20, July 25–30, 2020, Virtual Event, China

ACM ISBN 978-1-4503-8016-4/20/07…$15.00
https://doi.org/10.1145/3397271.3401217

recommendation. Specifically, HLN treats each preference group as a subsequence of the session, in which the items are related to a same preference. A Leap Recurrent Unit (LRU) is proposed to automatically learn the preference subsequences. The main idea of LRU is to skip the items that are irrelevant to current preference. To ensure the difference between learned preferences, HLN carries out the preference identification progressively, and takes existing preferences into account when making skip decisions. Then we introduce a Preference Manager (PM) to control the aggregated representation of existing preferences. Before processing the next preference, PM aggregates the information of existing preferences and passes it to LRU. Finally, the final output of PM contains multiple preferences of the user and is used to make recommendations. Our main contributions are summarized as follows:

- We propose a Hierarchical Leaping Network (HLN) to model users' multiple preferences with subsequent group learning. Each group adaptively identifies a subset of items in the session that reflects one preference distinguishing from other groups, and items that carry several preferences could cross groups.
- In HLN, we propose a Leap Recurrent Unit (LRU) with a preference gate filtering out existing preferences and a leap gate deciding whether to skip the current item. A module named Preference Manager (PM) helps to incrementally generate the existing group representations and fuse hierarchical preferences for the next recommendation.
- We conduct experiments on two benchmark datasets, the results demonstrate the effectiveness of HLN.

2 RELATED WORK

Session-based recommendation is a subtask of recommender system, in which earlier researches mainly focus on the mining of users' general interests. In particular, item-based collaborative filtering methods [10] use the item-to-item similarity matrix for recommendation, which is constructed from the co-occurrence information of items. Matrix factorization based techniques [4] factorize the user-item rating matrix into two low-rank matrices, each of which represents the latent factors of users or items. To obtain the ability of modeling sequential behaviors, Rendle et al. [9] propose to combine matrix factorization with Markov chains and achieve better performance.

With the rise of deep learning, most recent works are neural-based. Hidasi et al. [1] first propose a recurrent neural network approach and make marked improvements over previous methods. Tan et al. [11] further enhance the performance by introducing data augmentation techniques and taking temporal shifts in user behavior into account. Jannach and Ludewig [3] combine the recurrent neural network with a co-occurrence-based KNN method. To emphasis the user's main purpose in current session, Li et al. [5] employ an attention mechanism on recurrent neural network. Liu et al. [6] take a similar idea but replace recurrent neural network with simple MLP networks. Wang et al. [12] propose a parallel memory network to utilize collaborative information from neighborhood sessions. Wu et al. [13] treat sessions as graphs and apply a gated graph neural network to capture complex item transitions.

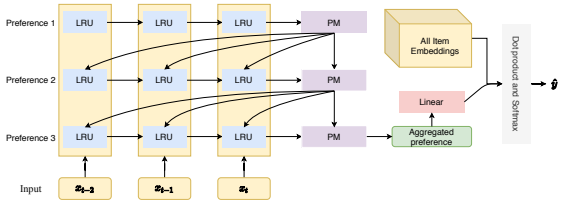

Figure 2: The architecture of HLN. LRU and PM are the two main modules. The LRU module tries to learn the user preferences one at a time by grouping items that reflect the same user preference. Specifically, LRU goes through the session in a recurrent fashion and skips items that are irrelevant to current preference. The PM module aggregates learned preferences and passes it to LRU to ensure that current preference is newly learned.

3 PROPOSED MODEL

We first formulate the session-based recommendation problem as follows. Let $\mathcal{I} = \{i_1, i_2, ..., i_{|N|}\}$ denote the set of all unique items in the session-based recommender system and N be the total number of items. An anonymous session sequence s can be represented by $s = [i_{s,1}, i_{s,2}, \ldots, i_{s,t}]$, where $i_{s,t}$ is the item that user interacted at timestamp t. The task of session-based recommendation is to predict the item at timestamp $t+1$.

Let $\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_{|N|}\}$ denote the embedding vectors with respect to the item set $\mathcal{I}$ and $\boldsymbol{x}_i \in \mathbb{R}^d$ for each of the item i. As shown in Figure 2, the proposed HLN model uses the corresponding list of item embedding vectors of a given session sequence as input and outputs scores $\hat{\boldsymbol{y}} = \{\hat{\boldsymbol{y}}_1, \hat{\boldsymbol{y}}_2, \ldots, \hat{\boldsymbol{y}}_{|N|}\}$ for all items. Finally, items with top-K scores are picked as candidate recommendations.

3.1 Leap Recurrent Unit

The leap recurrent unit (LRU) aims to identify the user's various preferences in current session. As previously analyzed, each preference of the user can be represented by a subsequence of current session. To identify the subset of items belonging to the same preference, LRU skips when the item is irrelevant to current preference. Besides, LRU also considers existing preferences to ensure current preference is newly learned.

Technically, LRU extends GRU with a preference gate and a leap gate. The preference gate is used to filter out existing preferences from current item. There are two reasons for that: 1) an item can carry several preferences of the user, 2) to avoid repeat learning of preferences. It's defined as:

$$\boldsymbol{g}_p = \sigma(\boldsymbol{W}_p \boldsymbol{x}_t + \boldsymbol{U}_p \boldsymbol{a}_{l-1}), \tag{1}$$

where $\boldsymbol{x}_t$ is the item vector at timestamp t, $\boldsymbol{a}_{l-1} \in \mathbb{R}^n$ is the aggregated representation of previous $l-1$ preferences, σ is the sigmoid function, $\boldsymbol{W}_p \in \mathbb{R}^{d \times d}$ and $\boldsymbol{U}_p \in \mathbb{R}^{d \times n}$ are weight matrices. The leap gate is used to skip timestamps when the preference-gated item is irrelevant to current preference. To generate the leap gate, we first output the probability of skipping:

$$p_{l,t} = \sigma(\boldsymbol{w}_q^\top \boldsymbol{h}_{l,t-1} + \boldsymbol{u}_q^\top (\boldsymbol{g}_p \odot \boldsymbol{x}_t) + b), \tag{2}$$

where $\boldsymbol{h}_{l,t-1} \in \mathbb{R}^n$ is the hidden state of LRU at timestamp $t-1$, b is a constant bias, $\boldsymbol{w}_q \in \mathbb{R}^n$ and $\boldsymbol{u}_q \in \mathbb{R}^d$ are weight vectors. Then

we sample the leap gate from the Bernoulli distribution:

$$g_{l,t} \sim \text{Bernoulli}(p_{l,t}), \tag{3}$$

where $g_{l,t} \in \{0, 1\}$. The current timestamp will be skipped when $g_{l,t} = 1$ and be kept when $g_{l,t} = 0$. Finally, we formulate LRU as follows:

$$\text{LRU}(\boldsymbol{h}_{l,t-1}, \boldsymbol{x}_t, \boldsymbol{a}_{l-1}) = \begin{cases} \text{GRU}(\boldsymbol{h}_{l,t-1}, \boldsymbol{x}_t) & \text{if } g_{l,t} = 0 \\ \boldsymbol{h}_{l,t-1} & \text{if } g_{l,t} = 1. \end{cases} \tag{4}$$

We use the last hidden state of LRU as the representation of each learned user preference.

3.2 Preference Manager

The preference manager (PM) manages learned user preferences in current session and assists the progressive identification of all user preferences. Before the processing of the next user preference, PM generates an aggregated representation of existing preferences and passes it to LRU. Then LRU can determine whether items contain preferences that haven't been learned with the preference gate.

The preference aggregation method can be implemented in many ways, such as *sum*, *max-pooling* or *attention*. Since it's not our major concern, we choose *sum* as the implementation. Then the aggregated representation of previous l preferences is

$$\boldsymbol{a}_l = \boldsymbol{h}_{1,last} + \boldsymbol{h}_{2,last} + \cdots + \boldsymbol{h}_{l,last}, \tag{5}$$

where $\boldsymbol{h}_{l,last}$ is the representation of l-th preference. Suppose L is the total number of learned user preferences, we use $\boldsymbol{a}_L$ to make recommendations.

3.3 Prediction and Training

For prediction, we first transform the aggregated preference vector $\boldsymbol{a}_L$ into the item embedding space, which generates an ideal item that matches all of the user preferences:

$$\boldsymbol{x}_{ideal} = \boldsymbol{W}\boldsymbol{a}_L, \tag{6}$$

where $\boldsymbol{W} \in \mathbb{R}^{d \times n}$. Then we apply dot product and a softmax function to get the match scores for all items with the ideal item:

$$\hat{\boldsymbol{y}}_i = \text{softmax}(\boldsymbol{x}_i^\top \boldsymbol{x}_{ideal}), \tag{7}$$

where $\hat{\boldsymbol{y}}_i$ is the score for item i.

When training, HLN is not differentiable since the leap gate has discrete values of 0 and 1. To solve this, we apply Gumbel-Softmax [2, 7] distribution as a surrogate of Bernoulli distribution to the leap gate when computing gradients during backpropagation.

The loss function is defined as the cross-entropy of the prediction and the ground truth:

$$\mathcal{L}(\hat{\boldsymbol{y}}) = -\sum_{i=1}^{N} \boldsymbol{y}_i \log(\hat{\boldsymbol{y}}_i) + (1 - \boldsymbol{y}_i) \log(1 - \hat{\boldsymbol{y}}_i), \tag{8}$$

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

4 EXPERIMENTS

4.1 Datasets

We evaluate our proposed method on two real-world datasets: (1)**Yoochoose**[1]: RecSys'15 Challenge dataset from an e-commerce website, (2) **LastFM**[2]: a music recommendation dataset obtained through the LastFM API.

For the Yoochoose dataset, following [13] and [5], we filter out sessions of length 1 and items that appear less than 5 times. We train the model on the most recent 1/64 and 1/4 of Yoochoose as done in [5, 12, 13]. For the LastFM dataset, we follow [12] except that we recommend songs and use last 3 months' data with the last week for testing. After the preprocessing step, there are 23,726,880 sessions and 37,483 items remaining in the Yoochoose dataset, while 159,898 sessions and 28,946 items in the LastFM dataset.

4.2 Experimental Settings

Baselines We compare and analyze our model with nine representative methods, including conventional methods and deep learning algorithms. On the one hand, the methods of conventional approaches are: popularity-based **S-POP**, KNN-based method **SKNN** [3], **BPR-MF** [8] which applies matrix factorization with Bayesian Personalized Ranking learning optimization, and **FPMC** [9] that combines Matrix Factorization and Markov Chain. On the other hand, the neural approaches we compare with include two RNN-based methods (**GRU4REC** [1], **NARM**[5]), a session graph learning method (**SR-GNN** [13]), an MLP approach (**STAMP** [6]), and a memory augmented approach (**CSRM** [12]).

Evaluation Metrics In our experiments, we evaluate our model with the classical Recall@k and MRR@k metrics and employ k = 5 and k=10 to report the evaluation metrics.

Experimental Setup The batch size, embedding size and hidden size are set to 64, 100 and 200 respectively. The layers of HLN is 3. We optimize the model with Adam. The initial learning rate is 0.001 and decays by 0.1 after every 3 epochs.

4.3 Performance Comparison

Table 1 presents the detail performance of all methods on three datasets, with the best results highlighted in boldface. From the table, we have the following observations :1) HLN outperforms both conventional methods and recent neural approaches and achieves state-of-the-art performance on benchmark datasets. This verifies the power of considering subsequence group learning and multiple preference identification. 2) HLN achieves better performance than session interest modeling approaches such as NARM, SR-GNN and STAMP, as they take a session as a fusion of global and current interests, which is insufficient in capturing diverse but clear-cut differentiated aspects of session preference. 3) The neural approaches outperform the conventional methods. SKNN and CSRM are both competitive in traditional and neural approaches, respectively. This demonstrates that collaborative neighborhood information is still a strong signal in this problem. 4) FPMC and RNN-based methods (GRU4Rec and NARM) significantly outperform simple models like BPR-MF and SPOP, thus proving the effectiveness of sequential

[1] http://2015.recsyschallenge.com/challege.html
[2] https://www.dtic.upf.edu/ocelma/MusicRecommendationDataset/lastfm-1K.html

Table 1: Performance comparison of HLN with baseline methods. The best performing method is boldfaced, and the second best method is underlined.

Methods	YOOCHOOSE 1/64				YOOCHOOSE 1/4				LastFM			
	MRR		Recall		MRR		Recall		MRR		Recall	
	@5	@10	@5	@10	@5	@10	@5	@10	@5	@10	@5	@10
SPOP	1.23	1.79	2.21	3.24	0.38	0.46	0.89	1.52	2.15	2.64	3.77	4.63
SKNN	22.34	24.94	38.48	48.69	24.12	25.78	39.29	50.50	26.65	27.30	25.89	27.48
BPR-MF	17.65	18.89	24.16	31.30	15.37	16.24	21.77	22.93	14.19	15.37	19.06	20.14
FPMC	19.28	20.05	28.91	37.44	17.33	18.54	27.09	36.12	17.60	17.91	23.62	24.58
GRU4REC	22.82	24.53	39.67	52.43	23.29	26.05	42.38	55.49	25.73	25.92	20.65	20.95
STAMP	23.55	25.17	40.09	52.96	25.26	28.32	43.16	57.67	26.03	26.57	30.31	30.66
NARM	25.73	27.42	45.21	57.83	26.77	28.51	45.09	57.98	26.02	26.69	31.68	36.27
CSRM	27.13	28.24	45.82	57.96	28.64	29.14	48.09	60.54	32.22	**33.47**	31.61	36.46
SR-GNN	27.26	28.92	45.39	58.01	29.34	31.08	48.15	61.06	27.34	27.91	32.04	36.37
HLN	**28.52**	**29.86**	**46.53**	**58.91**	**29.64**	**31.37**	**48.52**	**61.47**	**32.57**	32.97	**34.18**	**38.09**

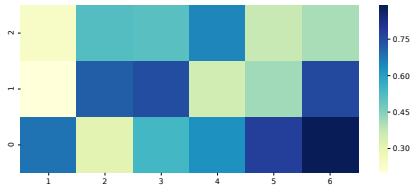

Figure 3: Visualization of retention rate from group-0 to group-2. The square scores represent the average retention rates of the items at each position.

behavior modeling. The time order in each subsequence group we considered helps improve the ability to express sequence behaviors.

4.4 Analysis

To understand the influence of subsequence learning and preferance leaping mechanism on next recommendation, we select all sessions of length 6 (according to the statistics of datasets [13]) and calculate the the retention rate of each item in all groups. The retention rate is the average of the probability that each item belong to a subsequence group without leaping. Figure 3 shows the results on the Yoochoose 1/64 dataset. We have the following observations: 1) There are obvious different distribution between different groups, which proves that our model can learn multiple session preferences. 2) In the first subsequence group, users' behavior sequences have significant interest transfer changes, and the value of the last item is larger, which is shows that the click on the last item plays a great role for next item prediction. 3) As the number of groups increases, the occurrence of bypass tends to increase, which is probably due to the less useful information is learned in the subsequence. This shows that our model has a gradual learning process, and 3 subsequence groups can learn well for session multiple preferences.

5 CONCLUSION

In this paper, we propose a Hierarchical Leaping Network (HLN) to model users' multi-preferences by grouping items share some relationships. Two modules named Leaping Recurrent Unit (LRU) and Preference Manager (PM) in HLN are introduced. LRU automatically identifies whether to skip preference-unrelated items and PM controls the existing preferences. Empirical experiments show that our model achieves state-of-the-art performance on all datasets.

ACKNOWLEDGMENTS

This work was supported by Beijing Municipal Science and Technology Project Z201100001820003.

REFERENCES

[1] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2015. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939* (2015).
[2] Eric Jang, Shixiang Gu, and Ben Poole. 2016. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144* (2016).
[3] Dietmar Jannach and Malte Ludewig. 2017. When recurrent neural networks meet the neighborhood for session-based recommendation. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*. 306–310.
[4] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 42, 8 (2009), 30–37.
[5] 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*. 1419–1428.
[6] 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.
[7] Chris J Maddison, Andriy Mnih, and Yee Whye Teh. 2016. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712* (2016).
[8] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2012. BPR: Bayesian personalized ranking from implicit feedback. *arXiv preprint arXiv:1205.2618* (2012).
[9] 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.
[10] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*. 285–295.
[11] 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.
[12] Meirui Wang, Pengjie Ren, Lei Mei, Zhumin Chen, Jun Ma, and Maarten de Rijke. 2019. A collaborative session-based recommendation approach with parallel memory modules. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 345–354.
[13] Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan. 2019. Session-based recommendation with graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 346–353.