RepeatNet: A Repeat Aware Neural Recommendation Machine for Session-Based Recommendation

Pengjie Ren,[1,2] Zhumin Chen,[1] Jing Li,[1] Zhaochun Ren,[3] Jun Ma,[1] Maarten de Rijke[2]
[1]Shandong University, Jinan, China
[2]University of Amsterdam, Amsterdam, The Netherlands
[3]Data Science Lab, JD.com, Beijing, China

Abstract

Recurrent neural networks for session-based recommendation have attracted a lot of attention recently because of their promising performance. *repeat consumption* is a common phenomenon in many recommendation scenarios (e.g., e-commerce, music, and TV program recommendations), where the same item is re-consumed repeatedly over time. However, no previous studies have emphasized *repeat consumption* with neural networks. An effective neural approach is needed to decide when to perform repeat recommendation. In this paper, we incorporate a repeat-explore mechanism into neural networks and propose a new model, called RepeatNet, with an encoder-decoder structure. RepeatNet integrates a regular neural recommendation approach in the decoder with a new repeat recommendation mechanism that can choose items from a user's history and recommends them at the right time. We report on extensive experiments on three benchmark datasets. RepeatNet outperforms state-of-the-art baselines on all three datasets in terms of MRR and Recall. Furthermore, as the dataset size and the repeat ratio increase, the improvements of RepeatNet over the baselines also increase, which demonstrates its advantage in handling repeat recommendation scenarios.

Introduction

Session-based recommendations have received increasing interest recently, due to their broad applicability in many online services (e.g., e-commerce, video watching, music listening) (Cheng et al. 2017). Here, a session is a group of interactions that take place within a given time frame. Sessions from a user can occur on the same day, or over several days, weeks, or even months (Quadrana et al. 2017).

Conventional recommendation methods tackle session-based recommendations based on either the last interaction or the last session. Zimdars, Chickering, and Meek (2001) and Shani, Heckerman, and Brafman (2005) investigate how to extract sequential patterns to predict the next item using Markov models. Then, Chen et al. (2012) propose logistic Markov embeddings to learn the representations of songs for playlist prediction. A major issue for these models is that the state space quickly becomes unmanageable when trying to include all possible sequences of potential user selections over all items. Recurrent neural networks (RNNs) have recently been used for the purpose of session-based recommendations and attracted significant attention. Hidasi et al. (2016a) introduce RNNs with gated recurrent units (GRUs) for session-based recommendation. They introduce a number of parallel RNN (p-RNN) architectures to model sessions based on both clicks and features (images and text) of clicked items (Hidasi et al. 2016b). Quadrana et al. (2017) personalize RNN models with cross-session information transfer and devise a Hierarchical RNN model that relays and evolves latent hidden states of the RNNs across user sessions. Li et al. (2017b) introduce an attention mechanism into session-based recommendations and outperform (Hidasi et al. 2016a). Though the number of studies that apply deep learning to session-based recommendation is increasing, none has emphasized so-called *repeat consumptions*, which are a common phenomenon in many recommendation scenarios (e.g., e-commerce, music, and TV program recommendations), where the same item is re-consumed repeatedly over time.

Repeat consumption exists because people have regular habits. For example, we all buy the same things repeatedly, we eat at the same restaurants regularly, we listen to the same songs and artists frequently (Anderson et al. 2014). Table 1 shows the repeat consumption ratio for three benchmark datasets that are commonly used in related studies (Hidasi et al. 2016a; Li et al. 2017b). Repeat consumption not only exists but also accounts for a large proportion of the interactions in some applications. In this paper, we investigate *repeat consumption* by incorporating a repeat-explore mechanism into neural networks and propose a new model called RepeatNet with an encoder-decoder structure. Unlike existing work that evaluates a score for each item using a single decoder, RepeatNet evaluates the recommendation probabilities of each item with two decoders in a *repeat mode* and an

Table 1: Repeat ratio (%) on three benchmark datasets.

Datasets	**Train**	**Validation**	**Test**
YOOCHOOSE 1/4	25.52	25.51	26.02
DIGINETICA	19.94	20.06	20.49
LASTFM	20.72	20.42	20.95

explore mode, respectively. In the *repeat mode* we recommend an old item from the user's history while in the *explore mode* we recommend a new item. Specifically, we first encode each session into a representation. Then, we use a repeat-explore mechanism to learn the switch probabilities between repeat and explore modes. After that, we propose a *repeat recommendation decoder* to learn the probabilities of recommending old items in the *repeat mode* and an *explore recommendation decoder* to learn the probabilities of recommending new items under the *explore mode*. Finally, we determine the recommendation score for an item by combining the mode switch probabilities and the recommendation probabilities of each item under the two modes in a probabilistic way. The mode prediction and item recommendation are jointly learned in an end-to-end back-propagation training paradigm within a unified framework.

We carry out extensive experiments on three benchmark datasets. The results show that RepeatNet outperforms state-of-the-art baselines on all three datasets in terms of MRR and Recall. Furthermore, we find that as the dataset size and the repeat ratio increase, the improvements of RepeatNet over the baselines also increase, which demonstrates its advantages in handling repeat recommendation scenarios.

To sum up, the main contributions in this paper are:

- We propose a novel deep learning-based model named RepeatNet that takes into account the *repeat consumption* phenomenon. To the best of our knowledge, we are the first to consider this in the context of session-based recommendation with a neural model.
- We introduce a repeat-explore mechanism for session-based recommendation to automatically learn the switch probabilities between repeat and explore modes. Unlike existing works that use a single decoder, we propose two decoders to learn the recommendation probability for each item in the two modes.
- We carry out extensive experiments and analyses on three benchmark datasets. The results show that RepeatNet can improve the performance of session-based recommendation over state-of-the-art methods by explicitly modeling *repeat consumption*.

Related Work

We survey related work in two areas: session-based recommendations and repeat recommendations.

Session-based recommendation

Conventional methods for session-based recommendation are usually based on Markov chains that predict the next action given the last action. Zimdars, Chickering, and Meek (2001) propose a sequential recommender based on Markov chains and investigate how to extract sequential patterns to learn the next state using probabilistic decision-tree models. Mobasher et al. (2002) study different sequential patterns for recommendation and find that contiguous sequential patterns are more suitable for sequential prediction task than general sequential patterns. Shani, Heckerman, and Brafman (2005) present a Markov decision process (MDP) to provide recommendations in a session-based manner and the simplest MDP boils down to first-order Markov chains where the next recommendation can simply be computed through the transition probabilities between items. Yap, Li, and Yu (2012) introduce a competence score measure in personalized sequential pattern mining for next-item recommendations. Chen et al. (2012) model playlists as Markov chains, and propose logistic Markov embeddings to learn the representations of songs for playlists prediction. A major issue with applying Markov chains to the session-based recommendation task is that the state space quickly becomes unmanageable when trying to include all possible sequences of potential user selections over all items.

RNNs have proved useful for sequential click prediction (Zhang et al. 2014). Hidasi et al. (2016a) apply RNNs to session-based recommendation and achieve significant improvements over conventional methods. They utilize session-parallel mini-batch training and employ ranking-based loss functions for learning the model. Later, they introduce a number of parallel RNN (p-RNN) architectures to model sessions based on clicks and features (images and text) of clicked items (Hidasi et al. 2016b); they propose alternative training strategies for p-RNNs that suit them better than standard training. Tan, Xu, and Liu (2016) propose two techniques to improve the performance of their models, namely data augmentation and a method to account for shifts in the input data distribution. Jannach and Ludewig (2017) show that a heuristics-based nearest neighbor scheme for sessions outperforms the model proposed by Hidasi et al. (2016a) in the large majority of the tested configurations and datasets. Quadrana et al. (2017) propose a way to personalize RNN models with cross-session information transfer and devise a Hierarchical RNN model that relays end evolves latent hidden states of the RNNs across user sessions. Li et al. (2017b) explore a hybrid encoder with an attention mechanism to model the user's sequential behavior and intent to capture the user's main purpose in the current session.

Unlike the studies listed above, we emphasize the *repeat consumption* phenomenon in our models.

Repeat recommendation

Anderson et al. (2014) study the patterns by which a user consumes the same item repeatedly over time, in a wide variety of domains, ranging from check-ins at the same business location to re-watches of the same video. They find that recency of consumption is the strongest predictor of repeat consumption. Chen, Wang, and Wang (2015) derive four generic features that influence people's short-term *repeat consumption* behavior. Then, they present two fast algorithms with linear and quadratic kernels to predict whether a user will perform a short-term *repeat consumption* at a specific time given the context.

An important goal of a recommender system is to help users discover new items. Besides that, many real-world systems utilize lists of recommendation for a different goal, namely to remind users of items that they have viewed or consumed in the past. Lerche, Jannach, and Ludewig (2016) investigate this through a live experiment, aiming to quantify the value of such reminders in recommendation lists.

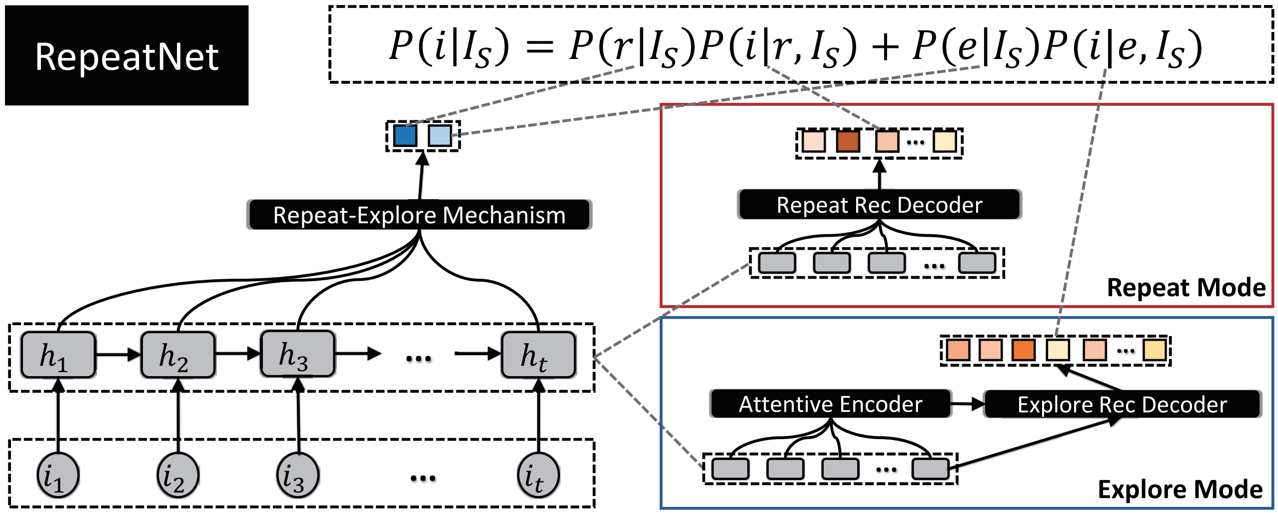

Figure 1: Overview of RepeatNet.

Benson, Kumar, and Tomkins (2016) identify two macroscopic behavior patterns of repeated consumptions. First, in a given user's lifetime, very few items live for a long time. Second, the last consumptions of an item exhibit growing inter-arrival gaps consistent with the notion of increasing boredom leading up to eventual abandonment. The main difference between our work and previous work on repeat recommendations is that we are the first to propose a neural recommendation model to explicitly emphasize *repeat consumption* in both conventional and session-based recommendation tasks.

RepeatNet

Given an action (e.g., clicking, shopping) session $I_S = \{i_1, i_2, \ldots, i_\tau, \ldots, i_t\}$, where i_τ refers to an item, session-based recommendation tries to predict what the next event would be, as shown in Eq. 1. Without loss of generality, we take *click actions* as our running example in the paper:

$$P(i_{t+1} \mid I_S) \sim f(I_S), \tag{1}$$

where $P(i_{t+1} \mid I_S)$ denotes the probability of recommending i_{t+1} given I_S. Conventional methods usually model $f(I_S)$ directly as a discriminant or probability function.

Framework

We propose RepeatNet to model $P(i_{t+1} \mid I_S)$ from a probabilistic perspective by explicitly taking *repeat consumption* into consideration, as shown in Eq. 2:

$$\begin{aligned} P(i_{t+1} \mid I_S) = P(r \mid I_S)P(i_{t+1} \mid r, I_S) + \\ P(e \mid I_S)P(i_{t+1} \mid e, I_S), \end{aligned} \tag{2}$$

where r and e denote *repeat mode* and *explore mode*, respectively. Here, $P(r \mid I_S)$ and $P(e \mid I_S)$ represent the probabilities of executing in *repeat mode* and *explore mode*, respectively. $P(i_{t+1} \mid r, I_S)$ and $P(i_{t+1} \mid e, I_S)$ refer to the probabilities of recommending i_{t+1} in *repeat mode* and in *explore mode*, respectively, given I_S.

As illustrated in Fig. 1, RepeatNet consists of four main components, a *session encoder*, a *repeat-explore mechanism*, a *repeat recommendation decoder*, and an *explore recommendation decoder*. The *session encoder* encodes the given session I_S into latent representations $H = \{h_1, h_2, \ldots, h_\tau, \ldots, h_t\}$, where h_t represents the session representation at timestamp t. The *repeat-explore mechanism* takes H as input and predicts the probabilities of executing *repeat mode* or *explore mode*, corresponding to $P(r \mid I_S)$ and $P(e \mid I_S)$ in Eq. 2. The *repeat recommendation decoder* takes H as input and predicts the repeat recommendation probabilities over clicked items in I_S, corresponding to $P(i_{t+1} \mid r, I_S)$ in Eq. 2. The *explore recommendation decoder* takes H as input and predicts the explore recommendation probabilities over unclicked items in $I - I_S$, where I refers to all items, corresponding to $P(i_{t+1} \mid e, I_S)$ in Eq. 2.

Session encoder

Like previous studies (Hidasi et al. 2016a; Li et al. 2017b), we use a GRU to encode I_S, where the GRU is defined as:

$$\begin{aligned} z_\tau &= \sigma(W_z[emb(i_\tau), h_{\tau-1}]) \\ r_\tau &= \sigma(W_r[emb(i_\tau), h_{\tau-1}]) \\ \widetilde{h_\tau} &= \tanh(W_h[emb(i_\tau), r_\tau \odot h_{\tau-1}]) \\ h_\tau &= (1 - z_\tau) \odot h_{\tau-1} + z_\tau \odot \widetilde{h_\tau}, \end{aligned} \tag{3}$$

where W_z, W_r, and W_h are weight matrices; $emb(i_\tau)$ is the item embedding of i_τ; σ denotes the sigmoid function. The initial state of the GRU is set to zero vectors, i.e., $h_0 = 0$. After the *session encoder*, each session I_S is encoded into $H = \{h_1, h_2, \ldots, h_\tau, \ldots, h_t\}$.

Repeat-explore mechanism

The *repeat-explore mechanism* can be seen as a binary classifier that predicts the recommendation mode based on $H = \{h_1, h_2, \ldots, h_\tau, \ldots, h_t\}$. To this end, we first apply an attention mechanism (Bahdanau, Cho, and Bengio 2015) to

H to get a fixed-length vector representation of I_S. Specifically, we first use the last hidden state h_t to match with each encoder hidden state $h_\tau \in H$ to get an importance score:

$$e_\tau^{re} = v_{re}^\top \tanh(W_{re} h_t + U_{re} h_\tau), \tag{4}$$

where v_{re}, W_{re}, and U_{re} are parameters. The importance scores are then normalized to get the context vector for I_S as a weighted sum in Eq. 5:

$$\begin{aligned} \alpha_\tau^{re} &= \frac{\exp(e_\tau^{re})}{\sum_{\tau=1}^{t} \exp(e_\tau^{re})} \\ c_{I_S}^{re} &= \sum_{\tau=1}^{t} \alpha_\tau^{re} h_\tau. \end{aligned} \tag{5}$$

We then employ a softmax regression to transform $c_{I_S}^{re}$ into a mode probability distribution, corresponding to $P(r \mid I_S)$ and $P(e \mid I_S)$ respectively, as shown in Eq. 6:

$$[P(r \mid I_S), P(e \mid I_S)] = \text{softmax}(W_{re}^c c_{I_S}^{re}), \tag{6}$$

where W_{re}^c is the weight matrix.

Repeat recommendation decoder

The *repeat recommendation decoder* evaluates the probability of re-clicking an item in I_S. Inspired by CopyNet (Gu et al. 2016), we use a modification of the attention model to achieve this. The probability of re-clicking item $i_\tau \in I_S$ is computed as follows:

$$e_\tau^r = v_r^\top \tanh(W_r h_t + U_r h_\tau) \tag{7}$$

$$P(i \mid r, I_S) = \begin{cases} \frac{\sum_i \exp(e_\tau^r)}{\sum_{\tau=1}^{t} \exp(e_\tau^r)} & \text{if } i \in I_S \\ 0 & \text{if } i \in I - I_S, \end{cases} \tag{8}$$

where v_r, W_r, and U_r are parameters; $\sum_i \exp(e_\tau^r)$ denotes the sum of all occurrences of item $i \in I_S$, because the same item might occur multiple times in different positions of I_S.

Explore recommendation decoder

The *explore recommendation decoder* evaluates the probability of clicking a new item that does not exist in I_S. To better capture the user's interest in session I_S, we employ an item-level attention mechanism that allows the decoder to dynamically select and linearly combine different parts of the input sequence (Li et al. 2017b):

$$\begin{aligned} e_\tau^e &= v_e^\top \tanh(W_e h_t + U_e h_\tau) \\ \alpha_\tau^e &= \frac{\exp(e_\tau^e)}{\sum_{\tau=1}^{t} \exp(e_\tau^e)} \\ c_{I_S}^e &= \sum_{\tau=1}^{t} \alpha_\tau^e h_\tau, \end{aligned} \tag{9}$$

where v_e, W_e, and U_e are parameters. The factors α_h^e determine which part of the input sequence should be emphasized or ignored when making predictions. We then combine the last hidden state and the attentive state into a hybrid representation c_{I_S} for I_S, which is the concatenation of vectors h_t and $c_{I_S}^e$: $c_{I_S} = [h_t, c_{I_S}^e]$.

Finally, the probability of clicking item $i_\tau \in I - I_S$ is computed as follows:

$$f_i = \begin{cases} -\infty & \text{if } i \in I_S \\ W_e^c c_{I_S} & \text{if } i \in I - I_S \end{cases} \tag{10}$$

$$P(i \mid e, I_S) = \frac{\exp(f_i)}{\sum_{\tau=1}^{t} \exp(f_\tau)}, \tag{11}$$

where W_e^c is the weight matrix and $-\infty$ means negative infinity. Since $\exp(-\infty) = 0$, we assume that if an item exists in I_S, then the probability of recommending it in the *explore mode* is zero.

Objective function

Our goal is to maximize the output prediction probability given the input session. Therefore, we optimize the negative log-likelihood loss function as follows:

$$L_{rec}(\theta) = -\frac{1}{|\mathbb{I}_\mathbb{S}|} \sum_{I_S \in \mathbb{I}_\mathbb{S}} \sum_{\tau=1}^{|I_S|} \log P(i_\tau \mid I_S), \tag{12}$$

where θ are all the parameters of RepeatNet, $\mathbb{I}_\mathbb{S}$ is the set of all sessions in the training set, and $P(i_\tau \mid I_S)$ is the item prediction probability as defined in Eq. 2.

RepeatNet incorporates an extra *repeat-explore mechanism* to softly switch between *repeat mode* and *explore mode*. We assume that if the next item exists in I_S, then it is generated under the *repeat mode*, otherwise *explore mode*. Here, we can jointly train another mode prediction loss as follows, which is also the negative log-likelihood loss:

$$\begin{aligned} &L_{mode}(\theta) \\ &= -\frac{1}{|\mathbb{I}_\mathbb{S}|} \sum_{I_S \in \mathbb{I}_\mathbb{S}} \sum_{\tau=1}^{|I_S|} \mathbb{1}(i_\tau \in I_S) \log P(r \mid I_S) + \\ &\qquad (1 - \mathbb{1}(i_\tau \in I_S)) \log P(e \mid I_S), \end{aligned} \tag{13}$$

where $\mathbb{1}(i_\tau \in I_S)$ is an indicator function that equals 1 if $i_\tau \in I_S$ and 0 otherwise.

In the case of joint training, the final loss is a linear combination of both losses:

$$L(\theta) = L_{rec}(\theta) + L_{mode}(\theta). \tag{14}$$

All parameters of RepeatNet as well as the item embeddings are learned in an end-to-end back-propagation training paradigm. Due to the full probability term in Eq. 2, the two modes probabilities $P(r \mid I_S)$, $P(e \mid I_S)$ and the item prediction probabilities $P(i \mid r, I_S)$, $P(i \mid e, I_S)$ are basically competing through a unified function.

Experiments

Datasets and evaluation metrics

We carry out experiments on three standard datasets, i.e., YOOCHOOSE, DIGINETICA, and LASTFM. YOOCHOOSE and DIGINETICA are frequently used in session-based recommendation studies (Hidasi et al. 2016a; Tan, Xu, and Liu 2016; Li et al. 2017b; Jannach and Ludewig

2017). Since they are both for e-commerce, we choose a third dataset in a different domain, music, Last.fm.[1] See Table 2. The splitting of the datasets are the same as (Li et al. 2017b).

- YOOCHOOSE[2] is a public dataset released by the RecSys Challenge 2015. We follow (Hidasi et al. 2016a; Li et al. 2017b) and filter out sessions of length 1 and items that appear less than 5 times. They note that the 1/4 version of the dataset is enough for the task and increasing the amount of data will not further improve the performance.
- DIGINETICA[3] is released by the CIKM Cup 2016. We again follow (Li et al. 2017b) and filter out sessions of length 1 and items that appear less than 5 times.
- LASTFM[4] is released by (Celma 2010) and widely used in recommendation tasks (Cheng et al. 2017). We use the dataset for music artist recommendation; we keep the top 40,000 most popular artists and filter out sessions that are longer than 50 or shorter than 2 items.

Recommender systems can only recommend a few items at a time, the actual item a user might pick should be amongst the first few items of the list (He et al. 2018b; Cheng et al. 2018). Therefore, commonly used metrics are MRR@20 and Recall@20 (He et al. 2018a; Mei et al. 2018). In this paper, we also report MRR@10 and Recall@10.

- Recall@k: The primary evaluation metric is Recall@k, which is the proportion of cases when the desired item is amongst the top-k items in all test cases.
- MRR@k: Another used metric is MRR@k (Mean Reciprocal Rank), which is the average of reciprocal ranks of the desire items. The reciprocal rank is set to zero if the rank is larger than k.

Table 2: Statistics of three datasets (number of sessions and items).

Dataset	Training	Validation	Test	Items
YOOCHOOSE	5,325,971	591,775	55,898	30,470
DIGINETICA	647,532	71,947	60,858	43,097
LASTFM	2,690,424	333,537	338,115	40,000

Implementation details

We set the item embedding size and GRU hidden state sizes to 100. We use dropout (Srivastava et al. 2014) with drop ratio $p = 0.5$. We initialize model parameters randomly using the Xavier method (Glorot and Bengio 2010). We use Adam as our optimizing algorithm. For the hyper-parameters of the Adam optimizer, we set the learning rate $\alpha = 0.001$, two momentum parameters $\beta 1 = 0.9$ and $\beta 2 = 0.999$, respectively, and $\epsilon = 10^{-8}$. We halve the learning rate α every 3 rounds. We also apply gradient clipping (Pascanu, Mikolov, and Bengio 2013) with range $[-5, 5]$ during training. To speed up the training and converge quickly, we use mini-batch size 1024 by grid search. We test the model performance on the validation set for every epoch. The model is written in Chainer (Tokui et al. 2015) and trained on a GeForce GTX TitanX GPU.

Methods used for comparison

Conventional methods We select the following conventional methods which are commonly used as baselines in session based recommendations (Hidasi et al. 2016a; Tan, Xu, and Liu 2016; Li et al. 2017b).

- POP: POP always recommends the most popular items in the training set. It is frequently used as baselines in recommender system domains (He et al. 2017).
- S-POP: S-POP recommends the most popular items of the current session. Ties are broken using global popularity values (Hidasi et al. 2016a).
- Item-KNN: Items similar to the actual item are recommended by this baseline. Similarity is defined as the co-occurrence number of two items in sessions divided by the square root of the product of the number of sessions in which either item occurs. Regularization is also included to avoid coincidental high similarities between rarely visited items (Davidson et al. 2010).
- BPR-MF: BPR-MF (Rendle et al. 2009) is a commonly used matrix factorization method. We apply it to session-based recommendation by representing a new session with the average latent factors of items that occurred in the session so far.
- FPMC: FPMC (Rendle, Freudenthaler, and Schmidt-Thieme 2010) is a state-of-the-art hybrid model for next-basket recommendation. To adapt it to session-based recommendation, we ignore the user latent representations when computing recommendation scores.
- PDP: Benson, Kumar, and Tomkins (2016) propose PDP and claim that they are the first to model sequential repeat consumption. This is the only recommendation model that considers sequential repeat consumption, to the best of our knowledge.

Deep learning methods No previous studies propose neural models that consider sequential repeat consumption. We select recent state-of-the-art neural session based recommendation models as baselines.

- GRU4REC: GRU4REC (Hidasi et al. 2016a) uses session-parallel mini-batch training process and also employs ranking-based loss functions for learning the model.
- Improved-GRU4REC: Improved GRU4REC (Tan, Xu, and Liu 2016) improves GRU4REC with two techniques, data augmentation and a method to account for shifts in the input data distribution.
- GRU4REC-TOPK: Hidasi and Karatzoglou (2017) further improve GRU4REC with a top-k based ranking loss.
- NARM: NARM (Li et al. 2017b) further improves Improved-GRU4REC with a neural attention mechanism.

[1] https://www.last.fm

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

[3] http://cikm2016.cs.iupui.edu/cikm-cup

[4] http://www.dtic.upf.edu/ ocelma /MusicRecommendationDataset/lastfm-1K.html

Table 3: Experimental results (%) on the three datasets.

Methods	YOOCHOOSE				DIGINETICA				LASTFM			
	MRR		Recall		MRR		Recall		MRR		Recall	
	@10	@20	@10	@20	@10	@20	@10	@20	@10	@20	@10	@20
POP	0.26	0.30	0.81	1.33	0.18	0.20	0.53	0.89	1.09	1.26	2.90	5.26
S-POP	17.70	17.79	25.96	27.11	13.64	13.68	20.56	21.06	8.36	8.71	18.08	22.59
Item-KNN	20.89	21.72	41.56	52.35	10.77	11.57	25.04	35.75	4.48	4.85	9.77	14.84
BPR-MF	1.90	1.97	3.07	4.05	1.86	1.98	3.60	5.24	4.88	5.19	9.87	14.05
FPMC	16.59	17.50	38.87	51.86	6.30	6.95	17.07	26.53	4.58	4.99	11.67	17.68
PDP	18.44	19.15	40.03	52.98	6.75	7.24	19.57	28.77	4.86	5.05	12.11	18.09
GRU4REC	21.64	22.60	46.67	59.56	7.59	8.33	19.09	29.45	4.92	5.39	11.56	17.90
Improved-GRU4REC	28.36	29.15	57.91	69.20	13.63	14.69	33.48	46.16	9.60	10.15	20.98	29.04
GRU4REC-TOPK	29.76	30.69	58.15	70.30	13.14	14.16	31.54	45.23	7.44	7.95	15.73	22.61
NARM	28.52	29.23	58.70	69.73	15.25	16.17	33.62	**49.70**	10.31	10.85	22.04	29.94
RepeatNet (no repeat)	30.02	30.76	59.62	70.21	12.71	13.52	30.96	42.56	9.92	10.47	21.81	29.96
RepeatNet	**30.50**†	**31.03**†	**59.76**†	**70.71**	**16.90**†	**17.66**†	**36.86**†	47.79	**11.46**†	**12.03**†	**24.18**†	**32.38**†

Bold face indicates the best result in terms of the corresponding metric. Significant improvements over the best baseline results are marked with † (t-test, $p < .05$). The scores reported in (Li et al. 2017b) on the DIGINETICA dataset differ because they did not sort the session items according to the “timeframe” field, which ignores the sequential information. We run the code released by (Hidasi et al. 2016a; Tan, Xu, and Liu 2016; Hidasi and Karatzoglou 2017; Li et al. 2017b) to obtain the results of GRU4REC, Improved-GRU4REC, GRU4REC-TOPK, and NARM.

Results and Analysis

Results

The results of all methods are shown in Table 3. We run the code released by (Hidasi et al. 2016a; Li et al. 2017b) to report the results of GRU4REC and NARM. We can get several insights from Table 3. First, RepeatNet outperforms both conventional methods and recent neural methods, including the strong baselines, GRU4REC-TOPK and NARM. The improvement of RepeatNet over NARM is even larger than the improvement of NARM over Improved-GRU4REC. The improvements mean that explicitly modeling *repeat consumption* is helpful, which gives RepeatNet more capabilities to model complex situations in session-based recommendations.

Second, as the repeat ratio increases, the performance of RepeatNet increases generally. We reach this conclusion based on the different improvements on YOOCHOOSE and DIGINETICA. Both datasets are from the e-commerce domain but YOOCHOOSE has a higher repeat ratio.

Third, the performance of RepeatNet varies with different domains. Table 3 shows that RepeatNet has a bigger advantage in the music domain than in the e-commerce domain; we believe this is due to different characteristics of the different domains. S-POP performs much better than Item-KNN on LASTFM, which means that popularity is very important on LASTFM. However, Item-KNN performs much better than S-POP on YOOCHOOSE, which means that collaborative filtering is more important on YOOCHOOSE. Besides, the neural models have substantial gains over the conventional methods in all evaluation metrics on all datasets generally. Similar conclusions have been formulated in other recent studies (Hidasi et al. 2016a; Tan, Xu, and Liu 2016; Li et al. 2017b).

Analysis of the repeat mechanism

Table 4: MRR@20 (%) of RepeatNet (with and without repeat mechanism) on repeat and non-repeat sessions.

RepeatNet		With repeat	No repeat
YOOCHOOSE	Rep	58.78	**60.18**
	Non-Rep	**21.60**	20.42
DIGINTICA	Rep	**56.27**	29.20
	Non-Rep	7.71	**9.48**
LASTFM	Rep	**41.63**	32.68
	Non-Rep	4.18	**5.06**

Rep: repeat sessions; *Non-Rep*: non-repeat sessions.

Generally, RepeatNet with repeat outperforms RepeatNet without repeat on all datasets, as shown in Table 3. The results of RepeatNet (with and without repeat) on repeated and non-repeated sessions are shown in Table 4 and 5. We can see that the improvements of RepeatNet mainly come from repeated sessions. Especially on DIGINTICA and LASTFM, RepeatNet improves by 33.91% and 24.16% respectively in terms of Recall@20 on repeated sessions. However, RepeatNet drops a little on non-repeated sessions. The results indicate that RepeatNet has more potential by explicitly modeling *repeat mode* and *explore mode*. But it also shows the limitation of RepeatNet that it seems inclined to repeat recommendations too much if we let it learn the mode probabilities totally from data. A mechanism should be added to incorporate prior knowledge.

Table 5: Recall@20 (%) of RepeatNet (with and without repeat mechanism) on repeat and non-repeat sessions.

RepeatNet		With repeat	No repeat
YOOCHOOSE	Rep	**97.41**	93.70
	Non-Rep	61.32	**61.95**
DIGINTICA	Rep	**99.09**	65.18
	Non-Rep	34.58	**36.73**
LASTFM	Rep	**91.22**	67.06
	Non-Rep	16.79	**20.10**

Analysis of the attention vs repeat mechanism

Neural attention has shown its potential on many tasks (Bahdanau, Cho, and Bengio 2015; Ren et al. 2017; Li et al. 2017a) and also on recommender systems recently (Li et al. 2017b; Xiao et al. 2017; Chen et al. 2017). We compare the results of RepeatNet with and without attention, with and without repeat in Table 6 and 7. The results show that both repeat and attention mechanisms can improve the results over Improved-GRU4REC. Importantly, the contributions of attention and repeat mechanisms are complementary as the combination brings further improvements, on all metrics and datasets, demonstrating the need for both. Besides, we can see that the attention mechanism helps to improve Recall while the repeat mechanism helps to improve MRR.

Table 6: MRR@20 (%) of RepeatNet with attention vs with repeat.

RepeatNet	YOOCHOOSE	DIGINTICA	LASTFM
No attention	28.65	16.03	11.10
No repeat	30.76	13.52	10.47
With both	**31.03**	**17.66**	**12.03**

Table 7: Recall@20 (%) of RepeatNet with attention vs with repeat.

RepeatNet	YOOCHOOSE	DIGINTICA	LASTFM
No attention	67.74	36.50	29.47
No repeat	70.21	42.56	29.96
With both	**70.71**	**47.79**	**32.38**

Analysis of joint learning

Interestingly, if we jointly train the recommendation loss L_{rec} and the mode prediction probability L_{mode}, the overall performance drops a little, as shown in Table 8. We believe that this is due to the following. First, L_{rec} is already a good supervisor for learning the mode prediction. This conclusion can be drawn from Table 4 and 5 where it shows that RepeatNet (with L_{rec} only) achieves large improvements on repeated sessions. And the room left for improvement on repeated sessions is relatively small. Second, RepeatNet (with L_{rec} only) is inclined to repeat recommendation. Adding L_{mode} further exacerbates the situation. Besides, L_{mode} assumes that if the next item exists in I_S, then it is generated in *repeat mode*, which is not always reasonable.

Table 8: MRR@20 and Recall@20 (%) of RepeatNet with and without joint learning.

Loss	YOOCHOOSE		LASTFM	
	MRR	Recall	MRR	Recall
L_{rec}	**31.03**	**70.71**	**12.03**	**32.38**
$L_{rec} + L_{mode}$	28.99	69.64	11.58	31.94

Conclusion and Future Work

We propose RepeatNet with an encoder-decoder architecture to address *repeat consumption* in the session-based recommendation task. By incorporating a *repeat-explore mechanism* into RNNs, RepeatNet can better capture the repeat-or-explore recommendation intent in a session. We conduct extensive experiments and analyses on three datasets and demonstrate that RepeatNet outperforms state-of-the-art methods in terms of MRR and Recall.

RepeatNet can be advanced and extended in several directions. First, prior knowledge of people can be incorporated to influence *repeat-explore mechanism*. Second, more information (e.g., metadata, text) and more factors (e.g., collaborative filtering) can be considered to further improve the performance. Besides, variants of RepeatNet can be applied to other recommendation tasks, such as content based recommendations.

Acknowledgments

This work is supported by the Natural Science Foundation of China (61672324, 61672322), the Natural Science Foundation of Shandong province (2016ZRE27468), the Fundamental Research Funds of Shandong University, Ahold Delhaize, Amsterdam Data Science, the Bloomberg Research Grant program, Elsevier, the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement nr 312827 (VOX-Pol), the Google Faculty Research Awards program, the Microsoft Research Ph.D. program, the Netherlands Institute for Sound and Vision, the Netherlands Organisation for Scientific Research (NWO) under project nrs CI-14-25, 652.002.001, 612.001.551, 652.-001.003, and Yandex. All content represents the opinion of the authors, which is not necessarily shared or endorsed by their respective employers and/or sponsors.

Code

To facilitate reproducibility of the results in this paper, we are sharing the code used to run the experiments in this paper at https://github.com/PengjieRen/RepeatNet.

References

Anderson, A.; Kumar, R.; Tomkins, A.; and Vassilvitskii, S. 2014. The dynamics of repeat consumption. In *WWW '14*.

Bahdanau, D.; Cho, K.; and Bengio, Y. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR '15*.

Benson, A. R.; Kumar, R.; and Tomkins, A. 2016. Modeling user consumption sequences. In *WWW '16*.

Celma, Ò. 2010. *Music Recommendation and Discovery in the Long Tail*. Springer.

Chen, S.; Moore, J. L.; Turnbull, D.; and Joachims, T. 2012. Playlist prediction via metric embedding. In *KDD '12*.

Chen, J.; Zhang, H.; He, X.; Nie, L.; Liu, W.; and Chua, T.-S. 2017. Attentive collaborative filtering: Multimedia recommendation with item- and component-level attention. In *SIGIR '17*.

Chen, J.; Wang, C.; and Wang, J. 2015. Will you "reconsume" the near past? Fast prediction on short-term reconsumption behaviors. In *AAAI '15*.

Cheng, Z.; Shen, J.; Zhu, L.; Kankanhalli, M.; and Nie, L. 2017. Exploiting music play sequence for music recommendation. In *IJCAI '17*, 3654–3660.

Cheng, Z.; Ding, Y.; Zhu, L.; and Kankanhalli, M. 2018. Aspect-aware latent factor model: Rating prediction with ratings and reviews. In *WWW '18*, 639–648.

Davidson, J.; Liebald, B.; Liu, J.; Nandy, P.; Van Vleet, T.; Gargi, U.; Gupta, S.; He, Y.; Lambert, M.; Livingston, B.; and Sampath, D. 2010. The YouTube video recommendation system. In *RecSys '10*.

Glorot, X., and Bengio, Y. 2010. Understanding the difficulty of training deep feedforward neural networks. *JMLR* 9:249–256.

Gu, J.; Lu, Z.; Li, H.; and Li, V. O. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *ACL '16*.

He, X.; Liao, L.; Zhang, H.; Nie, L.; Hu, X.; and Chua, T.-S. 2017. Neural collaborative filtering. In *WWW '17*.

He, X.; He, Z.; Du, X.; and Chua, T.-S. 2018a. Adversarial personalized ranking for recommendation. In *SIGIR '18*, 355–364.

He, X.; He, Z.; Song, J.; Liu, Z.; Jiang, Y.-G.; and Chua, T.-S. 2018b. Nais: Neural attentive item similarity model for recommendation. *TKDE* 30(12):2354–2366.

Hidasi, B., and Karatzoglou, A. 2017. Recurrent neural networks with top-k gains for session-based recommendations. *CoRR* abs/1706.03847.

Hidasi, B.; Karatzoglou, A.; Baltrunas, L.; and Tikk, D. 2016a. Session-based recommendations with recurrent neural networks. In *ICLR '16*.

Hidasi, B.; Quadrana, M.; Karatzoglou, A.; and Tikk, D. 2016b. Parallel recurrent neural network architectures for feature-rich session-based recommendations. In *RecSys '16*.

Jannach, D., and Ludewig, M. 2017. When recurrent neural networks meet the neighborhood for session-based recommendation. In *RecSys '17*.

Lerche, L.; Jannach, D.; and Ludewig, M. 2016. On the value of reminders within e-commerce recommendations. In *UMAP '16*.

Li, H.; Min, M. R.; Ge, Y.; and Kadav, A. 2017a. A context-aware attention network for interactive question answering. In *KDD '17*.

Li, J.; Ren, P.; Chen, Z.; Ren, Z.; Lian, T.; and Ma, J. 2017b. Neural attentive session-based recommendation. In *CIKM '17*.

Mei, L.; Ren, P.; Chen, Z.; Nie, L.; Ma, J.; and Nie, J.-Y. 2018. An attentive interaction network for context-aware recommendations. In *CIKM '18*, 157–166.

Mobasher, B.; Dai, H.; Luo, T.; and Nakagawa, M. 2002. Using sequential and non-sequential patterns in predictive web usage mining tasks. In *ICDM '02*.

Pascanu, R.; Mikolov, T.; and Bengio, Y. 2013. On the difficulty of training recurrent neural networks. In *ICML '13*.

Quadrana, M.; Karatzoglou, A.; Hidasi, B.; and Cremonesi, P. 2017. Personalizing session-based recommendations with hierarchical recurrent neural networks. In *RecSys '17*.

Ren, P.; Chen, Z.; Ren, Z.; Wei, F.; Ma, J.; and de Rijke, M. 2017. Leveraging contextual sentence relations for extractive summarization using a neural attention model. In *SIGIR '17*.

Rendle, S.; Freudenthaler, C.; Gantner, Z.; and Schmidt-Thieme, L. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *UAI '09*.

Rendle, S.; Freudenthaler, C.; and Schmidt-Thieme, L. 2010. Factorizing personalized markov chains for next-basket recommendation. In *WWW '10*.

Shani, G.; Heckerman, D.; and Brafman, R. I. 2005. An MDP-based recommender system. *JMLR* 6:1265–1295.

Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. 2014. Dropout: A simple way to prevent neural networks from overfitting. *JMLR* 15(1):1929–1958.

Tan, Y. K.; Xu, X.; and Liu, Y. 2016. Improved recurrent neural networks for session-based recommendations. In *DLRS '16*.

Tokui, S.; Oono, K.; Hido, S.; and Clayton, J. 2015. Chainer: a next-generation open source framework for deep learning. In *NIPS '15*.

Xiao, J.; Ye, H.; He, X.; Zhang, H.; Wu, F.; and Chua, T. 2017. Attentional factorization machines: Learning the weight of feature interactions via attention networks. In *IJCAI '17*.

Yap, G.-E.; Li, X.-L.; and Yu, P. S. 2012. Effective next-items recommendation via personalized sequential pattern mining. In *DASFAA '12*.

Zhang, Y.; Dai, H.; Xu, C.; Feng, J.; Wang, T.; Bian, J.; Wang, B.; and Liu, T.-Y. 2014. Sequential click prediction for sponsored search with recurrent neural networks. In *AAAI '14*.

Zimdars, A.; Chickering, D. M.; and Meek, C. 2001. Using temporal data for making recommendations. In *UAI '01*.