Neural Attentive Session-based Recommendation

Jing Li
Shandong University
Jinan, China
jingli.sdu@gmail.com

Pengjie Ren
Shandong University
Jinan, China
jay.ren@outlook.com

Zhumin Chen
Shandong University
Jinan, China
chenzhumin@sdu.edu.cn

Zhaochun Ren
Data Science Lab, JD.com
Beijing, China
renzhaochun@jd.com

Tao Lian
Shandong University
Jinan, China
liantao1988@gmail.com

Jun Ma
Shandong University
Jinan, China
majun@sdu.edu.cn

ABSTRACT

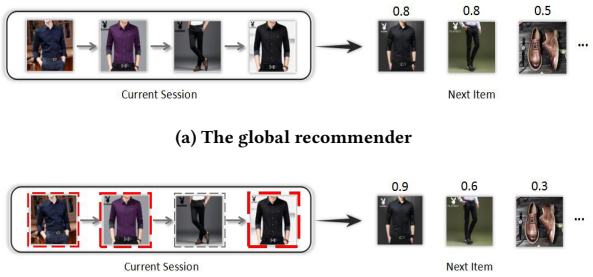
Given e-commerce scenarios that user profiles are invisible, session-based recommendation is proposed to generate recommendation results from short sessions. Previous work only considers the user's sequential behavior in the current session, whereas the user's main purpose in the current session is not emphasized. In this paper, we propose a novel neural networks framework, i.e., Neural Attentive Recommendation Machine (NARM), to tackle this problem. Specifically, we explore a hybrid encoder with an attention mechanism to model the user's sequential behavior and capture the user's main purpose in the current session, which are combined as a unified session representation later. We then compute the recommendation scores for each candidate item with a bi-linear matching scheme based on this unified session representation. We train NARM by jointly learning the item and session representations as well as their matchings. We carried out extensive experiments on two benchmark datasets. Our experimental results show that NARM outperforms state-of-the-art baselines on both datasets. Furthermore, we also find that NARM achieves a significant improvement on long sessions, which demonstrates its advantages in modeling the user's sequential behavior and main purpose simultaneously.

KEYWORDS

Session-based recommendation, sequential behavior, recurrent neural networks, attention mechanism

1 INTRODUCTION

A user session is kicked off when a user clicks a certain item; within a user session, clicking on the interesting item, and spending more time viewing it. After that, the user clicks another interesting one to start the view again. Such iterative process will be completed until the user's requirements are satisfied. Current recommendation research confronts challenges when recommendations are merely from those user sessions, where existing recommendation methods [1, 16, 39, 42] cannot perform well. To tackle this problem, session-based recommendation [33] is proposed to predict the next item that the user is probably interested in based merely on implicit feedbacks, i.e., user clicks, in the current session.



(a) The global recommender

(b) The local recommender

Figure 1: Two different recommenders. The global recommender models the user's whole sequential behavior to make recommendations while the local recommender captures the user's main purpose to make recommendations. The numbers above the items denote the recommendation scores produced by each recommender. In (b), the item in the red dashed box is more relevant to the current user's intention. And the red line is thicker when the item is more important.

Hidasi et al. [12] apply recurrent neural networks (RNN) with Gated Recurrent Units (GRU) for session-based recommendation. The model considers the first item clicked by a user as the initial input of RNN, and generates recommendations based on it. Then the user might click one of the recommendations, which is fed into RNN next, and the successive recommendations are produced based on the whole previous clicks. Tan et al. [40] further improve this RNN-based model by utilizing two crucial techniques, i.e., data augmentation and a method to account for shifts in the input data distribution. Though all above RNN-based methods show promising improvements over traditional recommendation approaches, they only take into account the user's sequential behavior in the current session, whereas the user's main purpose in the current session is not emphasized. Relying only on the user's sequential behavior is dangerous when a user accidentally clicks on wrong items or s/he is attracted by some unrelated items due

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
CIKM'17 , November 6–10, 2017, Singapore.
© 2017 ACM. ISBN 978-1-4503-4918-5/17/11…$15.00
DOI: https://doi.org/10.1145/3132847.3132926

to curiosity. Therefore, we argue that both the user's sequential behavior and main purpose in the current session should be considered in session-based recommendation.

Suppose that a user wants to buy a shirt on the Internet. As shown in Figure 1, during browsing, s/he tends to click on some shirts with similar styles to make a comparison, meanwhile s/he might click a pair of suit pants by accident or due to curiosity. After that, s/he keeps looking for suitable shirts. In this case, if we only consider about his/her sequential behavior, another shirt or suit pants even a pair of shoes might be recommended because many users click them after clicking some shirts and suit pants, as shown in Figure 1(a). Assume that the recommender is an experienced human purchasing guide, the guide could conjecture that this user is very likely to buy a short sleeve shirt at this time because most of his/her clicked items are related to it. Therefore, more attention would be paid to the short sleeve shirts that the user has clicked and another similar shirt would be recommended, as shown in Figure 1(b). Ideally, in addition to considering about the user's entire sequential behavior, a better recommender should also take into account the user's main purpose which is reflected by some relatively important items in the current session. Note that the sequential behavior and the main purpose in one session are complementary to each other because we can not always conjecture a user's main purpose from a session, e.g., when the session is too short or the user just clicks something aimlessly.

To tackle the above problem, we propose a novel neural networks framework, namely Neural Attentive Recommendation Machine (NARM). Specifically, we explore a hybrid encoder with an attention mechanism to model the user's sequential behavior and capture the user's main purpose in the current session, which are combined as a unified session representation later. With this item-level attention mechanism, NARM learns to attend differentially to more and less important items. We then compute the recommendation scores for each candidate item with a bi-linear matching scheme based on the unified session representation. NARM is trained by jointly learning the item and session representations as well as their matchings.

The main contributions of this work are summarized as follows:

- We propose a novel NARM model to take into account both the user's sequential behavior and main purpose in the current session, and compute recommendation scores by using a bi-linear matching scheme.
- We apply an attention mechanism to extract the user's main purpose in the current session.
- We carried out extensive experiments on two benchmark datasets. The results show that NARM outperforms state-of-the-art baselines in terms of recall and MRR on both datasets. Moreover, we find that NARM achieves better performance on long sessions, which demonstrates its advantages in modeling the user's sequential behavior and main purpose simultaneously.

2 RELATED WORK

Session-based recommendation is a typical application of recommender systems based on implicit feedbacks, where no explicit preferences (e.g., ratings) but only positive observations (e.g., clicks) are available [10, 23, 27]. These positive observations are usually in a form of sequential data as obtained by passively tracking users' behavior over a sequence of time. In this section, we briefly review the related work on session-based recommendation from the following two aspects, i.e., traditional methods and deep learning based methods.

2.1 Traditional Methods

Typically, there are two traditional modeling paradigms, i.e., general recommender and sequential recommender.

General recommender is mainly based on item-to-item recommendation approaches. In this setting, an item-to-item similarity matrix is pre-computed from the available session data. Items that are often clicked together (i.e., co-occurrence) in sessions are considered to be similar. Linden et al. [20] propose an item-to-item collaborative filtering method to personalize the online store for each customer. Sarwar et al. [32] analyze different item-based recommendation generation algorithms and compare their results with basic k-nearest neighbor approaches. Though these methods have proven to be effective and are widely employed, they only take into account the last click of the session, ignoring the information of the whole click sequence.

Sequential recommender is based on Markov chains which utilizes sequential data by predicting users' next action given the last action [36, 46]. Zimdars et al. [46] 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. Shani et al. [36] present a Markov Decesion Processes (MDP) aiming to provide recommendations in a session-based manner and the simplest MDP boil down to first-order Markov chains where the next recommendation can be simply computed through the transition probabilities between items. Mobasher et al. [25] study different sequential patterns for recommendation and find that contiguous sequential patterns are more suitable for sequential prediction task than general sequential patterns. Yap et al. [44] introduce a new Competence Score measure in personalized sequential pattern mining for next-item recommendations. Chen et al. [3] 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 in 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.

2.2 Deep Learning based Methods

Deep learning has recently been applied very successfully in areas such as image recognition [8, 17], speech recognition [2, 7, 13] and neural language processing [5, 18, 30, 37, 38]. Deep models can be trained to learn discriminative representations from unstructured data [9, 11, 19]. Here, we focus on the related work that uses deep learning models to solve recommendation tasks.

Neural network recommender is mostly focusing on the classical collaborative filtering user-item setting. Salakhutdinov et al. [31] first propose to use Restricted Boltzmann Machines (RBM) for Collaborative Filtering (CF). In their work, RBM is used to model

user-item interactions and to perform recommendations. Recently, denoising auto-encoders have been used to perform CF in a similar manner [34, 43]. Wang et al. [41] introduce a hierarchical representation model for the next basket recommendation which is based on encoder-decoder mechanism. Deep neural networks have also been used in cross-domain recommendations whereby items are mapped to a joint latent space [6]. Recurrent Neural Networks (RNN) have been devised to model variable-length sequence data. Recently, Hidasi et al. [12] apply RNN to session-based recommendation and achieve significant improvements over traditional methods. The proposed model utilizes session-parallel mini-batch training and employs ranking-based loss functions for learning the model. Tan et al. [40] further study the application of RNN in session-based recommendation. They propose two techniques to improve the performance of their model, namely data augmentation and a method to account for shifts in the input data distribution. Zhang et al. [45] also use RNN for the click sequence prediction, they consider historical user behaviors as well as hand-crafted features for each user and item.

Though a growing number of publications on session-based recommendation focus on RNN-based methods, unlike existing studies, we propose a novel neural attentive recommendation model that combines both the user's sequential behavior and main purpose in the current session, which to the best of our knowledge, is not considered by existing researches. And we apply the attention mechanism to session-based recommendation for the first time.

3 METHOD

In this section, we first introduce the session-based recommendation task. Then we describe the proposed NARM in detail.

3.1 Session-based Recommendation

Session-based recommendation is the task of predicting what a user would like to click next when his/her current sequential transaction data is given. Here we give a formulation of the session-based recommendation problem.

Let $[x_1, x_2, ..., x_{n-1}, x_n]$ be a click session, where $x_i \in \mathcal{I}$ $(1 \leq i \leq n)$ is the index of one clicked item out of a total number of m items. We build a model M so that for any given prefix of the click sequence in the session, $\mathbf{x} = [x_1, x_2, ..., x_{t-1}, x_t]$, $1 \leq t \leq n$, we get the output $\mathbf{y} = \mathsf{M}(\mathbf{x})$, where $\mathbf{y} = [y_1, y_2, ..., y_{m-1}, y_m]$. We view $\mathbf{y}$ as a ranking list over all the next items that can occur in that session, where y_j $(1 \leq j \leq m)$ corresponds to the recommendation score of item j. Since a recommender typically needs to make more than one recommendations for the user, thus the top-k $(1 \leq k \leq m)$ items in $\mathbf{y}$ are recommended.

3.2 Overview

In this paper, we propose an improved neural encoder-decoder architecture [26, 35] to address the session-based recommendation problem, named Neural Attentive Recommendation Machine (NARM). The basic idea of NARM is to build a hidden representation of the current session, and then generate predictions based on it. As shown in Figure 2, the encoder converts the input click sequence $\mathbf{x} = [x_1, x_2, ..., x_{t-1}, x_t]$ into a set of high-dimensional

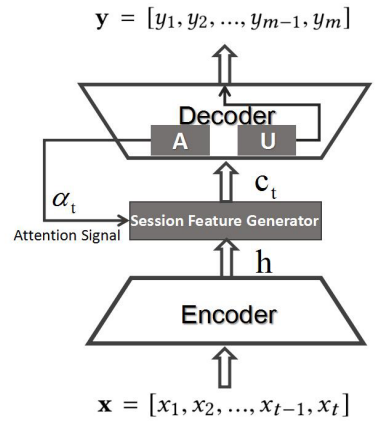


Figure 2: The general framework and dataflow of the encoder-decoder-based NARM.

hidden representations $\mathbf{h} = [\boldsymbol{h}_1, \boldsymbol{h}_2, ..., \boldsymbol{h}_{t-1}, \boldsymbol{h}_t]$, which along with the attention signal at time t (denoted as α_t), are fed to the session feature generator to build the representation of the current session to decode at time t (denoted as $\boldsymbol{c}_t$). Finally $\boldsymbol{c}_t$ is transformed by a matrix $\boldsymbol{U}$ (as part of the decoder) into an activate function to produce a ranking list over all items, $\mathbf{y} = [y_1, y_2, ..., y_{m-1}, y_m]$, that can occur in the current session.

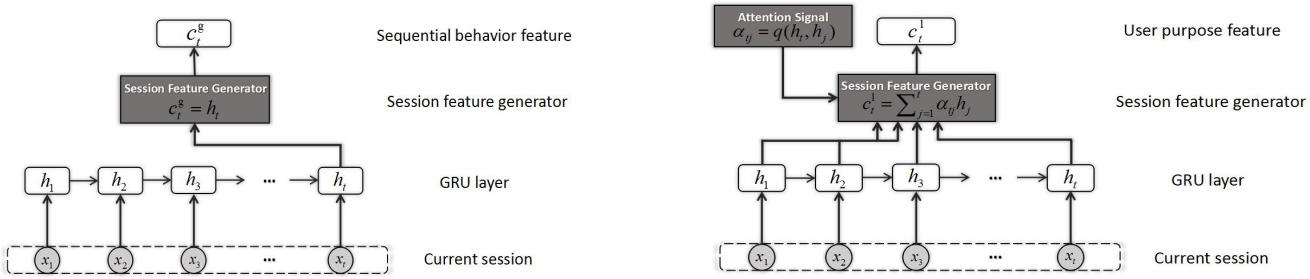
The role of α_t is to determine which part of the hidden representations should be emphasized or ignored at time t. It should be noted that α_t could be fixed over time or changes dynamically during the prediction process. In the dynamic setting, α_t can be a function of the representations of hidden states or the input item embeddings. We adopt the dynamic setting in our model, more details will be described in §3.4.

The basic idea of our work is to learn a recommendation model that takes into consideration both the user's sequential behavior and main purpose in the current session. In the following part of this section, we first describe the global encoder in NARM which is used to model the user's sequential behavior (§3.3). Then we introduce the local encoder which is used to capture the user's main purpose in the current session (§3.4). Finally we show our NARM which combines both of them and computes the recommendation scores for each candidate item by using a bilinear matching scheme (§3.5).

3.3 Global Encoder in NARM

In the global encoder, the inputs are entire previous clicks while the output is the feature of the user's sequential behavior in the current session. Both the inputs and output are uniformly represented by high-dimensional vectors.

Figure 3(a) shows the graphical model of the global encoder in NARM. We use a RNN with Gated Recurrent Units (GRU) rather than a standard RNN because Hidasi et al. [12] demonstrate that GRU can outperform the Long Short-Term Memory (LSTM) [14] units for the session-based recommendation task. GRU is a more elaborate RNN unit that aims at dealing with the vanishing gradient problem. The activation of GRU is a linear interpolation



(a) The graphical model of the global encoder in NARM, where the last hidden state is interpreted as the user's sequential behavior feature $c_t^{\mathrm{g}} = h_t$.

(b) The graphical model of the local encoder in NARM, where the weighted sum of hidden states is interpreted as the user's main purpose feature $c_t^{\mathrm{l}} = \sum_{j=1}^{t} \alpha_{tj} h_j$.

Figure 3: The global encoder and the local encoder in NARM.

between the previous activation $\boldsymbol{h}_{t-1}$ and the candidate activation $\widehat{\boldsymbol{h}}_t$,

$$\boldsymbol{h}_t = (1 - \boldsymbol{z}_t)\boldsymbol{h}_{t-1} + \boldsymbol{z}_t \widehat{\boldsymbol{h}}_t \ , \tag{1}$$

where the update gate $\boldsymbol{z_t}$ is given by

$$\boldsymbol{z}_t = \sigma(\boldsymbol{W}_z \boldsymbol{x}_t + \boldsymbol{U}_z \boldsymbol{h}_{t-1}) \ . \tag{2}$$

The candidate activation function $\widehat{\boldsymbol{h}}_t$ is computed as

$$\widehat{\boldsymbol{h}}_t = tanh[\boldsymbol{W}\boldsymbol{x}_t + \boldsymbol{U}(\boldsymbol{r}_t \odot \boldsymbol{h}_{t-1})] \ , \tag{3}$$

where the reset gate $\boldsymbol{r_t}$ is given by

$$\boldsymbol{r}_t = \sigma(\boldsymbol{W}_r \boldsymbol{x}_t + \boldsymbol{U}_r \boldsymbol{h}_{t-1}) \ . \tag{4}$$

With a trivial session feature generator, we essentially use the final hidden state $\boldsymbol{h_t}$ as the representation of the user's sequential behavior

$$\boldsymbol{c}_t^{\mathrm{g}} = \boldsymbol{h}_t \ . \tag{5}$$

However, this global encoder has its drawbacks such as a vectorial summarization of the whole sequence behavior is often hard to capture a preciser intention of the current user.

3.4 Local Encoder in NARM

The architecture of the local encoder is similar to the global encoder as shown in Figure 3(b). In this encoding scheme we also use RNN with GRU as the basic component. To capture the user's main purpose in the current session, we involve an item-level attention mechanism which allows the decoder to dynamically select and linearly combine different parts of the input sequence,

$$\boldsymbol{c}_t^{\mathrm{l}} = \sum_{j=1}^{t} \alpha_{tj} \boldsymbol{h}_j \ , \tag{6}$$

where the weighted factors α determine which part of the input sequence should be emphasized or ignored when making predictions, which in turn is a function of hidden states,

$$\alpha_{tj} = q(\boldsymbol{h}_t, \boldsymbol{h}_j) \ . \tag{7}$$

Basically, the weighted factor α_{tj} models the alignment between the inputs around position j and the output at position t, so it can be viewed as a specific matching model. In the local encoder, the function q specifically computes the similarity between the final hidden state $\boldsymbol{h}_t$ and the representation of the previous clicked item $\boldsymbol{h}_j$,

$$q(\boldsymbol{h}_t, \boldsymbol{h}_j) = \boldsymbol{v}^{\mathrm{T}} \sigma(\boldsymbol{A}_1 \boldsymbol{h}_t + \boldsymbol{A}_2 \boldsymbol{h}_j) \ , \tag{8}$$

where σ is an activate function such as sigmoid function, matrix $\boldsymbol{A}_1$ is used to transform $\boldsymbol{h}_t$ into a latent space, and $\boldsymbol{A}_2$ plays the same role for $\boldsymbol{h}_j$.

This local encoder enjoys the advantages of adaptively focusing on more important items to capture the user's main purpose in the current session.

3.5 NARM Model

For the task of session-based recommendation, the global encoder has the summarization of the whole sequential behavior, while the local encoder can adaptively select the important items in the current session to capture the user's main purpose. We conjecture that the representation of the sequential behavior may provide useful information for capturing the user's main purpose in the current session. Therefore, we use the representations of the sequential behavior and the previous hidden states to compute the attention weight for each clicked item. Then a natural extension combines the sequential behavior feature and the user purpose feature by concatenating them to form an extended representation for each time stamp.

As shown in Figure 4, we can see the summarization $\boldsymbol{h}_t^{\mathrm{g}}$ is incorporated into $\boldsymbol{c}_t$ to provide a sequential behavior representation for NARM. It should be noticed that the session feature generator in NARM will evoke different encoding mechanisms in the global encoder and the local encoder, although they will be combined later to form a unified representation. More specifically, the last hidden state of the global encoder $\boldsymbol{h}_t^{\mathrm{g}}$ plays a role different from that of the local encoder $\boldsymbol{h}_t^{\mathrm{l}}$. The former has the responsibility to encode the entire sequential behavior. The latter is used to compute the attention weights with the previous hidden states. By this hybrid encoding scheme, both the user's sequential behavior and main purpose in the current session can be modeled into a unified representation $\boldsymbol{c}_t$, which is the concatenation of vectors $\boldsymbol{c}_t^{\mathrm{g}}$

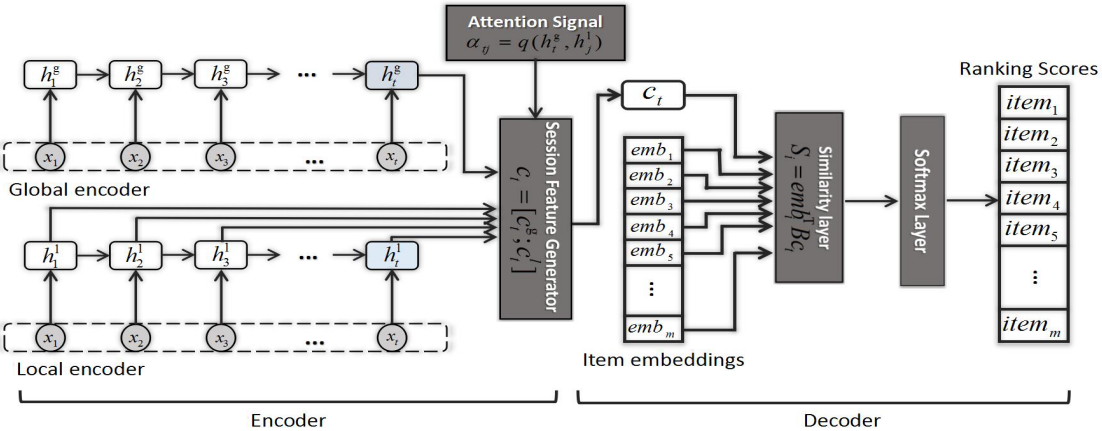


Figure 4: The graphical model of NARM, where the session feature c_t is represented by the concatenation of vectors c_t^g and c_t^l (as computed in equation (5) and (6)). Note that h_t^g and h_t^l play different roles, while they have the same values. The last hidden state of the global encoder h_t^g plays a role to encode the entire input clicks while the last hidden state of the local encoder h_t^l is used to compute attention weights with the previous hidden states.

and $\boldsymbol{c}_t^l$,

$$\boldsymbol{c}_t = [\boldsymbol{c}_t^g; \boldsymbol{c}_t^l] = [\boldsymbol{h}_t^g; \sum_{j=1}^{t} \alpha_{tj} \boldsymbol{h}_t^l] \ . \tag{9}$$

Figure 4 also gives a graphical illustration of the adopted decoding mechanism in NARM. Generally, a standard RNN utilizes fully-connected layer to decode. But using fully-connected layer means that the number of parameters to be learned in this layer is $|H| * |N|$ where $|H|$ is the dimension of the session representation and $|N|$ is the number of candidate items for prediction. Thus we have to reserve a large space to store these parameters. Though there are some approaches to reduce the parameters such as using a hierarchical softmax layer [24], and negative sampling at random [22], they are not the best choices for our model.

We propose an alternative bi-linear decoding scheme which not only reduces the number of the parameters, but also improves the performance of NARM. Specifically, a bi-linear similarity function between the representations of the current session and each candidate items is used to compute a similarity score S_i,

$$S_i = emb_i^{\mathrm{T}} \boldsymbol{B} \, \boldsymbol{c}_t \ , \tag{10}$$

where $\boldsymbol{B}$ is a $|D| * |H|$ matrix, $|D|$ is the dimension of each item embedding. Then the similarity score of each item is entered to a softmax layer to obtain the probability that the item will occur next. By using this bi-linear decoder, we reduce the number of parameters from $|N| * |H|$ to $|D| * |H|$, where $|D|$ is usually smaller than $|N|$. Moreover, the experiment results demonstrate that using this bi-linear decoder can improve the performance of NARM (as demonstrated in §4.4).

To learn the parameters of the model, we do not utilize the proposed training procedure in [12], where the model is trained in a session-parallel, sequence-to-sequence manner. Instead, in order to fit the attention mechanism in the local encoder, NARM process each sequence $[x_1, x_2, ..., x_{t-1}, x_t]$ separately. Our model can be trained by using a standard mini-batch gradient descent on the cross-entropy loss:

$$L(p, q) = -\sum_{i=1}^{m} p_i log(q_i) \tag{11}$$

where q is the prediction probability distribution and p is the truly distribution. At last, a Back-Propagation Through Time (BPTT) method for a fixed number of time steps is adopted to train NARM.

4 EXPERIMENTAL SETUP

In this section, we first describe the datasets, the state-of-the-art methods and the evaluation metrics employed in our experiments. Then we compare NARMs with different decoding schemes. Finally, we compare NARM with state-of-the-art methods.

4.1 Dataset

We evaluate different recommenders on two standard transaction datasets, i.e., YOOCHOOSE dataset and DIGINETICA dataset.

- YOOCHOOSE[1] is a public dataset released by RecSys Challenge 2015. This dataset contains click-streams on an e-commerce site. After filtering out sessions of length 1 and items that appear less than 5 times, there remains 7981580 sessions and 37483 items.
- DIGINETICA[2] comes from CIKM Cup 2016. We only used the released transaction data and also filtered out sessions of length 1 and items that appear less than 5 times. Finally the dataset contains 204771 sessions and 43097 items.

We first conducted some preprocesses over two datasets. For YOOCHOOSE, we used the sessions of subsequent day for testing and filtered out clicks from the test set where the clicked items did not appear in the training set. For DIGINETICA, the only difference is that we use the sessions of subsequent week for testing. Because we did not train NARM in a session-parallel manner [12], a

[1] http://2015.recsyschallenge.com/challenge.html
[2] http://cikm2016.cs.iupui.edu/cikm-cup

Table 1: Statistics of the datasets used in our experiments. (The avg.length means the average length of the complete dataset.)

Datasets	all the clicks	train sessions	test sessions	all the items	avg.length
YOOCHOOSE 1/64	557248	369859	55898	16766	6.16
YOOCHOOSE 1/4	8326407	5917746	55898	29618	5.71
DIGINETICA	982961	719470	60858	43097	5.12

sequence splitting preprocess is necessary. For the input session $[x_1, x_2, ..., x_{n-1}, x_n]$, we generated the sequences and corresponding labels $([x_1], V(x_2), ([x_1, x_2], V(x_3), ..., ([x_1, x_2, ..., x_{n-1}], V(x_n))$ for training on both YOOCHOOSE and DIGINETICA. The corresponding label $V(x_i)$ is the last click in the current session.

For the following reasons: (1) YOOCHOOSE is quite large, (2) Tan et al. [40] verified that the recommendation models do need to account for changing user behavior over time, (3) their experimental results showed that training on the entire dataset yields slightly poorer results than training on more recent fractions of the datasets. Thus we sorted the training sequences of YOOCHOOSE by time and reported our results on the model trained on more recent fractions 1/64 and 1/4 of training sequences as well. Note that some items that in the test set would not appear in the training set since we trained the model only on more recent fractions. The statistics of the three datasets (i.e., YOOCHOOSE 1/64, YOOCHOOSE 1/4 and DIGINETICA) are shown in Table 1.

4.2 Baseline Methods

We compare the proposed NARM with five traditional methods (i.e., POP, S-POP, Item-KNN, BPR-MF and FPMC) and two RNN-based models (i.e., GRU-Rec and Improved GRU-Rec).

- **POP**: Popular predictor always recommends the most popular items in the training set. Despite its simplicity, it is often a strong baseline in certain domains.
- **S-POP**: This baseline recommends the most popular items for the current session. The recommendation list changes during the session gains more items. Ties are broken up using global popularity values.
- **Item-KNN**: In 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 [4, 20].
- **BPR-MF**: BPR-MF [28] optimizes a pairwise ranking objective function via stochastic gradient descent. Matrix factorization can not be directly applied to session-based recommendation because new sessions do not have pre-computed latent representations. However, we can make it work by representing a new session with the average latent factors of items occurred in the session so far. In other words, the recommendation score can be computed as the average of the similarities between latent factors of a candidate item and the items in the session so far.
- **FPMC**: FPMC [29] is a state-of-the-art hybrid model on the next-basket recommendation. In order to make it work on session-based recommendation, we do not consider the user latent representations when computing recommendation scores.
- **GRU-Rec**: We denote the model proposed in [12] as GRU-Rec, which utilizes session-parallel mini-batch training process and also employs ranking-based loss functions for learning the model.
- **Improved GRU-Rec**: We denote the model proposed in [40] as Improved GRU-Rec. Improved GRU-Rec adopts two techniques which include data augmentation and a method to account for shifts in the input data distribution to improve the performance of GRU-Rec.

4.3 Evaluation Metrics and Experimental Setup

4.3.1 Evaluation Metrics.

As recommender systems can only recommend a few items at each time, the actual item a user might pick should be amongst the first few items of the list. Therefore, we use the following metrics to evaluate the quality of the recommendation lists.

- Recall@20: The primary evaluation metric is Recall@20 that is the proportion of cases when the desired item is amongst the top-20 items in all test cases. Recall@N does not consider the actual rank of the item as long as it is amongst the top-N and also usually correlates well with other metrics such as click-through rate (CTR) [21].
- MRR@20: Another used metric is MRR@20 (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 20. MRR takes the rank of the item into account, which is important in settings where the order of recommendations matters.

4.3.2 Experimental Setup.

The proposed NARM model uses 50-dimensional embeddings for the items. Optimization is done using Adam [15] with the initial learning rate sets to 0.001, and the mini-batch size is fixed at 512. There are two dropout layers used in NARM: the first dropout layer is between the item embedding layer and the GRU layer with 25% dropout, the second one is between the GRU layer and the bi-linear similarity layer with 50% dropout. We also truncate BPTT at 19 time steps as the setting in the state-of-the-art method [40] and the number of epochs is set to 30 while using 10% of the training data as the validation set. We use one GRU layer in our model and the GRU is set at 100 hidden units. The model is defined and trained in Theano on a GeForce GTX TitanX GPU. The source code of our model is available online[3].

[3]https://github.com/lijingsdu/sessionRec_NARM

Table 2: The comparison of different decoders in NARM.

Decoders	YOOCHOOSE 1/64		YOOCHOOSE 1/4		DIGINETICA	
	Recall@20(%)	MRR@20(%)	Recall@20(%)	MRR@20(%)	Recall@20(%)	MRR@20(%)
Fully-connected decoder	67.67	29.17	69.49	29.54	57.84	24.77
Bi-linear similarity decoder	**68.32**	28.76	**69.73**	29.23	**62.58**	**27.35**

Table 3: Performance comparison of NARM with baseline methods over three datasets.

Methods	YOOCHOOSE 1/64		YOOCHOOSE 1/4		DIGINETICA	
	Recall@20(%)	MRR@20(%)	Recall@20(%)	MRR@20(%)	Recall@20(%)	MRR@20(%)
POP	6.71	1.65	1.33	0.30	0.91	0.23
S-POP	30.44	18.35	27.08	17.75	21.07	14.69
Item-KNN	51.60	21.81	52.31	21.70	28.35	9.45
BPR-MF	31.31	12.08	3.40	1.57	15.19	8.63
FPMC*	45.62	15.01	-	-	31.55	8.92
GRU-Rec	60.64	22.89	59.53	22.60	43.82	15.46
Improved GRU-Rec	67.84	**29.00**	69.11	29.22	57.95	24.93
NARM	**68.32**	28.76	**69.73**	**29.23**	**62.58**	**27.35**

* On YOOCHOOSE 1/4, we do not have enough memory to initialize FPMC. Our available memory is 120G.

4.4 Comparison among Different Decoders

We first empirically compare NARMs with different decoders, i.e., fully-connected decoder and bi-linear similarity decoder. The results over three datasets are shown in Table 2. Here we only illustrate the results on 100-dimensional hidden states because we obtain the same conclusions on other dimension settings.

We make following observations from Table 2: (1) With regard to Recall@20, the performance improves when using the bi-linear similarity decoder, and the improvements are around 0.65%, 0.24% and 4.74% respectively over three datasets. (2) And with regard to MRR@20, the performance on the model using the bi-linear decoder becomes a little worse on YOOCHOOSE 1/64 and 1/4. But on DIGINETICA, the model with the bi-linear decoder still obviously outperforms the model with the fully-connected decoder.

For the session-based recommendation task, as the recommender system recommends top-20 items at once in our settings, the actual item a user might pick should be among the list of 20 items. Thus we consider that the recall metric is more important than the MRR metric in this task, and NARM adopts the bi-linear decoder in the following experiments.

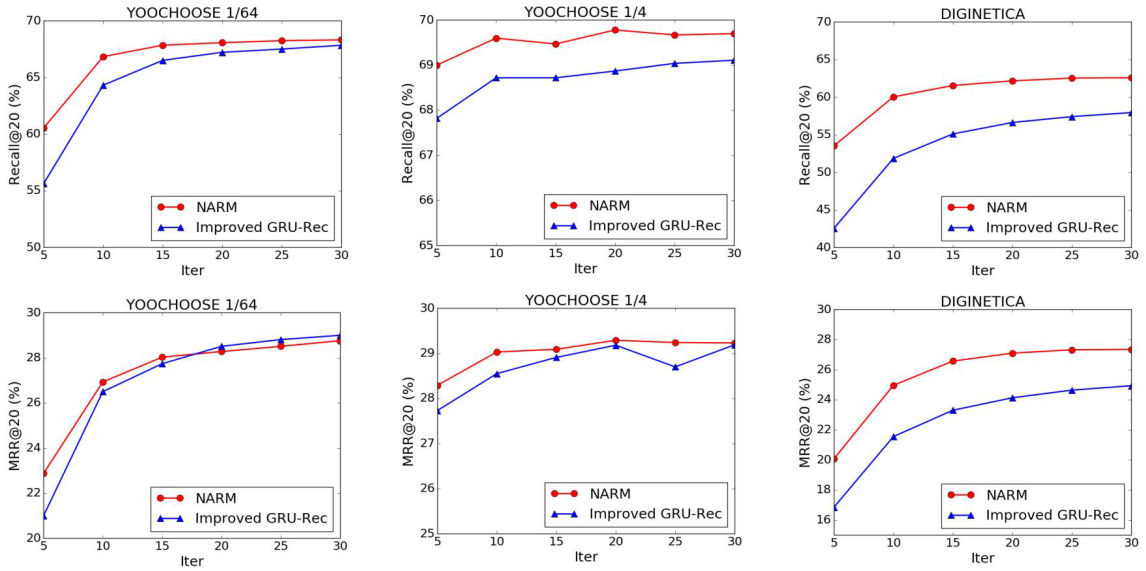
4.5 Comparison against Baselines

Next we compare our NARM model with state-of-the-art methods. The results of all methods over three datasets are shown in Table 3. And a more specific comparison between NARM and the best baseline (i.e., Improved GRU-Rec) over three datasets are illustrated in Figure 5.

We have the following observations from the results: (1) For YOOCHOOSE 1/4 dataset, BPR-MF does not work when we use the average of item factors occurred in the session to replace the user factor. Besides, since we regard each session as one user in FPMC, we do not have enough memory to initialize it. These problems indicate traditional user-based methods are no longer suitable for session-based recommendation. (2) Overall, three RNN-based methods consistently outperform the traditional baselines, which demonstrates that RNN-based models are good at dealing with sequence information in sessions. (3) By taking both the user's sequential behavior and main purpose into consideration, the proposed NARM can outperform all the baselines in terms of recall@20 over three datasets and can outperform most of the baselines in terms of MRR@20. Take DIGINETICA dataset as an example, when compared with the best baseline (i.e., Improved GRU-Rec), the relative performance improvements by NARM are around 7.98% and 9.70% respectively in terms of recall@20 and MRR@20. (4) As we can see, the recall values on two YOOCHOOSE datasets are not as significantly as the results on DIGINETICA and the obtained MRR values are very close to each other. We consider that one of the important reasons is when we split YOOCHOOSE dataset to 1/64 and 1/4, we do not filter out clicks from the test set where the clicked items are not in the training set in order to be consistent with the setting on Improved GRU-Rec [40]. While on DIGINETICA, we filter out these clicks from the test set, and hence NARM outperforms the baselines significantly in terms of both Recall@20 and MRR@20.

5 ANALYSIS

In this section, We further explore the influences of using different session features in NARM and analyze the effectiveness of the adopted attention mechanism.



(a) YOOCHOOSE1/64 (b) YOOCHOOSE1/4 (c) DIGINETICA

Figure 5: Performance comparison between NARM and the best baseline (i.e., Improved GRU-Rec) over three datasets.

Table 4: Performance comparison among three versions of NARM over three datasets.

(a) Performance comparison on YOOCHOOSE 1/64

	d=50		d=100	
Models	Recall@20	MRR@20	Recall@20	MRR@20
$NARM_{global}$	67.26	26.95	68.15	28.37
$NARM_{local}$	67.07	26.79	68.10	28.38
$NARM_{hybrid}$	**68.28**	**28.10**	**68.32**	**28.76**

(b) Performance comparison on YOOCHOOSE 1/4

	d=50		d=100	
Models	Recall@20	MRR@20	Recall@20	MRR@20
$NARM_{global}$	67.67	27.10	68.91	28.48
$NARM_{local}$	67.50	27.21	68.01	27.36
$NARM_{hybrid}$	**69.17**	**28.67**	**69.73**	**29.23**

(c) Performance comparison on DIGINETICA

	d=50		d=100	
Models	Recall@20	MRR@20	Recall@20	MRR@20
$NARM_{global}$	59.63	23.52	61.88	26.51
$NARM_{local}$	58.74	22.91	61.71	26.04
$NARM_{hybrid}$	**61.73**	**26.25**	**62.58**	**27.35**

5.1 Influence of Using Different Features

In this part, we refer to the NARM that uses the sequential behavior feature only, the NARM that uses the user purpose feature only, and the NARM that uses both two features as $NARM_{global}$, $NARM_{local}$ and $NARM_{hybrid}$ respectively. As shown in Table 4, (1) $NARM_{global}$ and $NARM_{local}$, which only use a single feature, do not perform well on three datasets. Besides, their performance are very close to each other in terms of two metrics. This indicates that merely considering the sequential behavior or the user purpose in the current session may not be able to learn a good recommendation model. (2) When we take into account both the user's sequential behavior and main purpose, $NARM_{hybrid}$ performs better than $NARM_{global}$ and $NARM_{local}$ in terms of Recall@20 and MRR@20 on different hidden state dimensions over three datasets. Take DIGINETICA dataset as an example, when compared with $NARM_{global}$ and $NARM_{local}$ with the dimensionality of the hidden state set to 50, the relative performance improvements by $NARM_{hybrid}$ are around 3.52% and 5.09% in terms of Recall@20 respectively. These results demonstrate the advantages of considering both the sequential behavior and the main purpose of the current user in session-based recommendation.

5.2 Influence of Different Session Lengths

Our NARM model is based on the assumption that when a user is browsing online, his/her click behavior frequently revolves his/her main purpose in the current session. However, we can hardly capture the user's main purpose when s/he just clicks a few items. Therefore, our NARM model should be good at modeling long sessions. To verify this, we make comparisons among sessions with different lengths on DIGINETICA. As shown in Table 5, (1) NARM performs better when the session lengths are between 4 and 17 in general. This indicates that NARM do capture the user's main purpose more accuracy on long sessions. In other words, it could

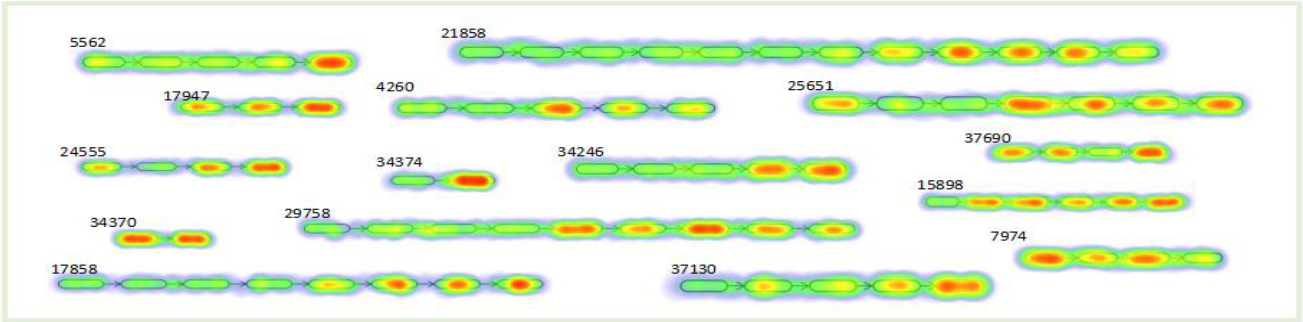


Figure 6: Visualization of items weights. The depth of the color corresponds to the importance of items given by equation (7). The numbers above the sessions is the session IDs. (Best viewed in color.)

Table 5: Performance comparison among different session lengths on DIGINETICA dataset. (The baseline method is Improved GRU-Rec [40].)

	DIGINETICA DATASET		
Length	Baseline correct	NARM correct	Performance
1	8747	9358	+6.98%
2	6601	7084	+7.31%
3	4923	5299	+7.63%
4	3625	3958	+9.18%
5	2789	3019	+8.24%
6	2029	2202	+8.52%
7	1520	1656	+8.94%
8	1198	1295	+8.09%
9	915	996	+8.85%
10	690	753	+9.13%
11	509	587	**+15.32%**
12	411	459	**+11.67%**
13	304	323	+6.25%
14	243	260	+6.99%
15	199	219	**+10.05%**
16	149	165	**+10.73%**
17	98	112	**+14.28%**
18	88	93	+5.68%
19	70	75	+7.14%

make a better prediction if NARM captures more user purpose features on the basis of the existing sequential behavior features. (2) When sessions are too long, the performance improvements of NARM are declined. We consider the reason is that when a session is too long, the user is very likely to click some items aimlessly, so that the local encoder in NARM could not capture the user's main purpose in the current session.

5.3 Visualize the Attention Weights

To illustrate the role of the attention mechanism intuitively, we present an example in Figure 6. The session instances are chosen randomly from DIGINETICA. The depth of the color corresponds to the importance of items given by equation (7). We have following observations from the example: (1) Overall, it is obvious that not all items are related to the next click and almost all the important items in the current session is continuous. This implies that the users' intentions in sessions are indeed localized, which is one of the reasons why NARM can outperform the general RNN-based model. (2) The most important items are often near the end of the session. This is in line with people's browsing behavior: a user is very likely to click other items that are related to what s/he has clicked just now. Recall that general RNN-based models are able to model this fact, thus they can achieve fairly good performance in session-based recommendation. (3) In some cases, the most important items appear in the beginning or middle of the session (e.g., in session 7974 or 4260). In this situation, we believe that our NARM can perform better than general RNN-based models because the attention mechanism could learn to pay more attention to more important items regardless of its position in one session.

6 CONCLUSION & FUTURE WORK

We have proposed the neural attentive recommendation machine (NARM) with an encoder-decoder architecture to address the session-based recommendation problem. By incorporating an attention mechanism into RNN, our proposed approach can capture both the user's sequential behavior and main purpose in the current session. Based on the sequential behavior feature and the user purpose feature, we have applied NARM to predict a user's next click in the current session. We have conducted extensive experiments on two benchmark datasets and demonstrated that our approach can outperform state-of-the-art methods in terms of different evaluation metrics. Moreover, we have performed an analysis on user click behaviors and found that users' intentions are localized in most sessions, which proves the rationality of our model.

As to future work, more item attributes, such as prices and categories, may enhance the performance of our method in session-based recommendation. Meanwhile, both the nearest neighbor sessions and the importance of different neighbors should give new insights. Finally, the attention mechanism can be used to explore the importance of attributes in the current session.

ACKNOWLEDGMENTS

The authors wish to thank the anonymous reviewers for their helpful comments. This work is supported by the Natural Science Foundation of China (61672322, 61672324), the Natural Science Foundation of Shandong province (2016ZRE27468) and the Fundamental Research Funds of Shandong University.

REFERENCES

[1] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17(6):734–749, 2005.

[2] D. Amodei, R. Anubhai, E. Battenberg, C. Case, J. Casper, B. Catanzaro, J. Chen, M. Chrzanowski, A. Coates, G. Diamos, et al. Deep speech 2: end-to-end speech recognition in english and mandarin. In *Proceedings of the 33rd. International Conference on Machine Learning*, pages 173–182, 2016.

[3] S. Chen, J. L. Moore, D. Turnbull, and T. Joachims. Playlist prediction via metric embedding. In *Proceedings of the 18th. ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 714–722, 2012.

[4] J. Davidson, B. Liebald, J. Liu, P. Nandy, T. Van Vleet, U. Gargi, S. Gupta, Y. He, M. Lambert, B. Livingston, et al. The youtube video recommendation system. In *Proceedings of the 4th. ACM Conference on Recommender Systems*, pages 293–296, 2010.

[5] L. De Vine, G. Zuccon, B. Koopman, L. Sitbon, and P. Bruza. Medical semantic similarity with a neural language model. In *Proceedings of the 23rd. ACM International Conference on Conference on Information and Knowledge Management*, pages 1819–1822, 2014.

[6] A. M. Elkahky, Y. Song, and X. He. A multi-view deep learning approach for cross domain user modeling in recommendation systems. In *Proceedings of the 24th. International Conference on World Wide Web*, pages 278–288, 2015.

[7] A. Graves, A.-r. Mohamed, and G. Hinton. Speech recognition with deep recurrent neural networks. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 6645–6649, 2013.

[8] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[9] X. He and T.-S. Chua. Neural factorization machines for sparse predictive analytics. In *Proceedings of the 40th. International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 355–364, 2017.

[10] X. He, H. Zhang, M.-Y. Kan, and T.-S. Chua. Fast matrix factorization for online recommendation with implicit feedback. In *Proceedings of the 39th. International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 549–558, 2016.

[11] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua. Neural collaborative filtering. In *Proceedings of the 26th. International Conference on World Wide Web*, pages 173–182, 2017.

[12] B. Hidasi, A. Karatzoglou, L. Baltrunas, and D. Tikk. Session-based recommendations with recurrent neural networks. In *Proceedings of the 4th. International Conference on Learning Representations*, 2016.

[13] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A. R. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, and T. N. Sainath. Deep neural networks for acoustic modeling in speech recognition: the shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97, 2012.

[14] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[15] D. Kingma and J. Ba. Adam: a method for stochastic optimization. In *Proceedings of the 4th. International Conference on Learning Representations*, 2015.

[16] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[17] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th. International Conference on Neural Information Processing Systems*, pages 1097–1105, 2012.

[18] P. Li, Z. Wang, W. Lam, Z. Ren, and L. Bing. Salience estimation via variational auto-encoders for multi-document summarization. In *Proceedings of the 31st. AAAI Conference on Artificial Intelligence*, pages 3497–3503, 2017.

[19] P. Li, Z. Wang, Z. Ren, L. Bing, and W. Lam. Neural rating regression with abstractive tips generation for recommendation. In *Proceedings of the 40th. International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 345–354, 2017.

[20] G. Linden, B. Smith, and J. York. Amazon.com recommendations: item-to-item collaborative filtering. *IEEE Internet Computing*, 7(1):76–80, 2003.

[21] Q. Liu, T. Chen, J. Cai, and D. Yu. Enlister: baidu's recommender system for the biggest chinese q&a website. In *Proceedings of the 6th. ACM Conference on Recommender Systems*, pages 285–288, 2012.

[22] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th. International Conference on Neural Information Processing Systems*, pages 3111–3119, 2013.

[23] A. Mild and T. Reutterer. An improved collaborative filtering approach for predicting cross-category purchases based on binary market basket data. *Journal of Retailing and Consumer Services*, 10(3):123–133, 2003.

[24] A. Mnih and G. Hinton. A scalable hierarchical distributed language model. In *Proceedings of the 21st. International Conference on Neural Information Processing Systems*, pages 1081–1088, 2008.

[25] B. Mobasher, H. Dai, T. Luo, and M. Nakagawa. Using sequential and non-sequential patterns in predictive web usage mining tasks. In *Proceedings of the IEEE International Conference on Data Mining*, pages 669–672, 2002.

[26] P. Ren, Z. Chen, Z. Ren, F. Wei, J. Ma, and M. de Rijke. Leveraging contextual sentence relations for extractive summarization using a neural attention model. In *Proceedings of the 40th. International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 95–104, 2017.

[27] Z. Ren, S. Liang, P. Li, S. Wang, and M. de Rijke. Social collaborative viewpoint regression with explainable recommendations. In *Proceedings of the 10th. ACM International Conference on Web Search and Data Mining*, pages 485–494, 2017.

[28] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme. Bpr: bayesian personalized ranking from implicit feedback. In *Proceedings of the 25th. Conference on Uncertainty in Artificial Intelligence*, pages 452–461, 2009.

[29] S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme. Factorizing personalized markov chains for next-basket recommendation. In *Proceedings of the 19th. International Conference on World Wide Web*, pages 811–820, 2010.

[30] O. Rsoy and C. Cardie. Deep recursive neural networks for compositionality in language. In *Proceedings of the 27th. International Conference on Neural Information Processing Systems*, pages 2096–2104, 2014.

[31] R. Salakhutdinov, A. Mnih, and G. Hinton. Restricted boltzmann machines for collaborative filtering. In *Proceedings of the 24th. International Conference on Machine Learning*, pages 791–798, 2007.

[32] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th. International Conference on World Wide Web*, pages 285–295, 2001.

[33] J. B. Schafer, J. Konstan, and J. Riedl. Recommender systems in e-commerce. In *Proceedings of the 1st. ACM Conference on Electronic Commerce*, pages 158–166, 1999.

[34] S. Sedhain, A. K. Menon, S. Sanner, and L. Xie. Autorec: autoencoders meet collaborative filtering. In *Proceedings of the 24th. International Conference on World Wide Web*, pages 111–112, 2015.

[35] L. Shang, Z. Lu, and H. Li. Neural responding machine for short-text conversation. In *Proceedings of the 53rd. Annual Meeting of the Association for Computational Linguistics*, pages 1577–1586, 2015.

[36] G. Shani, D. Heckerman, and R. I. Brafman. An mdp-based recommender system. *Journal of Machine Learning Research*, 6(1):1265–1295, 2005.

[37] R. Socher, C. Y. Lin, A. Y. Ng, and C. D. Manning. Parsing natural scenes and natural language with recursive neural networks. In *Proceedings of the 28th. International Conference on Machine Learning*, pages 129–136, 2011.

[38] H. Song, Z. Ren, S. Liang, P. Li, J. Ma, and M. de Rijke. Summarizing answers in non-factoid community question-answering. In *Proceedings of the 10th. ACM International Conference on Web Search and Data Mining*, pages 405–414, 2017.

[39] X. Su and T. M. Khoshgoftaar. A survey of collaborative filtering techniques. *Advances in Artificial Intelligence*, 2009.

[40] Y. K. Tan, X. Xu, and Y. Liu. Improved recurrent neural networks for session-based recommendations. In *Proceedings of the 1st. Workshop on Deep Learning for Recommender Systems*, pages 17–22, 2016.

[41] P. Wang, J. Guo, Y. Lan, J. Xu, S. Wan, and X. Cheng. Learning hierarchical representation model for nextbasket recommendation. In *Proceedings of the 38th. International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 403–412, 2015.

[42] M. Weimer, A. Karatzoglou, Q. V. Le, and A. Smola. Maximum margin matrix factorization for collaborative ranking. In *Proceedings of the 20th. International Conference on Neural Information Processing Systems*, pages 1–8, 2007.

[43] Y. Wu, C. Dubois, A. X. Zheng, and M. Ester. Collaborative denoising auto-encoders for top-n recommender systems. In *Proceedings of the 9th. ACM International Conference on Web Search and Data Mining*, pages 153–162, 2016.

[44] G. E. Yap, X. L. Li, and P. S. Yu. Effective next-items recommendation via personalized sequential pattern mining. In *Proceedings of the 17th. International Conference on Database Systems for Advanced Applications*, pages 48–64, 2012.

[45] Y. Zhang, H. Dai, C. Xu, J. Feng, T. Wang, J. Bian, B. Wang, and T.-Y. Liu. Sequential click prediction for sponsored search with recurrent neural networks. In *Proceedings of the 28th. AAAI Conference on Artificial Intelligence*, pages 1369–1375, 2014.

[46] A. Zimdars, D. M. Chickering, and C. Meek. Using temporal data for making recommendations. In *Proceedings of the 17th. Conference on Uncertainty in Artificial Intelligence*, pages 580–588, 2001.