Unclicked User Behaviors Enhanced SequentialRecommendation

Preprint · October 2020

7 authors, including:

Fei Sun
SIM University

Jin Taiwei
Alibaba Group

Unclicked User Behaviors Enhanced Sequential Recommendation

Fuyu Lv[1*], Mengxue Li[1*], Tonglei Guo[1*], Changlong Yu[2], Fei Sun[1], Taiwei Jin[1], Keping Yang[1]
[1]Alibaba Group, Hangzhou, China
[2]The Hong Kong University of Science and Technology, Hong Kong, China
{fuyu.lfy,lydia.lmx,tonglei.gtl,ofey.sf,taiwei.jtw,shaoyao}@alibaba-inc.com;{cyuaq}@cse.ust.hk

ABSTRACT

Deep learning-based sequential recommender systems have recently attracted increasing attention from both academia and industry. Among them, how to comprehensively capture sequential user interest is a fundamental problem. However, most existing sequential recommendation models take as input clicked or purchased behavior sequences from user-item interactions. This leads to incomprehensive user representation and sub-optimal model performance, since they ignore the complete user behavior exposure data, *i.e.,* impressed yet unclicked items. In this work, we attempt to incorporate and model those unclicked item sequences using a new learning approach in order to explore better sequential recommendation technique. An efficient triplet metric learning algorithm is proposed to appropriately learn the representation of unclicked items. Our method can be simply integrated with existing sequential recommendation models by a confidence fusion network and further gain better user representation. We name our algorithm SRU2B (short for ***S**equential **R**ecommendation with **U**nclicked **U**ser **B**ehaviors*). The experimental results based on real-world E-commerce data demonstrate the effectiveness of SRU2B and verify the importance of unclicked items in sequential recommendation.

KEYWORDS

User Behavior Modeling; Sequential Recommendation; Metric Learning

ACM Reference Format:
Fuyu Lv[1*], Mengxue Li[1*], Tonglei Guo[1*], Changlong Yu[2], Fei Sun[1], Taiwei Jin[1], Keping Yang[1]. 2021. Unclicked User Behaviors Enhanced Sequential Recommendation. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

In order to reduce information overload and satisfy customers' diverse online service needs (*e.g.,* E-commerce, music, and movies), personalized recommender systems (RS) have become increasingly important. Traditional recommendation algorithms (collaborative filtering [33] and content-based filtering [29]) only model users' long-term preference, while ignore dynamic interest in users' behavior sequences. Hence sequential recommendation (SR) is introduced to model sequential user behaviors in history to generate user representation by considering time dependency of user-item interactions.

The key to SR is understanding the evolution of users' preference. Most existing SR models (*e.g.,* GRU4REC [12], NARM [22], SDM [25], SASRec [19], Caser [37]) only take as input sequential clicked or purchased behaviors for user modeling. They pay little attention to model more abundant exposure data in users' complete behavior sequences, *i.e.,* those items that were impressed to users yet not clicked (referred to unclicked items in this paper). The unclicked items makes up a relatively significant share of the whole user-item exposure data. Those items also contain valuable signal on users' dynamic preference, which can complement the clicked data. On the one hand, unclicked items are of less interest to users, which influence users' future behaviors and can bring better understandings about users' preference. On the other hand, it is noteworthy that clicked items are inherently noisy [15]. For example, a user may accidentally click on some wrong items in a session. We can leverage the unclicked ones to distinguish more accurate positive signals from noises. Modeling users' preference ignoring the unclicked behavior sequences leads to incomprehensive user representation and limits the capacity and performance of SR.

In this work, we aim to integrate the valuable unclicked item sequences with clicked ones as complete user behaviors into SR models' input to enhance the model performance. Though it is novel in SR, prior works [6, 28, 46, 52] explore for the general recommendations. Compared with SR, they focus on quite different tasks (*i.e.,* matrix factorization [6], reinforcement recommender [52] or click-through rate prediction [28, 46]) and specially show different settings such as task definition, training/test sample construction and evaluation. Besides, their modelings of unclicked sequences remain at the feature level without complex interactions with clicked ones. It is believable that clicked and unclicked behaviors affect each other [42]. Naturally we start to think about effectively incorporating them together from the model level. Firstly, we derive three important characteristics observed from real-life cases. **1)** As introduced, it is obvious that unclicked items reflect users' dislikes to some extent compared with clicked ones as shown in Figure 1(A). **2)** On the other hand, this kind of items are not those that users particularly dislike compared with a random recommended item. The skipped unclicked items can be seen as an *intermediate feedback* between clicked and random recommended items. Because a modern RS recommends items in which users are probably interested by personalized algorithms. Users choose to skip items possibly due to many other complex factors, such as price of items displayed

∗ denotes equal contributions.

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/1122445.1122456

Figure 1: An example of one user's complete behaviors from a well-known E-commerce site. (A) Items belong to a same category (winter coat), but the user is only interested in women's wear rather than men's wear. (B) Items belong to the same product (blender) that the user is interested in, yet the user only selects a few of them to click.

nearby, seasonal nature of items or hot consumer trends. Illustrated in Figure 1(B), all of these items impressed to users at least partly conform to users' preference, but the user only select a few of them to click. Those unclicked items obviously are not random items. **3)** Clicked item sequences could contain some noises that belong to unclicked items. The model should be designed to help *denoise* in the clicked sequences with the help of unclicked ones.

Based on these observations, we propose a new metric learning algorithm to learn the "intermediate" representations of unclicked item sequences in SR. Specifically, we first project sequential clicked and unclicked behaviors as well as labels into the vector space by deep neural network (*e.g.,* LSTM, self-attention, and MLP), where Euclidean distance is used as metric measurement. The labels are next clicked items after the current user sequence, which represent the true vector of user interest. We consider triplet relations among those different vectors: **1)** clicked and unclicked item vectors, and **2)** clicked and label item vectors. The key idea is to regularize the model by enforcing that the representation of clicked sequence should be far away from the unclicked one. Meanwhile the accompanying direction of regularization is applied to clicked and label item representations, which pushes the correct optimization of clicked representation towards the label vector. Moreover, the properties of *intermediate feedback* of unclicked items are ensured by adding a predefined margin, which controls the maximum distance between clicked and unclicked vectors. Though metric learning gives unclicked sequences reasonable representations, it works on clicked or the final user representation in an implicit manner. We attempt an additional explicit way to further make use of *denoising functionality* of the unclicked sequence. The clicked and unclicked vectors are combined by a confidence fusion network, which dynamically learns the fusion weight of unclicked items, to get the final user representation. Our proposed algorithm could be flexibly integrated with any existing sequential recommender.

We name our algorithm SRU2B, a novel sequential recommendation model with unclicked user behaviors. The experimental results based on two real-world E-commerce datasets demonstrate the effectiveness of SRU2B. Further experiments have been conducted to understand the importance of unclicked items in sequential recommendation. The main contributions of this paper are summarized below:

- We identify the importance of unclicked items in SR and integrate them into models for complete sequential user behavior modeling.
- We propose an algorithm SRU2B based on triplet metric learning and a confidence fusion network to model users' unclicked together with clicked item sequences. It dynamically controls relationships between different representations to achieve accurate recommendation.
- We demonstrate the effectiveness of SRU2B on real-world E-commerce data for this topic, which would shed light on more research of incorporating unclicked item sequences.

2 RELATED WORK

2.1 Sequential Recommender Systems

Sequential recommender systems (SR) mainly leverage users' interaction data in a sequential manner to predict the next behavior. Recurrent Neural Networks (RNNs) are widely applied to capture the order relationship within behavior sequences. GRU4REC [12, 35] uses multi-layer GRU to model session-parallel mini-batches. NARM [22] proposes the neural attentive recommendation machine to learn the global and local representation in the current session. HRNN [32] proposes a hierarchical RNN model to extract latent hidden user states across user's historical sessions. On the top of RNNs, STAMP [24], SASRec [19] and BERT4Rec [34] incorporate attention mechanisms to capture more accurate preference from longer sequences. Tang and Wang [37] and You et al. [49] propose convolutional sequence embedding based recommendation model. Tang et al. [36] builds a model that can make use of different temporal ranges and dynamics depending on the request context. Moreover, Wu et al. [44] and Xu et al. [47] introduce graph neural networks in session-based recommendation. The multiple interests of users and multi-grained representations of items are emphasized in [21, 25] and [3, 9, 16, 17], respectively. And long-term stable preferences from users' historical behaviors are also considered in [1, 23, 25, 26, 45, 48, 50, 51]. However, all of these SR models do not consider the unclicked historical behaviors as model input in the whole user-item exposure data. In this paper, we explicitly model them for better understanding of users' preference.

2.2 Metric Learning based Recommendation

The goal of the metric learning algorithm is to find a more reasonable feature representation subspace of input data. It learns an alternative distance measurement which is used to optimize the performance of the model. When it's applied to recommendation domains, Hsieh et al. [14] propose Collaborative Metric Learning (CML), which learns a joint metric space to encode not only users' preferences but also the user-user and item-item similarity. Similar metric-based models can be found in [2, 7, 10], which use Euclidean distances for modeling transitional patterns. Tran et al. [39] utilize Mahalanobis distance-based metric learning algorithm for automatic playlist continuation of music. To overcome the limitation of geometrical congestion and instability in CML, Tay et al. [38] propose Latent Relational Metric Learning (LRML) to learn a latent relation vector for each given user-item pair. To jointly model heterogeneous user behaviors, different novel metric learning methods are proposed in [20] and [53]. These models criticize that existing score learning methods cannot correctly reflect user-user and item-item similarities in their latent spaces. The triplet metric has the good modeling ability of multivariate relations [13]. Therefore, in this paper, we use metric learning via triplet loss structure to encode clicked and unclicked behavior sequences simultaneously.

3 OUR APPROACH

In this section, we first introduce the notation used in the algorithm SRU2B and formulate our recommendation task. Then we review the base sequential recommendation models. At last, we will introduce and analyze the algorithm SRU2B in detail.

3.1 Problem Formulation

Let $\mathcal{U} = \{u_1, \ldots, u_m\}$ denote the set of users, and $\mathcal{I} = \{i_1, \ldots, i_n\}$ denote the set of items. Our task focuses on implicit recommender systems, such as E-commerce. Implicit recommender systems indirectly model users' preference through behaviors like watching videos, purchasing products, and clicking items. For a user $u \in \mathcal{U}$, we record the user's clicking interactions, then sort these records by interaction time t in the ascending order and get the clicked sequence, namely $\mathcal{S}_u^+ = \{i_1^+, \ldots, i_t^+, \ldots, i_{n_p}^+\}$[1]. The unclicked sequence (items impressed to u yet without clicking interactions) is formed by the same way, namely $\mathcal{S}_u^- = \{i_1^-, \ldots, i_t^-, \ldots, i_{n_n}^-\}$. The two sequences make up the complete sequential user behaviors $\mathcal{S}_u = \mathcal{S}_u^+ \cup \mathcal{S}_u^-$. In fact, clicked and unclicked items appear alternately in the same user sequence. We partition them into individual sequences to simplify problem definition in our work. Their strong dependence in a same sequence will remain for future work. We describe each item $i \in \mathcal{I}$ from different feature scales, *i.e.,* item ID, leaf category, first level category, brand and shop, which are denoted as side information set $\mathcal{F}$.

Based on these preliminaries, we can define the sequential recommendation task. Given the user u's complete historical behavior sequence $\mathcal{S}_{u,t}$, we would like to predict the items set $\mathcal{I}_{u,t}^{pre} \subset \mathcal{I}$ that the user will interact after t. In the process of modeling, all types of user behaviors are encoded into vectors of the same dimension L_e. Following [25, 37], we take next k clicked items after $\mathcal{S}_{u,t}$ as target items (labels) denoted as $C_{u,t} = \{c_1, \ldots, c_k\}$.

3.2 Base Sequential Recommendation

In real industrial recommender systems, users browse and interact with items in chronological order, where the containing items in the same sequence, *i.e.,* $\mathcal{S}_u^+$, may be closely relevant. This property facilitates a non-trivial recommendation task: sequential user modeling and successive item recommendation based on users' historical behaviors. Recently deep learning-based models have demonstrated great power in capturing and characterizing the temporal dependency in sequence data. Various effective sequential deep recommenders are proposed, where RNN [12, 32], CNN (Convolutional Neural Network) [37], self-attention [19, 24] or memory network [3, 30] are adopted to model behavior sequences.

Given the interaction sequence $\mathcal{S}_{u,t}^+ = \{i_1^+, i_2^+, \ldots, i_t^+\}$ of user u at time t, a deep sequential recommender computes the user representation vector $\boldsymbol{h}_{u,t} \in \mathbb{R}^{L_e}$ as:

$$\boldsymbol{h}_{u,t} = \mathrm{DSR}(\mathcal{S}_{u,t}^+, \boldsymbol{e_u}; \Theta) \tag{1}$$

where $\boldsymbol{e}_u \in \mathbb{R}^{L_e}$ is the user profile (gender, sex, etc.) embedding vector. DSR means **D**eep **S**equential **R**ecommenders for short. Θ denotes all the model parameters. As for the input of DSR, each item $i \in \mathcal{S}_{u,t}^+$ is mapped into a vector $\boldsymbol{q}_i \in \mathbb{R}^{L_e}$, which is called *item embedding* and can be learned or fixed. In addition to the item side, DSR encodes user u's clicked behavior sequence into vector $\boldsymbol{h}_{u,t}$, which represents the user's current preferences and hence is named by *sequential preference representation.*

To generate sequential recommendations for user u at time t, we rank a candidate item i by computing the recommendation score $\hat{y}_{u,i,t}$ according to:

$$\hat{y}_{u,i,t} = g(u, i, t) = \boldsymbol{h}_{u,t}^{\mathrm{T}} \cdot \boldsymbol{q}_i \tag{2}$$

where $g(\cdot)$ is the score function, implemented as the inner product between the sequential preference representation $\boldsymbol{h}_{u,t}$ and the i-th item embedding $\boldsymbol{q}_i$. After obtaining scores of all items regarding the user u, we can select Top-k items for recommendation.

3.3 Sequential Recommendation with Unclicked User Behaviors

The base deep sequential recommender (DSR) only take as input $\mathcal{S}_{u,t}^+$ and recommends items according to $\boldsymbol{h}_{u,t}$. They ignore the influence of unclicked user behavior sequences. We propose to model $\mathcal{S}_{u,t}^- = \{i_1^-, i_2^-, \ldots, i_t^-\}$ as a plug-in module on basis of DSR. Starting from a base DSR, the modeling of unclicked items is integrated into the base model for more accurate representation. Here we choose *SDM* [25] as the base DSR due to the following careful considerations:

- the state-of-the-art performance of several sequential recommendation benchmarks.
- capacity of handling with large-scale data for efficient deployed industry applications.

SDM considers short and long-term user interest when modeling sequential user behaviors. Multiple interest in a user's session is emphasized and modeled by LSTM and Transformer neural structures. The long-term interest is combined with the short one by an efficient gating mechanism, which takes correlation between short and long-term into account.

Metric learning for unclicked items. Compared with clicked ones, unclicked items reflect users' dislikes to some extent, but they are not those users particularly dislike compared to a random recommended item. Because a modern RS recommends items which at least partly conform to users' preference by personalized algorithms. Thus unclicked items can be intuitively treated as the *intermediate feedback* between clicked and random recommended items. Therefore, for user u's unclicked item sequence $\mathcal{S}_{u,t}^-$, it should have an intermediate representation of vector $\boldsymbol{n}_{u,t}$ between those of sequence $\mathcal{S}_{u,t}^+$ and random recommended items.

[1] We omit the subscript of u from the item indices without loss of clarity.

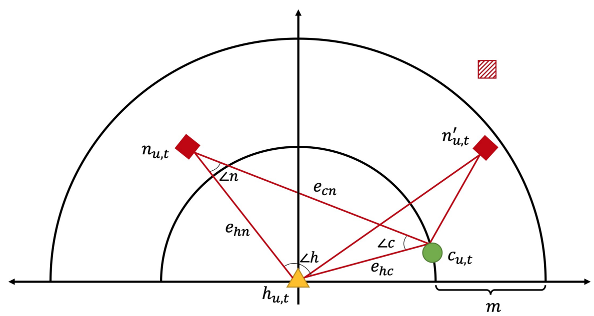


Figure 2: Triplet structure. Red hollow squares represent embedding that have met the constraints and no longer need to be optimized.

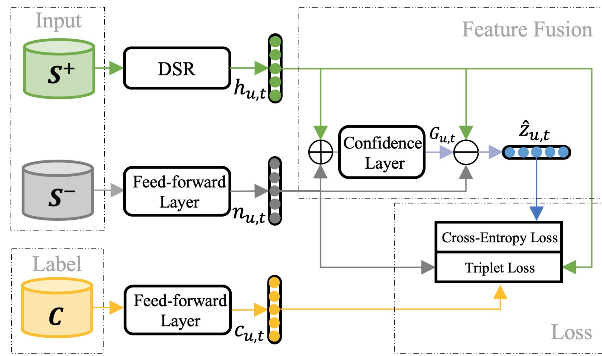


Figure 3: Network structure. Three types of data (in different colors) are mapped into embedding vectors with the same embedding size. We create the triplet loss based on these embedding vectors and combine the cross-entropy as the overall loss for model optimization.

To solve this problem, we introduce metric learning to control the representation of $\mathcal{S}^-_{u,t}$. Inspired by the success of metric learning for the tasks of visual retrieval [11, 27] and classification [4, 31, 43], we formally define the optimization in our recommendation scenarios and extend it with task-specific designs. By this means, we are able to differentiate the preference levels between unclicked items and others for modeling users' comprehensive representation in an accurate and flexible way.

Specifically the first step is to project the sequence $\mathcal{S}^-_{u,t}$ into vector space. With the reminder of *SDM* as our base DSR, we encode item $i \in \mathcal{S}^-_{u,t}$ from different feature scales in $\mathcal{F}$ denoted as $\boldsymbol{q}_i$, which is the same as $i \in \mathcal{S}^+_{u,t}$. On account of huge volume of unclicked items, we simply average all the $\boldsymbol{q}_i$ in $\mathcal{S}^-_{u,t}$ and then use feed-forward network to generate the embedding vector of unclicked items $\boldsymbol{n}_{u,t} \in \mathbb{R}^{L_e}$, described as:

$$\boldsymbol{n}_{u,t} = f\left(\frac{1}{|\mathcal{S}^-_{u,t}|}\sum_{i=1}^{|\mathcal{S}^-_{u,t}|} \boldsymbol{q}_i\right) \tag{3}$$

where $f(\cdot)$ represents non-linear function implemented by feed-forward network with tanh activation. More complex neural structures *e.g.,* Transformer, remain for future work and are not the major points in this paper.

Given a user u, now we have clicked behavior representation $\boldsymbol{h}_{u,t}$, unclicked behavior representation $\boldsymbol{n}_{u,t}$ and label representation $\boldsymbol{c}_{u,t}$. Here $\boldsymbol{c}_{u,t}$ generated from $C_{u,t}$ is embedded in the same way of $\boldsymbol{n}_{u,t}$. Then we use triplet metric learning [41] to construct triple structures among $\boldsymbol{h}_{u,t}$, $\boldsymbol{n}_{u,t}$ and $\boldsymbol{c}_{u,t}$. The optimization goal is to make $\boldsymbol{h}_{u,t}$ and $\boldsymbol{c}_{u,t}$ closer while to make $\boldsymbol{n}_{u,t}$ and $\boldsymbol{c}_{u,t}$ far away from each other. The overall triplet optimization is to minimize:

$$\mathcal{L}_{tri} = \sum_{u\in\mathcal{U}}\left[\left\|\boldsymbol{h}_{u,t}-\boldsymbol{c}_{u,t}\right\|_2^2 - \left\|\boldsymbol{h}_{u,t}-\boldsymbol{n}_{u,t}\right\|_2^2 + m\right]_+ \tag{4}$$

where $\|\boldsymbol{x}\|_2^2 = \sum_{i=1}^n x_i^2$ denotes the squared l_2 norm to measure the distance between vectors and the operator $[\cdot]_+ = \max(0,\cdot)$ denotes the hinge function. $m > 0$ is the relaxing parameter constraining the maximum margin distance.

To understand the role of triplet measurement in the feature space, we use an example from two-dimensional space to explain the intuition shown in Figure 2. The triplet $\mathcal{T} =< \boldsymbol{h}_{u,t}, \boldsymbol{n}_{u,t}, \boldsymbol{c}_{u,t} >$ forms a triangle $\triangle hcn$, whose edges are $e_{hn} = \left\|\boldsymbol{h}_{u,t}-\boldsymbol{n}_{u,t}\right\|_2^2$, $e_{cn} = \left\|\boldsymbol{c}_{u,t}-\boldsymbol{n}_{u,t}\right\|_2^2$, and $e_{hc} = \left\|\boldsymbol{h}_{u,t}-\boldsymbol{c}_{u,t}\right\|_2^2$, respectively. The triplet loss penalizes the shorter edge e_{hn}, so that difference between $\boldsymbol{h}_{u,t}$ and $\boldsymbol{n}_{u,t}$ are significantly large. At the same time, it will reward the shorter edge e_{hc} to make $\boldsymbol{h}_{u,t}$ more similar to $\boldsymbol{c}_{u,t}$. By introducing margin m, we control the maximum difference between e_{hn} and e_{hc} by enforcing $e_{hc}+m \le e_{hn}$. Hence distinction between $\boldsymbol{h}_{u,t}$ and $\boldsymbol{n}_{u,t}$ is constrained within m. It keeps the *intermediate feedback* property of unclicked items between clicked and random recommended items. The introduction of hinge function is to avoid the further correction of those "qualified" triplets.

However, we find that current optimization may lead to undesirable situations, as shown in the Figure 2. The movement of $\boldsymbol{n}_{u,t}$ to $\boldsymbol{n}'_{u,t}$ meets the optimization in Equation 4, but $\boldsymbol{n}'_{u,t}$ is closer to the $\boldsymbol{c}_{u,t}$, which leads to weak distinction between clicked and unclicked item representations. In order to eliminate the effect, we derive a symmetrical triplet constraint by increasing e_{hn} and e_{cn} at the same time, *i.e.,* adding constraint term $e_{hc} + m' \le e_{cn}$. Symmetrical constraints are incorporated into Equation 4 and the new optimization objective is defined as:

$$\begin{aligned}\mathcal{L}_{tri} &= \sum_{u\in\mathcal{U}}\left[\left\|\boldsymbol{h}_{u,t}-\boldsymbol{c}_{u,t}\right\|_2^2 - \left\|\boldsymbol{h}_{u,t}-\boldsymbol{n}_{u,t}\right\|_2^2 + m\right]_+ + \\ &\quad \sum_{u\in\mathcal{U}}\left[\left\|\boldsymbol{h}_{u,t}-\boldsymbol{c}_{u,t}\right\|_2^2 - \left\|\boldsymbol{c}_{u,t}-\boldsymbol{n}_{u,t}\right\|_2^2 + m'\right]_+ \\ &= \sum_{u\in\mathcal{U}}\left[2\left\|\boldsymbol{h}_{u,t}-\boldsymbol{c}_{u,t}\right\|_2^2 - \left\|\boldsymbol{h}_{u,t}-\boldsymbol{n}_{u,t}\right\|_2^2 - \left\|\boldsymbol{c}_{u,t}-\boldsymbol{n}_{u,t}\right\|_2^2 + m^*\right]_+\end{aligned} \tag{5}$$

here we use m^* to represent the addition of two margins in symmetrical losses.

Fusion network. To make better use of unclicked sequences, we attempt to explicitly combine $\boldsymbol{n}_{u,t}$ with base DSR to eliminate noises. We first come up a simple method which directly adopts the difference between $\boldsymbol{n}_{u,t}$ and $\boldsymbol{h}_{u,t}$. The fusion operation of $\boldsymbol{n}_{u,t}$ and $\boldsymbol{h}_{u,t}$ is performed on the top of the model. The final representation $z_{u,t}$ could be formulated as:

$$z_{u,t} = \boldsymbol{h}_{u,t} - \boldsymbol{n}_{u,t} \tag{6}$$

Table 1: Statistics of experimental datasets

Type	Taobao Dataset	Tmall Dataset
Num. of Users	358,978	446,464
Num. of Items	1,078,723	908,214
Num. of Train Data	1,041,094	1,229,271
Num. of Test Data	17,048	20,134
Avg Short-term Clicked Seq Len	13.49	12.30
Avg Long-term Clicked Seq Len	20.87	18.74
Avg Unclicked Seq Len	71.32	64.26

However, $\boldsymbol{z}_{u,t}$ does not contain any complex feature interaction for better modeling. Further we elaborately design a confidence neural network as an activation unit in the fusion process:

$$G_{u,t} = \boldsymbol{W}\text{concat}([\boldsymbol{h}_{u,t}, \boldsymbol{n}_{u,t}]) + \boldsymbol{b} \tag{7}$$

$G_{u,t} \in \mathbb{R}^{L_e}$ is used to determine the weight of each dimension, which indicates how to dynamically combine $\boldsymbol{h}_{u,t}$ and $\boldsymbol{n}_{u,t}$. The refined user representation $\hat{\boldsymbol{z}}_{u,t}$ is generated by subtracting weighted negative information from original sequential representation:

$$\hat{\boldsymbol{z}}_{u,t} = \boldsymbol{h}_{u,t} - G_{u,t} \odot \boldsymbol{n}_{u,t} \tag{8}$$

where $\odot$ is element-wise multiplication.

Overall structure. Figure 3 illustrates the model structure. Different colors represent different data resources, *i.e.,* clicked item sequences, unclicked item sequences and label data. The representation of $\mathcal{S}^+_{u,t}$ and $\mathcal{S}^-_{u,t}$ are concatenated ($\oplus$) as the input of the confidence network. Then the confidence network outputs the activation unit $G_{u,t}$ for feature fusion. The recommendations are made based on the final user representation $\hat{\boldsymbol{z}}_{u,t}$. The optimization of triplet metric learning for unclicked sequences is added to the final loss function.

Loss function. Besides the triplet loss $\mathcal{L}_{tri}$, we use the sampled-softmax [18] method to calculate the cross-entropy loss $\mathcal{L}_{ce}$ over the large amount of items in our real-world dataset for the sake of efficiency. Importance sampling (*e.g.,* log-uniform sampler *w.r.t.* items frequencies) are conducted to obtain j random negative samples $C^-_{u,t}$ from unobserved item set $\mathcal{I}/C_{u,t}$ as most DSR models do ([3, 17, 25, 36]). The model performs joint optimization according to the overall loss defined as follows:

$$\begin{aligned}
&\mathcal{L}_{\text{SRU2B}} = \mathcal{L}_{ce} + \lambda\mathcal{L}_{tri} = \\
&\sum_{u\in\mathcal{U}} \text{CrossEntropy}\big(C_{u,t}, \text{SampledSoftmax}(\hat{\boldsymbol{z}}_{u,t}, C_{u,t}, C^-_{u,t})\big) + \\
\lambda &\sum_{u\in\mathcal{U}} \Big[2\big\|\boldsymbol{h}_{u,t} - \boldsymbol{c}_{u,t}\big\|_2^2 - \big\|\boldsymbol{h}_{u,t} - \boldsymbol{n}_{u,t}\big\|_2^2 - \big\|\boldsymbol{c}_{u,t} - \boldsymbol{n}_{u,t}\big\|_2^2 + m^*\Big]_+
\end{aligned} \tag{9}$$

where $C_{u,t}$ is the positive labels from real behaviors of user u after time t. The sampled-softmax takes final user representation, positive and negative samples as input, which outputs the prediction probability distribution over items in $C_{u,t}$. It has the same motivation with large items candidates as the large vocabulary in language modeling [18]. λ is the trade-off coefficient of two loss terms.

4 EXPERIMENTAL SETUP

4.1 Datasets

As we have discussed, incorporating unclicked sequences into sequential recommendation is a novel exploration, where few of benchmark datasets exists. Hence we construct two large-scale datasets collected from the logs of running recommender systems from Mobile Taobao and Tmall platforms[2] within the time period from 2019/12/27 to 2020/01/03. The collected data contains user portrait features, user complete behavior sequences including clicked and unclicked ones. Each behavior is described from different feature scales, *i.e.,* item ID, leaf category, first level category, brand and shop. Those features are mapped into embedding space from one-hot vector for user/item representation as shown in Section 3.2. Note that dataset in [25] is also sampled from Taobao, but they do not contain unclicked items and the data is not available for the public. For the training/validation/test dataset split and evaluation pipeline, we directly followed the well-defined procedure in [25].

During data collection, one crucial issue is to prepare unclicked data, which is a kind of implicit feedback in our experimental platforms. They contain a large number of noises or uninformative signals, while they are the vast majority of all user behaviors. So we propose to use the following pre-defined filtering rules for selecting unclicked data with less noises

(1) Select latest impressed items of a user that were not clicked within the past three days as unclicked candidates.
(2) Only keep those unclicked items that are exposed to a user more than k times (set $k = 1$ in this work) in the above unclicked candidates.

Items impressed to a user many times yet not clicked indicate that his/her preference at this stage is not strong. We choose those items naturally to form the unclicked sequence for each user. The maximum length of unclicked sequences was set to 100 for the sake of effective sequential modeling. Table 1 shows the statistical information of our datasets.

4.2 Compared Methods

We used the following state-of-the-art sequential recommenders to compare with SRU2B:

- **DNN** [5]. A classic recommendation method proposed for YouTube based on the deep neural network.
- **GRU4REC** [12]. It firstly adopts the RNN in the session-based recommender system.
- **NARM** [22]. Based on GRU4REC, it adds a global and local attention-based module.
- **SHAN** [48]. SHAN combines users' historical preference with the current shopping demand by a hierarchical attention network.
- **BINN** [23]. BINN applies RNN to encode user's sessions, and connects the current consumption motivation and historical behaviors as a unified user representation.
- **SDM** [25]. SDM represents the user behaviors from different levels, which combines short-term sessions and long-term preferences via a gated fusion network.

[2]Popular E-commerce applications with tens of millions of active items (www.taobao.com and www.tmall.com).

Our methods. We conducted ablation experiments by gradually adding our proposed modules and compared with the baseline models above. We employ the model *SDM* (the best baseline) as the base DSR for modeling clicked sequences. We name several SRU2B variants with abbreviated terms.

- **SRU2B**. Proposed algorithm of this paper includes both symmetric triplet metric learning (Equation 5) and confidence fusion network (Equation 8).
- **SRU2B (w/o sym)**. The only difference with SRU2B is using asymmetric triplet metric learning algorithm (Equation 4).
- **SRU2B (w/o fusion+sym)**. SRUCB only employs asymmetric triplet metric learning algorithm (Equation 4) without any explicit feature fusion.
- **SRU2B (w/o metric)**. SRU2B only combines unclicked sequences via the confidence fusion network (Equation 8) to improve feature fusion without metric learning.
- **SRU2B (w/o conf+metric)**. SRU2B only combines unclicked sequences via simple feature fusion (Equation 6) without metric learning.

Although models in [28, 46, 52] are applied in other tasks, they also use unclicked sequences. But their methods are similar to *SRU2B (w/o conf+metric)*, which simply regard unclicked sequences as features of neural networks. Hence we do not involve them as the baselines for fair comparisons.

Evaluation metrics. To evaluate the effectiveness of different methods, we use HR (Hit Ratio), MRR (Mean Reciprocal Rank), R (Recall), and F_1 metrics for the Top-k recommendation results, which are also widely used in the previous works [3, 12, 25, 37]. We chose $k = \{50, 80\}$ to report the Top-k performance as [8]. The reason of setting larger k is the huge amount of item set $\mathcal{I}$ in our datasets and results over smaller k have larger variance thus uncomparable. We calculated these metrics for the test sets and then averaged all the results.

4.3 Implementation Details

We used the distributed Tensorflow[3] to implement all methods mentioned in this work. The training/test datasets were shared among all the models as well as item and user features. Results of the baselines and our models on test dataset are reported according to optimal hyper-parameters tuned on validation data split from training dataset. We used 2 parameter severs (PSs) and 5 GPU (Tesla P100-pcie-16GB) workers with average 30 global steps per second to conduct training and inference. For training, the learning rate was set to 0.1 and the sequences with similar length were selected in a mini-batch whose size is set to 256. AdaGrad was used as the optimizer and the gradient clipping technique was also adopted. Gradients were scaled to some extent when the gradient norm was greater than 5. The next $k = 5$ clicked items after a sequence were taken as the label items in $C_{u,t}$ in our experiments. The sampled-softmax used $j = 20,000$ random negative samples. All input feature representation and model parameters were initialized randomly without pre-training. For parameters of algorithm SRU2B, we set the margin parameter m^* in the triplet loss to 5, the trade-off parameter λ between cross-entropy loss and triplet loss to 10. These two parameters were the best results obtained by parameter selection experiment. All types of behaviors were embedded to vectors with the same size $L_e = 128$. A single layer feed-forward network with sigmoid activation function was used as the structure of the confidence network at the feature fusion process.

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

5 EXPERIMENT ANALYSIS

5.1 Overall Performances

The experimental results are reported in the Table 2 as well as the relative improvement based on the best baseline model. DNN performs worst since the average pooling operation ignores the inheritance correlation between items. The performance of GRU4REC and NARM are far beyond the original DNN by modeling the evolution of short-term behavior. Compared to GRU4REC, SHAN and BINN encode more personalized user information, which are significantly better than GRU4REC and beat NARM. *SDM* performs well due to jointly modeling long-term and short-term behavior. Also it simulates multiple interests in users' short-term session and combine the long-term preference using a gating network. *SRU2B* takes *SDM* as the base model. Two modules *i.e.,* confidence fusion network and symmetric triplet metric learning, are added to the base model. Results of all metrics are substantially improved. *SRU2B* outperforms it by **6.21%** in MRR@50 and **5.63%** in F_1@50 on the Taobao dataset. Similar trends are also observed on the Tmall dataset. This confirms the effectiveness of overall proposed method.

5.2 Ablation Analysis

To disentangle the capability of each module, we further conducted ablation study and results are also shown in Table 2. *SRU2B (w/o conf+metric)* attempts to eliminate noises contained in clicked sequences by using unclicked representation directly, as shown in the Equation 6. The results show that all indicators of this method are slightly improved compared with *SDM. SRU2B (w/o metric)* introduces a confidence network and applies it to weight the unclicked representation in feature fusion process. Results show that almost indicators increase by about 2%~3% on average compared with the base model *SDM* on two datasets. These two experiments demonstrate that the unclicked items does reflect negative interest of users, and it plays an important role of denoising in user preference modeling, though equipped with DSR for clicked sequences.

SRU2B (w/o fusion+sym) only adds an asymmetric triplet loss shown in Equation 4 without explicit feature fusion operation. The result is positive. It reveals the effect of metric learning. We considered the combination of two modules (confidence fusion and asymmetric metric learning) stated above denoted as *SRU2B (w/o sym)*. The average results increase about 3%~4% in almost indicators of two dataset, which indicates that the combination of metric learning and feature fusion can make better use of unclicked data. Metric learning provides higher-quality representations of unclicked sequences for the feature fusion network. Along this way, *SRU2B* further added symmetry constraints, as shown in the Equation 5. The average results show it achieve the highest improvement over *SDM* and *SRU2B (w/o sym)* in almost evaluations. This result shows that symmetric constraints are very important for model learning. From comparison results of all variants, we can conclude that significant improvement is produced by the introduction of the

Table 2: Top-k recommendation comparison of different methods. The relative improvements compared to the best baseline (SDM) are appended on the right starting with "+/-". (k is set to 50, 80). * indicates significant improvement of SRU2B over the personalization models DNN, GRU4REC, NARM, SHAN, BINN and SDM. (p<0.05 in two-tailed paired t-test).

Taobao Dataset

Methods	HR@50		MRR@50		R@50		F_1@50		HR@80		MRR@80		R@80		F_1@80	
DNN	29.95%	-17.42%	6.65%	-24.94%	1.44%	-20.44%	1.14%	-19.72%	36.74%	-15.93%	6.99%	-24.43%	2.01%	-18.29%	1.19%	-17.93%
GRU4REC	32.39%	-10.70%	7.85%	-11.40%	1.62%	-10.50%	1.26%	-11.27%	39.25%	-10.18%	8.20%	-11.35%	2.22%	-9.76%	1.30%	-10.34%
NARM	32.68%	-9.90%	8.20%	-7.45%	1.66%	-8.29%	1.30%	-8.45%	40.27%	-7.85%	8.52%	-7.89%	2.29%	-6.91%	1.34%	-7.59%
SHAN	34.00%	-6.26%	8.84%	-0.23%	1.81%	-0.00%	1.40%	-1.41%	40.93%	-6.34%	9.20%	-0.54%	2.45%	-0.41%	1.41%	-2.76%
BINN	36.24%	-0.08%	8.70%	-1.81%	1.73%	-4.42%	1.38%	-2.82%	43.30%	-0.92%	8.64%	-6.59%	2.34%	-4.88%	1.39%	-4.14%
SDM	36.27%	-	8.86%	-	1.81%	-	1.42%	-	43.70%	-	9.25%	-	2.46%	-	1.45%	-
SRU2B (w/o conf+metric)	36.77%	+1.38%	9.12%	+2.93%	1.82%	+0.55%	1.43%	+0.70%	44.43%	+1.67%	9.45%	+2.16%	2.48%	+0.81%	1.46%	+0.69%
SRU2B (w/o metric)	37.07%	+2.21%	9.12%	+2.93%	1.84%	+1.66%	1.45%	+2.11%	44.80%	+2.52%	9.53%	+3.03%	2.54%	+3.25%	1.49%	+2.76%
SRU2B (w/o fusion+sym)	37.13%	+2.37%	9.09%	+2.60%	1.89%	+4.42%	1.47%	+3.52%	44.87%	+2.68%	9.51%	+2.81%	2.58%	+4.88%	1.52%	+4.83%
SRU2B (w/o sym)	37.37%	+3.03%	9.29%	+4.85%	1.87%	+3.31%	1.47%	+3.52%	45.28%	+3.62%	9.71%	+4.97%	2.58%	+4.88%	1.52%	+4.83%
SRU2B	**37.97%***	**+4.69%**	**9.41%***	**+6.21%**	**1.92%***	**+6.08%**	**1.50%***	**+5.63%**	**45.44%**	**+3.98%**	**9.75%***	**+5.41%**	**2.61%***	**+6.10%**	**1.53%***	**+5.52%**

Tmall Dataset

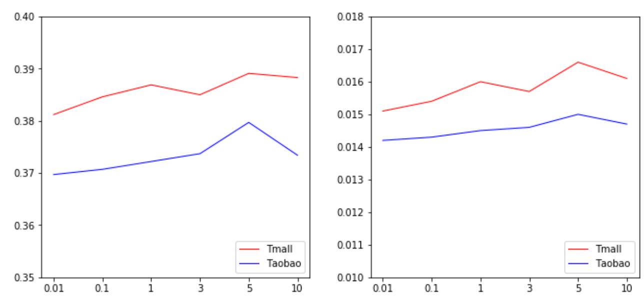
Methods	HR@50		MRR@50		R@50		F_1@50		HR@80		MRR@80		R@80		F_1@80	
DNN	30.45%	-19.47%	7.01%	-26.37%	1.68%	-21.86%	1.27%	-21.12%	37.64%	-16.71%	7.38%	-25.23%	2.36%	-18.34%	1.33%	-17.39%
GRU4REC	34.38%	-9.07%	8.94%	-6.09%	1.98%	-7.91%	1.49%	-7.45%	41.17%	-8.90%	9.14%	-7.40%	2.65%	-8.30%	1.48%	-8.07%
NARM	34.75%	-8.09%	8.96%	-5.88%	2.03%	-5.58%	1.52%	-5.59%	41.89%	-7.30%	9.30%	-5.78%	2.74%	-5.19%	1.52%	-5.59%
SHAN	35.12%	-7.11%	9.48%	-0.42%	2.15%	-0.00%	1.59%	-1.24%	42.29%	-6.42%	9.87%	-0.00%	2.83%	-2.08%	1.55%	-3.73%
BINN	37.20%	-1.61%	9.17%	-3.68%	2.04%	-5.12%	1.54%	-4.35%	45.10%	-0.20%	9.65%	-2.23%	2.83%	-2.08%	1.59%	-1.24%
SDM	37.81%	-	9.52%	-	2.15%	-	1.61%	-	45.19%	-	9.87%	-	2.89%	-	1.61%	-
SRU2B (w/o conf+metric)	38.22%	+1.08%	9.75%	+2.42%	2.19%	+1.86%	1.64%	+1.86%	45.80%	+1.35%	10.18%	+3.14%	2.97%	+2.77%	1.66%	+3.11%
SRU2B (w/o metric)	38.56%	+1.98%	9.67%	+1.58%	2.20%	+2.33%	1.64%	+1.86%	46.39%	+2.66%	10.11%	+2.43%	3.00%	+3.81%	1.67%	+3.73%
SRU2B (w/o fusion+sym)	38.69%	+2.33%	9.81%	+3.05%	**2.21%**	**+2.79%**	1.65%	+2.48%	46.37%	+2.61%	**10.25%**	**+3.85%**	3.00%	+3.81%	1.67%	+3.73%
SRU2B (w/o sym)	38.80%	+2.62%	9.81%	+3.05%	**2.21%**	**+2.79%**	**1.66%**	**+3.11%**	**46.81%**	**+3.58%**	**10.25%**	**+3.85%**	3.02%	+4.50%	1.68%	+4.35%
SRU2B	**38.91%***	**+2.91%**	**9.89%***	**+3.89%**	**2.21%***	+2.79%	**1.66%***	**+3.11%**	46.57%*	+3.05%	10.21%	+3.44%	**3.04%***	**+5.19%**	**1.69%***	**+4.97%**

confidence network and the triplet metric learning with symmetric constraint. Further, metric learning as one of the important modules will be analyzed.

5.3 Role of Metric Learning

Comparing the result of *SRU2B (w/o metric)* with *SRU2B (w/o sym)* and *SRU2B*, we can observe that the introduction of metric learning module improves the performance of *SRU2B* significantly. In this section, we will analyze it in detail. Metric learning algorithm is proposed to model unclicked sequences more appropriately with considering its relationship with clicked sequences. The predefined margin m^* comes into play in a way of threshold limit, and the hyper-parameter λ controls the importance ratio between different loss terms. The analysis is divided into three aspects: control margin effect, metric structure effect and hyper-parameter λ sensitivity. Note that the goal of experiments below is to show how model performance is affected by these experimental variables. It does not mean we optimize our model on test dataset, though results are reported on test.

5.3.1 The effect of margin. The threshold m^* is a parameter for distance control between clicked and unclicked sequences representation in a certain range, so that the unclicked representation has differentiation with random items. It also fits the hinge function to avoid correcting "already correct" triplets within the threshold. A comparison experiment is performed on changes caused by m^*. Parameter experiments follow the rule of coarse to fine. Figure 4 shows the change of the parameters m^*, which is taken from a finer set $\{0.01, 0.1, 1, 3, 5, 10\}$. Result is the best when the parameter m^* is 5, and it performs worse if m^* is too large or too small. Similar observations could be drawn for MRR@50 and R@50, thus omitted due to space limitation.



(a) Effect of m^* on HR@50 (b) Effect of m^* on F_1@50

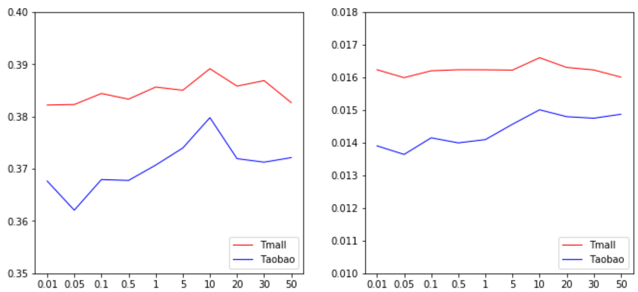
Figure 4: The effect of margin parameter m^*.

5.3.2 The effect of triplet structure. In order to explore the effectiveness of the triplet structure, we only kept one of the three feature relationship in the triplet loss by replacing triplet with pair loss: minimizing the distance between label and clicked representations (lab&clk), maximizing label and unclicked representations (unclk&lab), and maximizing unclicked and clicked representations (unclk&clk). Methods using asymmetric triplet loss (*SRU2B (w/o sym)*) and using symmetric triplet loss (*SRU2B*) are included as comparative experiments.

From Table 3, we can conclude that pair structure optimization with unclk&clk is better than the others (lab&clk and unclk&lab). It shows the introduction of unclicked sequences is effective and should be handled carefully, otherwise causing negative effect to the model. Comparing *SRU2B (w/o sym)* with the three pair experiments, it's clear that the triplet structure can model unclicked data well and produce a positive effect. This also conforms the necessity

Table 3: Triplet structure experiments on Taobao dataset.

models	HR@50	MRR@50	R@50	F_1@50
SRU2B-unclk&lab	36.69%	8.79%	1.83%	1.43%
SRU2B-lab&clk	36.91%	9.29%	1.85%	1.45%
SRU2B-unclk&clk	37.34%	9.40%	1.86%	1.46%
SRU2B (w/o sym)	37.37%	9.29%	1.87%	1.47%
SRU2B	**37.97%**	**9.41%**	**1.92%**	**1.50%**



(a) Effect of λ on HR@50 (b) Effect of λ on F_1@50

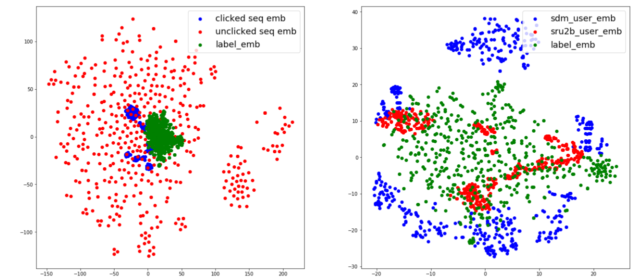
Figure 5: The effect of equilibrium coefficient λ which acts between cross-entropy loss and triplet loss.

of importing triplet metric. In addition, comparing *SRU2B (w/o sym)* with *SRU2B* we can infer the important role of symmetric constraints. To sum up, the triplet loss is effective for unclicked sequence modeling, and the added symmetric constraint makes the model learn a better user representation. The main reason why the triplet loss works well is that a dynamic balance is achieved by using metric learning that controls the distance among three representations.

5.3.3 The effect of hyper-parameter. The loss trade-off coefficient λ in *SRU2B* is an important hyper-parameter, which acts between the cross-entropy loss and the triplet loss to adapt different scale of two losses to a suitable range. Thus λ can directly determine the importance of the triplet loss in the model learning. A comparison experiment in Figure 5 was performed on changes caused by λ, where parameter experiments followed the same rule as parameter m^*. The parameter values are taken from a subdivided set $\{0.01, 0.05, 0.1, 0.5, 1, 5, 10, 20, 30, 50\}$. We find that the effect gets obvious when the parameter λ is greater than 1, and reach the optimal at point 10.

5.4 Visualization

Two modules we proposed (feature fusion and metric learning) are all placed in a learnable feature space. We are interested in whether the feature space is appropriate and whether the feature expression can correctly reflect different preferences. As a result, we did t-SNE [40] visualization for different types of feature representations which were reduced to a two-dimensional space, and explained the adaptability of feature space for *SRU2B*.



(a) Features visualization in triplet loss (b) User representation visualization

Figure 6: (a) Distribution of feature representation involved in triplet loss calculation; (b) User representation visualization generated by algorithm *SRU2B* and *SDM*.

We visualized various feature representations, *i.e.,* $\boldsymbol{h}_{u,t}$, $\boldsymbol{n}_{u,t}$, and $\boldsymbol{c}_{u,t}$ in the calculation of the triplet loss (Equation 5), as shown in Figure 6(a). The final user representations $\hat{\boldsymbol{z}}_{u,t}$ and $\boldsymbol{h}_{u,t}$ obtained from algorithm *SRU2B* and *SDM* respectively are presented in Figure 6(b). Figure 6(a) illustrates that the representations of clicked sequences and labels are closely clustered, and unclicked ones keep a certain distance around them. This result fits our optimization goal and makes the feature fusion in *SRU2B* more meaningful. Figure 6(b) shows that compared with *SDM*, user representation obtained from *SRU2B* is more closely clustered with labels. *SRU2B* can learn a better sequential recommender by introducing unclicked sequences.

5.5 Case Study

In order to show the real performance of our algorithm, as well as to explain what characteristics the unclicked data reflects, real cases from the Taobao dataset were analyzed in this section. We sampled a representative user and displayed her behaviors and top prediction results. Since it is hard to say whether filtering out all information contained in unclicked sequences will actually lead to actual online metric gains, we only compare top prediction results in this case. In Figure 7, the user's clicked item sequence concentrates on women's wear and children's wear, while the unclicked sequence concentrates mostly on shoes. From this point of view, the user does not have a strong preference for shoes currently, although she clicked shoes items in her clicked sequence. So we hope that the algorithm can learn this negative interest and does not recommend shoes at least in top positions. Comparing recommendation results of algorithm *SRU2B* and *SDM*, we find that *SRU2B* filters out shoes items well, while *SDM* still give priority to those items. It can be seen that noises in clicked behaviors could be eliminated by incorporating unlicked item sequences. Finally, comparing with the user's ground truth behaviors, we find it basically coincides with the result of algorithm *SRU2B* at category level, which indicates the importance of modeling unclicked items and the validity of our algorithm.

Figure 7: Items on the top are a user's historical behaviors, including clicked and unclicked sequences. Items on the bottom are top prediction results of SRU2B and SDM, and real behaviors (labels). Items in SDM results with red dashed box are similar to unclicked items.

6 CONCLUSION

In this paper, we study users' unclicked sequences modeling in sequential recommender systems in order to enrich user representation. The importance of unclicked items is emphasized and then incorporated into our new learning model. For modeling sequential behaviors with unclicked data, we design a novel algorithm SRU2B, which adopts a symmetric metric learning with a triplet structure as well as confidence fusion network. The experiment results demonstrate the effectiveness of the SRU2B and verify the importance of unclicked sequences in the sequential recommendation.

REFERENCES

[1] Ting Bai, Lixin Zou, Wayne Xin Zhao, Pan Du, Weidong Liu, Jian-Yun Nie, and Ji-Rong Wen. 2019. CTRec: A Long-Short Demands Evolution Model for Continuous-Time Recommendation. In *SIGIR*. 675–684.

[2] Shuo Chen, Josh L Moore, Douglas Turnbull, and Thorsten Joachims. 2012. Playlist prediction via metric embedding. In *KDD*. 714–722.

[3] Xu Chen, Hongteng Xu, Yongfeng Zhang, Jiaxi Tang, Yixin Cao, Zheng Qin, and Hongyuan Zha. 2018. Sequential recommendation with user memory networks. In *WSDM*. 108–116.

[4] Sumit Chopra, Raia Hadsell, and Yann LeCun. 2005. Learning a similarity metric discriminatively, with application to face verification. In *CVPR*, Vol. 1. IEEE, 539–546.

[5] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep neural networks for youtube recommendations. In *RecSys*. 191–198.

[6] Jingtao Ding, Yuhan Quan, Xiangnan He, Yong Li, and Depeng Jin. 2019. Reinforced negative sampling for recommendation with exposure data. In *IJCAI*. 2230–2236.

[7] Shanshan Feng, Xutao Li, Yifeng Zeng, Gao Cong, Yeow Meng Chee, and Quan Yuan. 2015. Personalized ranking metric embedding for next new POI recommendation. In *IJCAI*.

[8] Chen Gao, Xiangnan He, Danhua Gan, Xiangning Chen, Fuli Feng, Yong Li, Tat-Seng Chua, Lina Yao, Yang Song, and Depeng Jin. 2019. Learning to Recommend with Multiple Cascading Behaviors. *TKDE* (2019).

[9] Guibing Guo, Shichang Ouyang, Xiaodong He, Fajie Yuan, and Xiaohua Liu. 2019. Dynamic item block and prediction enhancing block for sequential recommendation. In *IJCAI*.

[10] Ruining He, Wang-Cheng Kang, and Julian McAuley. 2017. Translation-based recommendation. In *RecSys*. 161–169.

[11] Alexander Hermans, Lucas Beyer, and Bastian Leibe. 2017. In defense of the triplet loss for person re-identification. *arXiv preprint arXiv:1703.07737* (2017).

[12] Balzs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2015. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939* (2015).

[13] Elad Hoffer and Nir Ailon. 2015. Deep metric learning using triplet network. In *International Workshop on Similarity-Based Pattern Recognition*. Springer, 84–92.

[14] Cheng-Kang Hsieh, Longqi Yang, Yin Cui, Tsung-Yi Lin, Serge Belongie, and Deborah Estrin. 2017. Collaborative metric learning. In *WWW*. 193–201.

[15] Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative filtering for implicit feedback datasets. In *2008 Eighth IEEE International Conference on Data Mining*. Ieee, 263–272.

[16] Jin Huang, Zhaochun Ren, Wayne Xin Zhao, Gaole He, Ji-Rong Wen, and Daxiang Dong. 2019. Taxonomy-aware multi-hop reasoning networks for sequential recommendation. In *WSDM*. 573–581.

[17] Jin Huang, Wayne Xin Zhao, Hongjian Dou, Ji-Rong Wen, and Edward Y Chang. 2018. Improving sequential recommendation with knowledge-enhanced memory networks. In *SIGIR*. 505–514.

[18] Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2014. On using very large target vocabulary for neural machine translation. *arXiv preprint arXiv:1412.2007* (2014).

[19] Wang-Cheng Kang and Julian McAuley. 2018. Self-Attentive Sequential Recommendation. In *ICDM*. 197–206.

[20] Dongha Lee, Chanyoung Park, Hyunjun Ju, Junyoung Hwang, and Hwanjo Yu. 2019. Action space learning for heterogeneous user behavior prediction. In *IJCAI*. 2830–2836.

[21] Chao Li, Zhiyuan Liu, Mengmeng Wu, Yuchi Xu, Huan Zhao, Pipei Huang, Guoliang Kang, Qiwei Chen, Wei Li, and Dik Lun Lee. 2019. Multi-interest network with dynamic routing for recommendation at Tmall. In *CIKM*. 2615–2623.

[22] Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Tao Lian, and Jun Ma. 2017. Neural attentive session-based recommendation. In *CIKM*. 1419–1428.

[23] 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. In *KDD*. 1734–1743.

[24] Qiao Liu, Yifu Zeng, Refuoe Mokhosi, and Haibin Zhang. 2018. STAMP: short-term attention/memory priority model for session-based recommendation. In *KDD*. 1831–1839.

[25] Fuyu Lv, Taiwei Jin, Changlong Yu, Fei Sun, Quan Lin, Keping Yang, and Wilfred Ng. 2019. SDM: Sequential deep matching model for online large-scale recommender system. In *CIKM*. 2635–2643.

[26] Chen Ma, Peng Kang, and Xue Liu. 2019. Hierarchical gating networks for sequential recommendation. In *KDD*. 825–833.

[27] Hyun Oh Song, Yu Xiang, Stefanie Jegelka, and Silvio Savarese. 2016. Deep metric learning via lifted structured feature embedding. In *CVPR*. 4004–4012.

[28] Wentao Ouyang, Xiuwu Zhang, Li Li, Heng Zou, Xin Xing, Zhaojie Liu, and Yanlong Du. 2019. Deep spatio-temporal neural networks for click-through rate prediction. In *KDD*. 2078–2086.

[29] Michael J Pazzani and Daniel Billsus. 2007. Content-based recommendation systems. In *The adaptive web*. Springer, 325–341.

[30] Qi Pi, Weijie Bian, Guorui Zhou, Xiaoqiang Zhu, and Kun Gai. 2019. Practice on Long Sequential User Behavior Modeling for Click-Through Rate Prediction. In *KDD*.

[31] Qi Qian, Rong Jin, Shenghuo Zhu, and Yuanqing Lin. 2015. Fine-grained visual categorization via multi-stage metric learning. In *CVPR*. 3716–3724.

[32] Massimo Quadrana, Alexandros Karatzoglou, Balázs Hidasi, and Paolo Cremonesi. 2017. Personalizing Session-based Recommendations with Hierarchical Recurrent Neural Networks. In *RecSys*.

[33] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *WWW*. 285–295.

[34] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. 2019. BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer. In *CIKM*. 1441–1450.

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

[36] Jiaxi Tang, Francois Belletti, Sagar Jain, Minmin Chen, Alex Beutel, Can Xu, and Ed H Chi. 2019. Towards neural mixture recommender for long range dependent user sequences. In *WWW*. ACM, 1782–1793.

[37] Jiaxi Tang and Ke Wang. 2018. Personalized top-n sequential recommendation via convolutional sequence embedding. In *WSDM*. 565–573.
[38] Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. 2018. Latent relational metric learning via memory-based attention for collaborative ranking. In *WWW*. 729–739.
[39] Thanh Tran, Renee Sweeney, and Kyumin Lee. 2019. Adversarial Mahalanobis Distance-based Attentive Song Recommender for Automatic Playlist Continuation. In *SIGIR*. 245–254.
[40] Laurens Van Der Maaten. 2014. Accelerating t-SNE using tree-based algorithms. *JMLR* 15, 1 (2014), 3221–3245.
[41] Jiang Wang, Yang Song, Thomas Leung, Chuck Rosenberg, Jingbin Wang, James Philbin, Bo Chen, and Ying Wu. 2014. Learning fine-grained image similarity with deep ranking. In *CVPR*. 1386–1393.
[42] Menghan Wang, Mingming Gong, Xiaolin Zheng, and Kun Zhang. 2018. Modeling dynamic missingness of implicit feedback for recommendation. In *NIPS*. 6669–6678.
[43] Kilian Q Weinberger and Lawrence K Saul. 2009. Distance metric learning for large margin nearest neighbor classification. *JMLR* 10, Feb (2009), 207–244.
[44] Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan. 2019. Session-based recommendation with graph neural networks. In *AAAI*, Vol. 33. 346–353.
[45] Teng Xiao, Shangsong Liang, and Zaiqiao Meng. 2019. Hierarchical Neural Variational Model for Personalized Sequential Recommendation. In *WWW*. 3377–3383.
[46] Ruobing Xie, Cheng Ling, Yalong Wang, Rui Wang, Feng Xia, and Leyu Lin. 2020. Deep Feedback Network for Recommendation. In *IJCAI*. 2519–2525.
[47] Chengfeng Xu, Pengpeng Zhao, Yanchi Liu, Victor S Sheng, Jiajie Xu, Fuzhen Zhuang, Junhua Fang, and Xiaofang Zhou. 2019. Graph contextualized self-attention network for session-based recommendation. In *IJCAI*. 3940–3946.
[48] 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. In *IJCAI*.
[49] Jiaxuan You, Yichen Wang, Aditya Pal, Pong Eksombatchai, Chuck Rosenburg, and Jure Leskovec. 2019. Hierarchical Temporal Convolutional Networks for Dynamic Recommender Systems. In *WWW*. 2236–2246.
[50] Zeping Yu, Jianxun Lian, Ahmad Mahmoody, Gongshen Liu, and Xing Xie. 2019. Adaptive user modeling with long and short-term preferences for personalized recommendation. In *IJCAI*. 4213–4219.
[51] Wei Zhao, Benyou Wang, Jianbo Ye, Yongqiang Gao, Min Yang, and Xiaojun Chen. 2018. PLASTIC: Prioritize Long and Short-term Information in Top-n Recommendation using Adversarial Training.. In *IJCAI*. 3676–3682.
[52] Xiangyu Zhao, Liang Zhang, Zhuoye Ding, Long Xia, Jiliang Tang, and Dawei Yin. 2018. Recommendations with negative feedback via pairwise deep reinforcement learning. In *KDD*. 1040–1048.
[53] Xiao Zhou, Danyang Liu, Jianxun Lian, and Xing Xie. 2019. Collaborative Metric Learning with Memory Network for Multi-Relational Recommender Systems. *arXiv preprint arXiv:1906.09882* (2019).