Streaming Session-based Recommendation

Lei Guo
Shandong Normal University
China
leiguo.cs@gmail.com

Hongzhi Yin*
The University of Queensland
Australia
h.yin1@uq.edu.au

Qinyong Wang
The University of Queensland
Australia
qinyong.wang@uq.edu.au

Tong Chen
The University of Queensland
Australia
tong.chen@uq.edu.au

Alexander Zhou
The University of Queensland
Australia
alexander.zhou@uq.net.au

Nguyen Quoc Viet Hung
Griffith University
Australia
quocviethung1@gmail.com

ABSTRACT

Session-based Recommendation (SR) is the task of recommending the next item based on previously recorded user interactions. In this work, we study SR in a practical streaming scenario, namely Streaming Session-based Recommendation (SSR), which is a more challenging task due to (1) the uncertainty of user behaviors, and (2) the continuous, large-volume, high-velocity nature of the session data. Recent studies address (1) by exploiting the attention mechanism in Recurrent Neural Network (RNN) to better model the user’s current intent, which leads to promising improvements. However, the proposed attention models are based solely on the current session. Moreover, existing studies only perform SR under static offline settings and none of them explore (2).

In this work, we target SSR and propose a Streaming Session-based Recommendation Machine (SSRM) to tackle these two challenges. Specifically, to better understand the uncertainty of user behaviors, we propose a Matrix Factorization (MF) based attention model, which improves the commonly used attention mechanism by leveraging the user’s historical interactions. To deal with the large-volume and high-velocity challenge, we introduce a reservoir-based streaming model where an active sampling strategy is proposed to improve the efficiency of model updating. We conduct extensive experiments on two real-world datasets. The experimental results demonstrate the superiority of the SSRM method compared to several state-of-the-art methods in terms of MRR and Recall.

CCS CONCEPTS

• **Information systems → Recommender systems**;

KEYWORDS

Session Recommendation; Streaming Recommendation; Attention Model; Matrix Factorization

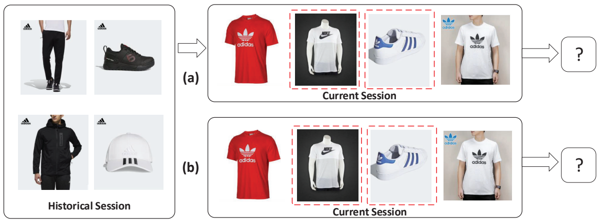


Figure 1: Two attention-based recommenders: (a) historical and current session-based attention, (b) current session-based attention.

ACM Reference Format:
Lei Guo, Hongzhi Yin, Qinyong Wang, Tong Chen, Alexander Zhou, and Nguyen Quoc Viet Hung. 2019. Streaming Session-based Recommendation. In *The 25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD ’19), August 4–8, 2019, Anchorage, AK, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3292500.3330839

1 INTRODUCTION

In many online systems (such as E-commerce and social media sites), user interactions are organized into sessions, a sequence of user behaviors that take place within a short time window. As, in a session, the user’s interactions are always accompanied by randomness and only limited actions can be exploited, making recommendations in sessions can be challenging.

Recently, Hidasi et al. [12] applied RNNs to SRs and achieved a significant improvement over traditional recommendation methods. However, as a user may accidentally click on wrong or irrelevant items, traditional RNNs that do not differentiate the importance of all the visited items may fail to capture the user’s main purpose of the current session. To address this issue, Li et al. [18] improved the RNN-based model by introducing the attention mechanism to help the recommender capture the user’s current intention. However, there are two limitations w.r.t. their work: (1) The attention model was explored based solely on the current user session, where user profiles are assumed invisible to the recommenders. But in many systems, the user’s identification is traceable (via log-ins or cookies). In these cases, it is reasonable to assume that we can obtain a more accurate attention model from the user’s past sessions. Although several methods have investigated how to leverage the history information to improve SRs [16, 19, 29], the user’s main attentions of the current session were not considered. (2) They only performed

[*] Corresponding author and contributing equally with the first author.

KDD ’19, August 4–8, 2019, Anchorage, AK, USA

ACM ISBN 978-1-4503-6201-6/19/08…$15.00
https://doi.org/10.1145/3292500.3330839

SR under a static offline setting, and the large-volume, high-velocity nature of the session data was not explored.

On one hand, capturing the user's main purpose from history is helpful for SRs, which is still largely unexplored. Suppose a user wants to buy a T-shirt on the E-commerce website. As shown in Figure 1, to make comparisons, a user tends to click on some T-shirts with similar styles, but she/he also clicks a pair of shoes out of curiosity. In this case, if we only consider the user's current interactions and interests, another similar T-shirt with different brands might be recommended because the user has clicked on more T-shirts than other products. We can derive that the user intends to buy a T-shirt with the attention model learned from the current session (denoted by Figure 1 (b)). However if we further explore this user's browsing history, we will know this user may be an 'Adidas' fan (as she/he has bought various 'Adidas' products), the T-shirt with the 'Adidas' brand will be recommended (as shown in Figure 1 (a)). The historical information allows us to model the users' current intention more accurately. However, if we only utilize the historical information without considering its relationship with the user's current purpose, other kinds of products (rather than T-shirt) with the 'Adidas' brand might be recommended.

On the other hand, in the real-world scenarios, most online systems generate massive amounts of session data at an unprecedented rate. This data is temporally ordered, continuous, large-volume and high-velocity, which determines the streaming nature of the session data. Hence it is more natural and realistic to conduct SR in streaming settings. Traditional SR cannot be directly applied to streaming settings because of the following two challenges: (1) Large-volume of the incoming data. As in reality, computational resources are limited, it is impractical to assume we can store all the sessions in memory. One alternative solution is to resort to online learning, which intends to update the pre-trained model by utilizing only new observations. But by doing so, the model will soon forget the user's past interactions, since they only learn from the recent data points. (2) High-velocity of the streaming data. As an inherent characteristic of the streaming data, the high arrival velocity requires the system to update the model in a very short time frame, which means the system needs to update and respond instantaneously to catch the user's instant intention and demands.

To address the above two issues, we propose a Streaming Session-based Recommendation Machine (SSRM), which consists of a hybrid session-based recommender and a reservoir-based model updating mechanism. More specifically, in the session-based recommender, we introduce the well-known MF to the RNN-based session encoder to model the user's long-term preferences, which are further utilized as the attention signals to capture the main purpose of the user's current session. At the output layer, the predictions from the MF and the session encoder are combined together. In the model updating component, the reservoir-based streaming technique is first introduced to tackle the long-term memory challenge. Then, to deal with the high velocity challenge, an active session selection strategy is exploited, where only sessions that can change the model most are selected for model updating.

The main contributions of this study are summarized as follows:

- To the best of our knowledge, we are the first to systematically address the SR problem in streaming settings, and propose a streaming session-based recommender to deal with the encountered challenges.
- We propose a MF-based attention model to capture the main intention of a user from her/his historical interactions, and furthermore propose a hybrid session-based recommender to model both a user's long and short-term preferences.
- We introduce a reservoir-based technique and an active sampling strategy to respectively tackle the large-volume and high-velocity challenges of the streaming data.
- We conduct extensive experiments on two real-world datasets to demonstrate the effectiveness of our proposed method.

2 RELATED WORK

This section briefly reviews the related work from two aspects: session-based and streaming-based recommendation methods.

2.1 Session-based Recommendation Method

As one of the emerging recommendation problems, SR has been widely studied in recent years [22, 38]. Compared with traditional user-item settings, this task focuses on sequential mining, and only limited user interactions within a short time frame can be exploited. Due to its sequential nature, the deep neural networks used in sequential recommendations are employed in session-based recommenders [13, 17, 21, 25]. For example, Hidasi et al. [12] applied Gated Recurrent Unit (GRU)-based RNN to SR and achieved a significant improvement over traditional methods. Tan et al. [28] improved their work by utilizing data augmentation and model pre-training techniques. Hidasi and Karatzoglou [11] further exploited the ranking loss function to improve the RNN-based recommender. Ren et al. [25] explored the repeated consumption phenomenon in session recommendation, and incorporated a repeat-explore mechanism into neural networks. Li et al. [18] adopted an attention mechanism to capture the user's main intention in the current session, since users' interactions are often accompanied by randomness. Liu et al. [20] used the embedding of the last-click to represent the user's current interests, and built the attention model on top of it to capture the user's short-term intention.

However, the above models can only derive the user's main purposes from the current session. As a result, they only reflect the user's short-term intentions. But in many situations [5, 7, 30], the long-term preferences can also be helpful for SRs. For example, Quadrana et al. [24] investigated the SR problem in domains where user profiles are available, and proposed a hierarchical RNN to simultaneously model the user-level and session-level interactions. In this work, the user-level GRU models the evolution of the user interests by across sessions. Ying et al. [37] modeled the user's general tastes in a hierarchical attention network, where the user representations are obtained by summing the items that the user has rated in the past. But their work is not designed for session recommendation, and no sequential information is learned. Li et al. [19] considered both historical and current information to solve the next item recommendation problem, and their method can discriminatively exploit different types of user behaviours. Nevertheless, their work ignores the situations where users may often click wrong/unrelated items due to mistakes or curiosity, and the current main purpose cannot be captured by their model.

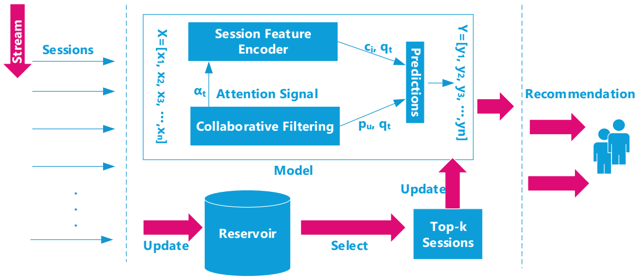


Figure 2: Overview of SSRM. It consists of two parts: 1) the attentive session recommender, and 2) reservoir-based online model updating technique.

2.2 Streaming-based Recommendation Method

Making recommendations in the continuous streaming data has been widely addressed in traditional recommendation problems. One kind of solution is to resort to online learning, which tries to capture the drift of a user’s interest [35, 36] by designing strategies to update the model only by new arrival data [10, 14, 26, 27]. For example, Chang et al. [3] proposed a variational Bayesian approach to permit efficient instantaneous online inference. Jugovac et al. [14] studied the online news recommendation problem and implemented a streaming-based evaluation protocol by retraining the offline model with incoming events. However, their models all intend to forget user’s long-term preferences, since only the most recent data points are utilized to update the model. Moreover, most of these online learning models overlook the overload problem by assuming they are able to learn sufficiently from all the new arrival data.

To address the above challenges, the technique of random sampling with a reservoir was employed to keep the model’s long-term memory [4, 8, 32, 34]. For example, Diaz-Aviles et al. [8] maintained the reservoir using the strategy proposed in [31], and then an active learning based sample selective strategy was utilized to perform online model updating. Wang et al. [34] proposed a streaming-based probabilistic MF to overcome the encountered challenges, where a positive sampling strategy that can sample informative subsets to conduct model updating was adopted. However, these existing streaming-based recommendation methods are all designed for the traditional recommendation problem, and the SR problem in streaming data is still unexplored.

Differences: Our method has significant differences with these existing methods. First, although the previous work [18] has introduced an attention mechanism to capture the user’s main purposes of the current session, they did not consider the user’s intention from their historical interactions, which might be useful to the session recommender. Our method is also different from the works in [24] and [19], which try to model both the user’s long and short-term interests, but the randomness of user interactions was not considered, and the user’s current intention cannot be captured. Second, although reservoir-based streaming techniques have been exploited in traditional recommender systems (such as the work in [34]), none of them were aimed at SR. Conducting SR in streaming data is still an unexplored and challenging problem, and the solutions can benefit the real-time recommender systems.

3 PROPOSED METHOD

This section first gives an overview of our recommendation framework, and then describes how to incorporate MF into a GRU-based session recommender. Finally, this section describes how we extend our model to the streaming scenario.

3.1 Task Definition

The task of SSR is to predict the items that a user would click next in the streaming session data, which is different from traditional SR problem in two aspects: First, it assumes the users’ profiles are visible, which means that both the user’s current and historical interactions can be employed. For simplicity, in this work only one type of user interactions is considered. Second, it focuses on the streaming setting, which assumes the session data is produced largely, continuously and rapidly. Additionally, two major challenges need to be tackled (the details can be seen in the “Introduction” Section). Here, we give a formulation of our recommendation problem.

Let $[x_1, x_2, ..., x_i, ..., x_t]$ be the current click sequence within a session, where x_i is the index of one clicked item out of a total number of n items. Let $\mathcal{R} = (r_{i,j})_{m \times n}$ be the observed interactions among m users and n items, where each entry $r_{i,j} \in \{0, 1\}$ indicates whether user i has clicked on item j. We build a model M so that for any given prefix $(\mathrm{x} = [x_1, x_2, ..., x_t](1 \leq t \leq n))$ of the clicked sequence, and the users’ historical interactions $\mathcal{R}$, we can get the output $y = M(\mathrm{x}, \mathcal{R}) = [y_1, y_2, ..., y_n]$, where y is the ranked list over all the next items that can occur in that session, and $y_j(1 \leq j \leq n)$ is the prediction score for item j. Compared with static session recommendation, the given prefix of a sequence in the streaming setting is in a time ordered and continuous manner. Moreover, since users usually only care about the first few items, the top-k $(1 \leq k \leq n)$ items are recommended.

3.2 Overview of SSRM

In this study, we propose a novel attentive session recommender SSRM to address the SSR problem, whose main work is to build an attention-based session recommender, and then extend it to the streaming setting. The system architecture of SSRM is shown in Figure 2. For the new incoming session, it is first stored to the reservoir with a certain probability. Then, to deal with the high arrival rate of streaming data, only informative sessions are selected to update the pre-trained model. The reservoir technique is introduced to avoid the model being updated only based on the most recent data, and soon forgetting the historical data. The role of the active sample strategy is to find which session is most representative and can correct the model most.

In the model updating component, SSRM first converts the given click sequence within session $s_i = [x_1, x_2, ..., x_t]$ into a set of low-dimensional hidden representations $\mathcal{H}_i = [h_1, h_2, ..., h_t]$. Then, it exploits MF as the attention signal (denoted as α_u) to determine which item is most concerned by user u, and formulates the representation (denoted as c_i) of the current session accordingly. Finally, by combining the outputs of MF and session encoder, the hybrid session recommender is achieved. The attention signal from MF not only indicates the user’s long-term preference, but also determines which part of the hidden representations should be emphasized.

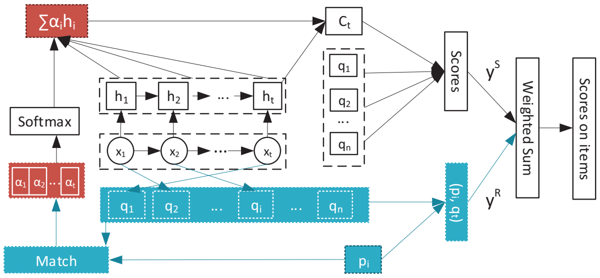


Figure 3: MF-based attentive session recommender.

Intuitively, the more a user 'likes' an item in history, the more attention should be paid to it.

3.3 MF-based Attentive Session Recommender

The attentive session recommender consists of two parts (as shown in Figure 3), the session encoder and MF model, where the session encoder is mainly responsible for modeling the user's current sequence behaviour, and MF is responsible for modeling their collaborative behaviour from historical data. In the following, we will first describe the basic session encoder, and then illustrate how we incorporate the MF as the attention signal to formulate our final hybrid session recommender.

3.3.1 Basic Session Encoder (BSE). We use the RNN with a single GRU layer [6] to encode the user's interactions within a session. The GRU layer has been proved by [12] that, for SR task, it can perform better than the Long Short-Term Memory (LSTM) units.

For each session $s_i = [x_1, x_2, ..., x_t]$, a standard GRU takes the current item ID in the session as an input (1-of-n encoding is utilized), and computes the hidden state h_t for each item as follows:

$$h_t = (1 - z_t)h_{t-1} + z_t \hat{h}_t, \tag{1}$$

where h_{t-1} and $\hat{h}_t$ are the previous and the current hidden state, respectively. z_t is the update gate, which controls how much information needs to be forgotten from h_{t-1} and how much information needs to be remembered from $\hat{h}_t$. It is updated by:

$$z_t = \sigma(W_z x_t + U_z h_{t-1}), \tag{2}$$

where $\sigma(x)$ is the sigmoid function. The current hidden state $\hat{h}_t$ is computed as:

$$\hat{h}_t = \tanh(W x_t + U(r_t \odot h_{t-1})), \tag{3}$$

where r_t is the reset gate, and it controls how much previous memory needs to be retained:

$$r_t = \sigma(W_r x_t + U_r h_{t-1}). \tag{4}$$

As the final hidden state is computed according to all the previous sequential information, a simple way is to use h_t as the representation of the current session i to indicate the user's short-term sequential interests [18]. The basic session representation c_i is thereafter defined as:

$$c_i = h_t. \tag{5}$$

However, this hidden state is only a summary of the whole sequence behaviour, and it is difficult to capture the current intent of the user. Moreover, h_t can only encode the short-term sequential interests of the user, and it has limited ability to model users' long-term interests. On the other hand, collaborative filtering (such as MF) as an effective modeling mechanism has been demonstrated in many recommendation tasks, which provides an efficient way to model the user's historical interests.

3.3.2 MF-based Attentive Session Encoder (ASE). To model the user's historical interactions, we introduce MF to enhance BSE and learn the user and item latent feature factors from his/her collaborative behaviors. Let $p_u \in \mathbb{R}^{1 \times D}$ and $q_t \in \mathbb{R}^{1 \times D}$ be the latent feature vector of user u and item t (D is the dimension of these two latent factors), respectively. Then, the inner product of p_u and q_t can be denoted as follows:

$$\hat{y}^R_{u,t} =< p_u, q_t >= p_u q_t \tag{6}$$

The value of $\hat{y}^R_{u,t}$ indicates how well p_u matches with q_t, which also denotes how much user u 'likes' item t. Incorporating MF to SR enables each item to be represented by the hidden state in the GRU layer and the latent feature factor in MF simultaneously. For any given item t, these two representations (denoted by h_t and q_t) play different roles, where h_t plays a role to summarize the sequence behavior based on the current input, and q_t plays a role to indicate whether an item is preferred by the user.

As the user interactions are often accompanied by randomness, capturing the user's main purpose of the current session can benefit the recommender. To achieve this goal, as shown in Figure 3, we involve the user's long-term preference as an attention signal to BSE, allowing the encoder to dynamically select different parts of the input sequence. For user u and session i, the attention-based encoder can be expressed as follows:

$$c_i = [\sum_{j=1}^{t} \alpha_{u,j} h_j; h_t], \tag{7}$$

where $\alpha_u = softmax(\hat{y}^R_u)$ is the weighting factor representing which part of the input sequence should be emphasized. It can be computed by any real-valued score function (such as inner product, generalized matrix factorization [9] or multi-layered perceptions [9]). In this work, we use the inner product of p_u and q_i as the scoring function for simplicity. Similar to the method in [18], the attention-based encoder is further concatenated with the basic encoder h_t to model the representation of the whole sequential behavior.

Note that our attention-based encoder is different from [18], which only learns the user's attentions from the current session, and the user's historical interests cannot be modeled.

3.3.3 Hybrid Attentive Session Recommender. To enable our method to learn from both the current and historical data simultaneously, we further combine the outputs of MF and attention-based encoder in a unified framework, and train them in a joint manner. In this framework, when making recommendations, our model not only considers the user's sequential behaviour in the current session, but also the user's long-term interests and the collaborative behaviour from other users.

To reduce the number of parameters, we follow the work in [18] and adopt a bi-linear similarity function between the representations of the current session and each candidate item to compute the predictions of the session encoder:

$$\hat{y}^S_{i,t} = q_t B c_i, \tag{8}$$

where $B \in \mathbb{R}^{D \times H}$ is the transformation matrix, D is the dimension of each item embedding, H is the dimension of the session representation. $q_t \in \mathbb{R}^{1 \times D}$ is the item embedding learnt from MF. The final prediction score $\hat{y}_{i,t}$ can be a weighted sum of these two outputs:

$$\hat{y}_{i,t} = w\hat{y}^R_{u,t} + (1-w)\hat{y}^S_{i,t}, \tag{9}$$

where w is the weighting factor, and it determines how much the recommender will be dependent on the MF model. Then, we treat the item ranking problem as a classification task, and try to optimize the following cross-entropy loss:

$$L(r_u, \hat{y}_u) = -\sum_{i=1}^{n} r_{u,i} \log(\hat{y}_{u,i}) \tag{10}$$

where r_u is the item's truly distribution of user u and $\hat{y}_u$ is the predicted probability distribution.

To learn the parameters of the model, instead of using parallel mini-batch training, we train it with the data augmentation technique [18, 28], that is, all prefixes of the original input sessions are treated as new sequences. Given an input session sequence $[x_i, x_2, ..., x_{n-1}]$, the generated sub-sequences and the corresponding labels can be denoted as $([x_1], V(x_2))$, $([x_1, x_2], V(x_3))$, ..., $([x_1, x_2, ..., x_{n-1}], V(x_n))$, where $V(x)$ represents x as the session label. A standard stochastic gradient descent on this cross-entropy loss via Backpropagation-Through-Time (BPTT) for a fixed number of time steps can be utilized in our training procedure.

3.4 Reservoir-based Model Updating Method

In this section, we extend our method to the streaming setting, and hope that the online updating can avoid retraining the model every time a new data instance arrives, but also maintain the model's historical memory. Since traditional online learning methods [26] only conduct updates based on recent arriving data, they can only capture the user's short-term interests. To avoid the model soon forgetting the past, we employ the widely-used reservoir technique [4, 8, 34] to well maintain the long-term memory of the model.

As the goal of a reservoir is to keep an accurate sketch of the history, we employ the random sampling technique [31] to maintain the session data in the reservoir. Let C be the set of session sequences stored in the reservoir, t be the time order of the arrived data instance. When $t > |C|$, the reservoir includes the t-th session instance with probability $|C|/t$, and replaces, uniformly at random, an instance from C. The resulting reservoir has been proven to be a random sample of the current dataset, and it has also been proven to maintain the model's long-term memory [8].

However, because the sampling strategy of a session instance is a time decay function $|C|/t$, it tends to overlook the recent data, and the user's recent interactions cannot be well perceived. But in reality, the user's interests may drift overtime. For example, a user would like to buy different kinds of products in different seasons. Moreover, new users and items arrive continuously in data streams. Hence, catching the patterns contained in the latest generated data is helpful for the recommender. To address this issue, we update the pre-trained model based on both the new input data C^{new} and the sessions maintained in the reservoir C [34]. In the online setting, the goal of our model updating method is to maximize the model's prediction accuracy for both of them.

Algorithm 1 Online training method of SSRM

```
Input: $\mathcal{M}_t$: pre-trained SSRM at time $t$; $C = \{s_1, s_2, ..., s_{|C|}\}$: the
    current reservoir; $s_i = \{x_1, x_2, ..., x_{|S|}\}$: the incoming session;
    $W$: the time window.
Output: The fine-tuned SSRM model at time $t + W$.
for each action $x_t \in C \cup s_i$ do
    Compute $r_{u,t}$ according to Eq. 11.
end for
Compute the sampling probability $\forall s \in C \cup s_i$.
while $W > 0$ do
    Sample sessions $set(s) \in C \cup s_i$ according to Eq. 13.
    Update model $\mathcal{M}_t$ based on batch SGD.
end while
Return the updated model $\mathcal{M}_{t+W}$.
```

3.5 Active Sampling Strategy

Though updating the model using the whole reservoir and the new arrival data can yield better results, the high-velocity of streaming data and limited computing resources may lead to very limited data instances which can be utilized. In many stream processing systems [23], this problem is known as system overload, and a natural solution is to reduce the updating complexity through a wise sample selection strategy. This problem is also similar to that of active learning [2, 33, 39], which selects the minimum set of samples that contributes the most to the system for the users to evaluate. Along this line, to make our model learn as much as possible in a specific time window, a sampling strategy that can select the top informative session instances from $C^{new} \cup C$ is designed.

In this work, we qualify the informativeness of each session by their ranking and select the top sessions that have the lowest rank as the updating instances. To reduce the updating time, we simply rank the sessions according to their scores computed by the prediction function MF, i.e., the inner product of the latent factors (p_u and q_t), with the current set of parameters. Then, for each session, we can derive a rank (denoted as $rank_{i,j}$) in the ranked list for it, where $rank_{i,j}$ denotes the session i is ranked in the j-th position. To reduce the time cost of sampling, we assume that the ranks for all sessions in the ranked list do not need to be updated during the model updating process. Note that, different from traditional active learning method [34] where each sample only has one action, in SRs each session is composed of several sub-actions. When we evaluate the informativeness of a session, we need to qualify its corresponding sub-actions and then use their numerical sum as the representation of that session. Given the i-th session $s_i = [x_1, x_2, ..., x_t]$ of the streaming data, the prediction score of the session is computed by:

$$r_{s_i} = \frac{\sum_{k=1}^{t} r_{u,k}}{t}, \tag{11}$$

where t is the length of the current session, $r_{u,k}$ represents the model's current prediction ability for user u to item k. The smaller the score $r_{u,k}$ is, the more informative the instance is to correct the model. By ranking the sessions in descending order, following the work in [34], we can derive the weight factor of the session s_i:

$$w_{s_i} = exp(\frac{rank_{s_i}}{|C \cup C^{new}|}), \tag{12}$$

which computes the weight of session s_i according to its ranking.

Thus, the sampling probability for a session instance is:

$$p(s_i) = \frac{w_{s_i}}{\sum_{s_i \in C \cup C^{new}} w_{s_i}}. \tag{13}$$

This ranking-based weighting schema favors the data instances at a lower rank much more than the ones at the top. The online training method of our model is listed in Algorithm 1.

4 EXPERIMENTAL SETUP

In this section, we describe the datasets, evaluation metrics and comparison methods utilized in experiments.

4.1 Datasets

We utilize two public datasets Lastfm and Gowalla that contain the user's identities as our experimental data source. Every record in these two datasets contains a user ID, timestamp and an item ID.

Lastfm[1] is released by Celma Herrada [1], and has been widely used in recommendation tasks. We use this dataset for music artist recommendation. In experiments, we keep the top 10,000 most popular artists as the recommended items, and treat user's transactions in 8 hours as a session. Following the work in [25] and [18], we filter out sessions that are longer than 20 or shorter than 2 items and resulted in a dataset that contains 298,919 sessions.

Gowalla[2] is a check-in dataset that is widely used for point-of-interest recommendation. In experiments, we keep the top 30,000 most popular items and treat a user's check-ins in a single day as a session. The sessions with more than 20 or less than 2 items are also removed. The resulting dataset contains 198,608 sessions.

4.2 Evaluation Metrics

Similar to the method in [3] and [34], we simulate the SSR scenario by the following steps: First, we order the session instances in the whole dataset $\mathcal{D}$ according to their ending time. Second, we divide all the data into two parts in chronological order (60% and 40%), and treat the first part as the "training set" (denoted as $\mathcal{D}^{train}$). The second part is treated as the "candidate set" (denoted as $\mathcal{D}^{candidate}$), which is used to mimic the online streaming inputs. More specifically, the candidate set is divided into five equal-sized slices by time with each slice treated as a test set $\mathcal{D}^{test}$. The first test set and 10% of $\mathcal{D}^{train}$ are used as the "validation set", which will determine the optimal parameters of the model. In experiments, all the test slices prior to the target test set are used for online training. For example, if we are going to predict the sequential behaviors in the i-th test set $\mathcal{D}_i^{test}$ then all the test sets ($\forall \mathcal{D}_j^{test} \in \mathcal{D}^{candidate}$) prior to i are used for online training.

To guarantee all the streaming methods are evaluated in the same overload setting, we assume the session sequences all arrive at the same rate, which means all the methods will have the same time window to update their model before the next test data arrives. As in recommender systems the users prefer to pick the first few items of the recommendation list, we utilize the commonly used Recall@20 and MRR@20 [12, 18] as our evaluation metrics. We conduct the one sample paired t-tests to verify that all improvements are statistically significant for $p < 0.01$.

4.3 Comparison Methods

We compare our method with the following baselines:

- **POP**: The basic method that always selects the most popular items of the training set as the recommendations.
- **S-POP**: This method recommends the items that are most popular in the current session, and the recommendation list changes when the new items are generated during the session.
- **BPR-MF**: This is a commonly used traditional recommendation method. We convert it for SR tasks by utilizing the average latent factors of the items that have occurred in the session so far to represent each new session [12].
- **HRNN**: This method [24] tries to exploit user's cross-session information to personalize the RNN model. However, it does not consider the uncertainty of the user behavior, and cannot capture the user's current intent.
- **GRU4Rec**: This is a recent state-of-the-art session recommendation method [11], which utilizes BPR-max ranking loss to increase the performance of their pre-proposed RNN model [12].
- **NARM**: This is another state-of-the-art recommendation method [18], which exploits the attention mechanism to improve the session recommender. But different from ours, their attention model was derived by only using the current interactions and no historical information was adopted.

Note that for all the above methods, to evaluate them in streaming settings, we report the results based on their online versions. To demonstrate the effectiveness of the MF-Attention mechanism, and the importance of the long-term user interests, we compare SSRM with NARM and the basic session encoder (**baseline**, in Section 3.3.1) in an online setting. In experiments, the baseline method only adopts the basic RNN network for recommendation, and the user's current intent is not investigated (more details are shown in Section 5.2).

To evaluate the streaming strategy in SSRM, we further conduct experiments on three different streaming settings. As all the existing streaming-based recommendation methods are not aimed for SR, we implement their streaming techniques in SSRM, and compare SSRM with its following variant versions:

- **SSRM-S1**: This is the simplified version without maintaining the reservoir, and updates the pre-trained model based on the sampled instances that only from newly incoming data.
- **SSRM-S2**: This version implements the reservoir maintaining strategy proposed by Diaz-Aviles et al. [8], and samples the training instances from C randomly.
- **SSRM-S3**: This version further improves SSRM-S2 by randomly selecting the samples from $C \cup C^{new}$ with the same reservoir maintaining technique [34].

[1] http://mtg.upf.edu/static/datasets/last.fm/lastfm-dataset-1K.tar.gz
[2] https://snap.stanford.edu/data/loc-gowalla.html

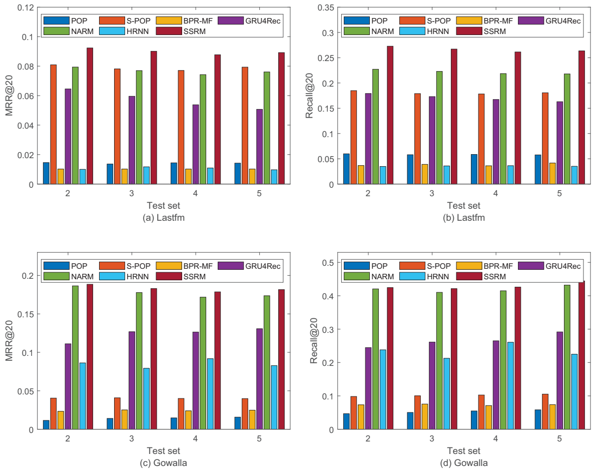


Figure 4: Recommendation performance on Lastfm and Gowalla.

4.4 Implementation Details

We implement SSRM using Python and theno library with NVidia GTX 1080 Ti GPU. The size of the reservoir is searched in $[\frac{|\mathcal{D}|}{100}, \frac{|\mathcal{D}|}{50}, \frac{|\mathcal{D}|}{30}, \frac{|\mathcal{D}|}{16}, \frac{|\mathcal{D}|}{8}]$ and for all the streaming methods the updating time is all set as $\frac{|C^{new}|}{2}$ iterations on each $C \cup C^{new}$. Note that, as the hardware condition has a big influence on the time cost, we do not use a fixed time frame (e.g., 10 seconds) as the updating time. Instead, we set its value as the number of the iterations. In our SSRM method, the optimization is done by using Adam [15] with initial learning rate 0.001. As with the settings in the compared method [18], we truncate BPTT at 19 time steps and set the batch size as 512. All the parameters are tuned on the validation set and the results with the best parameter settings are reported. The weight parameter w (as shown in Eq.9) is set as 0.4 and 0.7 for Lastfm and Gowalla respectively. For the dimension of the item factor q, we set its value as 50. For the user factor p that indicates the user's latent feature in historical sessions, we set its dimensions as 50 for both Lastfm and Gowalla. To prevent SSRM from over-fitting, we use dropout and set the dropout rate to 0.25 for these two datasets.

5 EXPERIMENTAL RESULTS AND ANALYSIS

This section first reports the comparison results on two datasets, and then analyzes the impact of important parameters.

5.1 Results

The experimental results are shown in Figure 4, from which we can see the results of SSRM are better than other baselines, and are statistically significant ($p < 0.01$). From this result, we can also observe that although S-POP is simple, it has very competitive results, and the RNN-based methods can perform better in most cases. But we also notice that the performances of these RNN-based methods cannot perform as well as reported in other works. One reason is that we only utilize half of the data for training. Another reason might be that we split the data into 5 slices according to the time order, and there are many new clicks appeared in the test sets. As the model does not see these clicks before, the recommendation performance will be affected. Attention-based recommendation methods (NARM and SSRM) achieve better performance over other baselines, demonstrating the usefulness of the attention model in SRs, and they are able to deal with the uncertainty of user behaviors. The HRNN method that models both users' long and short-term preferences does not work well, which is beyond our expectations, but this result also provides us with evidence that the attention information is more important than user's historical interactions. Our SSRM method achieves the best performance in both datasets, demonstrating that considering the attention signal from history is beneficial for the recommender, and it can work better than using only current or historical interactions. But we also notice that our SSRM method has a greater improvement in Lastfm than in Gowalla. The main reason is that Gowalla is a sparser dataset, where each user only has a few of clicks, and most of them only have one or few session data, which will result in insufficient model training.

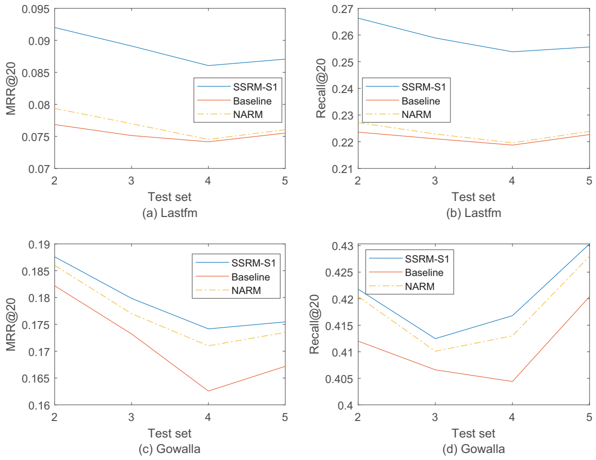
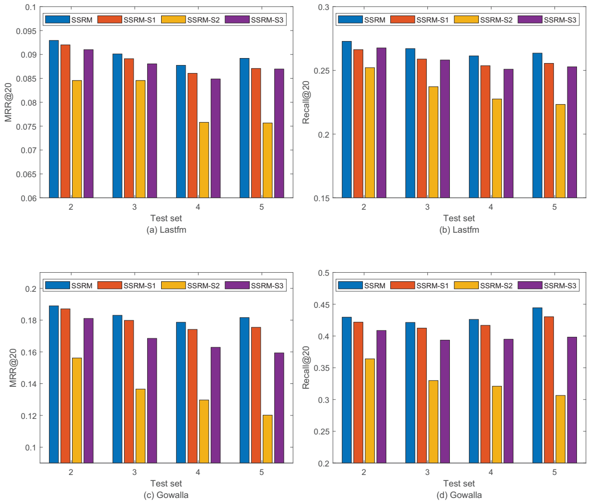


Figure 5: Evaluation of the MF-based attention mechanism.

5.2 Analysis of MF-Attention

The MF-attention models the user's intent by exploiting their interactions from both historical and current sessions. To validate the effectiveness of the attention model, we further conduct experiments with NARM and the baseline method, which only utilizes the basic session encoder (in Section 3.3.1) to make predictions.

To highlight the role of the attention model, in this experiment, we use the online version of SSRM (i.e. SSRM-S1) to reduce the impact of the streaming technique. Both of these three methods report their results based on the same online updating method. From the results shown in Figure 5, we can see that the role of the attention mechanism is obvious. It can improve the performance of the RNN-based model significantly, which demonstrates modeling the user's current intent is helpful for SRs. Our SSRM method achieves a better result than NARM, which indicates modeling user's intents from both history and current is beneficial. From this result, we also find that the sparsity of the dataset impacts our model greatly. When there is enough users interactions (as the Lastfm data), the MF-based attention model can build the user model with greater



(a) Lastfm

(b) Lastfm

(c) Gowalla

(d) Gowalla

Figure 6: Evaluation of different streaming strategies on Lastfm and Gowalla.

accuracy. Conversely, it will has insufficient session data to train the model (as the Gowalla data), and has less improvement than the baseline method.

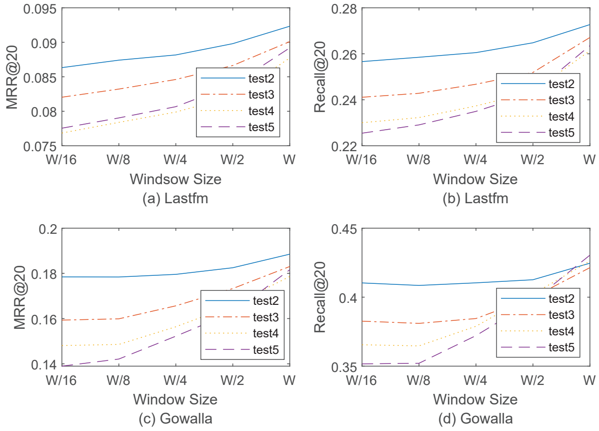
5.3 Analysis of the Streaming Strategy

The reservoir technique has been widely used in streaming systems to tackle the large-volume challenge as it helps the streaming methods to keep a long-term memory. The active sampling strategy borrows the idea from active learning, which tries to select the samples that will most greatly change the model. To evaluate the benefits brought by the reservoir and active sampling strategy, we compare SSRM with its three variant versions.

The experimental results are reported in Figure 6, from which we can observe that compared with the basic SSR recommender SSRM-S2, the method SSRM-S1 that only samples from the new input data performs better. This result indicates the session recommender can benefit more from the user's recent interactions compared to their historical long-term memory. The performance gap between SSRM-S3 and SSRM-S2 further validates this phenomenon. The main reason for this result is that new users and items are more likely to appear in session streams, and if we mainly update the model by historical interactions, it will know nothing about the incoming data. From the results, we can also see that the SSRM method achieves the best performance among all the variant versions, which indicates the benefit of the active sampling strategy. The strategy that sampling the most informative sub-sequences to conduct model updating is more effective over these three variant versions.

5.4 Analysis of the Overload Setting

In our SSRM method, we propose an active sampling strategy to reduce the number of samples needed for updating the model within a fixed time window W. The size of W determines how many samples can be sampled from $C \cup C^{new}$, which indicates the work load of the streaming system. A smaller size of the time window indicates a heavier work load, and only limited training sequences can be adopted. The experimental results are shown in Figure 7.



(a) Lastfm

(b) Lastfm

(c) Gowalla

(d) Gowalla

Figure 7: Evaluation of the overload setting.

From Figure 7, we can observe that with the increase of the time window, the values of Recall and MRR increase in all the test slices, which indicates that the overload of the system impacts our model significantly. In the heavy work load setting, as the model cannot be sufficiently updated, its performance becomes worse than the light work load setting. From the results, we also notice that the performance gaps among different test slices first decrease rapidly, and then the gaps tend to converge to a small value. These performance gaps happen because we split the test data by time order, which results in new users and items appearing in the recent streaming data. Our SSRM model can reduce the gaps quickly demonstrating its ability to deal with the new users and items even in a heavy work load system.

5.5 Analysis of the Reservoir Size

The SSRM method involves maintaining a fixed size reservoir C to retain the observed instances, which captures an accurate sketch of the history of a user's interactions. The size of C determines how much historical information it can keep, and the more observed behaviors it maintains, the greater the chance for SSRM to sample the historical interactions. The results are shown in Figure 8.

From these results, we can see with the decrease of the reservoir, the recommendation performance increases, but the improvement is limited when the size of the reservoir has reduced to a particular value (such as $|C|/8$), and the values of Recall and MRR become steady. For a bigger fixed reservoir, more samples that are distant from the user's recent behaviors are kept, and the model will use less current session information to update. In other words, the model will learn less for the users and items that do not appear in the past, which will result in relatively poor performance. This result is also consistent with the results that shown in Figure 6, that is, the SSRM method can benefit more from users' recent interactions than their history information. But due to the long-memory problem, if we only utilize the current session to retrain the model, it cannot achieve better results than considering them simultaneously.

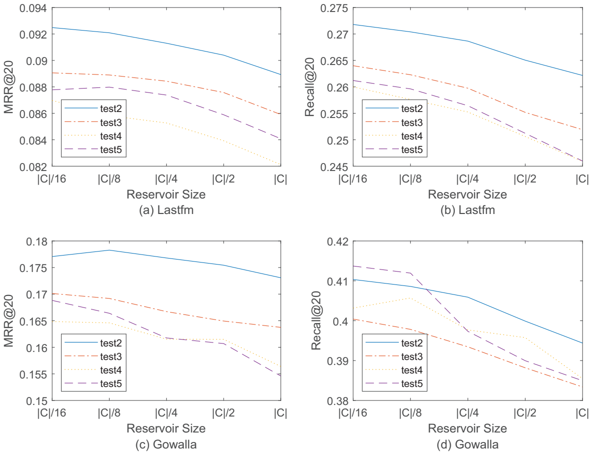


Figure 8: Test for the reservoir size.

6 CONCLUSIONS

In this work, we investigated the SSR problem, and proposed the SSRM to deal with the encountered challenges. Specifically, to deal with the uncertainty of user's behaviors, the attention mechanism that can consider the user's historical interactions was proposed, which introduces MF as the attention signal to guide the hybrid session encoder. To extend our method to the real-time scenarios, the reservoir technique was firstly introduced, and to improve the update efficiency, the active sampling strategy was then proposed. We conducted a wide range of experiments on two real-world datasets with the simulated streaming settings, and the experimental results demonstrate the effectiveness of our SSRM model in the SSR task.

ACKNOWLEDGMENTS

This work was supported by ARC Discovery Projects (Grant Nos. DP190101985, DP170103954), National Natural Science Foundation of China (No. 61602282), and China Postdoctoral Science Foundation (No. 2016M602181).

REFERENCES

[1] Òscar Celma Herrada. 2009. *Music recommendation and discovery in the long tail.* Ph.D. Dissertation. Universitat Pompeu Fabra.

[2] Shayok Chakraborty, Vineeth Balasubramanian, Qian Sun, Sethuraman Panchanathan, and Jieping Ye. 2015. Active batch selection via convex relaxations with guaranteed solution bounds. *IEEE trans. on PAMI* (2015).

[3] Shiyu Chang, Yang Zhang, Jiliang Tang, Dawei Yin, Yi Chang, Mark A Hasegawa-Johnson, and Thomas S Huang. 2017. Streaming recommender systems. *WWW* (2017).

[4] Chen Chen, Hongzhi Yin, Junjie Yao, and Bin Cui. 2013. Terec: A temporal recommender system over tweet stream. *VLDB Endowment* (2013).

[5] Xu Chen, Hongteng Xu, Yongfeng Zhang, Jiaxi Tang, Yixin Cao, Zheng Qin, and Hongyuan Zha. 2018. Sequential recommendation with user memory networks. *WSDM* (2018).

[6] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. *EMNLP* (2014).

[7] Robin Devooght and Hugues Bersini. 2017. Long and Short-Term Recommendations with Recurrent Neural Networks. *UMAP* (2017).

[8] Ernesto Diaz-Aviles, Lucas Drumond, Lars Schmidt-Thieme, and Wolfgang Nejdl. 2012. Real-time top-n recommendation in social streams. *RecSys* (2012).

[9] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. *WWW* (2017).

[10] Xiangnan He, Hanwang Zhang, Min-Yen Kan, and Tat-Seng Chua. 2016. Fast matrix factorization for online recommendation with implicit feedback. *SIGIR* (2016).

[11] Balázs Hidasi and Alexandros Karatzoglou. 2018. Recurrent neural networks with top-k gains for session-based recommendations. *CIKM* (2018).

[12] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2016. Session-based recommendations with recurrent neural networks. *ICLR* (2016).

[13] Liang Hu, Longbing Cao, Shoujin Wang, Guandong Xu, Jian Cao, and Zhiping Gu. 2017. Diversifying personalized recommendation with user-session context. *IJCAI* (2017).

[14] Michael Jugovac, Dietmar Jannach, and Mozhgan Karimi. 2018. Streamingrec: a framework for benchmarking stream-based news recommenders. *RecSys* (2018).

[15] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[16] Dejiang Kong and Fei Wu. 2018. HST-LSTM: A Hierarchical Spatial-Temporal Long-Short Term Memory Network for Location Prediction. *IJCAI* (2018).

[17] Duc Trong LE, Hady Wirawan LAUW, and Yuan Fang. 2018. Modeling contemporaneous basket sequences with twin networks for next-item recommendation. *IJCAI* (2018).

[18] Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Tao Lian, and Jun Ma. 2017. Neural attentive session-based recommendation. *CIKM* (2017).

[19] Zhi Li, Hongke Zhao, Qi Liu, Zhenya Huang, Tao Mei, and Enhong Chen. 2018. Learning from history and present: next-item recommendation via discriminatively exploiting user behaviors. *SIGKDD* (2018).

[20] Qiao Liu, Yifu Zeng, Refuoe Mokhosi, and Haibin Zhang. 2018. STAMP: short-term attention/memory priority model for session-based recommendation. *SIGKDD* (2018).

[21] Pablo Loyola, Chen Liu, and Yu Hirate. 2017. Modeling User Session and Intent with an Attention-based Encoder-Decoder Architecture. *RecSys* (2017).

[22] Malte Ludewig and Dietmar Jannach. 2018. Evaluation of Session-based Recommendation Algorithms. *arXiv preprint arXiv:1803.09587* (2018).

[23] Thao N Pham, Panos K Chrysanthis, and Alexandros Labrinidis. 2016. Avoiding class warfare: managing continuous queries with differentiated classes of service. *The VLDB Journal* (2016).

[24] Massimo Quadrana, Alexandros Karatzoglou, Balázs Hidasi, and Paolo Cremonesi. 2017. Personalizing session-based recommendations with hierarchical recurrent neural networks. *RecSys* (2017).

[25] Pengjie Ren, Zhumin Chen, Jing Li, Zhaochun Ren, Jun Ma, and Maarten de Rijke. 2019. RepeatNet: A Repeat Aware Neural Recommendation Machine for Session-based Recommendation. *AAAI* (2019).

[26] Steffen Rendle and Lars Schmidt-Thieme. 2008. Online-updating regularized kernel matrix factorization models for large-scale recommender systems. *RecSys* (2008).

[27] Karthik Subbian, Charu Aggarwal, and Kshiteesh Hegde. 2016. Recommendations for streaming data. *CIKM* (2016).

[28] Yong Kiam Tan, Xinxing Xu, and Yong Liu. 2016. Improved recurrent neural networks for session-based recommendations. *DLRS* (2016).

[29] Jiaxi Tang and Ke Wang. 2018. Personalized top-n sequential recommendation via convolutional sequence embedding. *WSDM* (2018).

[30] Kiewan Villatel, Elena Smirnova, Jérémie Mary, and Philippe Preux. 2018. Recurrent Neural Networks for Long and Short-Term Sequential Recommendation. *arXiv preprint arXiv:1807.09142* (2018).

[31] Jeffrey S Vitter. 1985. Random sampling with a reservoir. *ACM Trans. Math. Software* (1985).

[32] Qinyong Wang, Hongzhi Yin, Zhiting Hu, Defu Lian, Hao Wang, and Zi Huang. 2018. Neural memory streaming recommender networks with adversarial training. *SIGKDD* (2018).

[33] Weiqing Wang, Hongzhi Yin, Zi Huang, Xiaoshuai Sun, and Nguyen Quoc Viet Hung. 2018. Restricted Boltzmann Machine Based Active Learning for Sparse Recommendation. *DSAA* (2018).

[34] Weiqing Wang, Hongzhi Yin, Zi Huang, Qinyong Wang, Xingzhong Du, and Quoc Viet Hung Nguyen. 2018. Streaming Ranking Based Recommender Systems. *SIGIR* (2018).

[35] Hongzhi Yin, Bin Cui, Ling Chen, Zhiting Hu, and Xiaofang Zhou. 2015. Dynamic user modeling in social media systems. *TOIS* (2015).

[36] Hongzhi Yin, Bin Cui, Xiaofang Zhou, Weiqing Wang, Zi Huang, and Shazia Sadiq. 2016. Joint modeling of user check-in behaviors for real-time point-of-interest recommendation. *TOIS* (2016).

[37] Haochao Ying, Fuzhen Zhuang, Fuzheng Zhang, Yanchi Liu, Guandong Xu, Xing Xie, Hui Xiong, and Jian Wu. 2018. Sequential Recommender System based on Hierarchical Attention Networks. *IJCAI* (2018).

[38] Shuai Zhang, Yi Tay, Lina Yao, Aixin Sun, and Jake An. 2019. Next Item Recommendation with Self-Attentive Metric Learning. *RecNLP* (2019).

[39] Zongwei Zhou, Jae Y Shin, Lei Zhang, Suryakanth R Gurudu, Michael B Gotway, and Jianming Liang. 2017. Fine-Tuning Convolutional Neural Networks for Biomedical Image Analysis: Actively and Incrementally. *CVPR* (2017).