Information Sciences 569 (2021) 125–137

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Fusion of latent categorical prediction and sequential prediction for session-based recommendation

Zizhuo Zhang, Bang Wang *

School of Electronic Information and Communications, Huazhong University of Science and Technology (HUST), Wuhan, China

ARTICLE INFO

Article history:
Received 23 November 2020
Received in revised form 8 February 2021
Accepted 2 April 2021
Available online 09 April 2021

Keywords:
Latent category mining
Latent categorical representation
Sequential representation
Prediction fusion
Session-based recommendation

ABSTRACT

Session-based recommendation is to predict the next item for an anonymous item sequence. Most of recent neural models have focused on how to learn sessions' sequential representations based on the assumption that items can be projected into a single latent embedding space to describe their latent attributes. In this paper, we argue that an item can also be described by some latent categorical abstractions. To examine our argument, we first mine items' latent categorical distributions via random walk on an item graph constructed from sessions. We design a new neural model which consists of two prediction modules: One is to learn a session's latent categorical representation; The other is to learn a session's sequential representation. Each module independently makes a next item prediction, and their predictions are fused as the final recommendation result. Experiments on three public datasets validate that our model achieves performance improvements over the recent state-of-the-art algorithms.

1. Introduction

Recommendation systems have been becoming ubiquitous in most of online e-commerce platforms, news portals and etc. [1–5], which help users to easily discover their interested items in this era of information explosion. In some application scenarios, users' personal information may not be available beforehand, for example, the click behaviors of a user when browsing websites without logging-in. In such cases, we only have many anonymous users' browsing sequences. Recently, *session-based recommendation* (SBR) has been proposed to predict the next behavior of a user (e.g. the next item to click) for an ongoing anonymous session [6]. According to [7–10], the SBR task can be formally defined as follows.

An anonymous session is defined as an item sequence $s = \{v_1, v_2, \ldots, v_t\}$ ordered by timestamps, where $v_i \in V$ represents the i-th clicked item by a user, and V is the set including all m unique items. For a session s, the SBR task is to predict the next item to be clicked, i.e. to predict v_{t+1}. Generally, a SBR model outputs the preference scores for all candidate items in V, that is, $\hat{\mathbf{y}} = \{\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_m\}$. The items with the top-K scores in $\hat{\mathbf{y}}$ will be chosen to form a recommendation list.

The main challenges of the SBR task are from the anonymity, as users' profiles are not available and items' information are also limited. Nonetheless, for its wide applications, recent years have witnessed a surge of research on SBR [11]. Some early SBR algorithms adopt simple rules for next item prediction, like based on items' popularity or items' co-occurrence [12,13]. Although they exploit items' statistics in datasets, they ignore the *sequential information*, i.e. items and their orders in an ongoing session. Some have proposed to use Markov chains to model the transitions of sequential items in sessions

* Corresponding author.
E-mail addresses: zhangzizhuo@hust.edu.cn (Z. Zhang), wangbang@hust.edu.cn (B. Wang).

https://doi.org/10.1016/j.ins.2021.04.019
0020-0255/

[14,15]. However, such approaches suffer from the explosion of state space when considering all items [7]. Also their independent transition assumption might not be practical in the SBR task, leading to unsatisfactory performance [9].

Many recent approaches have employed powerful deep learning techniques [16]. Various models based on *recurrent neural networks* (RNNs) have been designed to capture the change of users' interests over time when learning sessions' sequential representations for next item prediction [17,7,18–21]. Some work have proposed to construct graph based on training sessions, and then adopted *graph neural networks* (GNNs) to learn items' embeddings from the graph [9,22]. These models have obtained great improvements over early algorithms, however, they have mainly focused on learning sessions' sequential representations from a single item embedding space.

In our daily life, it is common that we not only describe an item by its attributes, like shape, size, color and etc, but also by its categories, like plant, animal, insect and etc. Indeed, taxonomy is everywhere in our daily life for better understanding our world. In this regard, we may need different taxonomy systems to describe an item from different angles. Motivated from such considerations, besides the widely used items' embeddings, we propose to also put items against some categorical taxonomy system to mine some categorical information for items. As the SBR task exploits only anonymous item sequences, we have to assume some *latent categories*; While an item can belong to one or more latent categories. So we actually mine a kind of *latent categorical distributions* (LC distributions) for each item also. Furthermore, we treat items' LC distributions independent with their embeddings and exploit them to produce a kind of complementary prediction for final recommendation.

In this paper, we propose a novel FLCSP model, abbreviated from *fusion of latent categorical prediction and sequential prediction*, for session-based recommendation. The FLCSP model consists of three modules: the FLCSP-LC, FLCSP-S and fusion module. The fusion module fuses two kinds of next item predictions, that is, *latent categorical prediction* from FLCSP-LC and *sequential prediction* from FLCSP-S, to output final recommendation result. In FLCSP-LC, we first mine items' LC distributions by random walk with restart on an item graph constructed from training sessions and design an attention network to learn the LC representation of a session. In FLCSP-S, we design a neural model consisting of a recurrent network and a casual convolutional network to learn a session's sequential representation from both long-term and short-term perspective, respectively. Experiments on three public datasets show that our model can achieve performance improvements over the recent state-of-the-art algorithms.

In summary, our main contributions can be summarized as follows:

- Propose a FLCSP-LC module to mine items' LC distributions and learn sessions' LC representations for LC prediction.
- Propose a FLCSP-S module to learn sessions' sequential representations for sequential prediction.
- Propose a fusion mechanism to fuse the two kinds of predictions for final recommendation result.

The rest of the paper consists of the following sections: Section 2 reviews the related work. The proposed FLCSP model is presented in Section 3 and experimented in Section 4. Finally, the paper concludes in Section 5.

2. Related work

We review the mostly related work on the SBR task, which can be generally divided into conventional methods and deep learning-based methods.

2.1. Conventional methods

Some studies [12,13] in this group leverage simple decision rules for recommendation, like based on items' popularity or based on the item-to-item co-occurrence similarity. Some have applied the classic matrix factorization for the SBR task [23–31]. For example, the BPR-MF [23] executes matrix factorization based on the pairwise ranking optimization. Although these approaches are intuitive, they ignore the items' order, i.e. sequential information in a session. Some work have proposed to use Markov-chain to model the relations between two consecutive items [14,15,32]. For example, Shani et al. [14] compute items' preference probabilities for recommendation with the transition probabilities between items based on a Markov decision process. However, the state space of a Markov model could become unmanageable if all items and their orders in sessions have to be included [7]. Furthermore, the independence assumption of state transition also confines its prediction accuracy [9].

2.2. Deep learning methods

There are many successful cases of deep learning technologies in recommendation system [11]. Recurrent neural networks (RNNs) as the major tools for sequential data have been employed for solving the SBR task [17,7,18–21,33,34]. Hidasi et al. [17] apply the *gated recurrent unit* (GRU) [35] to model the sessions as time series. Based on GRU to extract the features of item sequences, Li et al. [7] add attention mechanism to learn both the global and local representations of sessions to predict the target items. Wang et al. [20] and Pan et al. [33] further leverage the collaborative relationships between sessions to augment sessions' representations, except that they select the neighbor sessions for each session in a different way. Ren et al.

[21] argue that users often repeat their behaviors in an ongoing session, so they propose a repeat-explore mechanism for the SBR task to automatically learn the switch probabilities between their so-called repeat and explore modes.

Besides RNN models, some other neural networks have also been designed for the SBR task [9,8,22,36–38]. For example, Liu et al. [8] leverage a MLP network with an attentive mechanism to capture users' long-term and short-term interests for sessions' representations. Some recent works have employed GNN models for learning items' embeddings [9,36,38,39]. For example, Wu et al. [9] first construct an item graph from sessions, and they adopt a GNN to learn items' embeddings, based on which another attentive network is used to learn sessions' representations. Qiu et al. [36] convert each session into a graph and transfer the recommendation problem as a graph classification problem, where a weighted attention graph network is designed to learn items' embeddings.

In this paper, we also learn sessions' sequential representations, yet employing both a GRU model and a causal convolutional network [40] with an attentive mechanism. Furthermore, we propose to also mine items' LC distributions for prediction and design a fusion mechanism to fuse two types of next item predictions.

3. The FLCSP model

Fig. 1 presents our proposed FLCSP model, which consists of three modules: (1) *Latent categorical prediction* (FLCSP-LC): It mines all items' LC distributions and learns the LC representation of a session for prediction; (2) *Sequential prediction* (FLCSP-S): It learns the sequential representation of a session for prediction; (3) *Prediction Fusion*: It fuses the above two predictions to output the final preference scores for recommendation.

3.1. Latent categorical prediction

In this module, we propose to mine items' LC distributions based on graph random walk and design an attentive neural network to learn the LC representation of a session for prediction.

3.1.1. Mining items' LC distribution

Fig. 2 illustrates the basic idea of our LC distribution mining. We first construct an item graph from all sessions to describe all item-to-item transitions as well as the transitions' frequencies. We would like to mine latent but collaborative relationships among anonymous users from items' transition patterns in such an item graph. Based on this item graph, we design an anchor selection strategy to select a few of representative items and perform random walk with restart to compute the LC distribution for each item.

Item-graph construction: We construct an item graph from all training sessions as follows: Each node represents an item v_i; each edge (v_i, v_j) denotes that the item v_j is right after v_i in at least one session. The edge weight w_{ij} is computed as the occurrences of the pair (v_i, v_j) in all training sessions divided by the out-degree of the node v_i in the item graph.

Anchor selection: We argue that each item can be generally included into a few of latent categories, as per our discussion in Section 1. On the other hand, we also have no a priori knowledge about appropriate definitions and characteristics for latent categories. To solve this dilemma, we propose to select some items each representing one latent category. This is motivated from our observations on the constructed item graph, from which some item nodes are with many edges connecting to other nodes. This suggests that such items, called *anchors*, might be more important or representative than other items. Indeed, this observation is in accordance with the core idea of the well-known PageRank web-page ranking algorithm [41]. So we compute the items' PageRank values in the item-graph to select anchors.

For all items in V, we sort their PageRank values in a decreasing order into a list Γ_V. We design a threshold-based anchor selection strategy as follows. For the i-th item v_i with its Pagerank value $\gamma_{v_i} \in \Gamma_V$, if γ_{v_i} is not much larger than its subsequent items' PageRank values $\gamma_{v_j}, j > i$, it suggests that this item is with similar importance with its subsequent items, in other words, not representative enough. As such, we compute a *representative difference* for $\gamma_{v_i} \in \Gamma_V$ by

$$\Delta_{v_i} = \gamma_{v_i} - \frac{1}{k}\sum_{j=i+1}^{i+k}\gamma_{v_j}, \tag{1}$$

where k is used to control how many items are taken after v_i in the list Γ_V to compute the representative difference Δ_{v_i}. An item is called a *cutoff item*, if $\Delta_v < \Delta_{th}$, where Δ_{th} is a predefined threshold. The items before the cutoff item in the list Γ_V are selected as anchors. We let A denote the set of selected anchors, and let $n = |A|$.

LC distribution mining: As we regard anchors as latent categories, we would like to know how an item can be included into which category and with what probability. To do so, we first apply a *random walk with restart* (RWR) for an anchor v_a to obtain the *convergence probabilities* of other items to the anchor by iteratively computing the following equation:

$$\mathbf{v}^{(t+1)} = (1-\alpha)\cdot\mathbf{P}_{vv}\mathbf{v}^{(t)} + \alpha\cdot\mathbf{r},\ \ k = 0, 1, \ldots \tag{2}$$

where $\mathbf{v}^{(t+1)}$ is the probability vector that the item node is visited in the t-th iteration. $\mathbf{P}_{vv} \in \mathbb{R}^{m\times m}$ is the transition matrix by row-normalizing the item graph adjacency matrix $\mathbf{A}_{vv} \in \mathbb{R}^{m\times m}$. And $\mathbf{r}$ is defined as a *restart vector*: For anchor $v_a, \mathbf{r}(v_j) = 1$, if $v_j = v_a$; Otherwise, $\mathbf{r}(v_j) = 0$. The *restart probability* $\alpha \in [0, 1]$ as a hyper-parameter means the probability α of returning to

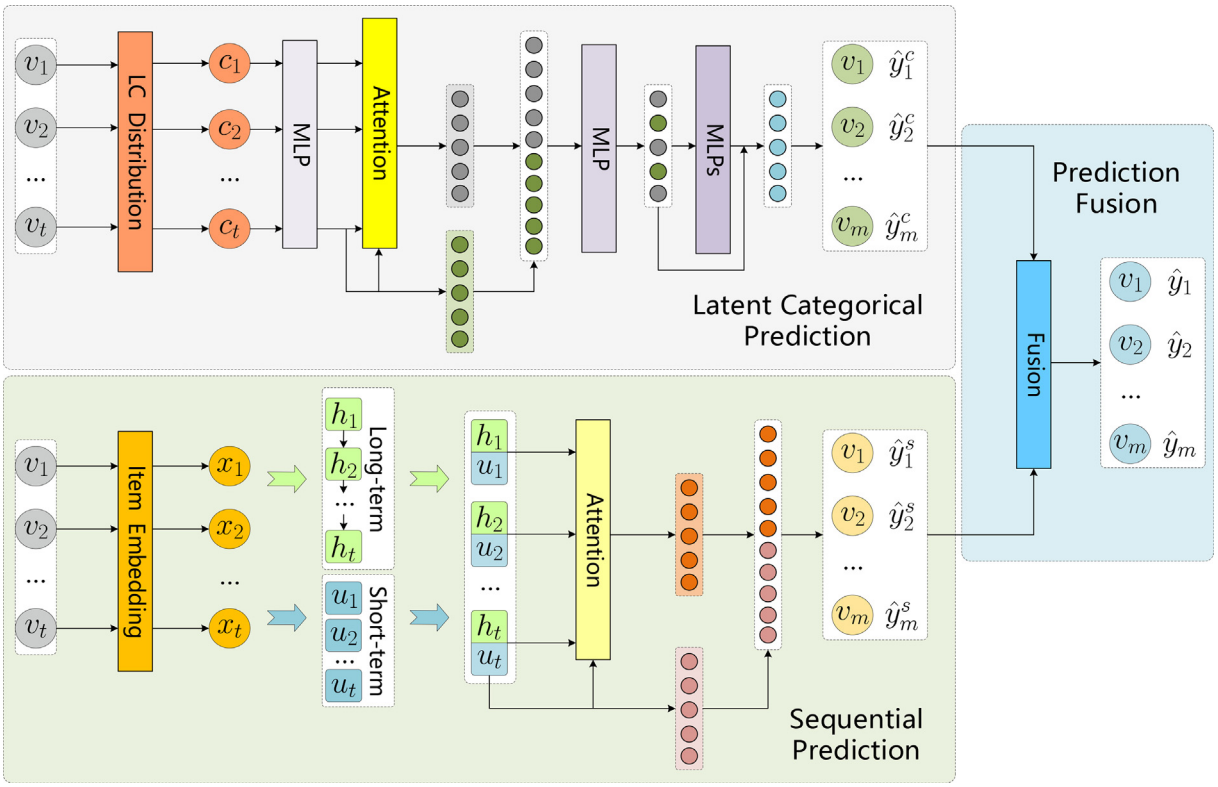

Fig. 1. Overview of the proposed FLCSP model. The left-top is the LC prediction part, the left-bottom is the sequential prediction part, and the right is the fusion mechanism to fuse two types of predictions.

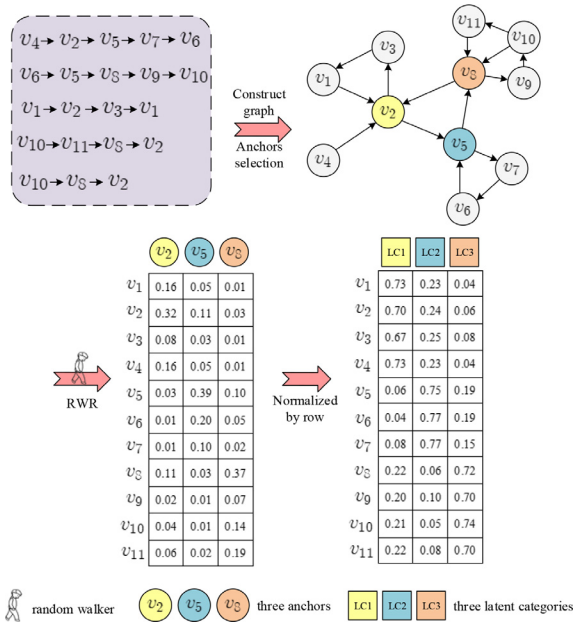

Fig. 2. An example of item-graph construction, anchor selection and LC distribution mining. For the five sessions, we first construct an item-graph and then select v_2, v_5, v_8 as anchors with the top-3 highest PageRank values to represent three latent categories. We next apply RWR for each anchor to compute the convergence probabilities and row-wisely normalize them to obtain LC distributions.

the restart node v_a and $1-\alpha$ probability walking to its adjacent nodes in each iteration. The initial probability vector $\mathbf{v}^{(0)}$ is randomly initialized. The iteration is terminated when $\|\mathbf{v}^{(t+1)}-\mathbf{v}^{(t)}\|<10^{-8}$.

Let $\mathbf{v}_a$ denote the *convergence probability vector* for anchor v_a after RWR termination, which could be interpreted as the *relevance degrees* between the anchor and other items. A larger element in $\mathbf{v}_a$ could indicate that the corresponding item is more relevant to the anchor, in other words, more likely to be included into the latent category represented by this anchor. We normalize the relevance degrees for an item against all anchors to compute its LC distribution as follows: We first construct a convergence matrix $\mathbf{C}_V \in \mathbb{R}^{m\times n}$ with the a-th column being $\mathbf{v}_a$ for anchor v_a. We next row-wisely normalize $\mathbf{C}_V$ to obtain a LC distribution matrix $\mathbf{C}'_V \in \mathbb{R}^{m\times n}$, where the i-th row is regarded as the LC distribution of the item v_i among all latent categories, denoted by $\mathbf{c}_i$ in what follows.

3.1.2. Learning a session's LC representation for prediction

To leverage items' LC distributions, we propose a FLCSP-LC neural model consisting of an attentive network with MLP layers to learn the LC representation of a session for prediction. For a session s, let $s_c=\{\mathbf{c}_1,\mathbf{c}_2,\ldots,\mathbf{c}_t\}$ denote the sequence of its items' LC distributions. The FLCSP-LC model takes s_c as input, and learns the LC representation $\mathbf{s}_c$ in order to output its latent categorical prediction $\hat{\mathbf{y}}_c$.

The FLCSP-LC first uses a fully connected layer to reduce the input dimensionality from n to $d_1(d_1<n)$:

$$\mathbf{c}\prime_j = F_1(\mathbf{c}_j) = ReLU(\mathbf{W}_1\mathbf{c}_j+\mathbf{b}_1) \tag{3}$$

where $F_1(\cdot)$ denotes the fully connected layer function, $\mathbf{W}_1$ and $\mathbf{b}_1$ are trainable parameters. We use $F_*(\cdot)$ to denote the function of fully connected layers in the rest of the paper. After dimension reduction, we employ an attention network to compute the shallow LC representation for a session by weighting items with different priorities:

$$\mathbf{c} = F_2\left(\left[\sum_{j=1}^{t} q_{tj}\mathbf{c}\prime_j, \mathbf{c}\prime_t\right]\right) \tag{4}$$

$$q_{tj} = \mathbf{v}_1^{\mathrm{T}}\sigma(\mathbf{W}_2\mathbf{c}\prime_t+\mathbf{W}_3\mathbf{c}\prime_j) \tag{5}$$

where $\mathbf{c}\in\mathbb{R}^{d_1}$ is the shallow LC representation for session $s, q_{tj}, j=1,\ldots,t$ is the attentive weights, $\mathbf{v}_1$ and $\mathbf{W}_*$ are the trainable parameters, $\sigma(\cdot)$ is the sigmoid function.

We adopt the MLP network with residual connection at the last layer for deep LC representation abstraction. Finally, we directly compute a LC preference probability distribution $\hat{\mathbf{y}}_c\in\mathbb{R}^m$ for all m candidate items $v_i\in V$ as LC prediction:

$$\mathbf{s}_c = F_5(F_4(F_3(\mathbf{c}))) + \mathbf{c} \tag{6}$$

$$\hat{\mathbf{y}}_c = softmax(\mathbf{W}_4\mathbf{s}_c) \tag{7}$$

where $\mathbf{s}_c\in\mathbb{R}^{d_1}$ is the LC representation for session $s, \mathbf{W}_4\in\mathbb{R}^{m\times d_1}$ is the trainable parameter of output layer for prediction.

3.2. Sequential prediction

We design a FLCSP-S neural model to learn a session's sequential representation $\mathbf{s}_s$ and ouput its sequential prediction $\hat{\mathbf{y}}_s$. The input of the FLCSP-S model is the items' embeddings of a session $s: s_s=\{\mathbf{x}_1,\ldots,\mathbf{x}_t\}$, which will be fine-tuned during the model training. We propose to learn $\mathbf{s}_s$ from both long-term and short-term perspective for each session, which will be learned by different neural networks.

3.2.1. Learning long-term representation

We argue that a long-term representation could reflect the changes of the user's interests over a relatively long time. To this end, we employ a GRU network that takes into consideration of all items before the current timestamp to learn a long-term representation $\mathbf{H}=(\mathbf{h}_1,\ldots,\mathbf{h}_t)$ for a session as follows:

$$\mathbf{h}_t = (1-\mathbf{g}_t)\odot\mathbf{h}_{t-1} + \mathbf{g}_t\odot\widehat{\mathbf{h}}_t \tag{8}$$

$$\mathbf{g}_t = \sigma\left(\mathbf{W}_g^{(1)}\mathbf{x}_t + \mathbf{W}_g^{(2)}\mathbf{h}_{t-1}\right) \tag{9}$$

$$\widehat{\mathbf{h}}_t = \tanh\left(\mathbf{W}_h^{(1)}\mathbf{x}_t + \mathbf{W}_h^{(2)}(\mathbf{r}_t\odot\mathbf{h}_{t-1})\right) \tag{10}$$

$$\mathbf{r}_t = \sigma\left(\mathbf{W}_r^{(1)}\mathbf{x}_t + \mathbf{W}_r^{(2)}\mathbf{h}_{t-1}\right) \tag{11}$$

where $\mathbf{g}_t, \mathbf{r}_t$ are the update gate and reset gate, $\mathbf{W}_*$ the trainable parameter. $\widehat{\mathbf{h}}_t\in\mathbb{R}^{d_2}$ is the candidate hidden state, and $\mathbf{h}_t\in\mathbb{R}^{d_2}$ is the hidden state of the t-th timestamp.

3.2.2. Short-term representation learning

We argue that a short-term representation could be useful to reflect a kind of up-to-date user's interests, which considers only a few of preceding items before the current timestamp. To this end, we adopt a *causal convolutional network* to learn a short-term representation $\mathbf{U} = (\mathbf{u}_1, \ldots, \mathbf{u}_t)$ at each timestamp. As illustrated by Fig. 3, a causal network has two important characteristics: One is that no information "leakage" from the future to the past; The other is that the output layer is with the same length as the input layer. To these aims, we add zero padding of length $k-1$ in front of the sequence, and k is kernel size. We can see that for each timestamp, the output of a causal convolutional network only includes embeddings of k items before the current timestamp. The causal convolutional layer as a function $\Phi(\cdot)$ computes the short-term representation $\mathbf{u}_i \in \mathbb{R}^{d_2}$ at the i-th timestamp by:

$$\mathbf{u}_i \equiv \Phi(\mathbf{x}_i) = \sum_{j=0}^{k-1} \phi(j) \cdot \mathbf{x}_{i-j}, \mathbf{x}_{\leqslant 0} = 0, \tag{12}$$

where $\phi(\cdot)$ is the convolution filter function, k is the kernel size, $\mathbf{x}_i \in \mathbb{R}^{d_2}$ is the item embedding at the i-th timestamp.

3.2.3. Sequential representation for prediction

After learning the long-term representation $\mathbf{H}$ and short-term representation $\mathbf{U}$, we concatenate them at each timestamp as the sequential representation for a session:

$$(\mathbf{a}_1, \ldots, \mathbf{a}_t) = ([\mathbf{h}_1, \mathbf{u}_1], \ldots, [\mathbf{h}_t, \mathbf{u}_t]) \tag{13}$$

where $\mathbf{a}_i \in \mathbb{R}^{2d_2}$ denotes the sequential representation at the i-th timestamp. In order to automatically weigh the importance of representation from different timestamps, another attentive network is deployed to integrate them as the final sequential representation $\mathbf{s}_s \in \mathbb{R}^{4d_2}$ for one session:

$$\mathbf{s}_s = \left[\sum_{j=1}^{t} p_{tj}\mathbf{a}_j, \mathbf{a}_t\right] \tag{14}$$

$$p_{tj} = \mathbf{v}_2^{\mathrm{T}} \sigma(\mathbf{W}_5\mathbf{a}_t + \mathbf{W}_6\mathbf{a}_j) \tag{15}$$

where $p_{tj}, j = 1, \ldots, t$ is the attentive weight and $\sigma(\cdot)$ the sigmoid function. Based on $\mathbf{s}_s$, an inner product similarity is used to compute the sequential preference probabilities as the sequential prediction $\hat{\mathbf{y}}_s \in \mathbb{R}^m$ for all m candidate items $v_i \in V$ via a softmax function:

$$\hat{\mathbf{y}}_s = softmax(\mathbf{X}\mathbf{W}_7\mathbf{s}_s) \tag{16}$$

where $\mathbf{X} \in \mathbb{R}^{m \times d_2}$ is the item embedding matrix, $\mathbf{W}_3 \in \mathbb{R}^{d_2 \times 4d_2}$ the trainable parameters used to convert dimension.

3.3. Fusion mechanism

So far, we have obtained two predictions $\hat{\mathbf{y}}_c$ and $\hat{\mathbf{y}}_s$ from the session's LC representation and sequential representation, respectively. Per our aforementioned discussion, we regard the LC prediction as a complementary for final recommendation and still prioritize sequential information in the next item prediction. So we take the cosine similarity between the two predictions to weight the LC prediction. We make a *decision fusion* based on $\hat{\mathbf{y}}_c$ and $\hat{\mathbf{y}}_s$ as follows:

$$\hat{\mathbf{y}} = \hat{\mathbf{y}}_s + \frac{< \hat{\mathbf{y}}_s, \hat{\mathbf{y}}_c >}{|\hat{\mathbf{y}}_s||\hat{\mathbf{y}}_c|} \cdot \hat{\mathbf{y}}_c \tag{17}$$

where $\hat{\mathbf{y}} \in \mathbb{R}^m$ denotes the final preference scores for all m candidate items, and the top-K items with the highest scores are selected to form a recommendation list.

To train the model, we use the cross-entropy objective functions for the two predictions $\hat{\mathbf{y}}_s$ and $\hat{\mathbf{y}}_c$ respectively as follows:

$$\mathscr{L}_c(\hat{\mathbf{y}}_c) = -\sum_{i=1}^{m} y_i \log\left(\hat{y}_i^{(c)}\right) \tag{18}$$

$$\mathscr{L}_s(\hat{\mathbf{y}}_s) = -\sum_{i=1}^{m} y_i \log\left(\hat{y}_i^{(s)}\right) \tag{19}$$

where $\mathbf{y}$ is the ground truth, that is, the real item clicked by user, and y_i is the i-th value in $\mathbf{y}$. Similarly, $\hat{y}_i^{(c)}$ and $\hat{y}_i^{(s)}$ are the i-th value respectively in $\hat{\mathbf{y}}_c$ and $\hat{\mathbf{y}}_s$.

In our model training, we train the two modules independently but synchronously by using the same training data in each epoch.

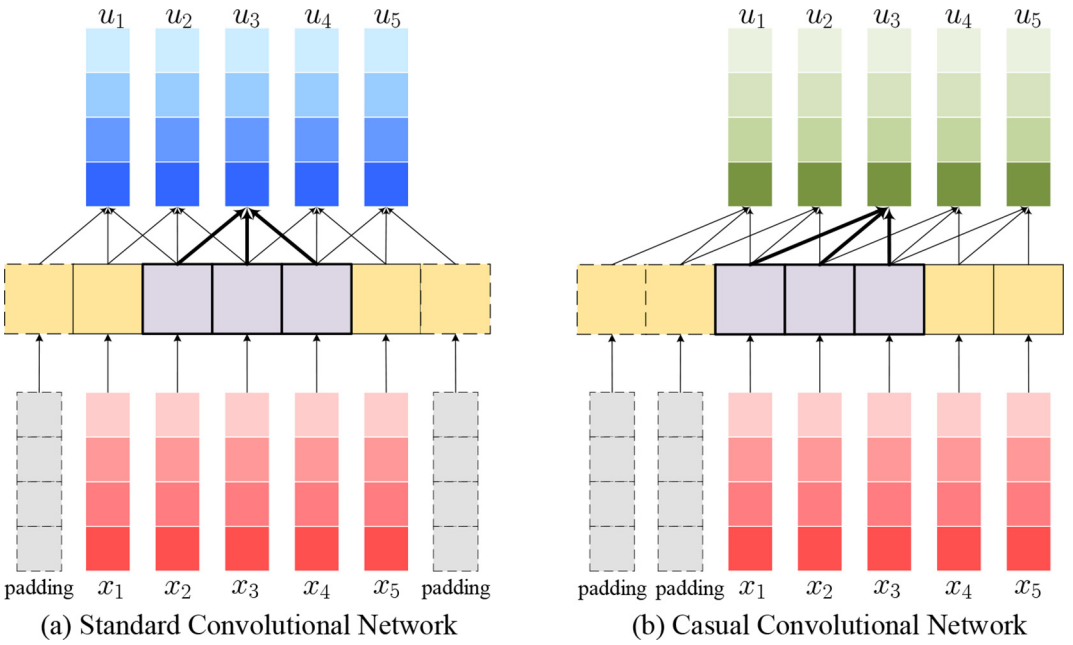

Fig. 3. Comparison between (a) a standard convolutionary network and (b) a causal convolutional network. The convolutionary operation in (a) for u_3 needs to look from a future item x_4; The convolutionary operation in (b) does not look into any future item, but is restricted to preceding (old) items.

4. Experiments and analysis

4.1. Experiment settings

4.1.1. Datsets

We conduct experiments on three real-world datasets: Yoochoose,[1] Gowalla,[2] and Last.fm,[3] which have been commonly used in the SBR task [7–9,21,36]. The statistics of the three datasets are presented in Table 1.

- Yoochoose: It is from Recsys Challenge 2015, which consists of six months of clicks from an e-commerce website in Europe. Following [7–9,36], the sessions of the last day are used for testing, and the recent fractions 1/64 of sessions are used for training.
- Gowalla: It is a check-in dataset widely used for the SBR task. Following [42–44], the users' check-in records are grouped into disjoint sessions by splitting at intervals between adjacent records that are longer than one day. The last 20% of the sessions are used as the testing set.
- Last.fm: It is widely used for music artist recommendation. Following [42,21,44], we set the splitting interval to eight hours. The most recent 20% of sessions are used as the testing set.

4.1.2. Competitors

We consider the following competitors for performance comparison:

- **POP** recommends the top-*K* popular items in the training set.
- **Item-KNN** [13] recommends items with higher cosine similarities in previous sessions.
- **FPMC** [15] is a next-basket recommendation method based on Markov chain.
- **NextItNet** [45] designs a CNN neural model for the next item recommendation.
- **NARM** [7] uses a GRU network with an attention mechanism to capture sequential behavior of a session.
- **STAMP** [8] employs a MLP network with attention to capture users' general interests of current session and short-term interests of the last clicked item.
- **SR-GNN** [9] models sessions as a graph and uses a GNN model to learn items' embeddings, based on which an attentive network is adopted to learn sessions' representations.

[1] https://2015.recsyschallenge.com/challenge.html
[2] https://snap.stanford.edu/data/loc-gowalla.html
[3] http://ocelma.net/MusicRecommendationDataset/lastfm-1K.html

Table 1
Statistics of the datasets.

Datasets	Yoo 1/64	Gowalla	Last.fm
# clicks	557,248	1,122,788	3,835,706
# train sessions	369,859	675,561	2,837,644
# test sessions	55,898	155,332	672,519
# all items	17,745	29,510	38,615

4.1.3. Performance metrics

The following widely used performance metrics are also used in our experiments:

HR@20 (Hit Rate) is used to evaluate whether a target item is in the top-K recommended list:

$$HR@K = \frac{n_{hit}}{N} \tag{20}$$

where N is the number of testing sessions and n_{hit} the number of sessions that have hit the target items in a recommended list.

MRR@20 (Mean Reciprocal Rank) is the average of reciprocal ranks of the correct recommended items, and the reciprocal rank is set zero if the rank is larger than K:

$$MRR@K = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{rank_i} \tag{21}$$

where $rank_i$ is the rank of a target item in the i-th session.

4.1.4. Parameter setting

We set the dimensionality $d_1 = 200$ and $d_2 = 50$ respectively. We set the hyper-parameter $k = 100$ and $\Delta_{th} = 2 \times 10^{-6}$ for anchor selection. The restart probability α is set to 0.5 to balance the depth and breadth of the random walk. The kernel size is set to 3 for causal convolutions. Besides, the initial learning rate is set as $l_r = 0.01$ with 0.1 decay rate for Yoochoose 1/64, Gowalla at epoch [10,15], for Last.fm at epoch [5,10]. Sequential model FLCSP-S and latent categorical model FLCSP-LC both use Adam optimizer for training with $\beta_1, \beta_2 = [0.9, 0.999]$ and the L2 penalty is 10^{-5}. Meanwhile, the batch size is set to 512 for Yoochoose 1/64 and Last.fm, 100 for Gowalla. We use the Pytorch[4] as experiment framework.

Anchor Selection: In our FLCSP-LC, based on the proposed anchor selection strategy as shown in Fig. 5, the numbers of selected anchors are $\{1538, 1038, 1801\} \approx \{1500, 1000, 1800\}$ respectively for the Yoochoose 1/64, Gowalla and Last.fm datasets. We note such approximations are to simplify the computations in our model, which do not impact much on the final recommendation performance. We take Yoochoose 1/64 as an example and vary n in the range of [1000, 2000] with a step of 100-anchor. Note that only FLCSP and FLCSP-LC could be impacted by the anchor numbers. Fig. 4 presents their HR@20 and MRR@20 against the number of anchors. It can be observed that they do not change much for the given anchor numbers. So we will choose these anchor numbers in our comparison experiments.

4.2. Experiment results

4.2.1. Overall performance

Table 2 compares our FLCSP with the competitors, where ours outperforms the others in terms of higher HR@20 and MRR@20 in all the three datasets.

The conventional methods, i.e. POP, Item-KNN and FPMC, perform not better than the deep learning-based methods. POP uses a simple popularity rule without differentiating sessions that often consist of different items. Item-KNN only calculates similarity between items and ignores the sessions' sequential information. Although FPMC takes sequential relation in sessions into account based on a Markov chain model, its strict assumption of items' independence is not appropriate for the SBR task.

The deep learning-based models achieve significant performance improvement over the conventional methods, validating the strong capability of their representation learning techniques for the SBR task. It is observed that NextItNet as a CNN-based model perform worse than the RNN-based NARM, the attention network-based STAMP and the GNN-based SR-GNN model. This might be due to that the CNN models are not good at capturing the sequential long-term representation of a session. On the other hand, both NARM and STAMP, as the state-of-the-art since 2017 and 2018, achieve high HR@20 and MRR@20 performance, because they can well handle the long-term sequential dependencies and discriminate items' importance in a session via attention mechanism. The mostly recent state-of-the-art method SR-GNN in 2019 plays the second best performance, which might be attributed to its use of a powerful GNN on an item graph to capture diverse item-to-item transition characteristics from sessions.

[4] https://pytorch.org/

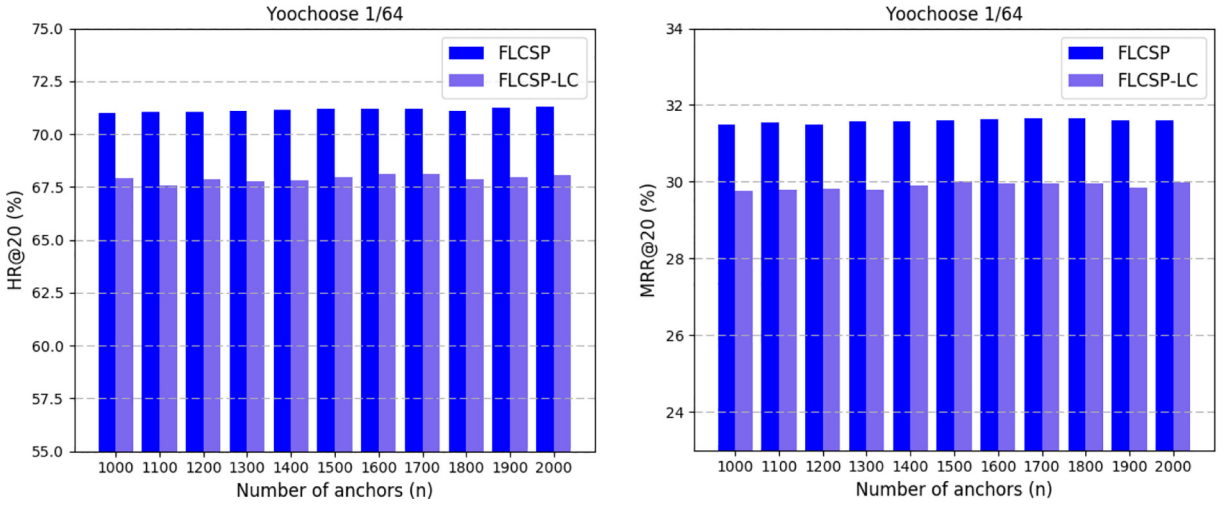

Fig. 4. The results for influence of anchors' number n in Yoochoose 1/64.

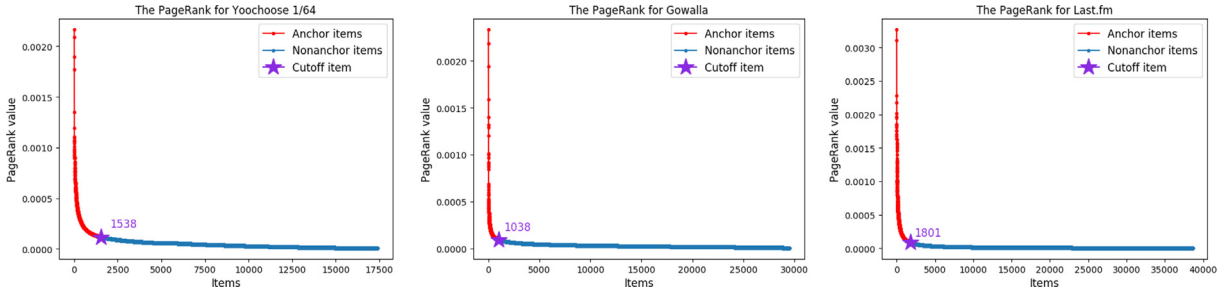

Fig. 5. The PageRank values of the three datasets. For each dataset, we sort items' PageRank values into a decreasing list. We compute a representative difference to find the cutoff item (the purple star). The items before the cutoff item are selected as anchors (the red part), and the others are non-anchor items (the blue part).

Table 2
Overall performance comparison.

Method	Yoochoose 1/64		Gowalla		Last.fm	
	HR@20 (%)	MRR@20 (%)	HR@20 (%)	MRR@20 (%)	HR@20 (%)	MRR@20 (%)
POP	6.71	1.65	4.56	0.79	4.98	1.27
Item-KNN	51.60	21.81	38.60	16.66	14.90	4.04
FPMC	45.62	15.01	29.91	11.45	12.86	3.78
NextItNet	–	–	45.15	21.26	20.12	7.18
NARM	68.32	28.63	50.07	23.92	21.83	7.59
STAMP	68.74	29.67	50.21	25.54	22.17	7.35
SR-GNN	70.57	30.94	50.32	24.25	22.33	8.23
FLCSP	**71.19**	**31.61**	**52.54**	**25.71**	**23.92**	**8.27**
Improv.	0.88%	2.17%	4.41%	0.67%	7.12%	0.49%

We have run 3 times in each dataset, and report the mean performances metrics, and the variances of different runs are consistently smaller than 0.01% in our model.
In each column of performance metric, the bolded item indicates the best performance among all competetors; while the underlined item the second best.

Our proposed model FLCSP outperforms all the competitors, suggesting that besides extracting sequential representations, mining items' LC distributions and learning sessions' LC representations can also assist in accomplishing the next item prediction from another viewpoint of items' categorical abstractions. This rationale can also be collaborated by our ablation studies in the next subsection.

4.2.2. Ablation studies

Table 3 presents the ablation results. It is observed that FLCSP-S performs better than FLCSP-LC. This is not unexpected as we argue that the most important characteristics in the SBR task is the sequential information embedded within a session; While our design philosophy is to mine and exploit latent categorical information to boost the next item prediction. On the one hand, it is good to observe that the FLCSP-LC itself can also achieve performance close to the state-of-the-art NARM, STAMP and SR-GNN models. On the other hand, the FLCSP-S itself achieves better performance than these models. This suggests that our sequential representation learned from the two neural networks is more effective for the next item prediction. Finally, the ablation results also validate the effectiveness of our prediction fusion, as the FLCSP achieves the best performance by fusing the two predictions.

Table 3 presents the ablation experiments on using an average operation in the FLCSP-LC, denoted as FLCSP-LC-Avg. It can be observed that its performance is not better than that of the FLCSP-LC using our attentive mechanism. This is not unexpected. The attentive network in the FLCSP-LP module can further differentiate the importance of each item's LC representation in a session according to its back-propagated prediction error in the training phase; While the average operation simply treats all items equally.

Table 3 compares the ablation study for the two components in the FLCSP-S module. The FLCSP-S-Short only uses the causal convolutional network with attention to learn short-term representation; While the FLCSP-S-Long only uses the GRU with attention to learn long-term representation. It is glad to observe their performances are close to those of the NARM and STAMP. However, both can not outperform their combined version FLCSP-S. This again validates the necessaries for learning and fusing both short-term and long-term representation for a session's sequential representation.

4.2.3. Influence of Top-K and decreasing training data

In general, users always focus on the top items in a recommendation list, however, the shorter length of recommendation list, the more difficult to hit the target item, that is, the worse recommendation performance. Besides, since the deep learning-based models are data hungry, getting worse performance when decreasing the training data could be expected. To investigate these influences, we conduct the following experiments. We set the list length $K = \{5, 10, 20\}$ in the SBR task. Furthermore, we decrease the original training data of Yoochoose by taking smaller and smaller subsets of latest sessions. The statistics of the smaller training datasets are provided in Table 4.

Fig. 6 plots HR@{5, 10, 20} and MRR@{5, 10, 20} against the size of training data and Table 5 presents all the results. It can be observed that our model achieves almost all the best results in all cases, except only the one achieving the second best in terms of HR@20 in the Yoochoose 1/128 dataset. Obviously, all the four neural models suffer from fewer training data. This is not much surprised, as neural models are with a great number of trainable parameters that generally can be better trained with more training data. When having a close comparison on the performance degradation, it can be observed that our FLCSP model can achieve greater performance improvements in those smaller datasets. In particular, the improvements of HR@20 and MRR@20 over the second best model in the Yoochoose 1/64 dataset is about 0.88% and 2.17%, respectively; While they are 4.76% and 3.45% in the Yoochoose 1/512 dataset. This can also be attributed to our FLCSP-LC model, as it complements the final prediction from another viewpoint of items' categorical abstractions.

Table 3
The performances of ablation experiments.

Method	Yoochoose 1/64		Gowalla		Last.fm	
	HR@20	MRR@20	HR@20	MRR@20	HR@20	MRR@20
FLCSP-LC-Avg	61.12	24.09	36.45	14.32	20.36	6.55
FLCSP-LC	67.96	30.02	42.05	17.73	20.18	6.70
FLCSP-S-Short	68.43	28.81	49.65	23.38	21.69	7.02
FLCSP-S-Long	68.32	28.63	50.07	23.92	21.83	7.59
FLCSP-S	71.01	31.11	51.97	25.63	23.41	8.04
FLCSP	**71.19**	**31.61**	**52.54**	**25.71**	**23.92**	**8.27**

Table 4
The number of training sessions in decreased Yoochoose datasets. The number of test sessions are correspondingly 55,898 for every datasets.

Datasets	# train sessions
Yoochoose 1/64	369,859
Yoochoose 1/128	184,929
Yoochoose 1/256	92,464
Yoochoose 1/215	46,232
Yoochoose 1/1024	23,116

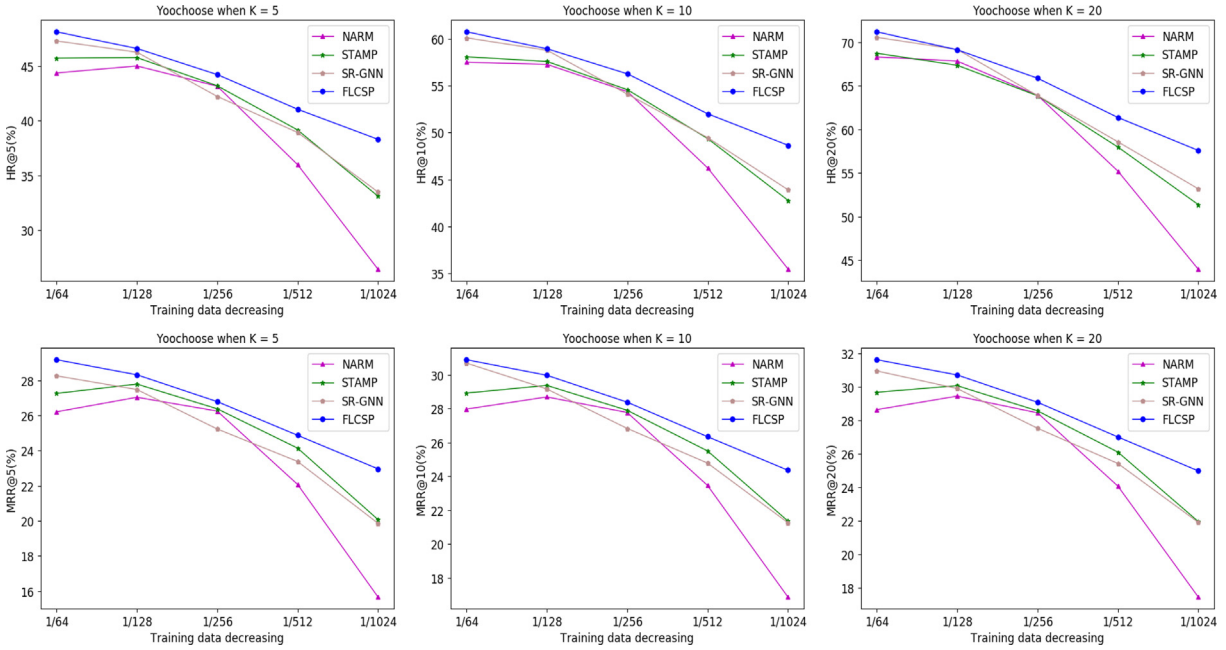

Fig. 6. Performance comparison for decreasing data when K = {5, 10, 20}.

Table 5
The performances of analysis when K = {5, 10, 20} with decreasing training data {Yoochoose 1/64, Yoochoose 1/128, Yoochoose 1/256, Yoochoose 1/512, Yoochoose 1/1024}.

Method	Yoochoose 1/64					
	HR@5	MRR@5	HR@10	MRR@10	HR@20	MRR@20
NARM	44.34	26.21	57.50	27.97	68.32	28.63
STAMP	45.69	27.26	58.07	28.92	68.74	29.67
SR-GNN	47.26	28.26	60.09	30.69	70.57	30.94
FLCSP	**48.07**	**29.14**	**60.75**	**30.85**	**71.19**	**31.61**
Method	Yoochoose 1/128					
	HR@5	MRR@5	HR@10	MRR@10	HR@20	MRR@20
NARM	44.98	27.04	57.26	28.69	67.85	29.44
STAMP	45.74	27.79	57.57	29.37	67.37	30.06
SR-GNN	46.23	27.47	58.76	29.16	**69.18**	29.89
FLCSP	**46.57**	**28.32**	**58.92**	**29.97**	69.14	**30.70**
Method	Yoochoose 1/256					
	HR@5	MRR@5	HR@10	MRR@10	HR@20	MRR@20
NARM	43.13	26.25	54.36	27.77	63.92	28.44
STAMP	43.17	26.38	54.56	27.90	63.87	28.56
SR-GNN	42.20	25.23	54.09	26.83	63.89	27.52
FLCSP	**44.19**	**26.79**	**56.25**	**28.38**	**65.88**	**29.07**
Method	Yoochoose 1/512					
	HR@5	MRR@5	HR@10	MRR@10	HR@20	MRR@20
NARM	35.98	22.07	46.21	23.46	55.23	24.07
STAMP	39.14	24.14	49.34	25.50	57.99	26.10
SR-GNN	38.91	23.38	49.44	24.78	58.59	25.42
FLCSP	**41.03**	**24.87**	**51.99**	**26.34**	**61.38**	**27.00**
Method	Yoochoose 1/1024					
	HR@5	MRR@5	HR@10	MRR@10	HR@20	MRR@20
NARM	26.45	15.66	35.44	16.87	43.98	17.46
STAMP	33.11	20.07	42.76	21.36	51.39	21.96
SR-GNN	33.49	19.86	43.89	21.26	53.20	21.91
FLCSP	**38.29**	**22.96**	**48.64**	**24.36**	**57.60**	**24.98**

5. Conclusion

In this paper, we have argued that a kind of latent categorical information could be mined and exploited for session-based recommendation. Following our arguments, we have proposed the FLCSP neural model which not only learns sequential representation but also latent categorical representation for a session. For learning the sequential representation, we have designed a neural model consisting a recurrent network and a casual convolutional network with an attentive mechanism. For learning the latent categorical representation, we have mined items' latent categorical distributions based on graph random walk as input to our attentive network. Furthermore, we have proposed to fuse the two kinds of predictions based on the two representations as the final next item prediction. Experiments on three public datasets have validated that our FLCSP model outperforms the recent state-of-the-art models.

In this paper, our argument about items' latent category actually assigns additional semantic abstractions to items to further enrich their descriptions. We believe that some latent semantic abstractions could also exist in the transition from one item to another, like from a shirt item to a jacket item indicating from a clothes category also to a clothes category. While such clothes categorical transition type could be used to differentiate other types of transition, like from a clothes category to a cosmetics category. In our future work, we will further investigate the existence of such potentials for the SBR task.

CRediT authorship contribution statement

Both authors have contributed equally to the problem definition, algorithm design, as well as experiment result analysis. Mr. Zhang conducted the experiments and drafted the manuscript; Dr. Wang rewrote the manuscript.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This work is supported in part by National Natural Science Foundation of China (Grant No: 61771209).

References

[1] C. Sun, H. Liu, M. Liu, Z. Ren, T. Gan, L. Nie, Lara: Attribute-to-feature adversarial learning for new-item recommendation, in: Proceedings of the 13th International Conference on Web Search and Data Mining, 2020, pp. 582–590.
[2] Y. Li, M. Liu, J. Yin, C. Cui, X.-S. Xu, L. Nie, Routing micro-videos via a temporal graph-guided recommendation system, in: Proceedings of the 27th ACM International Conference on Multimedia, 2019, pp. 1464–1472.
[3] X. Song, F. Feng, J. Liu, Z. Li, L. Nie, J. Ma, Neurostylist: Neural compatibility modeling for clothing matching, in: Proceedings of the 25th ACM International Conference on Multimedia, 2017, pp. 753–761.
[4] S. Liu, B. Wang, M. Xu, Event recommendation based on graph random walking and history preference reranking, in: Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2017, pp. 861–864.
[5] S. Liu, B. Wang, M. Xu, L.T. Yang, Evolving graph construction for successive recommendation in event-based social networks, Future Generation Computer Systems 96 (2019) 502–514.
[6] S. Wang, L. Cao, Y. Wang, A survey on session-based recommender systems, arXiv preprint arXiv:1902.04864 (2019)..
[7] J. Li, P. Ren, Z. Chen, Z. Ren, T. Lian, J. Ma, Neural attentive session-based recommendation, in: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, 2017, pp. 1419–1428.
[8] Q. Liu, Y. Zeng, R. Mokhosi, H. Zhang, 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, 2018, pp. 1831–1839.
[9] S. Wu, Y. Tang, Y. Zhu, L. Wang, X. Xie, T. Tan, Session-based recommendation with graph neural networks, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, 2019, pp. 346–353..
[10] Z. Zhang, B. Wang, Learning sequential and general interests via a joint neural model for session-based recommendation, Neurocomputing 415 (2020) 165–173.
[11] S. Zhang, L. Yao, A. Sun, Y. Tay, Deep learning based recommender system: A survey and new perspectives, ACM Computing Surveys (CSUR) 52 (1) (2019) 1–38.
[12] G. Linden, B. Smith, J. York, Amazon. com recommendations: Item-to-item collaborative filtering, IEEE Internet Computing 7 (1) (2003) 76–80.
[13] B. Sarwar, G. Karypis, J. Konstan, J. Riedl, Item-based collaborative filtering recommendation algorithms, in: Proceedings of the 10th International Conference on World Wide Web, 2001, pp. 285–295.
[14] G. Shani, D. Heckerman, R.I. Brafman, An mdp-based recommender system, Journal of Machine Learning Research 6 (Sep) (2005) 1265–1295.
[15] S. Rendle, C. Freudenthaler, L. Schmidt-Thieme, Factorizing personalized markov chains for next-basket recommendation, in: Proceedings of the 19th International Conference on World Wide Web, 2010, pp. 811–820.
[16] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (7553) (2015) 436–444.
[17] B. Hidasi, A. Karatzoglou, L. Baltrunas, D. Tikk, Session-based recommendations with recurrent neural networks, in: Proceedings of the 4th International Conference on Learning Representations, ICLR '16, 2016..
[18] Y.K. Tan, X. Xu, Y. Liu, Improved recurrent neural networks for session-based recommendations, in: Proceedings of the 1st Workshop on Deep Learning for Recommender Systems, 2016, pp. 17–22.
[19] M. Quadrana, A. Karatzoglou, B. Hidasi, P. Cremonesi, Personalizing session-based recommendations with hierarchical recurrent neural networks, in: Proceedings of the Eleventh ACM Conference on Recommender Systems, 2017, pp. 130–137.
[20] M. Wang, P. Ren, L. Mei, Z. Chen, J. Ma, M. de Rijke, 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, 2019, pp. 345–354.

[21] P. Ren, Z. Chen, J. Li, Z. Ren, J. Ma, M. de Rijke, Repeatnet: A repeat aware neural recommendation machine for session-based recommendation, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, 2019, pp. 4806–4813..
[22] C. Xu, P. Zhao, Y. Liu, V.S. Sheng, J. Xu, F. Zhuang, J. Fang, X. Zhou, Graph contextualized self-attention network for session-based recommendation, IJCAI (2019) 3940–3946.
[23] S. Rendle, C. Freudenthaler, Z. Gantner, L. Schmidt-Thieme, Bpr: Bayesian personalized ranking from implicit feedback, in: Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence, UAI'09, AUAI Press, 2009, pp. 452–461..
[24] Y. Koren, R. Bell, C. Volinsky, Matrix factorization techniques for recommender systems, Computer 42 (8) (2009) 30–37.
[25] D. Bokde, S. Girase, D. Mukhopadhyay, Matrix factorization model in collaborative filtering algorithms: A survey, Procedia Computer Science 49 (2015) 136–146.
[26] A. Mnih, R.R. Salakhutdinov, Probabilistic matrix factorization, in: Advances in Neural Information Processing Systems, 2008, pp. 1257–1264..
[27] Y. Koren, R. Bell, Advances in collaborative filtering, in: Recommender Systems Handbook, Springer, 2015, pp. 77–118..
[28] X. Yang, B. Wang, Local matrix approximation based on graph random walk, in: Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, 2019, pp. 1037–1040.
[29] X. Yang, B. Wang, Local ranking and global fusion for personalized recommendation, Applied Soft Computing 96 (2020) 106636.
[30] X. Yang, B. Wang, Local matrix approximation via automatic anchor selection and asymmetric neighbor inclusion based on feature divergence measure, Neurocomputing 383 (2020) 368–379.
[31] X. Yang, B. Wang, Destructure-and-restructure matrix approximation, Information Sciences 514 (2020) 434–448.
[32] A. Zimdars, D.M. Chickering, C. Meek, Using temporal data for making recommendations, in: Proceedings of the Seventeenth conference on Uncertainty in artificial intelligence, UAI'01, Morgan Kaufmann Publishers Inc., 2001, pp. 580–588..
[33] Z. Pan, F. Cai, Y. Ling, M. de Rijke, An intent-guided collaborative machine for session-based recommendation, in: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, 2020, pp. 1833–1836.
[34] J. Song, H. Shen, Z. Ou, J. Zhang, T. Xiao, S. Liang, Islf: Interest shift and latent factors combination model for session-based recommendation, IJCAI (2019) 5765–5771.
[35] J. Chung, C. Gulcehre, K. Cho, Y. Bengio, Empirical evaluation of gated recurrent neural networks on sequence modeling, arXiv preprint arXiv:1412.3555 (2014)..
[36] R. Qiu, J. Li, Z. Huang, H. Yin, Rethinking the item order in session-based recommendation with graph neural networks, in: Proceedings of the 28th ACM International Conference on Information and Knowledge Management, 2019, pp. 579–588.
[37] A. Luo, P. Zhao, Y. Liu, F. Zhuang, D. Wang, J. Xu, J. Fang, V.S. Sheng, Collaborative self-attention network for session-based recommendation, in: Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence (IJCAI), 2020, pp. 2591–2597, main track.
[38] Z. Pan, F. Cai, W. Chen, H. Chen, M. de Rijke, Star graph neural networks for session-based recommendation, in: Proceedings of the 29th ACM International Conference on Information & Knowledge Management, 2020, pp. 1195–1204.
[39] Z. Wang, W. Wei, G. Cong, X.-L. Li, X.-L. Mao, M. Qiu, Global context enhanced graph neural networks for session-based recommendation, in: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, 2020, pp. 169–178.
[40] S. Bai, J.Z. Kolter, V. Koltun, An empirical evaluation of generic convolutional and recurrent networks for sequence modeling, arXiv preprint arXiv:1803.01271 (2018)..
[41] L. Page, S. Brin, R. Motwani, T. Winograd, The pagerank citation ranking: Bringing order to the web, Tech. rep, Stanford InfoLab, 1999.
[42] L. Guo, H. Yin, Q. Wang, T. Chen, A. Zhou, N. Quoc Viet Hung, Streaming session-based recommendation, in: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2019, pp. 1569–1577.
[43] J. Tang, K. Wang, Personalized top-n sequential recommendation via convolutional sequence embedding, in: Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining, 2018, pp. 565–573.
[44] T. Chen, R.C.-W. Wong, Handling information loss of graph neural networks for session-based recommendation, in: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2020, pp. 1172–1180.
[45] F. Yuan, A. Karatzoglou, I. Arapakis, J.M. Jose, X. He, A simple convolutional generative network for next item recommendation, in: Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining, 2019, pp. 582–590.