The Thirty-Second AAAI Conference
on Artificial Intelligence (AAAI-18)

Attention-Based Transactional Context Embedding for Next-Item Recommendation

[†]Shoujin Wang, [†]Liang Hu, [†]Longbing Cao, [‡]Xiaoshui Huang, [*]Defu Lian, [†]Wei Liu
[‡]GBDTC, FEIT, Univeristy of Technology Sydney
[†]Advanced Analytics Institute, Univeristy of Technology Sydney
[*]Big Data Research Center, University of Electronic Science and Technology of China
shoujin.wang@student.uts.edu.au, rainmilk@gmail.com,
{longbing.cao, xiaoshui.huang, wei.liu}@uts.edu.au, dove.ustc@gmail.com

Abstract

To recommend the next item to a user in a transactional context is practical yet challenging in applications such as marketing campaigns. *Transactional context* refers to the items that are observable in a transaction. Most existing transaction-based recommender systems (TBRSs) make recommendations by mainly considering recently occurring items instead of all the ones observed in the current context. Moreover, they often assume a rigid order between items within a transaction, which is not always practical. More importantly, a long transaction often contains many items irreverent to the next choice, which tends to overwhelm the influence of a few truely relevant ones. Therefore, we posit that a good TBRS should not only consider all the observed items in the current transaction but also weight them with different relevance to build an attentive context that outputs the proper next item with a high probability. To this end, we design an effective *attention-based transaction embedding model* (ATEM) for context embedding to weight each observed item in a transaction without assuming order. The empirical study on real-world transaction datasets proves that ATEM significantly outperforms the state-of-the-art methods in terms of both accuracy and novelty.

Introduction

Nowadays, recommender systems (RSs) play an important role in real-world business especially in the e-commerce domain. However, most existing RS theories face various issues (Cao 2016) such as tending to repeat items that are similar to what users may have already chosen (Deshpande and Karypis 2004). In reality, users may prefer items that are novel and different from that already in hand. To address this aspect, new recommendation paradigm (Cao 2016) needs to be made on a transactional context, i.e., what has already been chosen in a transaction. On one hand, transaction-based RSs (TBRSs) (Huang and Zeng 2011) incorporate previous transactions, i.e., inter-transactions, to generate more sensible and reliable new transactional recommendations, such as next-basket and next-item recommendations (Wang et al. 2015) through analyzing inter-transaction coupling relationships (Cao 2015). These are quite different from the typical RS approaches built on user preferences and item property. On the other, however, it is still unclear what next-item should be recommended when a collection of items has been placed into a transaction. This generates the need to recommend the next item under a transactional context by analyzing intra-transaction dependency. Here, the *context* for recommending the next item refers to the corresponding item-related transaction, e.g., a shopping-basket record consisting of multiple chosen items.

Let us illustrate the above problem with an example. A user first puts three items $\{milk, apple, orange\}$ into a cart and then adds $bread$ to the same cart. Subsequently, the transaction is finalized as $\{milk, apple, orange, bread\}$. If we take the first three items as the context and the last one as the target to recommend, existing methods may suggest vegetables like $green\ salad$ due to the nearest contextual items ($orange$ and $apple$). However, the choice of the target item $bread$ may depend on the first item ($milk$). In this case, a TBRS should pay more attention to $milk$ than to $orange$ and $apple$, because $milk$ may be more related to the next choice $bread$. This example shows the importance of next-item recommendation which can be misled by irrelevant items in a transaction. Moreover, real-world transactional data often only indicates those items co-appear in a transaction with the order (e.g., the item timestamps) between items. Therefore, it may not be possible and realistic to recommend transactional items with a rigid order.

It is quite challenging to learn the relevance and transition between items in a transactional context. In TBRSs, a general challenge is to build an attentive context which outputs the real next choice with a high probability (Verbert et al. 2012). Some existing approaches aim to generate recommendations by taking a transaction as the context. However, most existing TBRSs utilize a partial context with an ordering assumption. Sequential pattern mining (Yap, Li, and Philip 2012) is used to predict the next item using associations between items with a rigid order assumption. However, items in a context may be arbitrary, which may fail to match any mined patterns. Markov chain (MC) (Rendle, Freudenthaler, and Schmidt-Thieme 2010; Cao, Ou, and Yu 2012) is another way to model sequential data. However, MC only captures the transition from one item to the next one rather than from a contextual sequence, i.e., it only captures the first-order transition. Recently, a matrix factorization (MF) based approach (Chou et al. 2016) factorizes the matrix of transition probability from the current item to the next one into the latent factors. However, MF easily suffers from sparsity issues due to the power-law distributed data in the real world (Hu et al. 2016). Inspired by great success of deep networks, (Hidasi et al. 2015) applied deep recurrent neural networks (RNN) to model the transaction of sequential data but the high computational cost caused by the complex structures prevents its application to large data. Moreover, MC, MF and RNN were originally designed for time-series data with a rigid natural order, hence they do not fit unordered transactions. For example, it makes no difference whether $milk$ or $bread$ is put into the cart first. In addition, existing methods do not effectively weight the items within a context, namely paying more attention to those relevant items. Such attention distinction is quite important especially for long transactions which often contain many items irrelevant to the next choice.

This paper addresses the above issues by proposing an attention-based transaction embedding model (ATEM). ATEM builds an attentive context embedding over the embeddings (Jian et al. 2017) of all the observed items in a transaction by identifying the contextual items with high relevance to the next choice. Considering the large number of items, usually over 10^5, in real-world business, we build a shallow wide-in-wide-out network (Goth 2016) to reduce the time and space cost. Specifically, we incorporate the attention mechanism (Shaonan, Jiajun, and Chengqing 2017) into the shallow network to build an attentive context over all the observed items in a transaction without the rigid ordering assumption. Thanks to the attention mechanism, the proposed model is able to pay greater attention to more relevant items and less attention to less relevant ones. As a result, ATEM is more effective and robust to predict the next item in a transaction with less constraints. The main contributions of this work are as follows:

- An attention-based model learns an attentive context embedding that intensifies relevant items but downplays those irrelevant to the next choice. Our method does not involve a rigid ordering assumption over items in a transaction.
- A shallow wide-in-wide-out network implements ATEM, which is more effective and efficient for learning and prediction over a large number of items.
- Our empirical study shows that (1) ATEM significantly outperforms the state-of-art TBRSs on two real-world datasets in both accuracy and novelty; and (2) the attention mechanism makes a significant difference to TBRSs by comparing the methods with and without the attention mechanism.

Related Work

Pattern mining-based approaches are an intuitive solution to TBRSs. (Adda et al. 2005) introduced relation rule mining to discover the relations between different objects for recommendations. Considering the order between items, (Yap, Li, and Philip 2012) introduced a personalized sequential pattern mining-based recommendation framework by applying a novel Competence Score measure for accurate personalized recommendation. Although simple and effective, these approaches usually lose those infrequent items (Hu et al. 2017a) due to the minimum support constraint. In addition, the dynamic context containing arbitrary items may fail to match any mined frequent patterns (Wang, Bao, and Zhou 2017).

Markov chain (MC) models are another solution to capture transitions in sequential data (Cao, Ou, and Yu 2012). (Wu et al. 2013) proposed Personalized Markov Embedding (PME) to first embed users and songs into a Euclidean space by modeling sequential singing behaviours and then generate recommendations based on the embeddings. Recently, a personalized ranking metric embedding method (PRME) was proposed to precisely model personalized check-in sequences for next POI recommendation (Feng et al. 2015). Both PME and PRME are first-order MC models built on rigid ordered data to model the transition between sequential items from the same transaction. They may lose higher-order dependencies and the assumed rigid ordered data may not always be real-world cases. Inspired by the great power of matrix factorization (MF), Factorized Personalized Markov Chains (FPMC) (Rendle, Freudenthaler, and Schmidt-Thieme 2010) combines the power of MF and MC to factorize the transition matrix over underlying MC to model personalized sequential behaviours for next-basket recommendation. Similar to MC and MF, FPMC also suffers from the unrealistic rigid order assumption and data sparsity issue.

Recently, the prosperous deep learning technology has begun to be applied in RS. (Hidasi et al. 2015) adopted RNN that consists of gated recurrent units into transaction-based RS to effectively model the long sequences of transactions. Compared to deep architectures (Wang et al. 2016), shallow networks are more efficient in dealing with such kinds of issues, especially on large datasets. Particularly, the Word2vec model has achieved great success in learning the probability distribution of candidate words conditional on a bag of words using a shallow and wide network (Goth 2016).

Lately, inspired by the psychological cognition scheme, the attention mechanism has shown surprising potential in context learning-related areas. (Yang et al. 2016a) presented stacked attention networks (SANs) for image question answering by searching for the regions in an image that are related to the answer. Another novel model learns sentence representation with the guidance of human attention (Shaonan, Jiajun, and Chengqing 2017). In view of the great success of the attention mechanism for context learning in CV and NLP, we incorporate some ideas and propose ATEM to model the attentive context for next item recommendation.

Problem Statement

Before going into the details of our proposed model, we first define the problem and define basic concepts.

Generally, transaction-based recommendations are built on shopping basket-based transaction data. For a given transactional dataset, let $T = \{t_1, t_2...t_{|T|}\}$ be the set of all transactions, and each transaction $t = \{i_1, i_2...i_{|t|}\}$ consists of a subset of items, where $|T|$ denotes the number of elements in set T. All the items occurring in all transactions constitute

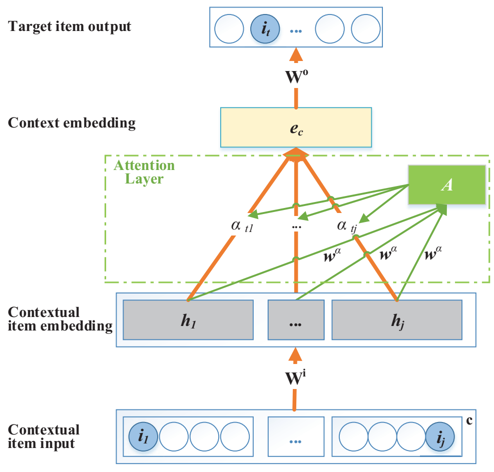

Figure 1: The ATEM architecture, which first learns item embeddings and then integrates them into the context embedding for target item prediction, where 'A' represents the attention model.

the whole item set $I = \{i_1, i_2 ... i_{|I|}\}$. Note that the items in a transaction t may not have a rigid order. For a given target item $i_s \in t$, all the items in t except i_s are picked up as its corresponding context $\mathbf{c}$, namely $\mathbf{c} = t \backslash i_s$. Particularly, an attentive context means items within the context contribute differently to the context embedding for next-item recommendation. Given the context $\mathbf{c}$, our ATEM is constructed and trained as a probabilistic classifier that learns to predict a conditional probability distribution $P(i_s|\mathbf{c})$. A total of $|t|$ training instances are built for each transaction t by picking up each item as the target one each time.

Therefore, TBRS is boiled down to rank all candidate items in terms of their conditional probability over the given context. Note that in the prediction stage, the conditional probability is computed based on the attentive embedding of the context $\mathbf{c}$. Such embedding is built on all the contextual items included in $\mathbf{c}$ by utilizing the attention mechanism to learn the weight of each contextual item.

Modeling and Learning

In this section, we first demonstrate the architecture of the proposed ATEM model, and then discuss how to train the model and learn the parameters. Finally, we show how to make predictions and accordingly generate recommendations using the trained model.

Attention-based Transaction Embedding Model

Overall, from bottom to top, the proposed ATEM model consists of an input layer, an item embedding layer, a context embedding layer, an output layer, plus an attention layer between the item and context embedding layers, as shown in Figure 1. Next, we explain the working mechanism of the model layer by layer from the input to the output.

Item Embedding Giving a contextual itemset $\mathbf{c}$ to the input layer, the input units in the bottom of Figure 1 constitute a one-hot encoding vector where only the unit at position i_j ($i_j \in \mathbf{c}$) is set to 1 and all others are set to 0. For each $i \in \mathbf{c}$, we encode it in the same way as i_j. Therefore, a vector with length $|I|$ is achieved to represent each item in the context and a total of $|\mathbf{c}|$ vectors can be achieved for a given context $\mathbf{c}$.

The information delivered by the sparse one-hot vectors is limited. In ATEM, we create an embedding mechanism to map these vectors to an informative and lower-dimensional vector representation in the item embedding layer, where a K-dimension real-valued vector $\mathbf{h}_j \in \mathbb{R}^K$ is used to represent the embedding of item i_j. The input weight matrix $\mathbf{W}^i \in \mathbb{R}^{K \times |I|}$ is used to fully connect the input-layer and item embedding-layer. Note that the j^{th} column of the weight matrix $\mathbf{W}^i_{:,j}$ actually encodes the one-hot vector of item i_j to the real-valued embedding $\mathbf{h}_j$, namely:

$$\mathbf{h}_j = \mathbf{W}^i_{:,j} \tag{1}$$

Transactional Context Embedding with Attention When the embeddings of all items in context $\mathbf{c}$ are ready, we can obtain the embedding $\mathbf{e}_c \in \mathbb{R}^K$ of context $\mathbf{c}$ by integrating the embeddings of all items in $\mathbf{c}$. Specifically, the attentive context embedding is built as a weighted sum of $\mathbf{h}_j$:

$$\mathbf{e}_c = \sum_{i_j \in \mathbf{c}} \alpha_{tj} \mathbf{h}_j, \quad s.t. \sum_{i_j \in \mathbf{c}} \alpha_{tj} = 1 \tag{2}$$

where α_{tj} is the integration weight of contextual item i_j w.r.t. the target item i_t, which indicates the contribution scale of i_j to the occurrence of i_t. In our model, to better capture the different contribution scale of various contextual items, we develop an attention layer to learn the integration weights automatically and effectively. Compared to assigning the weights manually under certain assumptions, or directly learning the weights without the attention mechanism, our method not only works more flexibly without such assumptions but also focuses more on the key items and reduces the interference from irrelevant items in a long context. Next, we demonstrate how the attention model achieves the integration weights.

Specifically, we use a softmax layer to determine the weights of different contextual items. In this case, those contextual items more relevant to the target item are given larger weights, while the input of the softmax is a transformation of each item embedding.

$$\alpha_{tj} = \frac{exp(e(\mathbf{h}_j))}{\sum_{s \in \mathbf{c}_t} exp(e(\mathbf{h}_s))} \tag{3}$$

$$e(\mathbf{h}_j) = \mathbf{w}^\alpha \mathbf{h}_j^T \tag{4}$$

where $\mathbf{w}^\alpha$ is an item-level context vector shared by all contextual items as shown in Figure 1. The shared context vector $\mathbf{w}^\alpha$ can be seen as a high level representation of a fixed query "which is the informative item" over the contextual items like that used in memory networks (Kumar et al. 2016). It is randomly initialized and jointly learned during the training stage. As $\mathbf{w}^\alpha$ serves as a weight vector to connect the item embedding layer to the attention model, we also refer to it as attention weight in the following section to keep it consistent with input and output weights.

Essentially, we measure the importance of each item i_j as the similarity of its embedding $\mathbf{h}_j$ with the item level context vector $\mathbf{w}^\alpha$ and get a normalized importance weight α_{tj} of item i_j w.r.t. the target item i_t through a softmax function (Yang et al. 2016b). Consequently, the attentive context representation vector can be computed using Eq. (2).

Target Item Prediction After getting the representation of context $\mathbf{c}$, we feed it into the output layer for the prediction task, which is shown in the top of Figure 1. Here the output weight matrix $\mathbf{W}^o \in \mathbb{R}^{|I| \times K}$ is used to fully connect the context embedding layer and output layer. With the embedding of the given context $\mathbf{c}$ plus the weight matrix $\mathbf{W}^o$, the score S_t of a target item i_t w.r.t. the given context $\mathbf{c}$ is computed as:

$$S_t(\mathbf{c}) = \mathbf{W}^o_{t,:}\mathbf{e}_\mathbf{c} \tag{5}$$

where $\mathbf{W}^o_{t,:}$ denotes the t^{th} row of $\mathbf{W}^o$. The resultant score $S_t(\mathbf{c})$ quantifies the relevance of the target item i_t w.r.t. the given context $\mathbf{c}$. As a result, the conditional probability distribution $P_\Theta(i_t|\mathbf{c})$ can be defined in terms of the softmax function, which is commonly used in neural network or regression model.

$$P_\Theta(i_t|\mathbf{c}) = \frac{exp(S_t(\mathbf{c}))}{Z(\mathbf{c})} \tag{6}$$

where $Z(\mathbf{c}) = \sum_{i \in I} exp(S_i(\mathbf{c}))$ is the normalization constant and $\Theta = \{\mathbf{W}^i, \mathbf{w}^\alpha, \mathbf{W}^o\}$ is the model parameters. Therefore, a probabilistic classifier modeled by our proposed ATEM is obtained to predict the target item.

Learning and Prediction

In the previous subsection, we have described the construction of a probability classifier over the transaction data $d = \langle \mathbf{c}, i_c \rangle$, where $\mathbf{c}$ is the input, namely the context constructed on the items within a transaction, and i_c is the observed output, namely the corresponding relevant item conditional on this context. Given a training dataset $D = \{\langle \mathbf{c}, i_c \rangle\}$, the joint probability distribution can be obtained as:

$$P_\Theta(D) \propto \prod_{d \in D} P_\Theta(i_c|\mathbf{c}) \tag{7}$$

Therefore, the model parameters Θ can be learned by maximizing the conditional log-likelihood (cf. Eq. (6)):

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

Note that, both the evaluation of L_Θ and the computation of its corresponding log-likelihood gradient involve the normalization term $Z(\mathbf{c})$, which needs to sum $exp(S_{i_c}(\mathbf{c}))$ over the entire item set for each training instance. This means, it takes $O(|I| \times |D|)$ time of computation to get the normalization constant for each iteration to train this model. Unfortunately, $|I|$ and $|D|$ are usually quite large in real-world business. For example, the Amazon dataset contains millions of transactions for more than ten thousand of products. Such a high computation cost makes the training process intractable.

Noise Contrastive Estimation To tackle the aforementioned issue, we adopt a subsampling approach to deal with the softmax layer, namely noise-contrastive estimation (NCE) (Gutmann and Hyvärinen 2012) which was proposed for training unnormalized probabilistic models and has been broadly used to handle similar issues in NLP etc. NCE does not directly compute the normalization constant of the softmax to avoid the high computation cost, instead it works with the other approximate objective which is much cheaper to compute.

The main idea of NCE is to use a binary classifier to distinguish samples from the data distribution from those with a known noise distribution Q. In our case, given a training example $\langle \mathbf{c}, i_c \rangle$, the probability of sampling from either a positive example or K noise examples is represented as a mixture of these two distributions (Mnih and Teh 2012):

$$P_\Theta(y, i_c|\mathbf{c}) = \frac{1}{K+1} P_\Theta(i_c|\mathbf{c}) + \frac{K}{K+1} Q(i_c) \tag{9}$$

Then the posterior probability of a sample i_c coming from the data distribution, namely the probability of a positive example, is calculated as:

$$P_\Theta(y=1|i_c, \mathbf{c}) = \frac{P_\Theta(i_c|\mathbf{c})}{P_\Theta(i_c|\mathbf{c}) + KQ(i_c)} \approx \frac{exp(S_{i_c}(\mathbf{c}))}{exp(S_{i_c}(\mathbf{c})) + KQ(i_c)} \tag{10}$$

where the normalization term $Z(\mathbf{c})$ is dropped from $P_\Theta(i_c|\mathbf{c})$. This is because the NCE is a normalized estimator where the objective encourages $P_\Theta(i_c|\mathbf{c})$ to be approximately self-normalized (Gutmann and Hyvärinen 2012). Hence, the probability of i_c coming from the noise samples is $P_\Theta(y = 0|i_c, \mathbf{c}) = 1 - P_\Theta(y = 1|i_c, \mathbf{c})$. Subsequently, instead of maximizing the original log-likelihood in Eq. (8), we can maximize the likelihood of the training samples against K noise samples as (Mnih and Kavukcuoglu 2013):

$$\begin{aligned} J_\Theta(i_c, \mathbf{c}) &= log P_\Theta(y=1|i_c, \mathbf{c}) + K\mathbb{E}_{i_k \sim Q}[log P_\Theta(y=0|i_c, \mathbf{c})] \\ &\approx log P_\Theta(y=1|i_c, \mathbf{c}) + \sum_{k=1}^{K} log P_\Theta(y=0|i_k, \mathbf{c}) \end{aligned} \tag{11}$$

Substituting Eq. (10) into Eq. (11), the gradient of $J_\Theta(i_c, \mathbf{c})$ can be immediately obtained. It approaches the original maximum likelihood (Eq. (8)) gradient when K increases (Mnih and Teh 2012). K is empirically set to 8 in our experiments.

$$\begin{aligned} \nabla J_\Theta(i_c, \mathbf{c}) = &\frac{KQ(i_c)}{exp(S_{i_c}(\mathbf{c})) + KQ(i_c)} \nabla S_{i_c}(\mathbf{c}) \\ &- \sum_{k=1}^{K} \frac{exp(S_{i_k}(\mathbf{c}))}{exp(S_{i_k}(\mathbf{c})) + KQ(i_k)} \nabla S_{i_k}(\mathbf{c}) \end{aligned} \tag{12}$$

Learning and Ranking After we get the gradient of $J_\Theta(i_c, \mathbf{c})$ as illustrated in Eq. (12), all the parameters Θ are learned by back-propagation. Algorithm 1 briefly summarizes the learning process. Note that $\nabla_{e(\mathbf{h}_j)}\alpha_{tj}$ is the gradient of a softmax function (cf. Eq. (3)) and it can be well computed (Buntine and Weigend 1994).

In Algorithm 1, $\odot$ denotes the element-wise product. Index t corresponds to the output item i_t which includes both the positive sample i_c and all noise ones $\{i_k\}$ and $i_j \in \mathbf{c}$

Algorithm 1 ATEM Parameter Learning Using SGD

```
1: $l \leftarrow 0$
2: while not converged do
3:     Compute output weight $w^o_{t,:}$-gradient (Eq. (5)):
       $g_{w^o_{t,:}} \leftarrow \mathbf{e}_c$
4:     Compute attention weight $w^\alpha_{tj}$-gradient (Eq. (2-5)):
       $g_{w^\alpha_{tj}} \leftarrow \mathbf{W}^o_{t,:} \odot \mathbf{h}^2_j \odot \nabla_{e(\mathbf{h}_j)}\alpha_{tj}$
5:     Compute input weight $w^i_{:,j}$-gradient (Eq. (1-5)):
       $g_{w^i_{:,j}} \leftarrow \mathbf{W}^{o\top}_{t,:} \odot (\alpha_{tj} + \nabla_{e(\mathbf{h}_j)}\alpha_{tj} \odot \mathbf{w}^\alpha \odot \mathbf{h}_j)$
6:     Perform SGD-updates for $w^o_{t,:}$, $w^\alpha_{tj}$ and $w^i_{:,j}$ :
       $w^o_{t,:} \leftarrow w^o_{t,:} + S^l_t(g) g_{w^o_{t,:}}$ (output weight update),
       $w^\alpha_{tj} \leftarrow w^\alpha_{tj} + S^l_{tj}(g) g_{w^\alpha_{tj}}$ (attention weight update),
       $w^i_{:,j} \leftarrow w^i_{:,j} + S^l_j(g) g_{w^i_{:,j}}$ (input weight update)
7:     $l \leftarrow l + 1$
8: end while
```

is one of the input items from the context. $w_{:,j}$ is the corresponding input weight (cf. Eq. (1)). A specific gradient-based update process for the parameter is achieved after we substitute the gradients demonstrated in Algorithm 1 (cf. Steps 3-5) into Eq. (12). To reduce the high computation cost caused by the large number of training samples, we adopt a mini-batch scheme to train the model, where each batch contains 50 training instances. The details to build the instances are given in the experimental part. Our experimental results are achieved by using Adam (Kingma and Ba 2014) for the specific gradient descent operation.

After all the parameters have been learned by the training process, the model can be used as a transaction-based recommender system, which is ready to make predictions and accordingly generate recommendations. To be specific, given an arbitrary transaction-based context **c** containing all the items chosen in a certain transaction, the probabilities of choosing each next candidate item can be calculated according to Eq. (6) immediately, and then the ranking over all of them can be achieved accordingly.

Experiments and Evaluation

The empirical study of the proposed ATEM is given in this section. Specifically, we first setup the experiments by preparing the experimental datasets and introducing the comparison methods, and then evaluate the performance in terms of recommendation accuracy and novelty.

Experimental Setup

Data Preparation We evaluate our method on two real-world transaction data sets: IJCAI-15 [1] and Tafang [2].

First, a shopping-basket-based transaction table is extracted from each of the original datasets []. The transaction table contains multiple transactions and each transaction consists of multiple items. Note that those transactions containing only one item are removed as they do not fit our model as we use at least one item as context and another as the target. Second, the transaction table is split into training and testing sets. Specifically, we randomly choose 20% from the transactions happened in last 30 days as the testing set, while the remainder is for training. Finally, to build the training and testing instances of format $d = \langle \mathbf{c}, i_c \rangle$ as illustrated in last section, for a transaction t, each time one out of which is picked up as the target item i_c and all the remaining ones are used as the corresponding context **c**. Subsequently, for a transaction containing $|t|$ items, $|t|$ instances are built in total. The characteristics of the datasets are shown in Table 1.

Table 1: Statistics of experimental datasets

Statistics	**IJCAI-15**	**Tafang**
#Transactions	144,936	19,538
#Items	27,863	5,263
Avg. Transaction Length	2.91	7.41
#Training Transactions	141,840	18,840
#Training Instances	412,679	141,768
#Testing Transactions	3,096	698
#Testing Instances	9,030	3,150

During the training stage, transactions in the training set are imported into the model in batches to learn the context embeddings. In the testing process, the learned embeddings are used to predict the target item. The true target item is used as the ground truth. We calculate the accuracy measures Recall@K and MRR (Chou et al. 2016) by comparing the predicted results to the ground truth. However, it is not enough to evaluate a recommender system only using accuracy metrics (Ge, Delgado-Battenfeld, and Jannach 2010). Considering the fact that an increasing number of customers prefer to enjoy a more surprising experience by discovering novel products which they have not chosen before, we also measure the recommendation novelty by comparing the recommendation list to the corresponding contextual itemset.

Comparison Methods We use the following representative start-of-the-art methods as the baselines for the experiments.

- ***PBRS***: A typical pattern-based recommender system which uses mined frequent patterns to guide the recommendations (Li et al. 2008).
- ***FPMC***: A model that combines matrix factorization and first-order Markov chains for next-basket recommendation. The model factorizes the personalized transition matrix between items with a pairwise interaction model (Rendle, Freudenthaler, and Schmidt-Thieme 2010).
- ***PRME***: A personalized ranking metric embedding method (PRME) to model personalized check-in sequences in a Markov chain framework. The learned PRME is used to recommend the next POI of users (Feng et al. 2015).

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

[2]http://stackoverflow.com/questions/25014904/download-link-for-ta-feng-grocery-dataset

Table 2: Accuracy comparisons on IJCAI-15

Model	REC@10	REC@50	MRR
PBRS	0.0780	0.0998	0.0245
FPMC	0.0211	0.0602	0.0232
PRME	0.0555	0.0612	0.0405
GRU4Rec	0.2283	0.3021	0.1586
ATEM	**0.3542**	**0.5134**	**0.2041**
TEM	0.3177	0.3796	0.1918

Table 3: Accuracy comparisons on Tafang

Model	REC@10	REC@50	MRR
PBRS	0.0307	0.0307	0.0133
FPMC	0.0191	0.0263	0.0190
PRME	0.0212	0.0305	0.0102
GRU4Rec	0.0628	0.0907	0.0271
ATEM	**0.1089**	**0.2016**	**0.0347**
TEM	0.0789	0.1716	0.0231

- ***GRU4Rec***: A RNN-based approach for session-based recommendations by modeling the session using a deep RNN which consists of GRU units (Hidasi et al. 2015).
- ***TEM***: A model similar to ATEM except that it utilizes distance-based exponential decay (Hu et al. 2017b) to replace the attention mechanism to assign the weights manually. The contextual items near to the target one are given larger weights. This model is built by us to test the effect of the attention mechanism.

Performance Evaluation

In this section, the accuracy evaluation is first given, followed by the novelty evaluation.

Accuracy Evaluation The following commonly used accuracy metrics for transaction-based RS are used for evaluation. Note that rating-based RS evaluation metrics, e.g., MSE, are not applicable in our work as we do not work on rating data to predict the ratings.

- ***REC@K***: It measures the recall of the top-K ranked items in the recommendation list over all the testing instances. Recall that in the real world, most customers are only interested in the items recommended on the first one or two web pages, so here we choose $K \in \{10, 50\}$. In practice, it is a significant challenge to exactly find the one true item from thousands of candidates.
- ***MRR***: It measures the mean reciprocal rank of the predictive position of the true target item on all the testing instances.

Table 2 and Table 3 demonstrate the results of REC@10, REC@50 and MRR over the testing sets on two real-world datasets, respectively. For PBRS, we empirically set the minimum support to 0.01 and 0.008 on IJCAI-15 and Tafang dataset respectively. As it only focuses on those frequent items and filters out infrequent ones, the performance is not so good. The number of factors is set to 10 for training the FPMC to achieve the best performance. However, the accuracy performance of FPMC on both datasets is quite poor. This is due to the fact that both datasets are extremely sparse and thus a very large but quite sparse item transition matrix is constructed on each dataset to train the MF model. For example, in IJCAI-15 dataset, each transaction only contains an average of 2.91 items from over 27,000 ones (cf. Table 1). This indicates each row of the built matrix contains less than two items. In practice, the non-empty entries account for less than 0.01%. We set the embedding dimensions to 60 as suggested in (Feng et al. 2015) when training the PRME model. Compared to FPMC, the performance of PRME is a little better, but it is still poor. This is because PRME is a first-order MC model, which learns the transition probability over the successive item rather than the whole context. This may lead to information loss. Furthermore, in the real word, the purchase of goods does not always follow a rigid sequence assumed by such kind of models. Benefiting from the deep structure, GRU4Rec achieves much better performance compared to FPMC and PRME.

For our ATEM model, the batch size is empirically set to 50 and the number of hidden units for item embeddings is set to 128 and 40 on IJCAI-15 and Tafang dataset respectively. We run 20 epochs to train the model. It clearly achieves a better performance than GRU4Rec, where the REC@10 and REC@50 exceed 35% and 50% respectively on IJCAI-15 dataset. The highest MRR also proves that our model can effectively put the users' desired items in the front of the recommendation list, the reason being that, different from the previous models which either capture only first-order dependency between items or capture the dependency between each contextual item and target one respectively, ATEM builds an embedding of the context by treating all the contextual items as a whole. Therefore, the complex dependency relations (e.g., intra-context dependency, context-target dependency) can be better captured. More importantly, the attention mechanism is applied here to discriminate the contributions of different contextual items to the prediction of a certain target item. This actually helps greatly to build a more informative context embedding for different target items. From the point of view of real-world applications, our model has a very shallow and concise structure for easy training, which makes it more efficient to recompute the scores for ranking all candidate items when contexts keep updating in online recommendations, compared to those complex and deep models (e.g., GRU4Rec).

The settings of TEM are kept the same as ATEM and the parameter λ in exponential decay is set to 0.75 to obtain the best performance. The performance of TEM is obviously weaker than ATEM. This is caused by the distance assumption used in TEM, which essentially still has an order assumption over items within a transaction. This may not be consistent with real-word cases, as previously stated.

The Effect of Context Length Although longer transactional contexts consisting of more items may be more informative, they may be more fragile and contain irrelevant

Table 4: Accuracy on disordered IJCAI-15

Model	REC@10	REC@50	MRR
PBRS	0.0500	0.0559	0.0185
FPMC	0.0151	0.0412	0.0183
PRME	0.0346	0.0389	0.0351
GRU4Rec	0.1636	0.2121	0.1022
ATEM	**0.3423**	**0.4981**	**0.1960**
TEM	0.2660	0.3012	0.1431

items, resulting in reduced recommendation accuracy when these are not identified. To show the advantages of our model under varies lengths of contexts, we test the effect of context length. Figure 2 illustrates that longer contexts benefit accuracy, and our method clearly outperforms the others under longer contexts, such as a context consisting of four items (denoted as Len-4). Note that PBRS, FPMC and PRME are mainly first-order dependency based and thus are not sensitive to context length. Hence, they are not included in this test.

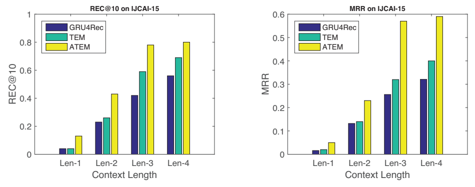

Figure 2: ATEM achieves higher REC@10 and MRR than the other approaches, especially under long contexts.

The Effect of Item Order To test the effect of the order of items within a transaction on recommendation accuracy, we randomize the default item order in the IJCAI-15 data to build a disordered dataset. Table 4 shows the accuracy of different methods on this new data. Compared to the results in Table 2, other approaches experience much more performance degradation than ATEM. This indicates the stronger ability of ATEM compared to the other methods in handling disordered data.

Novelty Evaluation Except for accuracy, novelty is another important quality which should be considered in real-world RS (Wang, Hu, and Cao 2017). Recall that by considering what a customer has already chosen in a transaction (transactional context), our proposed TBRS can effectively avoid recommending duplicate items and suggest some novel items which can result in a surprising experience. Therefore, we define a novelty measure to quantify the difference between the given context and the recommendation list. The larger the difference, the higher the novelty. It should be noted that the aforementioned accuracy guarantees the relevance of recommended items, so highly novel items are also of high relevance.

MCAN@K: In our model, the items which have already been chosen correspond to the context $\mathbf{c}$ used for recommendation R. Subsequently, this novelty measures the mean

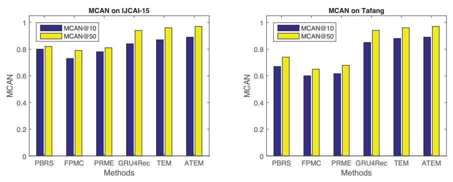

Figure 3: ATEM achieves higher novelty than the other approaches.

non-overlap ratio between each context-recommendation pair $\langle \mathbf{c}_i, R_i \rangle$ over all N top-K recommendations.

$$MCAN = \frac{1}{N}\sum_{i=1}^{N}(1 - \frac{|R_i \cap \mathbf{c_i}|}{|R_i|}) \tag{11}$$

Figure 3 illustrates the results of novelty comparison on the two datasets. PBRS only results in frequent patterns, hence it is difficult to match the whole context exactly, especially for long contexts. Subsequently, low novelty is achieved without considering all the contextual items. FPMC does not learn its parameters well on such sparse datasets, so it outputs relatively random recommendations. Accordingly, FPMC obtain low novelty. PRME is a first-order MC model which makes recommendations by only considering the exact prior item while ignoring other contextual items, so it may recommend duplicate items and thus lead to low novelty. GRU4Rec can accumulate the influence of all the sequential items from the context to make relatively reliable and novel recommendations. Compared to the aforementioned methods, our model not only considers the whole context but also tries to build an attentive context embedding utilizing attention mechanism. Consequently, it is easier for us to generate novel and relevant recommendations.

In summary, the higher accuracy and novelty of the next-item recommended by ATEM than the baselines verifies the significance of weighting all the contextual items in building an attentive context embedding for transaction-based recommender systems. In addition, the power of the attention mechanism is further justified by the comparison between ATEM and TEM.

Conclusions

To effectively recommend the next item within a transactional context, which cannot be addressed by existing next-basket and next-items recommender systems, this work proposed an attention-based transaction embedding model ATEM. ATEM is a shallow wide-in-wide-out neural network. It learns an attentive context embedding that is expected to be the most relevant to the next choice over all the observed items in a transaction. The empirical evaluation on the real-world transactional data shows its significant superiority in addressing gaps in state-of-the-art approaches. We will explore the application of ATEM to other problems such as the author-topic relation learning (Rosen-Zvi et al. 2010).

References

Adda, M.; Missaoui, R.; Valtchev, P.; and Djeraba, C. 2005. Recommendation strategy based on relation rule mining. In *IJCAI*

Workshop on Intelligent Techniques for Web Personalization, 33–40.

Buntine, W. L., and Weigend, A. S. 1994. Computing second derivatives in feed-forward networks: A review. *IEEE Transactions on Neural Networks* 5(3):480–488.

Cao, L.; Ou, Y.; and Yu, P. S. 2012. Coupled behavior analysis with applications. *IEEE Trans. on Knowledge and Data Engineering* 24(8):1378–1392.

Cao, L. 2015. Coupling learning of complex interactions. *Information Processing & Management* 51(2):167–186.

Cao, L. 2016. Non-iid recommender systems: A review and framework of recommendation paradigm shifting. *Engineering* 2(2):212–224.

Chou, S.-Y.; Yang, Y.-H.; Jang, J.-S. R.; and Lin, Y.-C. 2016. Addressing cold start for next-song recommendation. In *Proceedings of the 10th ACM Conference on Recommender Systems*, 115–118.

Deshpande, M., and Karypis, G. 2004. Item-based top-n recommendation algorithms. *ACM Transactions on Information Systems* 22(1):143–177.

Feng, S.; Li, X.; Zeng, Y.; Cong, G.; Chee, Y. M.; and Yuan, Q. 2015. Personalized ranking metric embedding for next new poi recommendation. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence*, 2069–2075.

Ge, M.; Delgado-Battenfeld, C.; and Jannach, D. 2010. Beyond accuracy: evaluating recommender systems by coverage and serendipity. In *Proceedings of the 4th ACM Conference on Recommender Systems*, 257–260.

Goth, G. 2016. Deep or shallow, nlp is breaking out. *Communications of the ACM* 59(3):13–16.

Gutmann, M. U., and Hyvärinen, A. 2012. Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. *Journal of Machine Learning Research* 13(2):307–361.

Hidasi, B.; Karatzoglou, A.; Baltrunas, L.; and Tikk, D. 2015. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939*.

Hu, L.; Cao, L.; Cao, J.; Gu, Z.; Xu, G.; and Yang, D. 2016. Learning informative priors from heterogeneous domains to improve recommendation in cold-start user domains. *ACM Transactions on Information Systems (TOIS)* 35(2):13.

Hu, L.; Cao, L.; Cao, J.; Gu, Z.; Xu, G.; and Wang, J. 2017a. Improving the quality of recommendations for users and items in the tail of distribution. *ACM Transactions on Information Systems (TOIS)* 35(3):25.

Hu, L.; Cao, L.; Wang, S.; Xu, G.; Cao, J.; and Gu, Z. 2017b. Diversifying personalized recommendation with user-session context. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, 1937–1943.

Huang, Z., and Zeng, D. D. 2011. Why does collaborative filtering work? transaction-based recommendation model validation and selection by analyzing bipartite random graphs. *INFORMS Journal on Computing* 23(1):138–152.

Jian, S.; Cao, L.; Pang, G.; Lu, K.; and Gao, H. 2017. Embedding-based representation of categorical data by hierarchical value coupling learning. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, 1937–1943.

Kingma, D., and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kumar, A.; Irsoy, O.; Ondruska, P.; Iyyer, M.; Bradbury, J.; Gulrajani, I.; Zhong, V.; Paulus, R.; and Socher, R. 2016. Ask me anything: dynamic memory networks for natural language processing. In *Proceedings of the 33rd International Conference on Machine Learning*, 1378–1387.

Li, H.; Wang, Y.; Zhang, D.; Zhang, M.; and Chang, E. Y. 2008. Pfp: parallel fp-growth for query recommendation. In *Proceedings of the 2nd ACM Conference on Recommender Systems*, 107–114.

Mnih, A., and Kavukcuoglu, K. 2013. Learning word embeddings efficiently with noise-contrastive estimation. In *Advances in Neural Information Processing Systems*, 2265–2273.

Mnih, A., and Teh, Y. W. 2012. A fast and simple algorithm for training neural probabilistic language models. In *Proceedings of the 29th International Conference on Machine Learning*, 419–426.

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

Rosen-Zvi, M.; Chemudugunta, C.; Griffiths, T.; Smyth, P.; and Steyvers, M. 2010. Learning author-topic models from text corpora. *ACM Transactions on Information Systems* 28(1):1–38.

Shaonan, W.; Jiajun, Z.; and Chengqing, Z. 2017. Learning sentence representation with guidance of human attention. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, 4137–4143.

Verbert, K.; Manouselis, N.; Ochoa, X.; Wolpers, M.; Drachsler, H.; Bosnic, I.; and Duval, E. 2012. Context-aware recommender systems for learning: a survey and future challenges. *IEEE Transactions on Learning Technologies* 5(4):318–335.

Wang, L.; Bao, X.; and Zhou, L. 2017. Redundancy reduction for prevalent co-location patterns. *IEEE Transactions on Knowledge and Data Engineering* 29:1–14.

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

Wang, S.; Liu, W.; Wu, J.; Cao, L.; Meng, Q.; and Kennedy, P. J. 2016. Training deep neural networks on imbalanced data sets. In *Proceedings of the 29th International Joint Conference on Neural Networks (IJCNN)*, 4368–4374.

Wang, S.; Hu, L.; and Cao, L. 2017. Perceiving the next choice with comprehensive transaction embeddings for online recommendation. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 1–16.

Wu, X.; Liu, Q.; Chen, E.; He, L.; Lv, J.; Cao, C.; and Hu, G. 2013. Personalized next-song recommendation in online karaokes. In *Proceedings of the 7th ACM Conference on Recommender Systems*, 137–140.

Yang, Z.; He, X.; Gao, J.; Deng, L.; and Smola, A. 2016a. Stacked attention networks for image question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 21–29.

Yang, Z.; Yang, D.; Dyer, C.; He, X.; Smola, A. J.; and Hovy, E. H. 2016b. Hierarchical attention networks for document classification. In *HLT-NAACL*, 1480–1489.

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