# Diversifying Personalized Recommendation with User-session Context

**Liang Hu[†], Longbing Cao[†], Shoujin Wang[†], Guandong Xu[†], Jian Cao[‡], Zhiping Gu[*]**
[†]University of Technology, Sydney
[‡]Shanghai Jiao Tong University
[*]Shanghai Technical Institute of Electronics & Information
rainmilk@gmail.com, {longbing.cao, guandong.xu}@uts.edu.au
shoujin.wang@student.uts.edu.au, cao-jian@sjtu.edu.cn, guzhiping@stiei.edu.cn

## Abstract

Recommender systems (RS) have become an integral part of our daily life. However, most current RS often repeatedly recommend items to users with similar profiles. We argue that recommendation should be diversified by leveraging session contexts with personalized user profiles. For this, current session-based RS (SBRS) often assume a rigidly ordered sequence over data which does not fit in many real-world cases. Moreover, personalization is often omitted in current SBRS. Accordingly, a personalized SBRS over relaxedly ordered user-session contexts is more pragmatic. In doing so, deep-structured models tend to be too complex to serve for online SBRS owing to the large number of users and items. Therefore, we design an efficient SBRS with shallow wide-in-wide-out networks, inspired by the successful experience in modern language modelings. The experiments on a real-world e-commerce dataset show the superiority of our model over the state-of-the-art methods.

## 1 Introduction

Nowadays, the ways we make choices in our daily life are highly influenced by recommender systems (RS). Thanks to the suggestions from e-commerce websites and smartphone apps, we spend much less time browsing and deciding what to eat from a huge number of options. The RS behind these websites and apps can be extremely helpful when one is making a choice. However, items which are similar to those we have previously purchased are often repeatedly recommended. In fact, one may prefer more diversified options than those we have had before. For example, it is unlikely that a consumer will purchase another a loaf of bread if they have recently purchased one, whereas butter or ham may be a more appealing recommendation. This indicates that RS would make more sensible and relevant recommendations if the session context was taken into consideration. In this paper, a session can refer to a transaction with clear boundaries, e.g., a purchase record consisting of multiple items or it can also refer to a period within a designated time window, e.g., a day or a week.

Factor models, such as matrix factorization (MF) [Koren *et al.*, 2009], and neighborhood methods, such as item-based collaborative filtering [Sarwar *et al.*, 2001], are the two most prevalent approaches in RS. However, these two approaches are not immediately applicable to session-based RS (SBRS) because they cannot capture the intra-session relevance over items and the inter-session similarity between users. As a result, these RS tend to produce homogeneous recommendations as the historical profile is largely built on the principle of "similarity". In practice, users tend to have different requirements in the context of changing sessions; therefore homogeneous recommendation may greatly degrade user satisfaction and business benefits. To diversify the recommendation results for different sessions, a RS needs to learn the "relevance" between candidate items and a given context instead of pure "similarity".

Sequential pattern mining (SPM) is a method at hand to capture the relevance between items [Han *et al.*, 2000]. SPM extracts a set of association rules according to the co-occurrence frequency in the historical data. However, a session context may consist of an arbitrary set of items in the recommendation problem so it probably fails to match any pattern from the extracted association rule space. Markov chain (MC) is another straightforward way to model sequential data [Gu *et al.*, 2014]. However, MC only captures the first-order dependency, i.e. it predicts the transition between a pair of items instead of that between an item and a contextual item set. Deep learning has achieved tremendous success in a number of tasks, including RS [Salakhutdinov *et al.*, 2007]. Recently, Hidasi *et al.* [2015] applied recurrent neural networks (RNN) for SBRS. Both MC and RNN are originally applied on time series data with a natural order. However, the choices of items in a session may not follow a "rigidly ordered sequence", for example, the order in which toast, milk and ham are put into a shopping cart makes no difference to the transaction. Moreover, most real-world datasets do not provide precise timestamps on items purchased. As a result, MC and RNN may produce misleading outputs.

Language modeling is the probability distribution over sequences of words in natural language processing (NLP). Both the number of items in RS and the size of vocabulary in language modeling are large, usually $10^5$-$10^7$, thereby it is too time-consuming to train a deep structure with a large number of input and output units. If we think of words as items, predicting a relevant word based on context is equivalent to recommending a relevant item according to the current session. This observation inspires us to incorporate language modeling ideas into SBRS. Word2vec [Mikolov *et al.*, 2013a; 2013b], which originated from neutral probabilistic language modeling [Bengio *et al.*, 2003], is a highly successful word embedding model in recent years. Although word2vec is regarded as a deep learning model due to its origin, it ignores all hidden layers but directly links the input and output layers with bilinear connections. Thus, word2vec is not deep, instead it is a *shallow and wide network* [Goth, 2016] which makes learning more efficient over this wide-in-wide-out (WIWO) structure. However, word2vec is not ready for SBRS due to the lack of an essential element for RS, that is, users. In this work, we propose to model user-session contexts with both users and items.

Considering the large number of items and users, we design a shallow network with double wide-in vectors, in which one indicates items and the other indicates users, and a wide-out vector to indicate the items relevant to the user-session context. This structure makes it more efficient to learn personalized preferences from users' session profiles. The main contributions of this paper include:

- We propose to diversify personalized recommendation results according to user-session contexts.
- Session-based WIWO (SWIWO) networks are designed to efficiently learn session profiles over the large number of users and items.
- We conducted empirical evaluations on a real-world data set. The results demonstrate the superiority of our approach in comparison with state-of-the-art methods.

## 2 Related Work

First, it needs to be noted that both time-aware RS [Campos *et al.*, 2014] and session-based RS are context-aware RS (CARS) but they target different goals. Time-aware RS consider temporal factors when making recommendations, for example, how time information (e.g. weekday vs. weekend) affects the recommendation [Baltrunas and Amatriain, 2009]. Temporal effects are another research area of time-aware RS. Koren [2009] studied the evolution of users and movies over time. In comparison, session-based RS do not need precise timestamps, as they focus on learning the relevance between items in a session scope.

Markov models are a straightforward way to model sequential data. Markov decision processes (MDP) [Shani *et al.*, 2005] are an early approach to provide recommendations in a session-based manner. Gu *et al.* [2014] introduce purchase sequence prediction based on the hidden Markov model (HMM) and purchase intervals. Since the number of items is large, an issue to apply MDP and HMM is that the state space quickly becomes unmanageable when evaluating all possible sequences over all items. Chen *et al.* [2012] model playlists as an MC, and propose to represent songs by logistic Markov embedding (LME). Personalized metric embedding (PME) [Wu *et al.*, 2013] and personalized ranking metric embedding (PRME) [Feng *et al.*, 2015] extend LME by adding personalization. LME, PME and PRME are first-order MC models built on rigid sequential data to model the transition between consequent choices.

Classic neighborhood methods and factor models do not consider session context, but researchers have developed extensions to incorporate with session information. Session-based collaborative filtering [Park *et al.*, 2011] firstly finds the *k*-nearest neighbors (*k*nn) for the current session and then calculates the score for each potential item to generate the rank. The number of historical sessions is huge so finding the *k*-nearest sessions online is infeasible for real RS. Factorized personalized Markov chains (FPMC) [Rendle *et al.*, 2010] combine the power of MF and MC to model personalized sequential behavior. The same as MF, FPMC easily suffers from the data sparsity issue.

In recent years, RS have begun to embrace the deep learning technology [Salakhutdinov *et al.*, 2007]. RNN have been devised to model variable-length sequence data, where the internal state of the network allows it to exhibit dynamic temporal behavior. Hidasi *et al.* [2015] applied RNN consisting of gated recurrent units as session-based RS, namely GRU4Rec. Twardowski [2016] used a similar RNN for ad click prediction. Although language modeling in NLP is not literally related to session-based RS, we find that they share common underlying problem, that is, learning the probability distribution over sequences of words is similar to learning that over sequences of items. Word embedding models [Mnih and Teh, 2012; Mnih and Kavukcuoglu, 2013], especially word2vec [Mikolov *et al.*, 2013a; 2013b], have achieved great success in NLP. Moreover, due to the large vocabulary size, word embedding models do not evolve to a deep structure; on the contrary, they become shallow and wide [Goth, 2016] to more efficiently adapt large-class data. In view of the large number of items and users in SBRS, we incorporate some successful ideas from word embedding models and propose SWIWO networks to learn and infer personalized preferences with user-session contexts.

## 3 Problem Formulation

Before introducing the specificities of our model, we first formulate the problem and clarify basic concepts in this section.

In general, user set $\mathbf{U} = \{u_1, u_2, \cdots, u_{|\mathbf{U}|}\}$ and item set $\mathbf{V} = \{v_1, v_2, \cdots, v_{|\mathbf{V}|}\}$ are two basic elements in RS. As to SBRS, $\mathbf{S} = \{s_1, s_2, \cdots, s_{|\mathbf{S}|}\}$ denotes session set containing all observed sessions, where each session consists of a subset of items, $s_i \subset \mathbf{V}$. Our SWIWO networks are constructed and trained as probabilistic classifiers that learn to predict a conditional probability distribution $P(v_t|\mathbf{c})$ where $\mathbf{c} \subseteq s \setminus v_t$ is the context of a session $s \in \mathbf{S}$ w.r.t. the relevant item $v_t$.

Similar to our problem, neural language models train a classifier to predict a conditional probability distribution $P(w_t|\mathbf{c}_t)$ where $\mathbf{c}_t$ is the context of the given word $w_t$. In word2vec models [Mikolov *et al.*, 2013a], a context window is used to include the words before and after $w_t$ as the context, $\mathbf{c}_t = \{w_{t-k}, \cdots, w_{t-1}, w_{t+1}, \cdots, w_{t+k}\}$. As stated at the beginning, a session could be generated by a moving time window $s = \{v_{t-k}, \cdots, v_{t+k}\}$, and then the corresponding context is $\mathbf{c} = \{v_{t-k}, \cdots, v_{t-1}, v_{t+1}, \cdots, v_{t+k}\}$ as that in word2vec. If sessions are split transactions, then the session context is $\mathbf{c} = s \setminus v_t$ w.r.t. each relevant item $v_t \in s$ (leave-one-out), therefore, we can construct $|s|$ possible training ex-

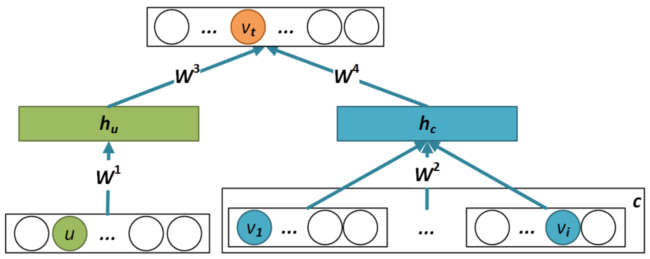

Figure 1: The architecture of SWIWO networks. This model predicts the relevant item based on the user-session context.

amples for each session $s \in \mathbf{S}$.

Since we are designing a personalized SBRS, we add user as a part of the context, $\mathbf{c}_u \equiv \langle u, \mathbf{c}\rangle$. Finally our SWIWO networks are refined to train the classifier over the conditional distribution $P(v_t|u, \mathbf{c})$ when this user-session context $\langle u, \mathbf{c}\rangle$ is given. Correspondingly, the personalized recommendation problem is reduced to generate the ranking over all candidate items given the online user-session context $\langle u', \mathbf{c}'\rangle$. Thus, we can do it by sorting $\{P(v|u', \mathbf{c}')|\forall v \in \mathbf{V}\}$ when the SWIWO networks are trained.

## 4 Modeling and Learning

In this section, we first demonstrate the model architecture of SWIWO networks, and then we discuss the details of parameter learning. Finally, we briefly present how to generate recommendation using the trained model.

### 4.1 SWIWO Architecture

Figure 1 illustrates the architecture of our SWIWO networks, which learn and train a probabilistic classifier to predict the most relevant item for a given user-session context. In brief, SWIWO is a three-layer network where the input layer encodes the raw user-session context, the hidden layer is the low-dimension embedding representation of the raw user-session context, and the output layer indicates the most relevant item based on the user-session context embeddings.

More specifically, we first illustrate how to encode a raw user-session context for the input layer. Given the user-session context $\langle u, \mathbf{c}\rangle$, the input units in the bottom-left corner of Figure 1 are a *one-hot* encoded vector, which means only the unit at position $u$ is set 1 and all other units are 0. Moreover, there are $|\mathbf{c}|$ items in the session context $\mathbf{c}$, for each $v \in \mathbf{c}$, it is encoded by a one-hot vector as depicted in the bottom-right corner of Figure 1. Thus, the input layer for each training example consists of a user vector with a length $|\mathbf{U}|$, and $|\mathbf{c}|$ item vectors with a length $|\mathbf{V}|$.

In RS, users and items are indexed by meaningless IDs, which is similar to the words in NLP systems. Word embedding models map per word from an ID space to a continuous lower-dimension vector space so as to work with this more informative representation of words instead of IDs. In fact, the latent-factor vector representation for users and items in MF is the counterpart of embedding in neural models although they have different names. In SWIWO, we create an embedding layer to map those sparse one-hot user and item vectors to an informative representation. In the hidden layer, we use $K$-dimension vector $\mathbf{h}_u \in [0,1]^K$ to represent the user embeddings. The weight matrix $\mathbf{W}^1 \in \mathbb{R}^{K\times|\mathbf{U}|}$ is fully connected between input-layer units and hidden-layer units, where the $u$-th column $\mathbf{W}^1_{:,u}$ encodes the one-hot vector w.r.t. user $u$ to the embeddings $\mathbf{h}_u$ in terms of logistic function $\sigma(\cdot)$.

$$\mathbf{h}_u = \sigma\left(\mathbf{W}^1_{:,u}\right) \tag{1}$$

Different from word2vec, we use nonlinear embeddings instead of linear ones. This is because the nonlinear embeddings are bounded in $[0, 1]$ which makes the training and prediction more stable. Furthermore, nonlinear representation is generally more expressive than linear one, but it may involve a little more computational overhead.

Similarly, we use the weight matrix $\mathbf{W}^2 \in \mathbb{R}^{L\times|\mathbf{V}|}$ to encode a one-hot item vector to the embeddings $\mathbf{h}_v \in [0,1]^L$.

$$\mathbf{h}_v = \sigma\left(\mathbf{W}^2_{:,v}\right) \tag{2}$$

Then, we can obtain the embeddings of session context, $\mathbf{h}_c \in [0,1]^L$, by combining all the embeddings of items in this context. As illustrated in Eq. 3, we construct the context embeddings $\mathbf{h}_v$ as a mixture of $\{\mathbf{h}_v|v \in \mathbf{c}\}$.

$$\mathbf{h}_c = \sum_{v\in\mathbf{c}} w_v \mathbf{h}_v \equiv \sum_{v\in\mathbf{c}} w_v \sigma\left(\mathbf{W}^2_{:,v}\right) \tag{3}$$

where $\sum_{v\in\mathbf{c}} w_v = 1$. These mixture weights $w_v$ are often assigned different values according to specific applications. For the experiments in this paper, we use the exponential decay:

$$w_v \propto \exp[-\lambda(|v-t|-1)] \tag{4}$$

where $|v - t|$ is the span between a context item $v_v$ and the target item $v_t$ in a session $s$ (cf. Section 3). Obviously, the context items previous and next to the target item $v_t$, i.e. $v_{t-1}$ and $v_{t+1}$, have the largest unnormalized weight 1, and those context items farther from $v_t$ are assigned smaller weights. We find that it produces good results in our experiments by setting $\lambda = 0.75$.

The weight matrices $\mathbf{W}^3 \in \mathbb{R}^{|\mathbf{V}|\times K}$ and $\mathbf{W}^4 \in \mathbb{R}^{|\mathbf{V}|\times L}$ fully connect the embedding layer to the output layer as depicted in the top of Figure 1. Then, we can compute the score $S_{v_t}$ of a target item $v_t$ w.r.t. the user-session context $\langle u, \mathbf{c}\rangle$ in terms of the context embeddings $\langle \mathbf{h}_u, \mathbf{h}_c\rangle$

$$S_{v_t}(u, \mathbf{c}) = \mathbf{W}^3_{t,:}\mathbf{h}_u + \mathbf{W}^4_{t,:}\mathbf{h}_c \tag{5}$$

where $\mathbf{W}_{t,:}$ denotes the $t$-th row of $\mathbf{W}$. This score function quantifies the compatibility of the target item $v_t$ with the user-session context. As a result, the conditional distribution $P_\Theta(v_t|u, \mathbf{c})$ can be defined in terms of a softmax function

$$P_\Theta(v_t|u, \mathbf{c}) = \frac{\exp(S_{v_t}(u, \mathbf{c}))}{Z(u, \mathbf{c})} \tag{6}$$

where $Z(u, \mathbf{c}) = \sum_{v\in\mathbf{V}} \exp(S_{v_t}(u, \mathbf{c}))$ is the normalizing constant and $\Theta = \left\{\mathbf{W}^1, \mathbf{W}^2, \mathbf{W}^3, \mathbf{W}^4\right\}$ defines the model parameter set. So far, we obtain a probabilistic classifier modeled by the SWIWO networks.

## 4.2 Learning and Inference

In the previous subsection, we describe the construction of a probabilistic classifier over the user session data $d = \langle \mathbf{c}_u, v_c \rangle$, where $\mathbf{c}_u \equiv \langle u_c, \mathbf{c} \rangle$ is the input, namely user-session context, and $v_c$ is the observed output, namely a relevant item to this context. Given a training dataset $\mathcal{D} = \{\langle \mathbf{c}_u, v_c \rangle\}$, we easily obtain the joint probability distribution over it

$$P_\Theta(\mathcal{D}) \propto \prod_{d \in \mathcal{D}} P_\Theta(v_c | u_c, \mathbf{c}) \tag{7}$$

Therefore, we can learn the model parameters $\Theta$ by maximizing the conditional log-likelihood (cf. Eq. 6)

$$L_\Theta = \sum_{d \in \mathcal{D}} \log P_\Theta(v_c | u_c, \mathbf{c}) = \sum_{d \in \mathcal{D}} S_{v_c}(u_c, \mathbf{c}) - \log Z(u_c, \mathbf{c}) \tag{8}$$

Both evaluating $L_\Theta$ and computing the corresponding log-likelihood gradient involves the normalizing term $Z(u_c, \mathbf{c})$, which needs to sum $\exp(S_{v_c}(u_c, \mathbf{c})$ over the entire item set for each training example. This means that training this model needs to take time $|\mathbf{V}| \times |\mathcal{D}|$ to compute the normalizing constants over all the training examples for each iteration. Unfortunately, $|\mathbf{V}|$ and $|\mathcal{D}|$ are always large in real RS, which makes the training process intractable.

**Softmax Approximation**

The vocabulary size in NLP systems is also large so language modeling runs into the same challenge as our problem [Mnih and Teh, 2012; Mikolov *et al.*, 2013b]. In this paper, we adopt a subsampling approach to do away with the softmax layer, namely noise-contrastive estimation (NCE) [Gutmann and Hyvärinen, 2012] which was proposed for training unnormalized probabilistic models. This approach avoids approximating the normalization of the softmax, instead it works with some other objective that is much cheaper to compute.

The basic idea of NCE is to apply a binary classifier to discriminate between samples from the data distribution and samples from a known noise distribution $Q$. Given a training example $\langle \mathbf{c}_u, v_c \rangle$, we can represent the probability of sampling from either a positive example or $K$ noise examples as a mixture of those two distributions [Mnih and Teh, 2012]:

$$P_\Theta(y | v_c, \mathbf{c}_u) = \frac{1}{K+1} P_\Theta(v_c | \mathbf{c}_u) + \frac{K}{K+1} Q(v_c)$$

Then the posterior probability that sample $v_c$ came from the data distribution, namely a positive example, is

$$\begin{aligned} P_\Theta(y = 1 | v_c, \mathbf{c}_u) &= \frac{P_\Theta(v_c | \mathbf{c}_u)}{P_\Theta(v_c | \mathbf{c}_u) + K Q(v_c)} \\ &\approx \frac{\exp(S_{v_c}(\mathbf{c}_u))}{\exp(S_{v_c}(\mathbf{c}_u)) + K Q(v_c)} \end{aligned} \tag{9}$$

where the normalizing term $Z(\mathbf{c}_u)$ is dropped from $P_\Theta(v_c | \mathbf{c}_u)$. We can do this because the NCE is an unnormalized estimator where the objective encourages $P_\Theta(v_c | \mathbf{c}_u)$ the model to be approximately self-normalized [Gutmann and Hyvärinen, 2012]. Then, the probability of $v_c$ coming from the noise samples is simply $P_\Theta(y = 0 | v_c, \mathbf{c}_u) = 1 - P_\Theta(y = 1 | v_c, \mathbf{c}_u)$. As a result, we can maximize this log-likelihood of the training example against $K$ noise samples [Mnih and Kavukcuoglu, 2013]:

$$\begin{aligned} & J_\Theta(v_c, \mathbf{c}_u) \\ &= \log P_\Theta(y = 1 | v_c, \mathbf{c}_u) + K \mathbb{E}_{\tilde{v}_k \sim Q} [\log P_\Theta(y = 0 | v_c, \mathbf{c}_u)] \\ &\approx \log P_\Theta(y = 1 | v_c, \mathbf{c}_u) + \sum_{k=1}^{K} \log P_\Theta(y = 0 | \tilde{v}_k, \mathbf{c}_u) \end{aligned} \tag{10}$$

Substituting Eq. 9 into Eq.10, we can immediately obtain the gradient of $J_\Theta(v_c, \mathbf{c}_u)$. This gradient approaches the maximum log-likelihood gradient (cf. Eq. 8) with $K$ increasing [Mnih and Teh, 2012].

$$\begin{aligned} \nabla J_\Theta(v_c, \mathbf{c}_u) &= \frac{K Q(v_c)}{\exp(S_{v_c}(\mathbf{c}_u) + K Q(v_c)} \nabla S_{v_c}(\mathbf{c}_u) \\ &- \sum_{k=1}^{K} \frac{\exp(S_{\tilde{v}_k}(\mathbf{c}_u))}{\exp(S_{\tilde{v}_k}(\mathbf{c}_u)) + K Q(\tilde{v}_k)} \nabla S_{\tilde{v}_k}(\mathbf{c}_u) \end{aligned} \tag{11}$$

Note that the selection of noise distribution may depend on the context. For example, users may click on some other similar items before they add a satisfying item to their cart. Then, these items which have been clicked most can be assigned lower probability as the noise samples against the item in the cart since users also have shown some interest in them. In this paper, we adopt a strategy that has often been used in RS to construct the noise distribution [Hu *et al.*, 2014b]

$$Q(v) \propto f(\#v) \tag{12}$$

where $\#v$ denotes the frequency of $v$ over all sessions and $f(\cdot)$ is a function to scale $\#v$ , such as log or squared root function. The assumption behind this is that the frequent items, e.g. water, tend to be less dependent on other items and users often intentionally choose them or not [Hu *et al.*, 2014b]. Similar strategy has been applied in word2vec to select negative words since high frequency words often provide little information [Mikolov *et al.*, 2013b].

**Learning and Ranking**

Given the gradient of $J_\Theta(v_c, \mathbf{c}_u)$ as illustrated in Eq. 11, we can learn all the parameters $\Theta$ by backpropogation. The gradient of $S_v(\mathbf{c}_u)$ w.r.t. each $\mathbf{W} \in \Theta$ is given below

$$\nabla_{\mathbf{W}^3_{v_o,:}} S_{v_o}(\mathbf{c}_u) = \mathbf{h}_u^\top \tag{13}$$

$$\nabla_{\mathbf{W}^4_{v_o,:}} S_{v_o}(\mathbf{c}_u) = \mathbf{h}_c^\top \tag{14}$$

$$\nabla_{\mathbf{W}^1_{:,u}} S_{v_o}(\mathbf{c}_u) = \mathbf{W}^{3\top}_{v_o,:} \odot \mathbf{h}_u \odot (1 - \mathbf{h}_u) \tag{15}$$

$$\nabla_{\mathbf{W}^2_{:,v_i}} S_{v_o}(\mathbf{c}_u) = w_{v_i} \mathbf{W}^{4\top}_{v_o,:} \odot \mathbf{h}_u \odot (1 - \mathbf{h}_u) \tag{16}$$

where $\odot$ denotes the element-wise product, $v_o$ is the output item which includes the positive example $v_c$ and all noise examples $\{\tilde{v}_k\}$, and $v_i \in \mathbf{c}$ is the input item in the session context and $w_{v_i}$ is the corresponding mixture weight (cf. Eq. 3, 4). We thus can substitute these gradients into Eq. 11 and obtain the gradient-based update equation for each parameter. Considering the strong power of modern GPUs on matrix multiplication, we design a GPU-based adaptive stochastic gradient descent (SGD, actually ascent) optimizer over mini-batches to speed up the training. Due to the limited

**Algorithm 1** A GPU-based SGD Optimizer for SWIWO

```
1: $\mathcal{M} \leftarrow$ getMinibatch();
2: $\mathbf{H}_u, \mathbf{H}_c \leftarrow$ Rearrange all the embeddings $\{\mathbf{h}_u, \mathbf{h}_c\}$ (cf. Eq. 1, 3) of $\mathcal{M}$ as two embedding matrices on GPUs;
3: $\nabla J_{\mathbf{W}}(\mathcal{M}) \leftarrow$ Using $\mathbf{H}_u, \mathbf{H}_c$ to perform matrix multiplication on GPUs to compute the gradient w.r.t. each parameter $\mathbf{W} \in \Theta$, cf. Eq. 10 and Eq. 13-16;
4: $\mathbf{W} \leftarrow \mathbf{W} + \Gamma(\nabla J_{\mathbf{W}}(\mathcal{M}))$, for each $\mathbf{W} \in \Theta$;
```

space, a brief scheme of this procedure on a mini-batch is demonstrated in Algorithm 1, where $\Gamma(\cdot)$ is a function to assign adaptive learning rate, e.g. AdaGrad, RMSProp, Adam [Ruder, 2016]. Our experimental results were obtained using Adam. The MATLAB implementation[1] of Algorithm 1 is provided online for more details.

When the model has been trained, we can immediately use it as a personalized SBRS. Given an arbitrary user-session context $\mathbf{c}_u$, we can compute the scores over all candidate items according to Eq. 5, and then rank them accordingly.

## 5 Experiments

We used the IJCAI-15 competition dataset[2] for our experiments. This real-world dataset was collected from Tmall.com which is the largest online B2C platform in China, and it contains anonymized users' shopping logs for the six months before and on the "Double 11" day (November 11th).

People have realized the harmfulness of evaluating RS only using accuracy metrics [Ge *et al.*, 2010]. Recall that we propose to diversify recommendation for different user-session context to replace similarity-based recommendation over history profile. Therefore, we respectively evaluated our model and other comparison methods from the perspectives of accuracy and diversity, but they often cannot be optimized simultaneously [Zhou *et al.*, 2010].

### 5.1 Comparison Methods

We used the following representative state-of-the-art methods for the experiments, most of them being introduced in related work, including our approach:

- **POP**: This recommender simply ranks items for recommendation according to occurrence frequency.
- **FPMC** [Rendle *et al.*, 2010]: This recommender is a combination of MF and first-order MC, which uses personalized MC for sequential prediction.
- **PRME** [Feng *et al.*, 2015]: This recommender learns personalized transition probability in a MC model by applying a pairwise embedding metric method to handle data sparsity.
- **GRU4Rec** [Hidasi *et al.*, 2015]: This recommender is a deep RNN which consists of GRU units.
- **SWIWO**: This is the full model proposed in this paper.
- **SWIWO-I**: This a sub-model of SWIWO which only models item-session contexts without considering users.

[1]https://github.com/rainmilk/ijcai17swiwo

[2]https://tianchi.shuju.aliyun.com/datalab/dataSet.htm?id=1

Table 1: Statistic of IJCAI-15 dataset for evaluation

| | |
|---|---|
| #users: | 50K |
| #items: | 52K |
| avg. session length: | 2.99 |
| #training sessions: | 0.20M |
| #training examples: | 0.59M |
| #testing cases (*LAST*): | 4.5K |
| #testing cases (*LOO*): | 11.9K |

### 5.2 Data Preparation

First, we removed those users who have less than three shopping sessions and we removed all singleton sessions, i.e. only containing one item, from the raw data. Note that this dataset only provides buying date without a specific time, so we treat a user's shopping records in a natural day as a session. For intra-day sequence, we retain the order given in the raw data. From the six-month shopping logs, we randomly held out 20% of the sessions from the last 30 days for testing, and the remaining data are used for training. In particular, we constructed two testing sets: *LAST* means that the last item in each testing sessions is used as ground truth, and *LOO* means each item in a testing session is held out in turn to serve as ground truth, i.e. leave-one-out. The statistics of this dataset for evaluation are summarized in Table 1.

### 5.3 Accuracy Evaluation

We use the following widely used accuracy metrics for SBRS to evaluate all the comparison methods.

- **REC@K**: This measures the recall from the top-*K* items over all the testing samples.
- **MRR**: This measures the mean reciprocal rank of the predictive positions over all the testing samples.

**Results**

Table 2 demonstrates the result of REC@10, REC@20 and MRR over the testing sets *Last* and *LOO*. Here, we chose $K \in \{10, 20\}$ because most users are only interest in viewing the recommendation on the first page in the real-world RS. In fact, finding exactly one true item from more than $10^5$ items is a big challenge. POP achieves reasonable results, which reflects common user behavior for online shopping, that is, users tend to choose items with high sales volume, i.e. popularity. We set the number of factors to 5 for training FPMC, and the results become worse when more factors are used. This is because the dataset is too sparse for MF methods where each row only contains less than two items (cf. the *avg. session length* in Table 1, note that one of the items needs to be used as the output) and the others are all empty (cf. *#items* in Table 1). Furthermore, most users only have three sessions although we have removed users with too few sessions. Therefore, this constructs a too sparse matrix to train the MF model. We trained PRME by setting the embedding dimension to 50. PRME lags far behind GRU4Rec and our models. This is because PRME is a first-order MC model, which learns the transitions over successive items instead of the whole context; moreover, the choice of items does not follow a rigid sequence. Thanks to deep technology, GRU4Rec

Table 2: The evaluation results for accuracy metrics

| *LAST* | | | |
|---|---|---|---|
| **Model** | **REC@10** | **REC@20** | **MRR** |
| *POP* | 0.0185 | 0.0317 | 0.0104 |
| *FPMC* | 0.0023 | 0.0068 | 0.0021 |
| *PRME* | 0.0670 | 0.0821 | 0.0363 |
| *GRU4Rec* | 0.2283 | 0.2464 | 0.1586 |
| *SWIWO-I* | **0.3223** | **0.3797** | **0.1918** |
| *SWIWO* | **0.3131** | **0.3689** | **0.1896** |

| *LOO* | | | |
|---|---|---|---|
| **Model** | **REC@10** | **REC@20** | **MRR** |
| *POP* | 0.0234 | 0.0420 | 0.0123 |
| *FPMC* | 0.0064 | 0.0117 | 0.0044 |
| *PRME* | 0.0757 | 0.0976 | 0.0431 |
| *GRU4Rec* | 0.2242 | 0.2425 | 0.1574 |
| *SWIWO-I* | **0.3177** | **0.3810** | **0.1903** |
| *SWIWO* | **0.3082** | **0.3703** | **0.1885** |

achieves a qualitative leap. In both testing cases the REC@10 of GRU4Rec are above 20%, which can result in very accurate recommendations in real-world systems.

We set 50 units for the context embeddings and 10 units for the user embeddings when training SWIWO-I and SWIWO models. Surprisingly, our models outperform GRU4Rec by clear margins, where the REC@10 are above 30% and the REC@20 are above 35% for both testing cases. The highest MRR also proves that our models can accurately list users' desired items on the first page. Most importantly, our models have a very concise structure which can be easily trained (cf. Algorithm 1). Furthermore, this shallow structure is efficient to recompute the scores over all candidate items when many session contexts keep updating in online SBRS. In comparison, GRU4Rec is a deep RNN consisting of GRU layers. As a result, it is more time expensive than SWIWO to recompute the scores for ranking in online SBRS. Moreover, SWIWO-I and SWIWO achieve very close performance, where SWIWO does not show improvement on accuracy by the incorporation of users. However, our main aim of incorporating users is to diversify personalized recommendations.

## 5.4 Diversity Evaluation

Since we aim to diversify recommendation with session context, we consider the following metrics.

- **DIV@K**: This diversity measures the mean non-overlap ratio between each pair of recommendations $\langle \mathbf{R}_i, \mathbf{R}_j \rangle$ over all $N$ top-$K$ recommendations (note that the number of all possible pairs is $N(N-1)/2$).

$$DIV@K = \frac{2}{N(N-1)} \sum_{i \neq j} \left( 1 - \frac{|\mathbf{R}_i \bigcap \mathbf{R}_j|}{|\mathbf{R}_i \bigcup \mathbf{R}_j|} \right)$$

- **F1@K**: The traditional F1 score is the harmonic mean of recall and precision. Here, we replace precision with diversity to jointly consider accuracy and diversity.

$$F1@K = \frac{2(REC@K \times DIV@K)}{REC@K + DIV@K}$$

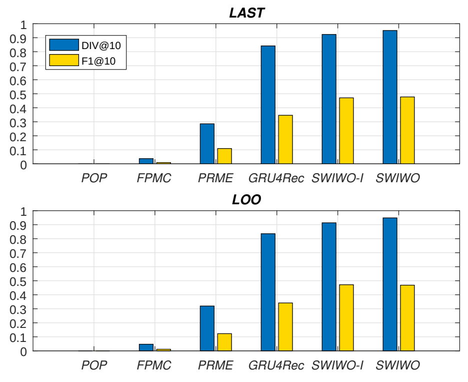

Figure 2: DIV@10, F1@10 on *LAST* and *LOO*

**Results**

Figure 2 illustrates the results of the top-10 recommendations over testing sets *Last* and *LOO*. POP always recommends the same items for all sessions so it has zero diversity and F1. FPMC does not learn its parameters well on this dataset, and it outputs very similar recommendations. Accordingly, we get low diversity and low F1 for FPMC. PRME is a first-order MC model which makes recommendations only considering the previous item in the session context, so it tends to generate low diversity results. The GRU units can accumulate the influence of the sequential items. Consequently, GRU4Rec is a good SBRS to generate high diversity recommendations.

Our models consider the whole session context so they more easily provide diverse recommendation results. Since the REC@10 scores of our models are also high as shown in the previous experiment, we accordingly get high F1@10 scores, which validates the effectiveness of SWIWO networks based SBRS. In particular, from Figure 2, we find that SWIWO generates more diverse recommendations than those generated by SWIWO-I. This is because SWIWO takes users into account, which makes personalized recommendations with the user-session contexts. In comparison, SWIWO-I does not model users, and ignores personalizing when making recommendations. As a result, we conclude that SWIWO is the best model to serve as real-world SBRS, jointly considering accuracy and diversity.

## 6 Conclusion

In this paper, we propose SWIWO networks to build a more efficient personalized SBRS. The empirical evaluation on a real-world e-commerce dataset proves the comprehensive superiority of our approach over other state-of-the-art methods. With minor modification, we can immediately apply SWIWO as a group-based RS [Hu *et al.*, 2014a] where the context embeddings are the representation of a group of users. Moreover, SWIWO is a general architecture so it can be applied in many other domains besides RS, for example the author-topic model [Rosen-Zvi *et al.*, 2004] in NLP.

## Acknowledgments

This work is partially sponsored China National Science Foundation (Granted Number 61472253).

## References

[Baltrunas and Amatriain, 2009] Linas Baltrunas and Xavier Amatriain. Towards time-dependant recommendation based on implicit feedback. In *Workshop on context-aware recommender systems (CARS)*, 2009.

[Bengio *et al.*, 2003] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. A neural probabilistic language model. *J. Mach. Learn. Res.*, 3:1137–1155, March 2003.

[Campos *et al.*, 2014] Pedro G. Campos, Fernando Dez, and Ivn Cantador. Time-aware recommender systems: a comprehensive survey and analysis of existing evaluation protocols. *User Modeling and User-Adapted Interaction*, 24(1):67–119, 2014.

[Chen *et al.*, 2012] Shuo Chen, Josh L. Moore, Douglas Turnbull, and Thorsten Joachims. Playlist prediction via metric embedding. KDD '12, pages 714–722, New York, NY, USA, 2012. ACM.

[Feng *et al.*, 2015] Shanshan Feng, Xutao Li, Yifeng Zeng, Gao Cong, Yeow Meng Chee, and Quan Yuan. Personalized ranking metric embedding for next new poi recommendation. In *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, pages 2069–2075. AAAI Press, 2015.

[Ge *et al.*, 2010] Mouzhi Ge, Carla Delgado-Battenfeld, and Dietmar Jannach. Beyond accuracy: evaluating recommender systems by coverage and serendipity. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 257–260. ACM, 2010.

[Goth, 2016] Gregory Goth. Deep or shallow, nlp is breaking out. *Commun. ACM*, 59(3):13–16, February 2016.

[Gu *et al.*, 2014] Wanrong Gu, Shoubin Dong, and Zhizhao Zeng. Increasing recommended effectiveness with markov chains and purchase intervals. *Neural Computing and Applications*, 25(5):1153–1162, 2014.

[Gutmann and Hyvärinen, 2012] Michael U Gutmann and Aapo Hyvärinen. Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. *Journal of Machine Learning Research*, 13(Feb):307–361, 2012.

[Han *et al.*, 2000] Jiawei Han, Jian Pei, Behzad Mortazavi-Asl, Qiming Chen, Umeshwar Dayal, and Mei-Chun Hsu. Freespan: Frequent pattern-projected sequential pattern mining. KDD '00, pages 355–359. ACM, 2000.

[Hidasi *et al.*, 2015] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. Session-based recommendations with recurrent neural networks. *CoRR*, abs/1511.06939, 2015.

[Hu *et al.*, 2014a] Liang Hu, Jian Cao, Guandong Xu, Longbing Cao, Zhiping Gu, and Wei Cao. Deep modeling of group preferences for group-based recommendation. In *Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2014.

[Hu *et al.*, 2014b] Liang Hu, Wei Cao, Jian Cao, Guandong Xu, Longbing Cao, and Zhiping Gu. Bayesian heteroskedastic choice modeling on non-identically distributed linkages. In *Data Mining (ICDM), 2014 IEEE International Conference on*, pages 851–856, 2014.

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

[Koren, 2009] Yehuda Koren. Collaborative filtering with temporal dynamics. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '09, pages 447–456. ACM, 2009.

[Mikolov *et al.*, 2013a] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013.

[Mikolov *et al.*, 2013b] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems*, pages 3111–3119. Curran Associates Inc., 2013.

[Mnih and Kavukcuoglu, 2013] Andriy Mnih and Koray Kavukcuoglu. Learning word embeddings efficiently with noise-contrastive estimation. In *Advances in Neural Information Processing Systems 26*, pages 2265–2273. 2013.

[Mnih and Teh, 2012] Andriy Mnih and Yee Whye Teh. A fast and simple algorithm for training neural probabilistic language models. In *In Proceedings of the International Conference on Machine Learning*, 2012.

[Park *et al.*, 2011] S. E. Park, S. Lee, and S. g. Lee. Session-based collaborative filtering for predicting the next song. In *2011 First ACIS/JNU International Conference on Computers, Networks, Systems and Industrial Engineering*, pages 353–358, May 2011.

[Rendle *et al.*, 2010] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. Factorizing personalized markov chains for next-basket recommendation. In *Proceedings of the 19th international conference on World wide web*, pages 811–820. ACM, 2010.

[Rosen-Zvi *et al.*, 2004] Michal Rosen-Zvi, Thomas Griffiths, Mark Steyvers, and Padhraic Smyth. The author-topic model for authors and documents. In *Proceedings of the 20th conference on Uncertainty in artificial intelligence*, pages 487–494. AUAI Press, 2004.

[Ruder, 2016] Sebastian Ruder. An overview of gradient descent optimization algorithms. *CoRR*, abs/1609.04747, 2016.

[Salakhutdinov *et al.*, 2007] Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey Hinton. Restricted boltzmann machines for collaborative filtering. In *Proceedings of the 24th international conference on Machine learning*, pages 791–798. ACM, 2007.

[Sarwar *et al.*, 2001] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, pages 285–295. ACM, 2001.

[Shani *et al.*, 2005] Guy Shani, David Heckerman, and Ronen I. Brafman. An mdp-based recommender system. *J. Mach. Learn. Res.*, 6:1265–1295, December 2005.

[Twardowski, 2016] Bartlomiej Twardowski. Modelling contextual information in session-aware recommender systems with neural networks. In *Proceedings of the 10th ACM Conference on Recommender Systems*, RecSys '16, pages 273–276. ACM, 2016.

[Wu *et al.*, 2013] Xiang Wu, Qi Liu, Enhong Chen, Liang He, Jingsong Lv, Can Cao, and Guoping Hu. Personalized next-song recommendation in online karaokes. In *Proceedings of the 7th ACM Conference on Recommender Systems*, RecSys '13, pages 137–140, New York, NY, USA, 2013. ACM.

[Zhou *et al.*, 2010] Tao Zhou, Zoltn Kuscsik, Jian-Guo Liu, Mat Medo, Joseph Rushton Wakeling, and Yi-Cheng Zhang. Solving the apparent diversity-accuracy dilemma of recommender systems. In *Proceedings of the National Academy of Sciences*, volume 107, pages 4511–4515, 2010.