A Collaborative Session-based Recommendation Approach with Parallel Memory Modules

Meirui Wang
Shandong University
wmeirui@gmail.com

Pengjie Ren
University of Amsterdam
p.ren@uva.nl

Lei Mei
Shandong University
lei.mei@outlook.com

Zhumin Chen
Shandong University
chenzhumin@sdu.edu.cn

Jun Ma
Shandong University
majun@sdu.edu.cn

Maarten de Rijke
University of Amsterdam
derijke@uva.nl

ABSTRACT

Session-based recommendation is the task of predicting the next item to recommend when the only available information consists of anonymous behavior sequences. Previous methods for session-based recommendation focus mostly on the current session, ignoring collaborative information in so-called *neighborhood sessions*, sessions that have been generated previously by other users and reflect similar user intents as the current session. We hypothesize that the collaborative information contained in such neighborhood sessions may help to improve recommendation performance for the current session.

We propose a Collaborative Session-based Recommendation Machine (CSRM), a novel hybrid framework to apply collaborative neighborhood information to session-based recommendations. CSRM consists of two parallel modules: an Inner Memory Encoder (IME) and an Outer Memory Encoder (OME). The IME models a user's own information in the current session with the help of Recurrent Neural Networks (RNNs) and an attention mechanism. The OME exploits collaborative information to better predict the intent of current sessions by investigating neighborhood sessions. Then, a fusion gating mechanism is used to selectively combine information from the IME and OME so as to obtain the final representation of the current session. Finally, CSRM obtains a recommendation score for each candidate item by computing a bilinear match with the final representation.

Experimental results on three public datasets demonstrate the effectiveness of CSRM compared to state-of-the-art session-based recommender systems. Our analysis of CSRM's recommendation process shows when and how collaborative neighborhood information and the fusion gating mechanism positively impact the performance of session-based recommendations.

CCS CONCEPTS

• **Information systems → Recommender systems**.

SIGIR '19, July 21–25, 2019, Paris, France

ACM ISBN 978-1-4503-6172-9/19/07…$15.00
https://doi.org/10.1145/3331184.3331210

KEYWORDS

Collaborative modeling, Session-based recommendation, Memory network

ACM Reference Format:
Meirui Wang, Pengjie Ren, Lei Mei, Zhumin Chen, Jun Ma, and Maarten de Rijke. 2019. A Collaborative Session-based Recommendation Approach with Parallel Memory Modules. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '19), July 21–25, 2019, Paris, France.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3331184.3331210

1 INTRODUCTION

Conventional recommendation methods are often based on explicit user-item preferences (e.g., ratings), for which information about users and items is essential [21, 22, 30]. For example, content-based methods generate lists of recommendations by evaluating similarities between user profiles and item features [26, 42]. However, in many realistic scenarios users are not logged in and no past interactions of users are known: no explicit preferences but only positive observations (e.g., clicks) are available. Under these circumstances, conventional recommendation methods [21, 24, 36] do not perform well. To address this issue, *session-based recommendation* has been proposed to generate recommendations based only on anonymous behavior sessions. Given the short-term history of a user's behavior in the current session, session-based recommendation aims to predict the next item that she/he might be interested in.

Early studies on session-based recommendation are mainly based on item-to-item recommendation [24, 36] and produce predictions relying on computing similarities (e.g., based on co-occurrence) between items within sessions, ignoring sequential information of click sequences in an ongoing session. Later work investigates the use of Markov chains to exploit sequential behavior data and predict a user's next action based on their last action [38, 46]. However, this work only models local sequential behavior between adjacent items. A major issue for Markov chain-based methods is that the state space quickly becomes unmanageable when trying to include all sequences of potential user selections over all items [23].

Recent studies use deep learning (e.g., RNNs) for session-based recommendations, so that information of the entire session can be considered [15, 23, 31]. For example, Hidasi et al. [15] use Gated Recurrent Units (GRUs) to model sequences of actions in a session. Li et al. [23] achieve further improvements by introducing an attention mechanism to capture the user's main purpose in the

current session. Although these RNN-based methods show noticeable improvements over traditional recommendation approaches, they usually have limited short-term memory and have difficulty in performing memorization. Memory networks [44] have been introduced to recommender systems motivated by the advantage of having a long-term memory module. Chen et al. [2] improve sequential recommendations by first introducing user memory networks to make use of a user's historical records and explicitly capture item- and feature-level sequential patterns. Huang et al. [16] propose a knowledge enhanced sequential recommender that integrates RNN-based networks with Key-Value Memory Networks (KV-MNs) [29].

Existing memory network-based methods for session-based recommendations have achieved encouraging results. But they only exploit a user's own information while ignoring the potential of collaborative information in so-called *neighborhood sessions*, which are sessions generated by any user (not necessarily the same user of the current session) that display similar behavior patterns and reflect similar user intents as the current session. Consider, for example, the session [$Phone_1$, $Phone_2$, $Phone_3$] and a later session [$Phone_1$, $Phone_3$, $Phone_4$]. While interacting with the recommender system, the users generating these sessions both click on some phones, presumably with similar specifications to make a comparison. The two users may have similar intents to find a suitable phone, and, hence, $Phone_2$ might be of interest to the user generating the second sequence. We hypothesize that neighborhood sessions may help to improve recommendations in the current session.

We propose a novel neural network framework, namely *Collaborative Session-based Recommendation Machine* (CSRM), for session-based recommendations that consists of two parallel modules: an *Inner Memory Encoder* (IME) and an *Outer Memory Encoder* (OME). The IME models the information contained within the current session with the help of an RNN's inherent dynamic memory. It learns a unified representation combining two encoding schemes: a global encoder and a local encoder. The OME exploits collaborative-filtering techniques to better predict the intent of a current session by employing an external memory module to investigate neighborhood sessions. The OME contains an associative addressing mechanism to automatically identify neighborhood sessions. Finally, CSRM introduces a fusion gating mechanism to combine the representations resulting from the IME and OME, and computes a recommendation score for each candidate item based on the fused representation.

We carry out extensive experiments on three benchmark datasets. The results show that CSRM outperforms state-of-the-art session-based recommendation baselines on all three datasets in terms of Recall and MRR. We conduct further experiments to analyze the IME and the OME in depth so as to explore CSRM's recommendation process and determine how collaborative neighborhood information and the fusion gating mechanism influence the performance of session-based recommendations.

We summarize our contributions as follows.

- To the best of our knowledge, we are the first to consider collaborative modeling in session-based recommendations with an end-to-end neural model.
- We propose a novel CSRM model that integrates an IME and OME to take information from the current session and neighborhood sessions into account for session-based recommendations.
- We introduce a fusion gating mechanism to selectively combine information from the current and neighborhood sessions for better recommendations.
- We carry out extensive experiments on three benchmark datasets. CSRM significantly outperforms state-of-the-art models in terms of Recall and MRR on the session-based recommendation task.

2 RELATED WORK

We survey related work on recommender systems in two areas: collaborative filtering and memory augmented neural networks.

2.1 Collaborative filtering

Collaborative filtering is a widely used recommendation method [3, 4, 20]. It identifies user preferences by modeling user-item interactions and recommends items to users based on the assumption that people who have similar preferences tend to make similar choices [7, 28]. Previous work on collaborative filtering can be classified into two groups: KNN-based methods [9, 13] and model-based methods [22, 30, 34, 37, 43].

KNN-based methods use predefined similarity functions to find similar users or items to promote recommendations. They can be further classified into *user-based* KNN methods [1] and *item-based* KNN methods [36]. User-based KNN methods compute similarities between users and recommend items that similar users might like [1]. Jin et al. [18] present an algorithm to learn weights for different items when identifying the similarity between two users. Liu et al. [25] propose a heuristic similarity measure approach to improve collaborative filtering that combines context information of user ratings and the preference of user behavior. Item-based collaborative filtering approaches calculate similarities between items and use these scores to predict ratings for user-item pairs [24, 36]. Sarwar et al. [36] propose to use the same correlation-based and cosine-based techniques to compute similarities between items. Deshpande and Karypis [6] extend item-to-item similarities to a notion of similarity between all consumed items of a user and a candidate item for top-N recommendation.

Given the user-item rating matrix, model-based methods map users and items into a shared latent factor space to characterize items and users, and generate predictions via the inner product of user and item latent factors. Example methods include matrix factorization [21], Singular Value Decomposition (SVD) and SVD++ [20]. Wu et al. [45] generalize collaborative filtering models by integrating a user-specific bias into an auto-encoder. He et al. [12] model user-item interactions by using deep learning instead of an inner product.

Although these methods have achieved promising results by exploring collaborative filtering, they all have limitations. On the one hand, KNN-based methods cannot identify sequential signals. On the other hand, model-based methods cannot be used for session-based recommendations when user profiles are not available. Recently, Jannach and Ludewig [17] have proposed to combine a GRU model and a co-occurrence-based KNN model to explore collaborative information in session-based recommendations. The differences between their method and our work are at least two-fold. First, our model is an end-to-end memory network, while they use a simple weighting scheme to combine the scores of a GRU model and a KNN

model with hand-crafted hyperparameters. Second, the similarities used to find k-nearest neighbors are learned in our model, while Jannach and Ludewig [17] use a heuristic co-occurrence-based similarity measure.

2.2 Memory augmented neural networks

Memory Augmented Neural Networks (MANNs) [40, 44] generally consist of two components: a memory that stores long-term historical information [10, 35] and a controller that performs read/write operations on the memory [2, 11, 40].

Recently, memory networks for recommender systems have received considerable attention because of their state-of-the-art performance [2, 8]. Chen et al. [2] are the first to propose to integrate matrix factorization with a MANN for sequential recommendations. They propose Recommendation with User Memory networks (RUMs) with two variants: item-level RUMs and feature-level RUMs [2]. An item-level RUM regards each item as a unit and directly stores the item embeddings in the memory matrix. Feature-level RUMs store the embeddings of user preferences on different latent features in the memory matrix. Huang et al. [16] propose a knowledge enhanced sequential recommender that incorporates knowledge base information to capture attribute-level user preferences. Ma et al. [27] combine MANNs with a cross-attention mechanism to perform the mention recommendation task for multimodal tweets, where they use a MANN to store the image and tweet history interests. Although these methods have achieved promising results, they all ignore collaborative neighborhood information.

The work most similar to ours concerns the Collaborative Memory Network (CMN) [7]. It unifies two classes of collaborative filtering models: matrix factorization and neighborhood-based methods. CMN exploits three memory states to model user-item interactions, including an internal user-specific memory, an item-specific memory, and a collective neighborhood state. The differences between our work and CMN are at least three-fold. First, CMN cannot be directly applied to session-based recommendations because there is no user information available. Although we can make it work with some changes, we find it is not effective for session-based recommendations (see below). Second, CMN performs collaborative filtering by finding similar user-item interaction patterns. In contrast, our model leverages collaborative information by exploring neighborhood sessions. Third, CMN simply combines the collaborative features and the user-item latent features to predict rating scores. Instead, we introduce a fusion gating mechanism to learn to selectively combine different sources of features.

3 METHOD

Let $\mathcal{I}$ denote the set of all items. We use $X_t = [x_1, x_2, \ldots, x_\tau, \ldots, x_n]$ to represent a session at timestamp t ($t \geq 1$) in the recommendation process, where each $x_\tau \in \mathcal{I}$ is an item that the user has interacted with during the session, e.g., listening to a song, watching a video. Given a session X, the task of a session-based recommender system is to predict the next item that users might interact with. Formally, given a current session $X_t = [x_1, x_2, \ldots, x_\tau, \ldots, x_n]$ as defined above, the goal is to predict x_{n+1} by recommending top-N items ($1 \leq N \leq |\mathcal{I}|$) from all items $\mathcal{I}$ that might be interesting to the user.

We propose a Collaborative Session-based Recommendation Machine (CSRM) framework to generate session-based recommendations. The key idea behind CSRM is to leverage information from neighborhood sessions to improve the recommendation performance of the current session. Neighborhood sessions are sessions that display similar behavior patterns as the current session. Specifically, we store the last m sessions in an outer memory as the potential neighborhood sessions. Given the current session, CSRM automatically finds neighborhood sessions from the memory and extracts useful features to promote the recommendation for this session.

As illustrated in Figure 1, CSRM consists of three main components: an *Inner Memory Encoder* (IME), an *Outer Memory Encoder* (OME), and a recommendation decoder. A fusion gating mechanism is used to control the flow of information between the IME and OME components. First, the IME converts the input session into two high-dimensional hidden representations: one is meant to summarize the whole session and the other is meant to select relatively important items in the current session when sessions contain items that are clicked accidentally or due to curiosity [23]. These two representations are concatenated into a unified session representation. Second, the OME encodes collaborative neighborhood information of the current session into a collaborative session representation with the help of an outer memory network. This process can be viewed as a session-based nearest neighbor method that places higher weights on specific subsets of sessions that have similar behavior patterns as the current session. Finally, the outputs of the IME and OME form the input for the recommendation decoder where a fusion gating mechanism is used to selectively combine information from the IME and OME for recommendation. The output is a recommendation score for each candidate item based on a bi-linear decoding scheme.

Next, we introduce each part in detail.

3.1 Inner memory encoder

The IME tries to encode useful information contained in the current session. It consists of two components: the global encoder and the local encoder following [23]. The former is used to model the sequential behavior in the whole session. The latter is used to pay attention to specific behavior, which is reflected by relatively important items in the current session [23]. We use a GRU as the global encoder because GRUs have shown better performance than a Long Short-Term Memory (LSTM) for session-based recommendations [15]. The activation of the GRU is a linear interpolation between the previous hidden state $\boldsymbol{h}_{\tau-1}$ and the candidate hidden state $\widehat{\boldsymbol{h}}_\tau$:

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

where the update gate $\boldsymbol{z}_\tau$ is determined by

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

and $\boldsymbol{x}_\tau$ is the item embedding of x_τ, and $\boldsymbol{W}_z$ and $\boldsymbol{U}_z$ are weight matrices. The candidate activation function $\widehat{\boldsymbol{h}}_\tau$ is computed as follows:

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

where the reset gate $\boldsymbol{r}_\tau$ is computed as

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

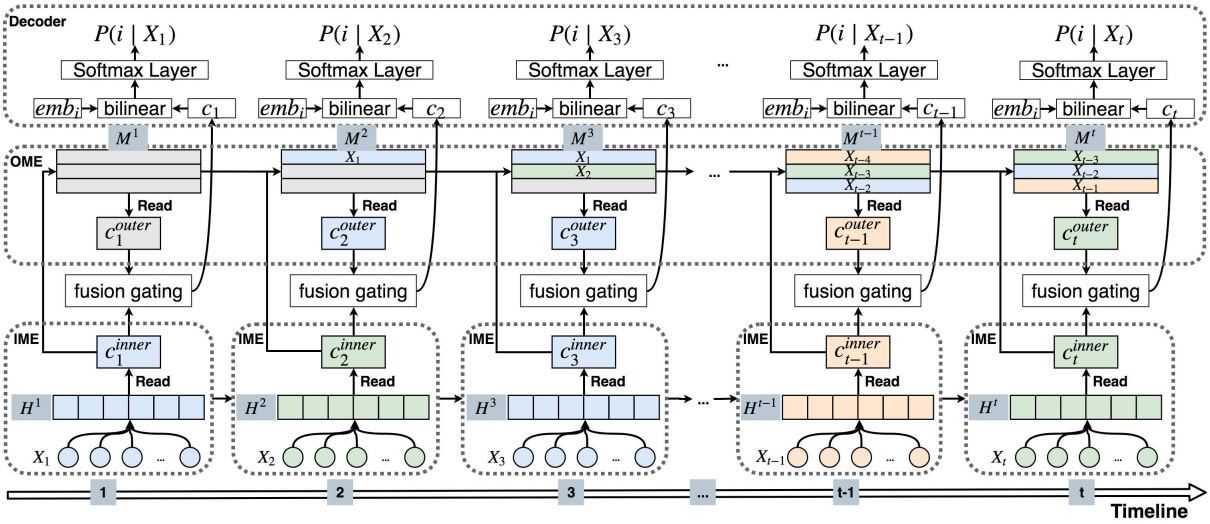

Figure 1: Overview of the *Collaborative Session-based Recommendation Machine* (CSRM). CSRM is composed of three main components: an *Inner Memory Encoder* (IME), an *Outer Memory Encoder* (OME) and a recommendation decoder. To reflect the collaborative process, we use different colors to indicate different session intents, where the same colors indicate similar session intents.

where $\boldsymbol{W}$, $\boldsymbol{U}$, $\boldsymbol{W}_r$ and $\boldsymbol{U}_r$ are weight matrices. Finally, we use the final hidden state $\boldsymbol{h}_n$ as the representation of the current session's sequential behavior:

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

Although this global encoder considers behavior reflected by the whole sequence, it is difficult to accurately capture the current session's behavior pattern due to the noise item [23]. To this end, we use another GRU with an item-level attention mechanism as the local encoder. After the global encoder, the current session X_t is encoded into the inner memory matrix $\mathbf{H}^t = [\boldsymbol{h}_1, \boldsymbol{h}_2, \ldots, \boldsymbol{h}_\tau, \ldots, \boldsymbol{h}_n]$. The local encoder dynamically reads from the inner memory network relying solely on the session's own information. The attention mechanism pays close attention to the current session's intent by emphasizing some specific behavior and ignoring other. We assign higher weights to relatively more important items.

For the current session X_t, the local encoder first finds out the contribution of each clicked item to session intent with the help of attention weights α_{nj}. Specifically, the weighting factor α_{nj} models the relation between the final hidden state vector $\boldsymbol{h}_n$ and the representation of the previous clicked item $\boldsymbol{h}_j$ by computing their similarity:

$$\alpha_{nj} = \boldsymbol{v}^{\mathrm{T}} \sigma(\boldsymbol{A}_1 \boldsymbol{h}_n + \boldsymbol{A}_2 \boldsymbol{h}_j), \tag{6}$$

where σ is an activation function such as the sigmoid function $\sigma(x) = 1/\big(1 + \exp(-x)\big)$, and matrices $\boldsymbol{A}_1$ and $\boldsymbol{A}_2$ transform $\boldsymbol{h}_n$ and $\boldsymbol{h}_j$ into latent spaces, respectively. Then we can capture the session's intent in the current session by adaptively focussing (or placing higher weights) on more important items:

$$\boldsymbol{c}_t^{\mathrm{l}} = \sum_{j=1}^{n} \alpha_{nj} \boldsymbol{h}_j. \tag{7}$$

In summary, the current session can be converted into two fixed-length vectors $\boldsymbol{c}_t^{\mathrm{g}}$ and $\boldsymbol{c}_t^{\mathrm{l}}$, respectively, by reading from the inner memory network $\mathbf{H}^t$. Then we concatenate $\boldsymbol{c}_t^{\mathrm{l}}$ and $\boldsymbol{c}_t^{\mathrm{g}}$ into a unified representation $\boldsymbol{c}_t^{\mathrm{inner}}$ as the final session representation:

$$\boldsymbol{c}_t^{\mathrm{inner}} = [\boldsymbol{c}_t^{\mathrm{l}}; \boldsymbol{c}_t^{\mathrm{g}}] = \left[\sum_{j=1}^{n} \alpha_{nj} \boldsymbol{h}_j; \boldsymbol{h}_n \right]. \tag{8}$$

3.2 Outer memory encoder

The global encoder and the local encoder in the IME only exploit information contained in the session on which it operates, which ignores the importance of collaborative information in neighboring sessions. To address this omission, we propose an Outer Memory Encoder (OME). The outer memory matrix $\mathbf{M}$ stores the representations of the m last sessions, in chronological order. The OME dynamically selects previous neighborhood sessions from $\mathbf{M}$ that share more similar behavior patterns with the current session. These neighborhood sessions are used as auxiliary information to help understand the current session. The OME uses the following reading and writing operations to access $\mathbf{M}$.

Reading operation. Intuitively, reading $\mathbf{M}$ can be considered as a session-based nearest neighborhood method that selectively weights the retrieved neighborhoods related to the current session.

Given the current session X_t, we first want to determine its k most similar previous sessions in $\mathbf{M}$. Specifically, we compute the cosine similarity between $\boldsymbol{c}_t^{\mathrm{l}}$ of the current session and each $\boldsymbol{m}_i$ of previous sessions stored in the memory matrix $\mathbf{M}$:

$$\mathrm{sim}(\boldsymbol{c}_t^{\mathrm{l}}, \boldsymbol{m}_i) = \frac{\boldsymbol{c}_t^{\mathrm{l}} \boldsymbol{m}_i}{\|\boldsymbol{c}_t^{\mathrm{l}}\| \times \|\boldsymbol{m}_i\|} \quad \forall \boldsymbol{m}_i \in \mathbf{M}. \tag{9}$$

According to the k largest similarity scores $[\mathrm{sim}_1^t, \mathrm{sim}_2^t, \ldots, \mathrm{sim}_{k-1}^t, \mathrm{sim}_k^t]$, we create a subsample of k sessions $\mathbf{M}^t = [\boldsymbol{m}_1^t, \boldsymbol{m}_2^t, \ldots, \boldsymbol{m}_{k-1}^t, \boldsymbol{m}_k^t]$ from $\mathbf{M}$ as the k nearest neighbors for the current session.

Then, the k largest similarity values are processed by a softmax function to obtain reading weights:

$$w_{tp} = \frac{\exp(\beta \mathrm{sim}_p^t)}{\sum_j \exp(\beta \mathrm{sim}_j^t)} \quad \forall p = 1, 2, \ldots, k, \tag{10}$$

where β is the strength parameter. These weights also reflect each neighbor's unique impact on the current session, which allows the model to place higher weights on more similar sessions in the neighborhood.

Finally, we derive the output of the OME module, $\boldsymbol{c}_t^{\text{outer}}$, by accessing other sessions' representations in the neighborhood memory network according to their influence on the current session:

$$\boldsymbol{c}_t^{\text{outer}} = \sum_{p=1}^{k} w_{tp} \boldsymbol{m}_p^t. \tag{11}$$

Writing operation. At the beginning of each epoch of our experiments, the outer memory matrix is empty. We adopt a first-in-first-out mechanism to update the memory matrix, which always stores the latest m sessions. When writing the outer memory, the earliest session is removed from the memory and the new one is added to the queue. Note that when the memory matrix is not full, the session is directly added without removing any existing sessions.

3.3 Recommendation decoder

The recommendation decoder evaluates the probability of clicking the next item based on the outputs of the IME and OME. As shown in Figure 1, to selectively combine information from the IME and OME, we construct the final session representation with a fusion gating mechanism that balances the importance of a user's own information and the collaborative neighborhood information:

$$\boldsymbol{c}_t = \boldsymbol{f}_t \boldsymbol{c}_t^{\text{inner}} + (\mathbf{1} - \boldsymbol{f}_t) \boldsymbol{c}_t^{\text{outer}}, \tag{12}$$

where the gate $\boldsymbol{f}_t$ is given by

$$\boldsymbol{f}_t = \sigma(\boldsymbol{W}_l \boldsymbol{c}_t^{\text{l}} + \boldsymbol{W}_g \boldsymbol{c}_t^{\text{g}} + \boldsymbol{W}_o \boldsymbol{c}_t^{\text{outer}}). \tag{13}$$

Then we utilize a bi-linear decoding scheme followed by a softmax to compute recommendation scores as suggested in Figure 1. Assuming that $\boldsymbol{emb}_i$ is the representation of a single candidate item, we obtain the final recommendation probability for each candidate item based on the final representation $\boldsymbol{c}_t$ of the current session X_t:

$$P(i \mid X_t) = \mathrm{softmax}(\boldsymbol{emb}_i^{\mathrm{T}} \boldsymbol{B}\, \boldsymbol{c}_t), \tag{14}$$

where $\boldsymbol{B} \in \mathbb{R}^{e \times D}$, e is the dimension of each item embedding, and D is the dimension of the final session representation $\boldsymbol{c}_t$.

3.4 Objective function

Our goal is to maximize the prediction probability of the actual item given the current session. Therefore, we adopt the cross-entropy loss function:

$$L = -\frac{1}{|\mathbb{X}|} \sum_{X \in \mathbb{X}} \sum_{i \in I} [\, 1(i, X) \log(P(i \mid X)) + (1 - 1(i, X)) \log(1 - P(i \mid X))], \tag{15}$$

where $\mathbb{X}$ is the set of all sessions in the training set; $P(i \mid X)$ is the predicted probability value for item i given the session X; $1(i, X)$ is the ground truth. $1(i, X) = 1$ if item i is contained in the top-N recommendations of session X, and $1(i, X) = 0$ otherwise. In the learning process, we adopt a Back-Propagation Through Time (BPTT) method to optimize CSRM.

4 EXPERIMENTAL SETUP

4.1 Research questions

We aim to answer the following research questions:

(RQ1) What is the performance of CSRM in session-based recommendation tasks? Does it outperform the state-of-the-art methods in terms of Recall and MRR? (See Section 5.1.)
(RQ2) How well does CSRM perform with different encoders instead of combining the IME and OME? (See Section 5.2.)
(RQ3) How well does CSRM perform with different aggregation operations instead of a fusion gating mechanism? (See Section 5.3.)
(RQ4) How well does CSRM perform with different numbers of neighbors in the OME? (See Section 5.4.)
(RQ5) How does the session length influence the session-based recommendation performance? (See Section 5.5.)
(RQ6) How does the collaborative neighborhood information influence the session-based recommendation performance? (See Section 5.6.)

4.2 Datasets

To answer our research questions we conduct experiments on two publicly available datasets: YOOCHOOSE[1] and LastFM.[2] We preprocess the data with the following steps. First, we make sure that all sessions are arranged in chronological order. Second, we filter out items that only appear in the test set. Third, we perform data augmentation to account for temporal shifts in sessions [41]. The statistics of the datasets (i.e., YOOCHOOSE 1/64, YOOCHOOSE 1/4 and LastFM) are summarized in Table 1.

- **YOOCHOOSE**: The YOOCHOOSE dataset is a publicly available dataset released by the RecSys Challenge 2015. It contains six months of click-streams in an e-commerce website. We use the sessions of the last day for testing and all others for training. We follow Tan et al. [41] and filter out sessions of length 1 and items that appear less than 5 times. Tan et al. [41] find that more recent fractions of YOOCHOOSE are enough for the task and increasing the amount of data does not further improve the performance. We use the most recent 1/64 and 1/4 fraction of the training sessions referred to as the YOOCHOOSE 1/64 and YOOCHOOSE 1/4 dataset, respectively. After preprocessing, the YOOCHOOSE 1/64 dataset contains 637,226 clicks on 17,702 items, which are randomly divided into 124,279 training sessions, 12,309 validation sessions, and 15,236 test sessions. The YOOCHOOSE 1/4 version contains 8,253,617 clicks on 30,672 items, which are randomly divided into 1,988,471 training sessions, 12,371 validation sessions, and 15,317 test sessions.
- **LastFM**: LastFM is a music recommendation dataset and contains (user, time, artist, song) tuples collected from 2004 to 2009

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

Table 1: Statistics of the datasets used in our experiments.

Datasets	all the clicks	training sessions	validation sessions	test sessions	all items
YOOCHOOSE 1/64	637,226	124,279	12,309	15,236	17,702
YOOCHOOSE 1/4	8,253,617	1,988,471	12,371	15,317	30,672
LastFM	3,804,922	269,847	5,996	5,771	39,163

obtained through the LastFM API. To simplify the dataset, we use the dataset for music artist recommendation.

In our experiments, we select the top 40,000 most popular artists as the item set. Then we manually partition the listening history into sessions through a 8-hours time interval. That is, if two consecutive actions happen within the time interval, they belong to the same session. Otherwise, we think they belong to different sessions. This time interval makes each session not only contain enough interactions, but also independent of each other to some extent. Finally, after we filter out sessions that are shorter than 2 and longer than 50 items, 281,614 sessions and 39,163 items remain. This dataset is referred to as LastFM. After preprocessing, the LastFM dataset contains 3,804,922 clicks on 39,163 items, which are randomly divided into 269,847 training sessions, 5,996 validation sessions, and 5,771 test sessions.

4.3 Baselines

We consider three groups of baseline methods for comparison: traditional, RNN-based, and memory network-based.

The commonly used traditional methods in previous studies on session-based recommendations [15, 23] are the following:

- **POP**: POP always recommends the most popular items in the training set. It is a very simple baseline, but it can perform well in certain domains [12].
- **S-POP**: S-POP recommends the most popular items of the current session. Ties are broken using global popularity values [15].
- **Item-KNN**: In this method, items similar to the actual item are recommended. Similarity is defined as the number of co-occurrences 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 included to avoid coincidental high similarities between rarely visited items [5].
- **BPR-MF**: BPR-MF [32] optimizes a pairwise ranking objective function by using stochastic gradient descent. As with previous studies [15, 23], we apply this method to session-based recommendations by averaging the latent factors of items that occurred in the session so far as its representation.
- **FPMC**: Factorizing Personalized Markov Chain (FPMC) [33] combines a Markov chain model and matrix factorization for the next-basket recommendation task. In order to make it work on session-based recommendations, we ignore the user latent representations when computing recommendation scores.

The RNN-based methods used for comparison are the following:

- **GRU-Rec**: We denote the model proposed in [14, 15] as GRU-Rec; it uses a session-parallel mini-batch training process and also employs ranking-based loss functions for learning the model.
- **RNN-KNN**: RNN-KNN [17] combines a heuristics-based nearest neighbor scheme and a GRU with hand-crafted weighting parameters for session-based recommendations, which demonstrates further performance gains over GRU-Rec. RNN-KNN uses item co-occurrences to determine the k-nearest neighbor sessions.
- **Improved GRU-Rec**: We denote the model proposed in [41] as Improved GRU-Rec. Improved GRU-Rec adopts two techniques: data augmentation and a method to account for shifts in the input data distribution to improve the performance of GRU-Rec.
- **NARM**: The Neural Attentive Recommendation Machine (NARM) [23] is an improved encoder-decoder architecture used for session-based recommendations. It improves over Improved GRU-Rec by incorporating an attention mechanism into RNN.

Finally, the memory network-based methods used for comparison are:

- **RUM**: RUM [2] has two specific implementations. The item-level RUM (RUM-I) directly stores item embeddings in the memory matrix, while the feature-level RUM (RUM-F) stores embeddings of user preferences on different latent features. We apply RUM-I to the session-based recommendation task by representing a new session with sequential latent factors of items through RNN-based methods. However, RUM-F still cannot be applied to session-based recommendations because it is hard to get a new session's preference on a specific feature in advance.
- **CMN**: CMN [7] exploits memory networks to address collaborative filtering with implicit feedback. We make this method work on session-based recommendations by representing a new session with the average latent representations of items occurring in the session when making predictions. CMN needs to evaluate a score for each candidate item one by one during testing, which is time consuming. Instead of evaluating all items, we only sample 1,000 negative items to speed up the process.

4.4 Implementation details

We implement CSRM with Tensorflow[3] and carry out experiments on a GeForce GTX TitanX GPU.[4] To alleviate overfitting, we employ two dropout layers [39]: the first dropout layer is between the item embedding layer and the GRU layer with 25% dropout, the second one is between the final representation layer and the bilinear decoding layer with 50% dropout.

During training, we randomly initialize model parameters with a Gaussian distribution (with a mean of 0 and standard deviation of 0.01), optimizing the model with mini-batch Adam [19]. For the hyper-parameters of the Adam optimizer, we set two momentum parameters $\beta 1 = 0.9$ and $\beta 2 = 0.999$, respectively, and $\epsilon = 10^{-8}$. The learning rate is determined by grid search in the range of [0.001, 0.0005, 0.0001] and the batch size is empirically set as 512. The embedding dimension of items and hidden units of the GRU are determined by grid search in the range of [50, 100, 150]. We vary the number of nearest neighbors in the OME from [128, 256,

[3]https://www.tensorflow.org

[4]Source code available at: https://github.com/wmeirui/CSRM_SIGIR2019

Table 2: Performance comparison of CSRM with baseline methods on three datasets.

Method		YOOCHOOSE 1/64		YOOCHOOSE 1/4		LastFM	
		Recall@20(%)	MRR@20(%)	Recall@20(%)	MRR@20(%)	Recall@20(%)	MRR@20(%)
Traditional	POP	4.51	0.72	1.34	0.31	4.43	1.15
	S-POP	29.30	18.07	27.10	17.77	22.38	8.73
	Item-KNN	52.13	21.44	51.52	21.21	11.59	4.19
	BPR-MF	29.99	8.99	3.76	1.77	13.38	5.73
	FPMC	57.01	21.17	61.14	23.84	24.08	8.23
RNN-based	GRU-Rec	66.70	28.50	65.66	28.24	21.42	8.21
	RNN-KNN	68.66	28.64	68.02	28.38	24.64	9.33
	Improved GRU-Rec	69.44	30.18	70.73	30.26	24.34	8.69
	NARM	70.13	29.34	70.96	30.47	25.64	9.18
Memory network-based	RUM-I	27.45	8.46	30.16	8.64	9.55	5.06
	CMN†	26.89	4.04	–	–	22.59	6.44
	CSRM‡	**71.45***	**30.36***	**73.01***	**31.42***	**27.55***	**9.71***

† On the YOOCHOOSE 1/4 dataset, we do not have enough memory to initialize CMN, so we cannot report the results of CMN on this dataset.

‡ The superscript * indicates that CSRM significantly outperforms the best baseline NARM.

512] to study its effects. All hyper-parameters are tuned according to the validation set, the size of which is a fraction of the size of the training data [23].

4.5 Evaluation metrics

Our task is to predict what a user will click next given their current session. Recommendation systems generate a recommendation list that usually contains N items sorted by predicted scores for each session at each time stamp. The actual item a user picks next should be contained in the list of recommendations. Therefore, we use the following metrics for evaluating the top-N recommendations:

- **Recall@20**: The primary evaluation metric is Recall@20, the proportion of cases when the groundtruth item is ranked amongst the top-20 items. It does not consider the actual rank of the item.
- **MRR@20**: We also use MRR@20 (Mean Reciprocal Rank), the average of reciprocal ranks of the desired 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 (e.g., the lower ranked items are only visible after scrolling).

For significance testing we use a paired t-test with $p < 0.05$.

5 RESULTS AND ANALYSIS

5.1 Performance comparison (RQ1)

Table 2 illustrates the performance of CSRM and the baseline methods on three datasets. It shows that CSRM consistently achieves the best performance in terms of both Recall@20 and MRR@20 metrics on all datasets. From the table, we have four main observations.

(1) Among the traditional methods, Item-KNN achieves significant improvements over POP, S-POP and BPR-MF in most cases. This means that KNN-based collaborative filtering methods can help to improve session-based recommendations. As for FPMC, considering that the key difference between BPR-MF and FPMC is that the latter models a user's historical records in a sequential manner, the observation that FPMC gets better results than BPR-MF confirms that sequential information contributes to recommendation performance.

(2) We observe that the five RNN-based methods (GRU-Rec, Improved GRU-Rec, RNN-KNN, NARM, and CSRM) outperform traditional methods. This indicates that RNN-based models are good at dealing with sequential information in sessions.

(3) CSRM significantly outperforms all RNN-based baseline methods. Generally, CSRM obtains improvements over the best baseline NARM of 1.88%, 2.89%, and 7.45% in Recall@20 on the three datasets, respectively. With regard to MRR@20, the relative improvements over the best baseline NARM are 3.48%, 3.12%, and 5.77%, respectively. Although both RNN-KNN and CSRM take collaborative filtering information into account, we notice that CSRM achieves consistent improvements over RNN-KNN. The reason is that RNN-KNN combines an RNN with a co-occurrence-based KNN through hand-crafted hyperparameters, which lacks the kind of nonlinear interaction that is able to capture more complex relations. These observations confirm that leveraging collaborative neighborhood information with memory networks leads to substantial performance improvements in session-based recommendations.

(4) As to the MANN-based methods, we find that CSRM outperforms RUM and CMN on all datasets. This is because there is no user information available in session-based recommendations and RUM and CMN are simply not applicable and effective in this case.

5.2 Influence of encoders (RQ2)

To further illustrate the effect of collaborative information and memory networks, we compare the performance of CSRM and two variants of CSRM. CSRM$_{ime}$ refers to CSRM without the OME that uses an RNN's inherent internal memory to model the sequential behavior in the current session. It actually amounts to the NARM model because CSRM further considers collaborative neighborhood

Table 3: Performance comparison of CSRM with different encoders.

Model	YOOCHOOSE 1/64		YOOCHOOSE 1/4		LastFM	
	Recall@20(%)	MRR@20(%)	Recall@20(%)	MRR@20(%)	Recall@20(%)	MRR@20(%)
CSRM$_{ime}$	70.13	29.34	70.96	30.47	25.64	9.18
CSRM$_{ome}$	65.91	24.32	65.97	24.12	21.37	6.92
CSRM	**71.45**	**30.36**	**73.01**	**31.42**	**27.55**	**9.71**

* CSRM further considers collaborative neighborhood information on the basis of NARM, so without the OME part, CSRM$_{ime}$ is actually NARM.

Table 4: Performance comparison of CSRM with different aggregation operations.

Method	YOOCHOOSE 1/64		YOOCHOOSE 1/4		LastFM	
	Recall@20(%)	MRR@20(%)	Recall@20(%)	MRR@20(%)	Recall@20(%)	MRR@20(%)
Max pooling	70.67	29.79	72.23	30.67	26.94	9.04
Average pooling	70.64	29.94	72.89	31.38	27.40	9.49
Concatenation	71.26	30.24	72.58	31.21	27.40	9.47
Fusion gating	**71.45**	**30.36**	**73.01**	**31.42**	**27.55**	**9.71**

Table 5: Performance comparison of CSRM with different numbers of neighbors k.

k	YOOCHOOSE 1/64		YOOCHOOSE 1/4		LastFM	
	Recall@20(%)	MRR@20(%)	Recall@20(%)	MRR@20(%)	Recall@20(%)	MRR@20(%)
$k = 128$	71.23	30.36	72.65	**31.43**	27.47	**9.76**
$k = 256$	71.35	**30.39**	72.76	31.34	**27.56**	9.65
$k = 512$	**71.45**	30.36	**73.01**	31.42	27.55	9.71

information on the basis of NARM. In this section, we use the result of NARM to represent CSRM$_{ime}$. CSRM$_{ome}$ refers to CSRM without IME that uses an external memory to encode collaborative neighborhood information. The empirical results on three datasets are summarized in Table 3.

First, we find that CSRM$_{ime}$ outperforms CSRM$_{ome}$. This demonstrates that a session's own sequential information is more important for the session-based recommendation task. Second, CSRM achieves better performance than CSRM$_{ime}$ and CSRM$_{ome}$ in terms of both metrics on three datasets. This confirms the usefulness of combining a user's own information and collaborative neighborhood information for better recommendations. Take the YOOCHOOSE 1/64 dataset as an example, when compared with CSRM$_{ime}$ and CSRM$_{ome}$, the relative performance improvements by CSRM are around 1.88% and 8.41% in terms of Recall@20, and 3.48% and 24.84% in terms of MRR@20, respectively.

5.3 Influence of aggregation operations (RQ3)

Next we compare CSRMs with different aggregation operations, i.e., max pooling, average pooling, concatenation, and a fusion gating mechanism. For max pooling, we aggregate the IME and OME by taking the maximum value of every dimension for each session. Then the i-th dimension of a session-representing vector $\boldsymbol{c}_{t_i}$ is formulated as

$$\boldsymbol{c}_{t_i} = \max(\boldsymbol{c}_{t_i}^{\text{inner}}, \boldsymbol{c}_{t_i}^{\text{outer}}) \,. \tag{16}$$

The average pooling is to aggregate the IME and OME by taking the average value of every dimension for each session, which can be formulated as

$$\boldsymbol{c}_t = \frac{1}{2}(\boldsymbol{c}_t^{\text{inner}} + \boldsymbol{c}_t^{\text{outer}}) \,. \tag{17}$$

For the concatenation operation, the final representation is the concatenation of vectors $\boldsymbol{c}_t^{\text{inner}}$ and $\boldsymbol{c}_t^{\text{outer}}$:

$$\boldsymbol{c}_t = [\boldsymbol{c}_t^{\text{inner}}; \boldsymbol{c}_t^{\text{outer}}] \,. \tag{18}$$

The fusion gating mechanism is actually a linear interpolation between the users' own representation $\boldsymbol{c}_t^{\text{inner}}$ and collaborative information $\boldsymbol{c}_t^{\text{outer}}$ as defined in Eq. 12 and Eq. 13.

As illustrated in Table 4, CSRM with a fusion gating mechanism outperforms the other three aggregation operations on three datasets in terms of Recall@20 and MRR@20. This indicates that the fusion gating mechanism is more helpful in modeling interactions between the IME and OME. In most cases, CSRM with the concatenation operation and CSRM with the average pooling achieve a similar performance and both outperform CSRM with the max pooling on three datasets. This demonstrates that average pooling and concatenation have advantage over max pooling in modeling interactions among multiple factors.

5.4 Influence of the number of neighbors (RQ4)

To investigate the influence of collaborative neighborhood information for session-based recommendations, we show the performance of CSRM with respect to the number of neighbors in Table 5. For the YOOCHOOSE 1/64 and YOOCHOOSE 1/4 dataset, performance improves as the number of neighbors increases in terms of Recall@20 and the best performance is almost the same as the performance

Table 6: Performance comparison of different session lengths on the LastFM dataset. Numbers in the middle represent the number of sessions whose groundtruth items are ranked amongst the top-20 items in each length interval.

Length	without OME	with OME	Performance
1–5	5047	5463	**+8.24%**
6–10	3854	4172	**+8.25%**
11–15	2726	2913	+6.86%
16–20	1932	2072	+7.25%
21–25	1478	1563	+5.75%
26–30	1048	1101	+5.06%
31–35	695	747	+7.48%
36–40	457	486	+6.35%
41–	311	334	+7.40%

at $k = 512$ in terms of MRR@20. As for LastFM dataset, we can get relatively good performance when $k = 512$. This shows that accurately predicting session intent depends on the degree to which the recommender system incorporates collaborative neighborhood information into the model.

5.5 Influence of session length (RQ5)

To understand the influence of session length on utilizing collaborative neighborhood information, we compare diverse sessions with different lengths on LastFM. We show the performance with respect to session length in Table 6. Note that since the number of session lengths is too large, we divide the test set into different groups by the number of items per session. From the table, we can observe that: (1) CSRM performs better overall. This indicates that collaborative neighborhood information does help to capture a session's intent and make a better prediction. (2) The performance improvements of CSRM are stronger when the session lengths are between 1 and 10 in general. This may be due to that it's hard to capture sessions' intent when sessions are short, where there are limited less clicked items for each session. Under these circumstances, our model is good at capturing a session's intent by means of collaborative neighborhood information.

5.6 Case study (RQ6)

To intuitively grasp the effect of collaborative information, we select some good and bad recommendation cases from the YOOCHOOSE 1/64 dataset, as shown in Figure 2. The numbers on the left are the current session IDs. The numbers on the right are the ranking of (positive) ground truths in the recommendation lists. The color scale represents the intensities of the neighbor similarities given by Eq. 9, where a darker color indicates a higher value and a lighter color indicates a lower value. Each row in the figure represents the top 10 neighbors for the current session. The difference between the similarities may be quite small and hence not visually distinguishable.

(1) Overall, it is obvious that sessions that have highly similar neighbors achieve better recommendation performance, i.e., session 12,475, 12,509 and 15,405 in Figure 2. Sessions that assign equal low similarities to neighbors achieve worse recommendation performance, i.e., session 12,600, 14,731 and 27,355 in Figure 2. This

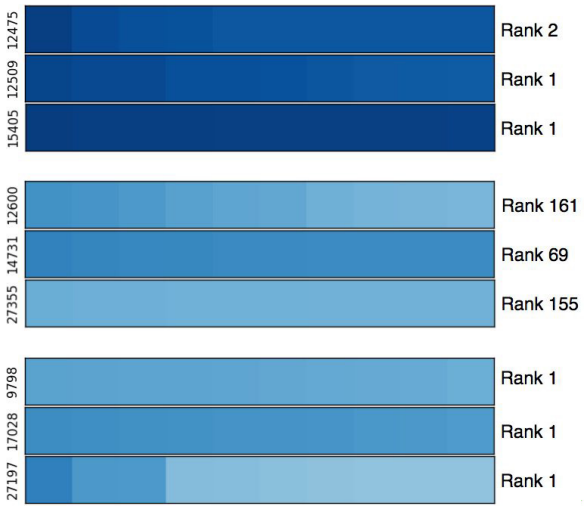

Figure 2: Visualization of neighbors' weights. The depth of color corresponds to the importance of neighborhood sessions given by Eq. 9. The numbers on the left are the session IDs. The numbers on the right are the ranking of ground truths in the recommendation lists.

confirms that collaborative neighborhood information has a great influence on user intents for session-based recommendations.

(2) Not all neighbors of the current session are equally important. E.g., the neighbors of session 12,475 have very different weights.

(3) Not all sessions need to rely on collaborative neighborhood information to achieve good performance. Some sessions assign equal low similarities to neighbors which means they use less collaborative information. But they still achieve good recommendation performance, e.g., session 9,798, 17,028 and 27,197 in Figure 2.

6 CONCLUSION

We have proposed a novel hybrid architecture CSRM to model collaborative filtering in session-based recommendations. CSRM incorporates two parallel memory modules, i.e., the IME and OME, to consider current session information and collaborative neighborhood information, respectively. Features from these two modules are selectively combined with a fusion gating mechanism to achieve better session-based recommendations. Extensive experiments on three datasets demonstrate significant improvements over state-of-the-art baselines in terms of different evaluation metrics. Qualitative experimental analyses on different features and the number of neighbors demonstrate the rationality of applying collaborative filtering to session-based recommendations.

Limitations of our work include the fact that the outer memory networks have limited slots and we can only select neighbors from a small number of recent sessions. In future work, we hope to incorporate a retrieval-based mechanism to enable OME to find neighbors based on all previous sessions. We also hope to improve CSRM by introducing user preferences information and item attributes information, such as user reviews and item categories. Besides, the attention mechanism and memory mechanism have a strong generalization ability. So we want to explore the usage of this framework in other application domains.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their helpful comments. This work is supported by the Natural Science Foundation of China (61672324, 61672322), the Natural Science Foundation of Shandong province (2016ZRE27468), the Tencent AI Lab Rhino-Bird Focused Research Program (JR201932), the Fundamental Research Funds of Shandong University, Ahold Delhaize, the Association of Universities in the Netherlands (VSNU), and the Innovation Center for Artificial Intelligence (ICAI).

REFERENCES

[1] Robert M Bell and Yehuda Koren. 2007. Improved neighborhood-based collaborative filtering. In *KDD Cup and Workshop at the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 7–14.

[2] Xu Chen, Hongteng Xu, Yongfeng Zhang, Jiaxi Tang, Yixin Cao, Zheng Qin, and Hongyuan Zha. 2018. Sequential recommendation with user memory networks. In *11th ACM International Conference on Web Search and Data Mining*. 108–116.

[3] Zhiyong Cheng, Ying Ding, Xiangnan He, Lei Zhu, Xuemeng Song, and Mohan Kankanhalli. 2018. A^3NCF: An adaptive aspect attention model for rating prediction. In *27th International Joint Conference on Artificial Intelligence*. 3748–3754.

[4] Zhiyong Cheng, Jialie Shen, Lei Zhu, Mohan Kankanhalli, and Liqiang Nie. 2017. Exploiting music play sequence for music recommendation. In *26th International Joint Conference on Artificial Intelligence*. 3654–3660.

[5] James Davidson, Benjamin Liebald, Junning Liu, Palash Nandy, Taylor Van Vleet, Ullas Gargi, Sujoy Gupta, Yu He, Mike Lambert, Blake Livingston, et al. 2010. The YouTube video recommendation system. In *4th ACM Conference on Recommender Systems*. 293–296.

[6] Mukund Deshpande and George Karypis. 2004. Item-based top-n recommendation algorithms. *ACM Transactions on Information Systems (TOIS)* 22, 1 (2004), 143–177.

[7] Travis Ebesu, Bin Shen, and Yi Fang. 2018. Collaborative memory network for recommendation systems. In *41st International ACM SIGIR Conference on Research and Development in Information Retrieval*. 515–524.

[8] Djordje Gligorijevic, Jelena Gligorijevic, Aravindan Raghuveer, Mihajlo Grbovic, and Zoran Obradovic. 2018. Modeling mobile user actions for purchase recommendation using deep memory networks. In *41st International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1021–1024.

[9] David Goldberg, David Nichols, Brian M Oki, and Douglas Terry. 1992. Using collaborative filtering to weave an information tapestry. *Commun. ACM* 35, 12 (1992), 61–70.

[10] Alex Graves, Greg Wayne, and Ivo Danihelka. 2014. Neural turing machines. *arXiv preprint arXiv:1410.5401* (2014).

[11] Alex Graves, Greg Wayne, Malcolm Reynolds, Tim Harley, Ivo Danihelka, Agnieszka Grabska-Barwińska, Sergio Gómez Colmenarejo, Edward Grefenstette, Tiago Ramalho, John Agapiou, et al. 2016. Hybrid computing using a neural network with dynamic external memory. *Nature* 538, 7626 (2016), 471–476.

[12] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *26th International Conference on World Wide Web*. 173–182.

[13] Jonathan L Herlocker, Joseph A Konstan, Al Borchers, and John Riedl. 1999. An algorithmic framework for performing collaborative filtering. In *22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. 230–237.

[14] Balázs Hidasi and Alexandros Karatzoglou. 2018. Recurrent neural networks with top-k gains for session-based recommendations. In *27th ACM International Conference on Information and Knowledge Management*. 843–852.

[15] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2016. Session-based recommendations with recurrent neural networks. In *4th International Conference on Learning Representations*.

[16] Jin Huang, Wayne Xin Zhao, Hongjian Dou, Ji-Rong Wen, and Edward Y Chang. 2018. Improving sequential recommendation with knowledge-enhanced memory networks. In *41st International ACM SIGIR Conference on Research and Development in Information Retrieval*. 505–514.

[17] Dietmar Jannach and Malte Ludewig. 2017. When recurrent neural networks meet the neighborhood for session-based recommendation. In *11th ACM Conference on Recommender Systems*. 306–310.

[18] Rong Jin, Joyce Y Chai, and Luo Si. 2004. An automatic weighting scheme for collaborative filtering. In *27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. 337–344.

[19] Diedirik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *4th International Conference on Learning Representations*.

[20] Yehuda Koren. 2008. Factorization meets the neighborhood: A multifaceted collaborative filtering model. In *14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 426–434.

[21] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 42, 8 (2009), 30–37.

[22] Daniel D Lee and H Sebastian Seung. 2001. Algorithms for non-negative matrix factorization. In *Advances in Neural Information Processing Systems*. 556–562.

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

[24] Greg Linden, Brent Smith, and Jeremy York. 2003. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing* 7, 1 (2003), 76–80.

[25] Haifeng Liu, Zheng Hu, Ahmad Mian, Hui Tian, and Xuzhen Zhu. 2014. A new user similarity model to improve the accuracy of collaborative filtering. *Knowledge-Based Systems* 56 (2014), 156–166.

[26] Pasquale Lops, Marco De Gemmis, and Giovanni Semeraro. 2011. Content-based recommender systems: State of the art and trends. In *Recommender Systems Handbook*. 73–105.

[27] Renfeng Ma, Qi Zhang, Jiawen Wang, Lizhen Cui, and Xuanjing Huang. 2018. Mention recommendation for multimodal microblog with cross-attention memory network. In *41st International ACM SIGIR Conference on Research and Development in Information Retrieval*. 195–204.

[28] Lei Mei, Pengjie Ren, Zhumin Chen, Liqiang Nie, Jun Ma, and Jian-Yun Nie. 2018. An attentive interaction network for context-aware recommendations. In *27th ACM International Conference on Information and Knowledge Management*. 157–166.

[29] Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-value memory networks for directly reading documents. In *2016 Conference on Empirical Methods in Natural Language Processing*. 1400–1409.

[30] Andriy Mnih and Ruslan R Salakhutdinov. 2008. Probabilistic matrix factorization. In *Advances in Neural Information Processing Systems*. 1257–1264.

[31] Pengjie Ren, Jing Li, Zhumin Chen, Zhaochun Ren, Jun Ma, and Maarten de Rijke. 2019. RepeatNet: A repeat aware neural recommendation machine for session-based recommendation. In *33rd AAAI Conference on Artificial Intelligence*.

[32] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *25th Conference on Uncertainty in Artificial Intelligence*. 452–461.

[33] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2010. Factorizing personalized markov chains for next-basket recommendation. In *19th International Conference on World Wide Web*. 811–820.

[34] Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey Hinton. 2007. Restricted boltzmann machines for collaborative filtering. In *24th International Conference on Machine Learning*. 791–798.

[35] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. 2016. Meta-learning with memory-augmented neural networks. In *33rd International Conference on Machine Learning*. 1842–1850.

[36] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *10th International Conference on World Wide Web*. 285–295.

[37] Suvash Sedhain, Aditya Krishna Menon, Scott Sanner, and Lexing Xie. 2015. Autorec: Autoencoders meet collaborative filtering. In *24th International Conference on World Wide Web*. 111–112.

[38] Guy Shani, David Heckerman, and Ronen I. Brafman. 2005. An MDP-based recommender system. *Journal of Machine Learning Research* 6, Sep (2005), 1265–1295.

[39] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research* 15, 1 (2014), 1929–1958.

[40] Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. In *Advances in Neural Information Processing Systems*. 2440–2448.

[41] Yong Kiam Tan, Xinxing Xu, and Yong Liu. 2016. Improved recurrent neural networks for session-based recommendations. In *1st Workshop on Deep Learning for Recommender Systems*. 17–22.

[42] Aaron Van den Oord, Sander Dieleman, and Benjamin Schrauwen. 2013. Deep content-based music recommendation. In *Advances in Neural Information Processing Systems*. 2643–2651.

[43] Hao Wang, Naiyan Wang, and Dit-Yan Yeung. 2015. Collaborative deep learning for recommender systems. In *21st ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 1235–1244.

[44] Jason Weston, Sumit Chopra, and Antoine Bordes. 2015. Memory networks. In *International Conference on Learning Representations*.

[45] Yao Wu, Christopher DuBois, Alice X. Zheng, and Martin Ester. 2016. Collaborative denoising auto-encoders for top-n recommender systems. In *9th ACM International Conference on Web Search and Data Mining*. 153–162.

[46] Andrew Zimdars, David Maxwell Chickering, and Christopher Meek. 2001. Using temporal data for making recommendations. In *17th Conference on Uncertainty in Artificial Intelligence*. 580–588.