# Perceiving the Next Choice with Comprehensive Transaction Embeddings for Online Recommendation

Shoujin Wang, Liang Hu and Longbing Cao

Advanced Analytics Institute, University of Technology Sydney, Sydney NSW 2007, Australia
{Shoujin.Wang,Liang.Hu-2}@student.uts.edu.au,
Longbing.Cao@uts.edu.au

**Abstract.** To predict customer's next choice in the context of what he/she has bought in a session is interesting and critical in the transaction domain especially for online shopping. Precise prediction leads to high quality recommendations and thus high benefit. Such kind of recommendation is usually formalized as transaction-based recommender systems (TBRS). Existing TBRS either tend to recommend popular items while ignore infrequent and newly-released ones (e.g., pattern-based RS) or assume a rigid order between items within a transaction (e.g., Markov Chain-based RS) which does not satisfy real-world cases in most time. In this paper, we propose a neural network-based comprehensive transaction embedding model (NTEM) which can effectively perceive the next choice in a transaction context. Specifically, we learn these comprehensive embeddings of both items and their features from relaxed ordered transactions. The relevance between items revealed by the transactions is encoded into such embeddings. With rich information embedded, such embeddings are powerful to predict the next choices given those already bought items. NTEM is a shallow wide-in-wide-out network, which is more efficient than deep networks considering large numbers of items and transactions. Experimental results on real-world datasets show that NTEM outperforms three typical TBRS models FPMC, PRME and GRU4Rec in terms of recommendation accuracy and novelty. Our implementation is available at https://github.com/shoujin88/NTEM-model

## 1 Introduction

### 1.1 Target Problem and Motivation

Nowadays, recommender systems (RS) play an important role in real-world business especially in the e-commerce domain. For example, the RS behind thousands of websites (e.g., Amazon) provide magic power to help end users to discover and make choices from a huge number of items. As a result, RS can not only improve customers' shopping experience but also increase the business profits. Although lots of work has been done to produce high quality recommendations, some issues are still challenging and need more efforts. One of them is to enable RS to dynamically perceive customers' next choice on thousands of candidate items online according to what they have just put in the shopping carts. In this paper, we call the next item to choose as the target item while those items having been added to cart are treated as its context. The challenges come from two sides: on one hand, the context is dynamic along with the shopping

transaction; on the other hand, the RS needs to keep updating the recommendations when the context was changed. For instance, suppose a customer Robin starts an online shopping transaction on Amazon: first, he bought a cellphone and then a protective film may be his next choice, so good RS should be capable of perceiving the protective film according to the context of cellphone. When Robin has bought the cellphone and protective film, his next choice is probably an earphone rather than buying another type of protective film again. Therefore, the recommended item should be changed from the protective film to the earphone accordingly. This dynamic recommendation process keeps updating until the transaction is finished.

This kind of recommender systems are also called transaction-based RS (TBRS) [11] as they work on transactional data, which are different from the rating-based RS (RBRS) [2] working on rating data. Although existing TBRS can recommend next items given the context, most of them treat the context as static rather than dynamic and can only work on static transactional data, an example is the pattern-based RS [23]. Moreover, it is quite hard for them to keep updating recommendations due to the long computational time, like deep network-based RS [10, 21]. As a result, they do not work efficiently for online recommendations. RBRS has been well studied but it cannot tackle our problems here due to the lack of consideration of context, existing TBRS cannot perform well either for online recommendations due to the aforementioned reasons. In this paper, we focus on TBRS and target at handling the dynamic context and producing online recommendations.

Due to the lack of effective approach to model the context of a transaction event, most current TBRS cannot capture the intra-session relevance over items perfectly and thus cannot produce high quality recommendations. They tend to recommend those popular and long-released items while ignore those less popular or newly-released ones. In practice, customers may not always need popular or similar items to form immutable shopping behaviors, instead, they want to explore something novel or unpopular. For example, when the first smart-phone $iPhone$ appeared in 2010, most customers prefer to it rather than $Nokia$, a popular function-phone (not smart-phone) brand lasted for more than a decade at that time. Therefore, more sensible RS which can bring surprising experience to users by recommending novel but relevant items is increasingly important. In this paper, novel items refer to unpopular or newly-released items while "relevant" infers the recommended items are strongly relevant to the context. It's clear that an effective approach to model the dynamic context in real-time is necessary to produce high quality online recommendations.

Content-based filtering (CBF) [14] and collaborative filtering (CF) [12] are two approaches that are most commonly used in RS. They are not applicable to TBRS directly though they perform well in RBRS. This is because these methods are actually designed to work on rating matrix in RBRS, which is quite different from the shopping-basket data in TBRS, thus it is hard for them to capture the relevance between items embedded in transactional data.

Pattern-based recommendation [23] is a straightforward solution to transaction-based recommendation issues. It first captures associations between items and then recommends items associated to the context items. Although simple and effective sometimes, patterns are extracted from those frequent items due to the "support" measure,

whereas those infrequent ones are missed. As a result, the discovered patterns can neither capture the relevance between all items nor achieve the goal in this work. In addition, the sequential pattern mining assumes a rigid order on transaction data [9]. For instance, it makes no difference whether milk or bread is put into the cart firstly. Markov Chain (MC) [18] is another straightforward way to model sequential data and thus can be used for TBRS on sequence data. However, MC can only capture the transition from one item to another rather than from a context item set to an item. Recently, matrix factorization (MF) [5] is used to factorize the transition probability from current item to the next one to the latent factors of each item and user. Similar to sequential patterns, both MC and MF were originally designed for time-series data with rigid natural order, which limit their applications in TBRS, where the order between items within a transaction usually makes no difference. Moreover, they cannot handle those novel items well as the transitions between them and other items tend to be weak due to their low frequencies.

The above illustrations reveal the difficulty of modeling the context especially for dynamic context. In practice, the next choice is not only affected by one item or part of items in front of the target item, but by all items bought in the transaction event (i.e., the whole context). It is important to model the whole context and learn the relevance between the whole context and the target item. Furthermore, the intra-transaction relevance over items are greatly driven by their intrinsic nature, in another words, there are complex coupling relations [4] between item features and item relevance. Such relations are particularly critical for those novel items with quite a few transaction records. For example, milk and bread are always bought together probably due to their different but closely related categories “drink” and “food”. This indicates that not only the context items but also the features of these items can affect the choice of the target item. To capture the indicators on the next choice as more as possible, we propose a neural-network-based comprehensive transaction embedding model (NTEM) to learn the embeddings of both items and their features when modeling the relevance between the context and the known choice. The model is comprehensive for several reasons: it models the relations between the target item and the whole context rather than part of it. It learns the embedding of the two important aspects (e.g., items and their features) of a transaction at the same time. During model training, the coupling relations between item relevance and item features are learnt and encoded into feature embeddings, which is useful for novel item discovery and recommendation. Though comprehensive, the embedding model has a shallow and wide network structure containing only one hidden layer, which guarantees its efficiency to find the best next choices over thousands of candidate items when the given context changes over time. This is suitable for online recommendation.

### 1.2 Our Design and Main Contributions

Inspired by the great success of modern word embedding models, such as Word2Vec [15], in natural language processing (NLP) domain, we propose a shallow and wide network-based transaction embedding model (NTEM) to learn the relevance between different items efficiently on a large number of items with its wide-in-wide-out structure. Such relevance is learnt by capturing both the explicit relevance between items

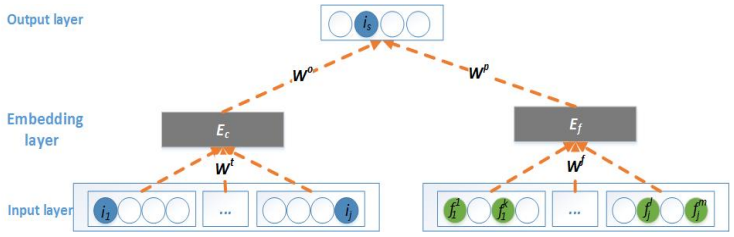



Fig. 1: The NTEM architecture, which learns item embeddings and feature embeddings for target item prediction based on contextual items and their features

from the shopping-basket data and the implicit relevance from item features together with the coupling relations between them and the item relevance. It is noted that a deep structure is not efficient for online recommendation due to the long computational time needed to deal with thousands of items in real time. The Word2Vec cannot be directly applied to RS for two reasons: on one hand, it lacks of necessary element to incorporate the item features. On the other hand, the words in NLP often have a strict sequence, which is different from our case.

Our model, NTEM has a three-layer network structure consisting of input layer, embedding layer and output layer as shown in Fig. 1. The input layer contains double wide-in data vectors, the contextual itemset is collected from one while their corresponding features are acquired from the other. The embedding layer learns the item embeddings and feature embeddings respectively. The target item is then predicted by the output layer taking the embeddings of contextual items and features as the input. The NTEM learns the relevance between items with comprehensive transaction embeddings using a concise network structure. The main contributions of this work are summarized below.

(1). We model the whole context using a comprehensive network-based transaction embedding model for the next choice prediction.

(2). A TBRS model is proposed, which does not require the strict order over items within one transaction. This is more consistent with the real-world case.

(3). We incorporate item features into the model and encode the feature-item relevance coupling relations into feature embeddings, which makes our model also work well on cold start cases.

(4). We propose a shallow and wide network, which recommends the next item efficiently on large number of items under dynamic context.

## 2 Related Work

Existing transaction-based recommender system (TBRS) can be roughly divided into pattern mining-based and model-based ones, we will briefly review the literature on these two.

Pattern mining-based approaches are an intuitive solution to TBRS and widely used in real-world business. [13] proposed to adjust the minimum support according to the

various frequencies of users and items. The mined association rules between users as well as between items are then used for making recommendations. [1] introduced relation rule mining to discover relations between different objects and such relations are then used for recommendations. Considering the order between items, some sequential pattern mining (SPM) based recommendation methods are proposed. [23] introduced a personalized sequential pattern mining-based recommendation framework. Using a novel Competence Score measure, the proposed framework effectively learns user-specific sequence importance knowledge for accurate personalized recommendation. Although simple and effective, such kind of approaches may lose information of those infrequent items due to the " minimum support" constraint. In addition, they can not be applied to online recommendation directly as the dynamic context may contain arbitrary items, it probably fail to match any mined pattern.

Overall, there are mainly two kinds of model-based approaches for TBRS: Markov Chain (MC) based ones and matrix-factorization (MF) based ones. [18] used Markov Chain to estimate the transition probability from current item to the next item and thus make prediction based on this probability. [22] proposed Personalized Markov Embedding (PME) to map the users and songs to an Euclidean space by modeling the sequential singing behaviours, the prediction and recommendation are conducted on the base of the embeddings. Although sequential behaviour prediction based on Markov Chain is effective for capturing the transition preferences of certain users and thus make good recommendations, it is essentially based on item order within a transaction, which is not always available in real-world business and such method only captures the first-order dependency between items. Recently, MF-based approaches are also developed for TBRS. [18] combined MF and MC for next-item recommendation, the latent user and item representations from the {user, item, last item} triplets are learned, and then the inner product of these latent vectors is used to perform next-item recommendation. Following factorization machine [17], a pairwise factorization model is proposed in [5] to learn the latent vectors of user, last item and item for next-song recommendation. Similar to MC-based model, MF-based ones depend strongly on the rigid order between items. The working mechanism limits it to model the relations between one item and another rather than between a contextual itemset and the target item. Furthermore, MF-based methods can not work well on sparse dataset, while the data on novel items usually has a large chance to be sparse due to their low frequencies.

## 3 Transaction Embeddings for Online Recommendation

In this section, we start with the problem formulation, then we talk about the proposed NTEM model including the network architecture in the model and the model construction, finally we illustrate how to train the model and how to make prediction and thus produce recommendations for online shopping using the trained model.

### 3.1 Problem Formalization

Let $T = \{t_1, t_2...t_{|T|}\}$ be a set of transactions, each transaction $t = \{i_1, i_2...i_{|t|}\}$ contains a set of items, where $|T|$ denotes the number of elements in set $T$. All the

items occurred in all transactions constitute the whole item set $I = \{i_1, i_2...i_{|I|}\}$. Let $F = \{f^1, f^2...f^{|F|}\}$ be a set a features which describe the items from $I$. Each item $i$ is described by a set of feature values $F_i = \{f_i^1, f_i^2...f_i^{|F|}\}$. Note that the items in one transaction $t$ may not have a rigid order, which is consistent with the real-world cases. Given the set of context itemset $\mathbf{c}$, our NTEM is constructed and trained as a probabilistic classifier that learns to predict a conditional probability distribution $P(i_s|\mathbf{c})$, where $\mathbf{c} \subseteq t \backslash i_s$ is the context from a transaction $t \in T$ w.r.t. the target item $i_s$. This is similar to the bag of word (BOW) model in natural language processing, which trains a classifier to learn a conditional probability distribution $P(w_j|w_I)$, where $w_I$ is the context consisting of several words of the target word $w_j$ [19]. Similar to BOW, for each target item $i_s \in t$, the transaction context is $\mathbf{c} = t \backslash i_s$, namely all the items except the target one in the transaction are picked up as the context. Totally $|t|$ training instances are built for each transaction $t$ by picking up one item as the target one each time.

Since we want to capture more information from the context for prediction, the features of items are added to the model as part of context, which result in $\mathbf{c}_f =< \mathbf{c}, F_c >$, where $F_c = \{F_i | i \in \mathbf{c}\}$ is the corresponding features of the items from $\mathbf{c}$. Thus our NTEM model is refined to predict the conditional probability distribution $P(i_s|\mathbf{c}, F_c)$ when the transaction-feature context $< \mathbf{c}, F_c >$ is given. We call $\mathbf{c}$ and $F_c$ as transaction context and feature context respectively in this paper. Thus, the TBRS is reduced to ranking all candidate items in terms of their conditional probability over the given transaction-feature context. Note that in the prediction stage, the conditional probability is computed based on the embeddings of item set $\mathbf{c}$ and its corresponding features $F_c$ learned in the training stage.

Particularly, the incorporation of features contributes greatly to the recommendation of novel items. Due to the low frequencies of novel items in training set, the embeddings of these items may not be learned well during the model training process and it leads to poor prediction. Thanks to the feature embeddings synchronously learned with the item embeddings, the intra-transaction item relevance can be partly encoded into feature value embeddings of novel items. In addition, part of the feature values of novel items may already be embedded when encoding those of frequent items as some feature values may be shared between frequent items and infrequent ones.

### 3.2 Neural-Network-based Transaction Embedding Model (NTEM)

In this section, we mainly talk about the details of constructing NTEM and learning its parameters.

Giving a context $< \mathbf{c}, F_c >$ to the input layer, the input units in the bottom left corner of Fig. 1 constitute a one-hot encoding vector where only the units at position $i_j$ ($i_j \in \mathbf{c}$) is set to 1 and all other units are set to 0. For each $i \in \mathbf{c}$, we encode it in the same way as $i_j$. Note that items may have both numerical and categorical features in real-world business. In this work, we only consider those categorical features. For a value from a categorical feature $f$ ($f \in F$) with $m$ different values, we transform it to a $1 \times m$ vector using one-hot encoding. Suppose $V = \sum n_k$ is the total number of distinct values of all features, where $n_k$ is the number of distinct values in feature $f^k$. For the features of a given item, a $1 \times V$ vector is achieved by doing the transformation

of all feature values first and then concatenate all the transformed vectors together. Thus the input layer for each training example consists of $|\mathbf{c}|$ item vector with length $|I|$ and $|\mathbf{c}|$ item feature vectors with length $V$.

In the input layer, items and item features are represented by sparse one-hot item and feature vectors. In NTEM, we create an embedding mechanism to map these vectors to an informative and lower-dimensional vector representation in the embedding layer, where a K-dimension vector $\mathbf{E}_i \in [0,1]^K$ is used to represent the item embedding. The transaction context weight matrix $\mathbf{W}^t \in \mathbb{R}^{K\times|I|}$ is used to fully connect between input-layer and embedding-layer. Where the $i^{th}$ column $\mathbf{W}^t_{:,i}$ encodes the one-hot vector of item $i$ to the embedding $\mathbf{E}_i$ using the commonly used logistic function $\sigma(\cdot)$.

$$\mathbf{E}_i = \sigma(\mathbf{W}^t_{:,i}) \tag{1}$$

To make the training and prediction more stable, here we use the nonlinear embeddings as they are bounded in [0,1] compared to the linear embeddings. Furthermore, the nonlinear embeddings are more expressive than linear one though they may involve a little more computation cost. After embedding all items in $\mathbf{c}$, we can obtain the embedding $\mathbf{E}_c \in [0,1]^L$ of transaction context $\mathbf{c}$ by combining all embeddings of items in such context. As illustrated in the following equation, the transaction context embedding is built as a combination of $\mathbf{E}_i, i \in \mathbf{c}$.

$$\mathbf{E}_c = \sum_{i\in\boldsymbol{c}} \omega_i \mathbf{E}_i \tag{2}$$

where $\sum_{i\in\boldsymbol{c}} \omega_i = 1$. The combination weight $\omega_i$ for each item $i$ in context $\mathbf{c}$ can be assigned to different values according to specific applications. For instance, in sequential data, the weights decay with time span to the target item. As illustrated in the introduction part, we treat the items within a transaction as unordered, so uniform weights are used in this paper, i.e. the items in context are equally important for the prediction of the target item.

Similarly, we use the feature context weight matrix $\mathbf{W}^f \in \mathbb{R}^{L\times V}$ to encode the one-hot item feature vector $F_i$ of item $i$ to the embedding $\mathbf{E}_{F_i} \in [0,1]^L$.

$$\mathbf{E}_{F_i} = \sigma(\mathbf{W}^f_{:,F_i}) \tag{3}$$

Similar to transaction context, we combine the embeddings of features of all items from $\mathbf{c}$ to construct the whole feature embedding $\mathbf{E}_{F_c}$ as below.

$$\mathbf{E}_{F_c} = \sum_{i\in\mathbf{c}} \omega_{F_i} \mathbf{E}_{F_i} \tag{4}$$

where $\sum_{i\in\mathbf{c}} \omega_{F_i} = 1$. Uniform weights are assigned to the feature embedding of each item for the same reason as $\omega_i$ illustrated above.

The output weight matrices $\mathbf{W}^o \in \mathbb{R}^{|I|\times K}$ and $\mathbf{W}^p \in \mathbb{R}^{|I|\times L}$ is used to fully connect the embedding-layger and output-layer as depicted in the top of Fig. 1. With the embeddings of given contextual itemset $\mathbf{c}$ and its features $F_c$, plus the weight metrics $\mathbf{W}^o$ and $\mathbf{W}^p$, the score $S_{i_s}$ of a target item $i_s$ w.r.t. the given context $< \mathbf{c}, F_c >$ is computed as:

$$S_{i_s}(\mathbf{c}, F_c) = \mathbf{W}^o_{s,:}\mathbf{E}_{\mathbf{c}} + \mathbf{W}^p_{s,:}\mathbf{E}_{F_c} \tag{5}$$

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

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

where $Z(\mathbf{c}, F_c) = \sum_{i \in I} exp(S_i(\mathbf{c}, F_c))$ is the normalization constant and $\Theta = \{\mathbf{W}^t, \mathbf{W}^f, \mathbf{W}^o, \mathbf{W}^p\}$ is the model parameters. Thus a probabilistic classifier modeled by our NTEM is obtained.

### 3.3 Learning and Prediction

In the above subsection, we have built a probabilistic classifier based on the transaction and item feature information data $b =< \mathbf{g}, i_g >$, where $\mathbf{g} =< \mathbf{c}, F_c >$ is the input data, namely the transaction-feature context, and $i_g$ is the corresponding observed output, namely an item bought together with the given transaction context $\mathbf{c}$. Given a training dataset $D = \{< \mathbf{g}, i_g >\}$, the joint probability distribution over it is obtained:

$$P_\Theta(D) \propto \prod_{b \in D} P_\Theta(i_g|\mathbf{c}, F_c) \tag{7}$$

As a result, the model parameters $\Theta$ can be learned by maximizing the conditional log-likelihood:

$$L_\Theta = \sum_{b \in D} logP_\Theta(i_g|\mathbf{c}, F_c) = \sum_{b \in D} S_{i_g}(\mathbf{c}, F_c) - logZ(\mathbf{c}, F_c) \tag{8}$$

Evaluating $L_\Theta$ and evaluating the corresponding log-likelihood gradient involve the normalization term $Z(\mathbf{c}, F_c)$, which needs to sum $exp(S_{i_g}(\mathbf{c}, F_c))$ over the whole itemset for each training instance. That is to say, training this model take $|I| \times |D|$ times of computation to get the normalization constant for each iteration, which makes the training process intractable. To tackle this problem, we adopt a sub-sampling approach, namely noise-constrictive estimation (NCE) [8] to deal with the normalization calculation of softmax function in the training process. We sample 50 negative items each time in the experiment.

All the parameters $\Theta$ are learned by back propagation. Algorithm 1 summarizes the learning process briefly. In Algorithm 1, $\odot$ denotes element-wise product operation, $i_o$ is the output item which includes both positive example and noise examples. $i_j \in \mathbf{c}$ is the input item from the context and $\omega_{i_j}$ is the corresponding combination weight used in Eq. 2. Similarly, $F_{i_j}$ is the corresponding input feature of item $i_j$ and $\omega_{F_{i_j}}$ is its corresponding combination weight used in Eq. 4.

Due to the large computation cost and the strength in matrix calculation of GPUs, a GPU-based adaptive stochastic gradient descent (SGD) optimizer is designed to speed up the training process.

**Algorithm 1** NTEM Parameter Learning Using Gradient Descent

1: $l \leftarrow 0$
2: **while** not converged **do**
3: Compute $w^o_{i_o}$-gradient (Eq.1): $g_{w^o_{i_o}} \leftarrow \mathbf{E}_c^\top$
4: Compute $w^p_{i_o}$-gradient (Eq.3): $g_{w^p_{i_o}} \leftarrow \mathbf{E}_{F_c}^\top$
5: Compute $w^t_{:,i_j}$-gradient (Eq.5):
$g_{w^t_{:,i_j}} \leftarrow \omega_{i_j} \mathbf{W}^{o\top}_{i_o,:} \odot \mathbf{E}_i \odot (1 - \mathbf{E}_i)$
6: Compute $w^f_{:,F_{i_j}}$-gradient (Eq.5):
$g_{w^f_{:,F_{i_j}}} \leftarrow \omega_{F_{i_j}} \mathbf{W}^{p\top}_{F_{i_o},:} \odot \mathbf{E}_{F_i} \odot (1 - \mathbf{E}_{F_i})$
7: Perform SGD-updates for $w^o_{i_o}$, $w^p_{i_o}$, $w^t_{:,i_j}$ and $w^t_{:,i_j}$:
$w^o_{i_o} \leftarrow w^o_{i_o} + S^l_{i_o}(g) g_{w^o_{i_o}}$, $w^p_{i_o} \leftarrow w^p_{i_o} + S^l_{i_o}(g) g_{w^p_{i_o}}$, $w^t_{:,i_j} \leftarrow w^t_{:,i_j} + S^l_{i_o}(g) g_{w^t_{:,i_j}}$, $w^f_{:,F_{i_j}} \leftarrow w^f_{:,F_{i_j}} + S^l_{i_o}(g) g_{w^f_{:,F_{i_j}}}$
8: $l \leftarrow l + 1$
9: **end while**

# 4 Experiments and Evaluation

## 4.1 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 and an item information table are extracted from each of the original data. 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 can not fit our model. This is because, we use at least one item as context for constructing the embeddings. The item information table contains the feature values of each item occurred in the transaction table, note that we only focus on those categorical features in this paper. Secondly, the transaction table is splitted into training and testing set. Specifically, we randomly choose $20\%$ from the transactions happened in last 30 days as the testing set, while others are used for training. The item information table is used in both training and testing processes. Finally, to test the performance of our proposed model under different cold-start levels, part of the transactions in training set are removed following certain rules. To be specific, we construct 4 different training sets with a drop rate of $0, 40\%, 80\%, 95\%$ respectively. Taking the one with drop rate of $40\%$ as an example, for each target item selected in the testing set, $40\%$ of all the transactions containing it in the training set are dropped. The characteristics of the datasets are shown in Table 1.

During the training, the transactions in the training set together with the corresponding item features are imported into the model in batches to learn embeddings of each item and each feature value. In the testing process, the learned embeddings are used to predict the target item given the $(n-1)$ ones. The real target item in the testing

[1] https://tianchi.aliyun.com/datalab/dataSet.htm?id=1
[2] http://stackoverflow.com/questions/25014904/download-link-for-ta-feng-grocery-dataset

Table 1: Statistics of experimental datasets

| **Statistics** | **IJCAI-15** | **Tafang** |
|---|---|---|
| #Transactions | 144,936 | 19,538 |
| #Items | 27,863 | 5,263 |
| #Features | 3 | 1 |
| 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 |

set is used as the ground truth. We calculate accuracy measures Recall@K [24] and MRR [5] by comparing the predicted results and the ground truth. We also calculate the recommendation novelty measures: global novelty and local novelty by comparing the recommendation list and the whole item population and the given context item set respectively. Finally, the performance of our proposed method is evaluated by comparing it and other related ones in terms of recommendation accuracy and novelty.

**Evaluated Methods** In the experiments, we compare our proposed NTEM with the following commonly used baselines:

• *FPMC:* A model that combines matrix factorization and Markov Chains for next-basket recommendation. The model factorize the personal transition matrix between items with a pairwise interaction model [18].

• *PRME:* A personalized ranking metric embedding method (PRME) to model personalized check-in sequences. The learned PRME is then used to recommend next new point of interest (POI) of users [6].

• *GRU4Rec:* A RNN-based approach for session-based recommendations by modeling the whole session using a modified RNN [10].

### 4.2 Performance Evaluation

Except for the traditional recommendation accuracy, we also evaluate the recommendation novelty to test the capability of the proposed method to recommend novel items.

**Accuracy Evaluation** We use the following widely used accuracy metrics for transaction-based recommendation to evaluate all the comparison approaches.The result of each approach is given in Table 2.

• *REC@K*:It measures the recall of the top-K ranked items in the recommendation list over all the testing instances [24]. Recall that in the real-world case most customers are only interested in the items recommended on the first one or two pages, so here we choose $K \in \{10, 50\}$. In practice, it's a big challenge to find the extract one true item from thousands of items.

• *MRR*:It measures the mean reciprocal rank of the predictive positions of the true target item on all the testing instances [5].

Table 2: Accuracy comparisons between different recommendation models

| | | IJCAI-15 | | | Tafang | | |
|---|---|---|---|---|---|---|---|
| **Scenario** | **Model** | **REC@10** | **REC@50** | **MRR** | **REC@10** | **REC@50** | **MRR** |
| drop 0 | FPMC | 0.0016 | 0.0025 | 0.0031 | 0.0189 | 0.0216 | 0.0089 |
| | PRME | 0.0555 | 0.0612 | 0.0405 | 0.0212 | 0.0305 | 0.0102 |
| | GRU4Rec | 0.1182 | 0.1566 | 0.0965 | 0.0428 | 0.0887 | 0.0221 |
| | NTEM | 0.2026 | 0.3224 | 0.1125 | 0.0689 | 0.1716 | 0.0231 |
| drop 40% | FPMC | 0.0012 | 0.0021 | 0.0026 | 0.0008 | 0.0010 | 0.0058 |
| | PRME | 0.0327 | 0.0411 | 0.0312 | 0.0102 | 0.0205 | 0.0095 |
| | GRU4Rec | 0.1108 | 0.1356 | 0.0868 | 0.0330 | 0.0659 | 0.0196 |
| | NTEM | 0.1928 | 0.2794 | 0.1117 | 0.0575 | 0.1049 | 0.0377 |
| drop 80% | FPMC | 0.0009 | 0.0017 | 0.0021 | 0.0005 | 0.0008 | 0.0020 |
| | PRME | 0.0212 | 0.0287 | 0.0215 | 0.0084 | 0.0125 | 0.0056 |
| | GRU4Rec | 0.0493 | 0.0611 | 0.0398 | 0.0110 | 0.0244 | 0.0054 |
| | NTEM | 0.1098 | 0.1450 | 0.0686 | 0.0254 | 0.0494 | 0.0072 |
| drop 95% | FPMC | 0.0003 | 0.0008 | 0.0012 | 0.0002 | 0.0004 | 0.0008 |
| | PRME | 0.0089 | 0.0113 | 0.0105 | 0.0071 | 0.0096 | 0.0043 |
| | GRU4Rec | 0.0233 | 0.0337 | 0.0173 | 0.0101 | 0.0172 | 0.0042 |
| | NTEM | 0.0318 | 0.0639 | 0.0173 | 0.0215 | 0.0305 | 0.0068 |

Table 2 demonstrates the results of REC@10, REC@50 and MRR over the testing sets of different cold-start levels. The number of factor is set to 10 for training FPMC as the best performance is achieved in such setting. The performance of FPMC on both data sets is quite poor, even in the warm start situation (e.g., REC@50=0.0025 when drop rate=0 on IJCAI-15). The main reason is that both data sets are extremely sparse and thus a very large but quite sparse matrix is constructed to train the MF model for each data set. For instance, in IJCAI-15 data set, each user only has an average of 3.6 transactions and each transaction only contains an average of 2.91 items of over 27,000 ones. This leads to that each row of the constructed matrix contains less than two items (cf. the avg. transaction length in Table 1, note that one out of the items need to be taken out as the output) and all other entries are empty. By calculation, we found the non-empty entries in the constructed matrix of IJCAI-15 account for less than 0.01%. We set the embedding dimensions to 60 as suggested in [6] when training PRME model. The performance of PRME is a little better than FPMC, but it's still poor especially in those cold start cases. This is because PRME is a fist-order MC model, which learns the transition probability over successive item instead of the whole context. Namely, this model only predict the target item based on its previous one while ignore all those bought before in the same transaction, which lost some information. Furthermore, in the real-word, the choice of items does not follow a rigid sequence assumed by such kind of models. Benefiting from the deep structure, GRU4Rec achieves much better performance compared with FPMC and PRME models. Especially, when we drop no more than 40% transactions, the REC@10 is above 10% on IJCAI-15, which can lead to a relative accurate recommendation in real-world business.

The batch size is empirically set to 200 and number of hidden units for the item and feature value embeddings are set to 50 and 20 respectively. We run 60 epochs to

train our NTEM model. It achieves much better performance than GRU4Rec, where the REC@10 and REC@50 exceed 20% and 30% respectively when drop nothing on IJCAI-15. In the extremely cold start case (drop 95% ), it also achieves obvious better performance than other methods on both data sets. The REC@10 and REC@50 of our model exceed 3% and 6% respectively, compared to around 2% and 3% of GRU4Rec on IJCAI-15. The higher MRR of our model also shows that we can accurately put the customers' desired items in the front of the recommendation list. The reason is multi-faceted. First of all, we use a complete context to predict one target item, and we donot assume a rigid order between the items within one transaction. This makes the input more informative and consistent with the reality compared with those models which only utilize part of the context and assume a rigid order between items. Second, more information is used by our model by incarnating the item features into it. More importantly, the coupling relations between item features and its transactions are learned and encoded into item and its feature embeddings in the back-forward propagation training mechanism. Thanks to the features and the coupling relations, additional information is provided to help with the item prediction and recommendation, especially for those novel items which lack of sufficient transaction information. What's more important, our model has a very concise structure which is easy to train. This shallow structure is efficient enough to retrain and recompute the score for ranking of all candidate items in an incremental dataset for online recommendations. However, GRU4Rec is a deep RNN consisting of GRU layers, which makes it more time consuming when new transaction records are added into the dataset.

**Novelty Evaluation** Except for accuracy, novelty is another important quality which should be considered in real-world recommendation scenarios [20]. Recall that in this paper we also try to recommend those infrequent or unpopular items, namely novel items, so the novelty is particularly important to measure the recommendation quality of our model. Specifically, we evaluate the recommendation novelty from both global perspective and local perspective by using different metrics.

• *Global novelty*: intuitively, the novelty of an item from the global perspective can be defined as the opposite of item popularity w.r.t the whole population. The item is novel if it occurs in few transactions, i.e., the item is far in the long tail of the popularity distribution [16]. Inspired by the inverse user frequency (IUF) proposed in [3] and [7], we define the concept of inverse transaction frequency (ITF) as $ITF = -log_2|T_i|/|T|$, where $T_i = \{t \in T | i \in t\}$ denotes the set of transactions containing item $i$. Similar to the novelty metric MIUF defined in [16], here we use the average of the $ITF$ (MITF) of the recommended items to measure the global novelty of a recommendation:

$$MITF = -\frac{1}{|R|}\sum_{i \in R} log_2 \frac{|T_i|}{|T|} \tag{9}$$

where $R$ is the set of recommended items in one recommendation and $T$ is the whole set of transactions. For all the $N$ recommendations on a certain testing dataset, the global recommendation novelty is defined as the mean of $MITF$ of each recommendation:

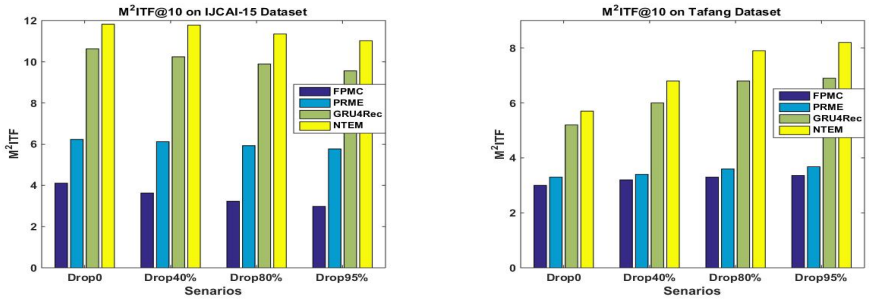
$$M^2ITF = \frac{1}{N}\sum MITF \tag{10}$$

• *Local novelty*: different from global novelty, local novelty refers to the difference of recommended items w.r.t the previous experience of the users. Generally, if the recommended items are more different from the already bought items, the more novel these items are. In our model, the already bought items correspond to the context $\mathbf{c}$ used for recommendation $R$. Given a recommendation list $R$, the more items in $R$ having been seen in the context $\mathbf{c}$, the less novelty the recommendation is. Based on this observation, the context-aware novelty ($CAN$) of recommendation $R$ can be defined as:

$$CAN = 1 - \frac{|R \cap \mathbf{c}|}{|R|} \tag{11}$$

Similarly, for all the $N$ recommendations on a certain testing set, the local recommendation novelty is defined as the mean of $CAN$ ($MCAN$) of all recommendations:

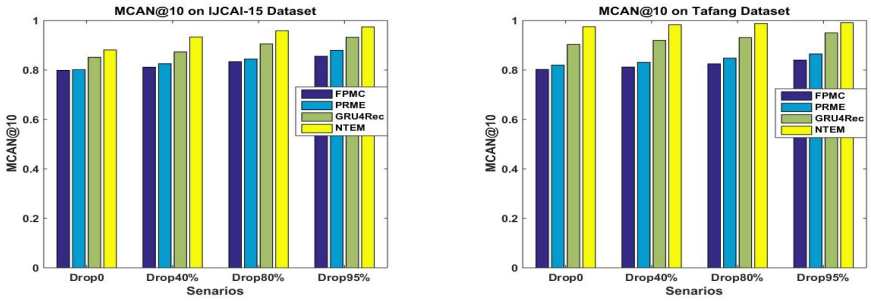
$$MCAN = \frac{1}{N} \sum CAN \tag{12}$$




(a) (b)

Fig. 2: $M^2ITF$ comparisons on top-10 items, NTEM achieves higher global novelty than other approaches

Fig. 2 and Fig. 3 show the results of global novelty and local novelty comparisons of top-10 recommendations over testing sets using the aforementioned measures $M^2ITF@10$ and $MCAN@10$ respectively on both data sets. Overally, our proposed NTEM achieves both higher global novelty and local novelty compared to other approaches. FPMC is not trained well on such sparse dataset and can not work well, so its recommendations are not precise and have a lot of randomness. Recall the fact that most of the candidate items in the dataset are frequent and not novel. Accordingly, we get low $M^2ITF$ and $MCAN$ on FPMC. PRME is a first-order Markov Chain model in which a sequential hypothesis is forced between the items within a transaction and it predicts the target item by only utilizing the one before it in the transaction. This is not always the case in the real world and it leads to information loss by ignoring other




(a) (b)

Fig. 3: $MCAN$ comparisons on top-10 items, NTEM achieves higher local novelty than other approaches

items in the transaction, so PRME also gets low novelties here. GRU units in GRU4Rec can accumulate the effect of all the sequential items in a transaction, it makes use of all other items in the transaction to predict the target one. Taking the advantage of the deep structure, it achieves relative high novelties. As a result, GRU4Rec is a relative good TBRS to make novel recommendations.

Our proposed NTEM do not have the sparse issue as FPMC due to its totally different work mechanism from matrix factorization used in FPMC, so it works well even on a sparse data set as shown in our experiment. Furthermore, NTEM does not assume a sequence between the items within a transaction, which is closer to the actual situation. More importantly, we make full use of all the other items in a transaction to predict the target one, which makes the prediction more solid by utilizing richer information. For the cold start issue, we incorporate the item features into the model and take the advantage of the information flow on feature values for prediction when the available transaction information is limited. So NTEM is more easily to provide novel recommendations and to achieve higher global and local novelties.

In practice, both the global and local novelties are somehow related to the recommendation accuracy here, specifically, some kind of positive relations exist between them. In the cold start scenarios, as all the target items are infrequent or novel, higher recommendation accuracy means more chance to get the target items into the recommendation list. Whereas more target items included in the recommendation list lead to higher novelty.

## 5 Discussions

Currently, we have to admit a weakness of our work on dealing with the purely cold items though it can get good results in the extremely high drop rate (i.e. 95%) case. We try to analyze the reasons and provide some suggestions in this section. On one hand, the model parameters are learned on training set, so they tend to reveal the existing patterns in training set naturally. However, the purely cold items never appear in the training

set, nothing is learned for them when training. On the other hand, for those purely cold items, only their features are used for predictions. This requires their feature values have occurred frequently in the training set and thus good embeddings can be learned. In addition, the item features also affect the prediction of new items greatly. In general, high-dimensional features have stronger capabilities to deliver information than low-dimensional ones. Unfortunately, the shopping-basket datasets in transaction domain usually have diverse feature values and low dimensions. For instance, the IJCAI-15 dataset only has three features: category, brand and seller. The average frequency of feature values in brand and seller is below five, which is quite infrequent. This leads to the feature values of new items in testing set occur too few times or even never occur, which leads to learning poor embeddings of them and thus poor information delivering capacity. Some potential solutions to improving our model for dealing with purely cold items may be: (1) finding more suitable datasets with high-dimensional features and intensive feature values, like audio data, image data or text data; (2) only using those features having already occurred in training set.

## 6 Conclusions

Perceiving the next choice in a dynamic context for onling recommendation is demanding but challenging. In this paper, we propose NTEM to build a more precise and efficient TBRS, it also shows great potential to improve the recommendation novelty. The empirical evaluation on real-world e-commerce datasets shows the com-prehensive superiority of our methods over other state-of-the-art ones. In the future, we will extend our model to other domains, like relational learning on social networks.

## 7 Acknowledgment

This work is partially sponsored by the China Scholarship Council (CSC Student ID 201406890033).

## References

1. Adda, M., Missaoui, R., Valtchev, P., Djeraba, C.: Recommendation strategy based on relation rule mining. In: IJCAI Workshop on Intelligent Techniques for Web Personalization (ITWP05). pp. 33–40 (2005)
2. Adomavicius, G., Tuzhilin, A.: Context-aware recommender systems. In: Recommender systems handbook, pp. 191–226. Springer (2015)
3. Breese, J.S., Heckerman, D., Kadie, C.: Empirical analysis of predictive algorithms for collaborative filtering. In: Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence. pp. 43–52. Morgan Kaufmann Publishers Inc. (1998)
4. Cao, L.: Coupling learning of complex interactions. Information Processing & Management 51(2), 167–186 (2015)
5. Chou, S.Y., Yang, Y.H., Jang, J.S.R., Lin, Y.C.: Addressing cold start for next-song recommendation. In: Proceedings of the 10th ACM Conference on Recommender Systems. pp. 115–118. ACM (2016)

6. Feng, S., Li, X., Zeng, Y., Cong, G., Chee, Y.M., Yuan, Q.: Personalized ranking metric embedding for next new poi recommendation. In: IJCAI. pp. 2069–2075 (2015)
7. Francesco Ricci, Lior Rokach, B.S.: Recommender Systems Handbook (2nd). Springer (2015)
8. Gutmann, M.U., Hyvärinen, A.: Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. Journal of Machine Learning Research 13(Feb), 307–361 (2012)
9. Han, J., Pei, J., Mortazavi-Asl, B., Chen, Q., Dayal, U., Hsu, M.C.: Freespan: frequent pattern-projected sequential pattern mining. In: Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 355–359. ACM (2000)
10. Hidasi, B., Karatzoglou, A., Baltrunas, L., Tikk, D.: Session-based recommendations with recurrent neural networks. arXiv preprint arXiv:1511.06939 (2015)
11. Huang, Z., Zeng, D.D.: 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 (2011)
12. Koren, Y.: Factorization meets the neighborhood: a multifaceted collaborative filtering model. In: Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 426–434. ACM (2008)
13. Lin, W., Alvarez, S.A., Ruiz, C.: Efficient adaptive-support association rule mining for recommender systems. Data mining and knowledge discovery 6(1), 83–105 (2002)
14. Melville, P., Mooney, R.J., Nagarajan, R.: Content-boosted collaborative filtering for improved recommendations. In: Aaai/iaai. pp. 187–192 (2002)
15. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Advances in neural information processing systems. pp. 3111–3119 (2013)
16. Park, Y.J., Tuzhilin, A.: The long tail of recommender systems and how to leverage it. In: Proceedings of the 2008 ACM conference on Recommender systems. pp. 11–18. ACM (2008)
17. Rendle, S.: Factorization machines with libfm. ACM Transactions on Intelligent Systems and Technology (TIST) 3(3), 57 (2012)
18. Rendle, S., Freudenthaler, C., Schmidt-Thieme, L.: Factorizing personalized markov chains for next-basket recommendation. In: Proceedings of the 19th international conference on World wide web. pp. 811–820. ACM (2010)
19. Rong, X.: word2vec parameter learning explained. arXiv preprint arXiv:1411.2738 (2014)
20. Vargas, S., Castells, P.: Rank and relevance in novelty and diversity metrics for recommender systems. In: Proceedings of the fifth ACM conference on Recommender systems. pp. 109–116. ACM (2011)
21. Wang, S., Liu, W., Wu, J., Cao, L., Meng, Q., Kennedy, P.J.: Training deep neural networks on imbalanced data sets. In: Neural Networks (IJCNN), 2016 International Joint Conference on. pp. 4368–4374. IEEE (2016)
22. Wu, X., Liu, Q., Chen, E., He, L., Lv, J., Cao, C., Hu, G.: Personalized next-song recommendation in online karaokes. In: Proceedings of the 7th ACM conference on Recommender systems. pp. 137–140. ACM (2013)
23. Yap, G.E., Li, X.L., Philip, S.Y.: Effective next-items recommendation via personalized sequential pattern mining. In: International Conference on Database Systems for Advanced Applications. pp. 48–64. Springer (2012)
24. Yuan, Q., Cong, G., Ma, Z., Sun, A., Thalmann, N.M.: Time-aware point-of-interest recommendation. In: Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval. pp. 363–372. ACM (2013)