An Attentive Interaction Network for Context-aware Recommendations

Lei Mei*
Shandong University
lei.mei@outlook.com

Pengjie Ren*
Shandong University
jay.ren@outlook.com

Zhumin Chen
Shandong University
chenzhumin@sdu.edu.cn

Liqiang Nie
Shandong University
nieliqiang@gmail.com

Jun Ma
Shandong University
majun@sdu.edu.cn

Jian-Yun Nie
Université de Montréal
nie@iro.umontreal.ca

ABSTRACT

Context-aware Recommender Systems (CARS) have attracted a lot of attention recently because of the impact of contextual information on user behaviors. Recent state-of-the-art methods represent the relations between users/items and contexts as a tensor, with which it is difficult to distinguish the impacts of different contextual factors and to model complex, non-linear interactions between contexts and users/items.

In this paper, we propose a novel neural model, named Attentive Interaction Network (AIN), to enhance CARS through adaptively capturing the interactions between contexts and users/items. Specifically, AIN contains an *Interaction-Centric Module* to capture the interaction effects of contexts on users/items; a *User-Centric Module* and an *Item-Centric Module* to model respectively how the interaction effects influence the user and item representations. The user and item representations under interaction effects are combined to predict the recommendation scores. We further employ effect-level attention mechanism to aggregate multiple interaction effects.

Extensive experiments on two rating datasets and one ranking dataset show that the proposed AIN outperforms state-of-the-art CARS methods. In addition, we alsofi nd that AIN provides recommendations with better explanation ability with respect to contexts than the existing approaches.

CCS CONCEPTS

• **Information systems** → **Recommender systems**;

KEYWORDS

Context-aware Recommendations; Interaction Networks; Explainable Recommendations

*Co-first author.

CIKM '18, October 22–26, 2018, Torino, Italy

ACM ISBN 978-1-4503-6014-2/18/10…$15.00
https://doi.org/10.1145/3269206.3271813

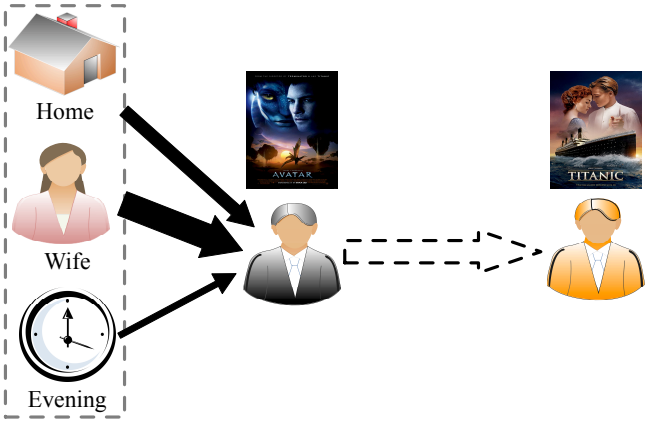


Figure 1: Illustration of context effects on user behaviors. The solid arrows represent the effects of contexts, and the thickness indicates the importance of context effects. Dashed arrow reveals the change of user interest with the influence of contexts.

ACM Reference Format:
Lei Mei, Pengjie Ren, Zhumin Chen, Liqiang Nie, Jun Ma, and Jian-Yun Nie. 2018. An Attentive Interaction Network for Context-aware Recommendations. In *The 27th ACM International Conference on Information and Knowledge Management (CIKM '18), October 22–26, 2018, Torino, Italy*. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3269206.3271813

1 INTRODUCTION

In Recommender Systems (RS), it is important to build models to capture the user's interests and the interactions with items. Traditional RS do not take into account the context such as time and location. However, the interest of a user may change depending on the context in which (s)he is. As illustrated in Figure 1, a user is interested in sciencefi ctions (e.g., Avatar) intrinsically. However, when he is at home with his wife in the evening (here the contexts are "home", "with his wife", and "evening"), his interest might change, and he might prefer to watching romantic movies (e.g., Titanic) at that moment. This example shows that it is essential to consider contexts in RS, since they have subtle but powerful effects on user behaviors [32]. Such recommender system is called Context-aware Recommender System (CARS).

The importance of context has been widely recognized in the area. Early studies [15, 30, 31] integrate contextual information by extending latent factor models, which usually treat contexts as

some additional dimensions similar to those of users and items, and capture relevances between contexts and users/items. A key problem with such an approach is the difficulty to explain some observations. For example, the latent factors would induce that a user has a stronger relation with "Friday" than with "Monday", but this relation is hard to interpret. To solve this problem, recent studies propose to generate context-aware or context-specific representations of users and items. Shi et al. [32] present a new formulation of latent factor models based on context-aware representations of users and items. These representations are learned by including an additional layer of latent space for users/items given a context. Liu et al. [23] and Wu et al. [38] propose Contextual Operating Tensor (COT) model, which represents a context as a latent vector and the effects of contexts as a tensor. Despite the fact that these models can cope with contexts to some extent, several issues remain. First, these approaches put all the contextual factors together, and expect that the latent model is capable of sorting out the impact of each of them. In other words, there are no explicit models for different types of interaction such as between context and user and between context and item. We believe that different types of interaction play different roles in CARS and they should be modeled in different ways explicitly. Second, the existing approaches rely on linear operations (e.g., matrix factorization) on the observation data to build models. The true influence of the context may be much more complex. Therefore, the technical approaches used may be limited for this purpose. Third, different contexts usually have different effects. For example, the context of "his wife's companionship" is more important than "evening" as shown in Figure 1. However, the existing approaches only consider the overall context effect, and they are unable to distinguish the different effects of different contexts.

In this paper, we will address the above problems. Wefi rst reformulate the CARS task by defining distinct objects (users, items, and contexts) and interactions (context-user interaction, context-item interaction, and user-item interaction), which makes a clear distinction between different types of interaction. Then, inspired by recent advance of Interaction Networks (IN) [7], we propose a novel neural model, called Attentive Interaction Network (AIN), which explores the use of deep neural network to model the interaction effects of contexts on user and item representations. IN is initially proposed for the physical reasoning task which involves physical objects and their relations. Here we treat the physical objects as users, items, and contexts and relations as the interactions among users, items, and contexts in CARS. Thus we can take advantage of IN to explicitly model the effects of different interactions (i.e., context-user interaction, context-item interaction, and user-item interaction) on users and items. This allows us to selectively process different types of interaction in different ways. Specifically, AIN consists of three modules, namely *Interaction-Centric Module*, *User-Centric Module*, and *Item-Centric Module*. *Interaction-Centric Module* is designed to explicitly model the interaction effects of contexts on users/items, which takes users/items and contexts as input, and returns the effects of context-user interactions and context-item interactions. *User-Centric Module* models how the effects of context-user interactions influence the user representations. It takes as input the aggregated interaction effects and the user representations, and returns the context-specific representations for the users. Similarly, *Item-Centric Module* models how the effects of context-item interactions influence the item representations and returns the context-specific representations for the items. Besides, we aggregate different interaction effects with attention mechanism [8, 20, 39] to capture the adaptive effects of different interactions. All neural parameters are trained in an end-to-end back-propagation paradigm.

To sum up, the main contributions in this paper are as follows:

- We enhance CARS by proposing AIN for explicitly modeling the high-order and non-linear interactions between contexts and users/items.
- We augment AIN with context-aware attention mechanism to enable it capture the adaptive effects of different contexts on users and items. With this mechanism, AIN can provide explainable recommendations by recognizing the most influential contexts.
- Through extensive experiments conducted on three real-world datasets, we show that AIN consistently outperforms the state-of-the-art CARS methods on both rating and ranking tasks.

2 RELATED WORK

2.1 Conventional methods

Conventional CARS methods can be grouped into three categories: *contextual pre-filtering*, *contextual post-filtering*, and *contextual modeling* [1].

Contextual pre-filtering methods use contextual information to drive data selection or data construction [2, 3, 27]. One popular technique for *contextual pre-filtering* is item splitting [4, 5], where each item is split into severalfi ctitious items based on the contexts. Similarly to item splitting, Baltrunas and Amatriain [3] propose user splitting, which splits the user profile into several sub-profiles and each sub-profile represents the user in a particular context. Zheng et al. [41] propose UI splitting, which is a combination of item splitting and user splitting. *Contextual post-filtering* methods use contexts to adjust the recommendation results [27]. Panniello et al. [27] introduce two different *contextual post-filtering* methods: *Weight* and *Filter*; the former reorders the recommended items by weighting the predicted rating with the probability of relevance in the specific context, while the latterfi lters out recommended items that have small probability of relevance in the specific context. *Contextual pre-filtering* and *post-filtering* methods may work in practice, but they require supervision andfi ne-tuning in all steps of recommendation [30].

Contextual modeling methods capture the contextual information directly in model constructions. Some studies are based on matrix factorization (MF) [6, 19, 42, 43]. Baltrunas et al. [6] present context-aware matrix factorization (CAMF), which extends MF by considering different influences of contextual information on the item bias. Li et al. [19] introduce two improved context-aware matrix factorization approaches (ICAMF), where context-user and context-item interactions are modeled by MF. Moreover, Zheng et al. [42, 43] extend a new matrix factorization method called sparse linear method (SLIM), and introduce a series of contextual SLIM (CSLIM) models for top-N recommendations. Some studies are based on tensor factorization (TF) [12, 15, 31, 40]. Karatzoglou

et al. [15] propose multiverse recommendation, which employs the Tucker decomposition [34] on the user-item-context rating tensor. To tackle the context-aware recommendation for implicit feedbacks, Shi et al. [31] introduce a ranking based TF method by directly optimizing mean average precision. Other studies are based on factorization machines [25, 30]. Rendle et al. [30] apply factorization machines to model the interactions between each pair of entities through their latent factors (e.g., user-user, user-item, and user-context interactions). However, these methods usually treat contexts as certain dimensions similar to those of users and items. The relevance between a user and a context (or an item and a context) is not intuitive and difficult to interpret [32].

To solve the above issues, recent studies propose to generate context-aware or context-specific representations of users and items [22, 23, 32, 38]. Shi et al. [32] propose a novel CARS^2 model which provides each user/item with not only a latent vector but also a context-aware representation. Liu et al. [23] and Wu et al. [38] represent the common semantic effects of contexts as a contextual operating tensor. A contextual operating matrix generated from the tensor and context latent vectors is used to model the semantic operations of contexts on users and items.

In summary, all above methods model the interactions between contexts and users/items in a linear way, which might be insufficient to capture the complex effects of contexts on users and items.

2.2 Deep learning methods

Deep learning has recently been applied to context-aware recommendation tasks and attracted significant attention.

Huang et al. [14] and Ebesu and Fang [9] study the task of citation recommendation, where context is defined as a sequence of words that appear around a particular citation. Rawat and Kankanhalli [28] study the problem of multi-label image tagging; they present a unified framework to integrate context information with image content for tag prediction. Kim et al. [16] employ convolutional neural network to capture contextual information such as surrounding words and word orders to get a deeper understanding of item description documents. Unger et al. [35, 36] utilize an autoencoder to extract latent contexts from a rich set of mobile sensors in a unsupervised manner. Liu et al. [24] introduce Context-Aware Recurrent Neural Networks (CA-RNN), which take into account the contextual information for sequential modeling tasks. He and Chua [11] propose Neural Factorization Machines (NFM) for sparse data prediction, which enhances FM by modeling high-order and nonlinear feature interactions. Xiao et al. [39] introduce Attentional Factorization Machines (AFM) that enhances FM by learning the importance of feature interactions with an attention network. Similar to FM, NFM and AFM can be applied to the task of context-aware recommendations by specifying the input data.

Although above studies explore the use of deep learning for modeling contextual information, they target different recommendation tasks from ours. Besides, most of the existing deep learning methods (including recently proposed NFM) fail to distinguish the different importance of different context effects in context-aware recommendations, which is one of the major concerns of our work. Though AFM can automatically discriminate the importance of feature interactions, it still models the feature interactions in a linear way. Besides, AFM models all types of interactions in the same way. These two limitations will limit the expressiveness of AFM in modeling the context effects on users and items (as we show in the experiments).

3 ATTENTIVE INTERACTION NETWORK

3.1 Problem formulation

The problem we deal with in this paper is item recommendation for users in a given context. We assume that there are M contextual factors $C_1, C_2, ..., C_M$, such as location, companion, and time . A context value c_m is a variable of the contextual factor C_m, where $m = 1, ..., M$. A unique combination $(c_1, ..., c_M)$ is named context combination $\boldsymbol{c}$. Let $u \in \mathcal{U}$ and $v \in \mathcal{V}$ denote a user and an item respectively. The task of CARS is to estimate the user-item preference score under a specific context combination $\boldsymbol{c}$ as expressed in Eq. 1.

$$\begin{aligned} \hat{r}_{u,v,\boldsymbol{c}} &= f(u, v, c_1, ..., c_M), \\ \boldsymbol{c} &= (c_1, ..., c_M), \end{aligned} \tag{1}$$

where $f(\cdot)$ denotes the predictive model. $\hat{r}_{u,v,\boldsymbol{c}}$ denotes the predicted score for the user-item interaction $r_{u,v,\boldsymbol{c}}$ under the context combination $\boldsymbol{c}$.

In this paper, we will model a user, an item and a context by a vector of latent factors. We use $\mathbf{p}_u \in \mathbb{R}^D$ to denote the latent vector of user u without any contexts, where D is the dimensionality of user latent vector. Similarly, $\mathbf{q}_v \in \mathbb{R}^D$ denotes the latent vector of item v without any contexts. We use $\mathbf{h}_{c_m} \in \mathbb{R}^{D_c}$ to denote the D_c-dimensional latent vector of context value c_m.

3.2 General framework

The overall framework of AIN is illustrated in Figure 2. It employs two symmetric pathways to model the effects of contexts on users and items respectively. Each pathway is composed of three core components: 1) *Interaction-Centric Module* captures the effects of interactions between contexts and users/items; 2) *Effect-Level Attention* aggregates the interaction effects with attention weights; 3) *User/Item-Centric Module* models how the aggregated interaction effects influence users/items. Through the three components, AIN learns the context-specific latent vector for user u, denoted by $\mathbf{p}_{u,\boldsymbol{c}} \in \mathbb{R}^D$, and the context-specific latent vector for item v, denoted by $\mathbf{q}_{v,\boldsymbol{c}} \in \mathbb{R}^D$. $\mathbf{p}_{u,\boldsymbol{c}}$ and $\mathbf{q}_{v,\boldsymbol{c}}$ are then used to predict user u's preference score $\hat{r}_{u,v,\boldsymbol{c}}$ for item v under context combination $\boldsymbol{c}$ as in Eq. 2.

$$\hat{r}_{u,v,\boldsymbol{c}} = \mu + b_u + b_v + \sum_{m=1}^{M} b_{c_m} + \mathbf{p}_{u,\boldsymbol{c}}{}^T \mathbf{q}_{v,\boldsymbol{c}}. \tag{2}$$

A key element in the above equation is the relation between a user and an item through context factors. This is formulated as the last element in the equation. The score is also influenced by a set of biases: μ is the global bias; b_u and b_v are the biases of user u and item v respectively; and b_{c_m} is the bias of context value c_m.

Next, wefi rst present the details of *Interaction-Centric Module*, *Effect-Level Attention*, and *User/Item-Centric Module*. Note that we only elaborate on the user-side pathway, since the item-side pathway is implemented in a similar way but with different parameters. Then, we introduce the parameter learning process of AIN.

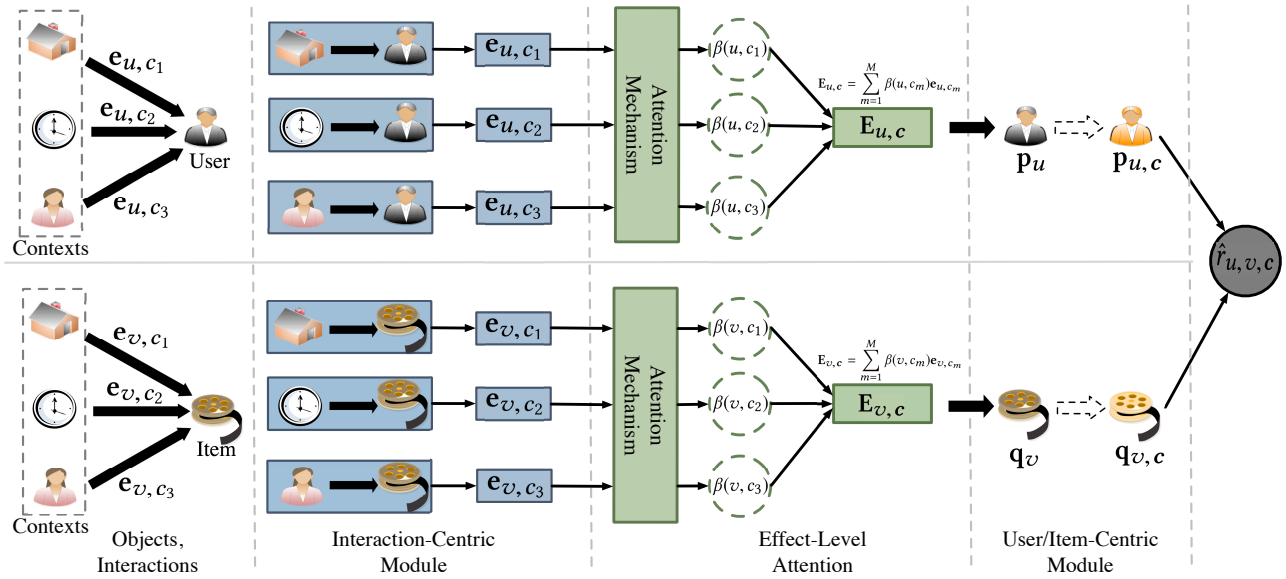


Figure 2: The framework of AIN. Two pathways are employed to model the effects of contexts on users and items. Each pathway is composed of three components: *Interaction-Centric Module, Effect-Level Attention,* and *User/Item-Centric Module.*

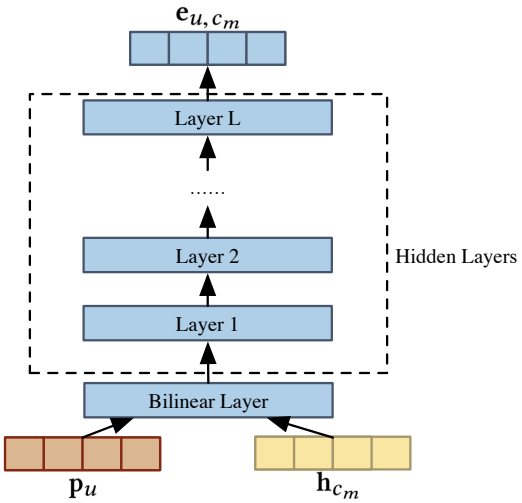


Figure 3: The architecture of the user-side Interaction-Centric Module.

3.3 Interaction-Centric Module

The goal of *Interaction-Centric Module* is to capture the effects of interactions between contexts and users. To achieve this, we employ an interaction-centric function as shown in Eq. 3.

$$\mathbf{e}_{u,c_m} = f_I(\mathbf{p}_u, \mathbf{h}_{c_m}), \tag{3}$$

where $f_I(\cdot)$ denotes an interaction-centric function of two objects, i.e., user u and context c_m. $\mathbf{e}_{u,c_m} \in \mathbb{R}^{D_e}$ denotes the D_e-dimensional latent vector of interaction effect.

As shown in Figure 3, we use a multi-layer perceptron network to model the interaction. Since user and context are different kinds of objects with different characteristics, wefi rst map user latent vector $\mathbf{p}_u$ and context latent vector $\mathbf{h}_{c_m}$ to a shared hidden space using a bilinear layer, as shown in Eq. 4.

$$\mathbf{z}^{\varphi} = \sigma(\mathbf{W}^{\varphi}_{uz}\mathbf{p}_u + \mathbf{W}^{\varphi}_{cz}\mathbf{h}_{c_m} + \mathbf{b}^{\varphi}_{z}), \tag{4}$$

where $\mathbf{W}^{\varphi}_{uz}$ and $\mathbf{W}^{\varphi}_{cz}$ are the mapping matrices for user latent vector and context latent vector respectively, and $\mathbf{b}^{\varphi}_{z}$ is the bias term. The superscript φ refers to model parameters related to *Interaction-Centric Module*. $\sigma(\cdot)$ is the Rectifier (ReLU) activation function [10], which is more biologically plausible and proven to be non-saturated.

We then place a stack of fully connected layers above the bilinear layer as follows.

$$\begin{aligned} \mathbf{z}^{\varphi}_1 &= \sigma_1(\mathbf{W}^{\varphi}_1\mathbf{z}^{\varphi} + \mathbf{b}^{\varphi}_1), \\ \mathbf{z}^{\varphi}_2 &= \sigma_2(\mathbf{W}^{\varphi}_2\mathbf{z}^{\varphi}_1 + \mathbf{b}^{\varphi}_2), \\ &\cdots\cdots \\ \mathbf{z}^{\varphi}_L &= \sigma_L(\mathbf{W}^{\varphi}_L\mathbf{z}^{\varphi}_{L-1} + \mathbf{b}^{\varphi}_L), \end{aligned} \tag{5}$$

where L denotes the number of hidden layers. $\mathbf{W}^{\varphi}_l$, $\mathbf{b}^{\varphi}_l$, $\sigma_l(\cdot)$, and $\mathbf{z}^{\varphi}_l$ denote the weight matrix, bias vector, ReLU activation function, and output vector of the l-th hidden layer respectively. As for the structure of hidden layers (i.e., size of each layer), we set all hidden layers with the same size.

Finally, the output vector of the last hidden layer $\mathbf{z}^{\varphi}_L$ is transformed to the latent vector of the interaction effect between user u and context c_m, as shown in Eq. 6.

$$\mathbf{e}_{u,c_m} = \mathbf{W}^{\varphi}_{ze}\mathbf{z}^{\varphi}_L + \mathbf{b}^{\varphi}_{ze}, \tag{6}$$

where $\mathbf{W}^{\varphi}_{ze}$ and $\mathbf{b}^{\varphi}_{ze}$ denote the weight matrix and bias vector of the output layer respectively.

3.4 Effect-Level Attention

To differentiate the importance of interaction effects, we propose *Effect-Level Attention*. Specifically, given effect latent vector $\mathbf{e}_{u,c_m}$

and user latent vector $\mathbf{p}_u$, we use a single-layer perceptron to compute the attention score $a(u, c_m)$ as in Eq. 7.

$$a(u, c_m) = \sigma(\mathbf{W}_1^\tau \mathbf{e}_{u,c_m} + \mathbf{W}_2^\tau \mathbf{p}_u + b^\tau), \quad (7)$$

where $\mathbf{W}_1^\tau$ and $\mathbf{W}_2^\tau$ denote the weight matrices, and b^τ is bias. The superscript τ refers to model parameters related to the attention calculation component. Then, thefi nal weights are obtained by normalizing the above attentive scores using softmax function, which can be interpreted as the importance of effect $\mathbf{e}_{u,c_m}$ in influencing user u, as shown in Eq. 8.

$$\beta(u, c_m) = \frac{\exp(a(u, c_m))}{\sum_{n=1}^{M} \exp(a(u, c_n))}. \quad (8)$$

After we obtain the effect-level attention $\beta(u, c_m)$, the latent vector of aggregated effects acting on user u is calculated by Eq. 9.

$$\mathbf{E}_{u,\boldsymbol{c}} = \sum_{m=1}^{M} \beta(u, c_m)\mathbf{e}_{u,c_m}. \quad (9)$$

3.5 User/item-Centric Module

The goal of *User-Centric Module* is to predict how the effects of context-user interactions influence the user representations. Given the user latent vector $\mathbf{p}_u$ and the latent vector of aggregated effects $\mathbf{E}_{u,\boldsymbol{c}}$, we employ a user-centric function to learn the context-specific latent vector $\mathbf{p}_{u,\boldsymbol{c}}$ for user u under context combination $\boldsymbol{c}$ as in Eq. 10.

$$\mathbf{p}_{u,\boldsymbol{c}} = f_{\mathrm{U}}(\mathbf{p}_u, \mathbf{E}_{u,\boldsymbol{c}}), \quad (10)$$

where $f_{\mathrm{U}}(\cdot)$ denotes a user-centric function.

Similar to *Interaction-Centric Module*, wefi rst use a bilinear layer to combine user latent vector $\mathbf{p}_u$ and the latent vector of aggregated context effects $\mathbf{E}_{u,\boldsymbol{c}}$, as shown in Eq. 11.

$$\mathbf{z}^\psi = \sigma(\mathbf{W}_{uz}^\psi \mathbf{p}_u + \mathbf{W}_{ez}^\psi \mathbf{E}_{u,\boldsymbol{c}} + \mathbf{b}_z^\psi), \quad (11)$$

where $\mathbf{W}_{uz}^\psi$ and $\mathbf{W}_{ez}^\psi$ are the weight matrices. b_z^ψ and $\sigma(\cdot)$ denote the bias vector and ReLU function respectively. Note that the superscript ψ refers to model parameters related to *User-Centric Module*.

To endow non-linearity to learn the complex effects on the user, we apply a stack of fully connected layers above the bilinear layer as in Eq. 5. Assume that $\mathbf{z}_L^\psi$ is the output of the last hidden layer. The output layer maps $\mathbf{z}_L^\psi$ into a D-length context-specific latent vector for user u, as shown in Eq. 12.

$$\mathbf{p}_{u,\boldsymbol{c}} = \mathbf{W}_{zu}^\psi \mathbf{z}_L^\psi + \mathbf{b}_{zu}^\psi, \quad (12)$$

where $\mathbf{W}_{zu}^\psi$, $\mathbf{b}_{zu}^\psi$ denote the weight matrix and bias vector of the output layer respectively.

3.6 Objective function

The proposed method can be used for predicting the rating or the ranking of an item for a user. We define the respective objective functions for the tasks. For rating task where $r_{u,v,\boldsymbol{c}}$ is a real value, we minimize the following squared loss:

$$\min_{\Theta} \sum_{(u,v,\boldsymbol{c})\in\mathcal{R}_{train}} (r_{u,v,\boldsymbol{c}} - \hat{r}_{u,v,\boldsymbol{c}})^2, \quad (13)$$

where Θ denotes parameters to be learned and $\mathcal{R}_{train}$ denotes the training set.

Table 1: Statistics of the evaluation datasets.

Dataset	Users	Items	Contexts	Interactions	Scale
Food	212	20	2	6,360	1-5
Frappe	953	4073	3	95,889	Raw frequency
Yelp	96,143	49,482	4	2,283,913	1-5

For ranking task, we optimize the following pairwise ranking loss of BPR [29]:

$$\arg\min_{\Theta} \sum_{(u,v,v',\boldsymbol{c})\in\mathcal{R}_B} -\ln g(\hat{r}_{u,v,\boldsymbol{c}} - \hat{r}_{u,v',\boldsymbol{c}}), \quad (14)$$

where v is an item preferred to item v' for user u under context combination $\boldsymbol{c}$ and $g(\cdot)$ is the logistic function. The training data $\mathcal{R}_B$ is generated as:

$$\mathcal{R}_B = \left\{(u, v, v', \boldsymbol{c}) | v \in \mathcal{R}_{train}(u) \wedge v' \in \mathcal{V} \setminus \mathcal{R}_{train}(u)\right\}, \quad (15)$$

where $\mathcal{R}_{train}(u)$ represents the set of items that are interacted by user u in training set $\mathcal{R}_{train}$, and $\mathcal{V}$ is the set of all items.

4 EXPERIMENTAL SETUPS

4.1 Datasets

We conduct our experiments on three publicly accessible datasets: Food [26], Yelp[1] and Frappe. The characteristics of the three datasets are summarized in Table 1.

(1) **Food.** This dataset was used by Ono et al. [26] and provided by authors. It contains 6,360 5-scale ratings by 212 users on 20 menu items, and each rating is associated with 2 contextual factors. One factor describes if the situation in which the user rates is virtual or real (2 values: real and virtual), and the second factor captures how hungry the user is (3 values: hungry, normal and full). The dataset is used to test the rating task.

(2) **Yelp.** To validate the scalability of AIN model, we use Yelp dataset in our experiment due to the lack of large-scale contextual datasets. The original data is very large but highly sparse. As such, we further process the dataset by retaining users and items with at least 10 interactions. This results in a subset of data that contains 96,143 users, 49,482 items, and 2,283,913 interactions. We use the following contextual factors: year, month, day of the week, and city. This dataset is used to evaluate the rating task.

(3) **Frappe.** This implicit feedback dataset is collected from Frappe[2], which is a context-aware personalized recommender of mobile apps. In our experiments, we use 3 contextual factors: daytime (with 7 possible values: e.g., morning or afternoon), day of the week (with 2 possible values: weekday and weekend), and location (with 3 possible values: work, home, and other place). This dataset is used for the ranking task.

4.2 Parameter settings

Our framework is implemented with Tensorflow[3]. During training, we randomly initialize model parameters with a Gaussian distribution (with a mean of 0 and standard deviation of 0.01), optimizing

[1] https://www.yelp.com/dataset/challenge
[2] http://frappe.cc/
[3] https://www.tensorflow.org.

the model with mini-batch Adam [17]. The learning rate is set to 0.001 and the batch size is set to 256. To prevent overfitting, we employ dropout [33] with drop ratio 0.4. We set $D = D_c = D_e = 64$ for user/item latent vectors, context latent vectors and effect latent vectors. The size of the bilinear layer is set to 128, and the size of the hidden layers is set to 128. Without special mention, we employ 2 hidden layers for Interaction-Centric Module, and 1 hidden layer for User/Item-Centric Module. Wefi nd the optimal parameter settings for each baseline with grid search.

5 EVALUATION OF RATING PREDICTION WITH EXPLICIT FEEDBACK

5.1 Baselines

We compare with the following methods on the rating prediction task.

- **MF [18].** This is the standard matrix factorization method that characterizes both users and items by latent vectors inferred from observed ratings. We use MF as a context-unaware baseline to show the performance gap between context-aware and context-unaware methods.
- **Multiverse Recommendation [15].** This method applies the Tucker decomposition to factorize the user-item-context n-dimensional tensor.
- **CAMF-C [6].** CAMF extends MF by considering different influences of contextual information on the item bias. CAMF-C is a variant of CAMF which assumes that each context variable has a global influence on rating, unrelated with the specific item.
- **CAMF-CI [6].** Similar to CAMF-C, but CAMF-CI assumes that each context variable has different impacts on the rating of each item.
- **FM [30].** Factorization machines are easily applicable to a wide variety of context by specifying only the input data. We use LibFM[4] to implement the method.
- **COT [23].** This method models the semantic operation of contexts on users and items using a contextual operating matrix, which is generated from a contextual operating tensor and latent vectors of contexts.
- **AFM [39].** AFM extends FM by utilizing the attention mechanism to learn the importance of each feature interaction.
- **NFM [11].** This is a state-of-the-art neural network model for sparse data prediction. It deepens FM under the neural framework to learn high-order feature interactions.

5.2 Evaluation protocols

We randomly split each explicit feedback dataset into three portions: 80% for training, 10% for validation, and 10% for testing. The validation set is used for tuning hyper-parameters and thefinal performance comparison is conducted on the test set. For the evaluation of rating prediction, we employ two metrics: root mean squared error (RMSE) and mean absolute error (MAE), which are conventional metrics for quantifying the rating prediction error. The smaller RMSE and MAE, the better the performance.

[4]http://www.libfm.org

Table 2: Performance comparison of AIN and baselines on Food and Yelp datasets. * indicates that our model AIN performs significantly better than COT as given by paired t-test with $p < 0.05$.

	Food		Yelp	
	RMSE	MAE	RMSE	MAE
MF	1.167	0.950	1.128	0.889
Multiverse	1.119	0.901	1.104	0.878
CAMF-C	1.121	0.900	1.120	0.879
CAMF-CI	1.123	0.902	1.123	0.878
FM	1.065	0.882	1.109	0.872
COT	1.055	0.828	1.100	0.865
AFM	1.051	0.839	1.102	0.853
NFM	1.013	0.806	1.103	0.854
AIN	**0.998***	**0.791***	**1.097***	**0.852***

Table 3: Performance comparison of AIN and baselines on Frappe dataset. The results of BPR, TFMAP, and CARS^2-Pair are obtained directly from [32]. * indicates that AIN performs significantly better than FM+BPR as given by paired t-test with $p < 0.05$.

	MAP@10	P@5	R@5	P@10	R@10
BPR	0.121	0.061	0.160	0.037	0.184
TFMAP	0.127	0.061	0.159	0.039	0.195
CARS^2-Pair	0.140	0.068	0.183	0.044	0.226
FM+BPR	0.112	0.071	0.200	0.053	0.280
AFM+BPR	0.138	0.081	0.219	0.056	0.283
NFM+BPR	0.144	0.085	0.227	0.056	0.285
AIN	**0.167***	**0.086***	**0.230***	**0.057***	**0.288***

5.3 Rating prediction results

Table 2 illustrates the rating prediction performance w.r.t. RMSE and MAE on two explicit feedback datasets. It shows that our model consistently achieves the best performance on both MAE and RMSE metrics on all datasets. From the table, we have four main observations. First, context-aware models outperform the context-unaware model MF. This confirms the importance of utilizing the contextual information in recommender systems. Second, our proposed AIN consistently outperforms all comparative context-aware methods. Generally, AIN obtains improvements over the best baseline NFM by 1.5% in RMSE and 1.9% in MAE on Food dataset. On Yelp dataset, AIN improves over the best baseline by 0.3% in RMSE and 0.1% in MAE. This observation confirms that considering the context effects on user behavior and item properties and modeling their interactions nonlinearly with AIN leads to substantial performance in context-aware recommendations. Third, we observe that AIN outperforms COT on both datasets. This reflects the value of modeling context-user interactions and context-item interactions in a non-linear way, since COT only models higher-order interactions linearly. At last, we can see that AIN achieves improvements over NFM and AFM. This observation indicates the importance of modeling different types of interaction (context-user interaction, context-item interaction, and user-item interaction) in different ways, since NFM and AFM still models all types of interaction in the same way.

Table 4: Performance of AIN and AIN variants. AIN-USER refers to AIN without modeling the context effects on items. AIN-ITEM refers to AIN without modeling the context effects on users.

Model	Food		Yelp		Frappe				
	RMSE	MAE	RMSE	MAE	MAP@10	P@5	R@5	P@10	R@10
AIN-USER	1.024	0.827	1.098	0.859	0.165	0.091	0.246	**0.061**	**0.307**
AIN-ITEM	1.034	0.842	1.100	0.858	**0.168**	**0.093**	**0.251**	0.060	0.303
AIN	**0.998**	**0.791**	**1.097**	**0.852**	0.167	0.086	0.230	0.057	0.288

6 EVALUATION OF RANKING WITH IMPLICIT FEEDBACK

6.1 Baselines

We compare with the following methods on the ranking task.

- **BPR [29].** This method optimizes the MF model with a pair-wise ranking loss, which is tailored to learn from implicit feedback. Note that BPR is a context-unaware method.
- **TFMAP [31].** This method uses tensor factorization to model implicit feedback data with contextual information, and is learned by directly optimizing MAP.
- **CARS^2-Pair [32].** It is an extended version of CARS^2 method, which is adapted to implicit feedback by using a pair-wise ranking loss function.
- **FM+BPR.** FM [30] is designed for the rating prediction task. Here we adapt FM for the ranking task on implicit feedback by optimizing it with the pair-wise ranking loss of BPR.
- **AFM+BPR and NFM+BPR.** To apply deep learning method AFM [39] and NFM [11] to the ranking task on implicit feedback, we optimize AFM and NFM with the BPR pair-wise learning objective function respectively.

6.2 Evaluation protocols

To evaluate the performance of top-N recommendation, we adopt the widely ranking evaluation metrics: recall, precision, and mean average precision (MAP). The higher the values of precision, recall and MAP, the better the recommendation performance. Note that we follow the convention in [32] and randomly sample 1000 items that have no feedback as irrelevant ones for each user under each context in the test set, and rank the test items among the 1000 items.

6.3 Ranking performance

Table 3 shows the ranking performance of our AIN and comparative models on Frappe dataset. We can make the following observations. First, context-aware methods generally outperform the context-unaware method BPR by integrating contextual information into the model. This further confirms the usefulness of contextual information for better recommendations. Second, AIN consistently outperforms the other context-aware methods. This demonstrates the effectiveness of AIN in exploiting the context effects on users and items. Third, we observe that CARS^2-Pair shows improvement over TFMAP, which is attributed to its modeling of context-aware representations. Meanwhile, the performance gap between CARS^2-Pair and our AIN indicates that AIN can learn high-quality context-specific representations of users and items with contexts, which is conducive to estimating the interactions between contexts and users/items precisely. At last, AIN outperforms NFM and AFM. This further verifies the substantial influence of modeling different types of interactions in different ways and the advantage of our proposed framework.

Table 5: Performance of AIN and AIN variants on Frappe dataset when recommending only new items to users.

Model	MAP@10	P@5	R@5	P@10	R@10
AIN-USER	0.0186	0.0117	**0.0269**	0.0090	0.0345
AIN-ITEM	0.0153	0.0105	0.0225	0.0087	0.0322
AIN	**0.0199**	**0.0121**	0.0263	**0.0094**	**0.0356**

7 ANALYSIS AND CASE STUDY

7.1 Impact of context

To investigate the effects of contexts on users and items, we compare the performance of AIN and two variants of AIN. AIN-USER refers to AIN without modeling the context effects on items, and AIN-ITEM refers to AIN without modeling the context effects on users. The empirical results on three datasets are summarized in Table 4. We have the followingfindings.

First, AIN-USER slightly outperforms AIN-ITEM on Food and Yelp datasets. This indicates that considering the effects of contexts on users is more effective than considering the effects of contexts on items to improve recommendation performance. This is reasonable because user interests are more easily influenced by contexts as compared to item properties. Second, AIN outperforms both AIN-USER and AIN-ITEM on Food and Yelp datasets. This may be due to the fact that contexts characterize the situation where users interact with items, and are associated with both of them at the same time. Therefore, the performance can be further improved by taking into account the effects of contexts on both users and items. At last, on Frappe dataset, it is interesting tofi nd that AIN-USER and AIN-ITEM achieve similar performance, while AIN performs worse. The reason behind this observation may be related to the particularity of the app usage scenario, where a user might use the same app repeatedly under various contexts. Moreover, compared with AIN, AIN-USER and AIN-ITEM are more likely to recommend a user the items that (s)he has already interacted with under other context combinations. To verify this, we restrict AIN to recommending new items to users, and conduct the same experiment on Frappe dataset in Table 5. From the table, we have two interestingfi ndings: 1) The performance of AIN, AIN-USER, and AIN-ITEM is much worse than before. This confirms the repetitiveness of app usage scenarios in Frappe dataset; 2) The performance of AIN is better than that of AIN-USER and AIN-ITEM (with the exception of AIN-USER on $R@5$). This indicates that the generalization ability of our AIN is better than that of AIN-USER and AIN-ITEM.

Table 6: Performance of AIN w.r.t. different context combinations on Food dataset.

Model	Context Combination	Food	
		RMSE	MAE
AIN	**Virtuality**	1.151	0.930
	Hunger	1.081	0.877
	Virtuality+Hunger	**0.998**	**0.791**

Table 7: Performance of AIN w.r.t. different context combinations on Yelp dataset.

Model	Context Combination	Yelp	
		RMSE	MAE
AIN	**Year**	1.107	0.869
	Month	1.107	0.870
	Year+Month	1.106	0.867
	Year+Month+Day of the week	1.103	0.860
	Year+Month+Day of the week+City	**1.097**	**0.852**

7.2 Impact of context combinations

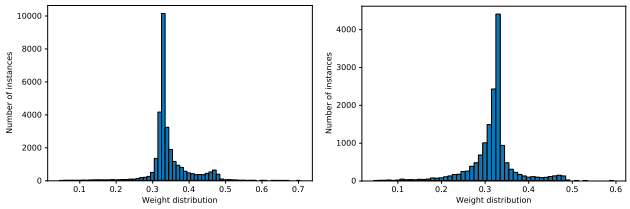
To evaluate the contribution of each contextual factor to the performance, we further conduct experiments based on different combination of these contextual factors on Food and Yelp datasets in Table 6 and Table 7. We only show the combinations containing both **Year** and **Month** on Yelp dataset. Similar observations can be obtained from other context combinations. The following observations can be made:

First, as can be seen from Table 6, the performance of considering **Hunger** is better than that of considering **Virtuality**. This indicates that when the user rates one kind of food, **Hunger** is more relevant to the rating results than **Virtuality**. Second, as shown in Table 7, considering only **Year** or **Month** achieve similar performance, and the performance can be improved by taking into account both of them. Besides, when we take more contextual factors into account, the performance can be further improved. This clearly indicates the necessity of taking more useful contexts into consideration. At last, when all contexts are taken into account, AIN achieves the best performance on both datasets. This observation further helps to reach the same conclusion that accurate prediction of user preferences for items undoubtedly depends upon the degree to which the recommender system has incorporated the relevant contextual information into the model [1].

7.3 Impact of attention mechanism

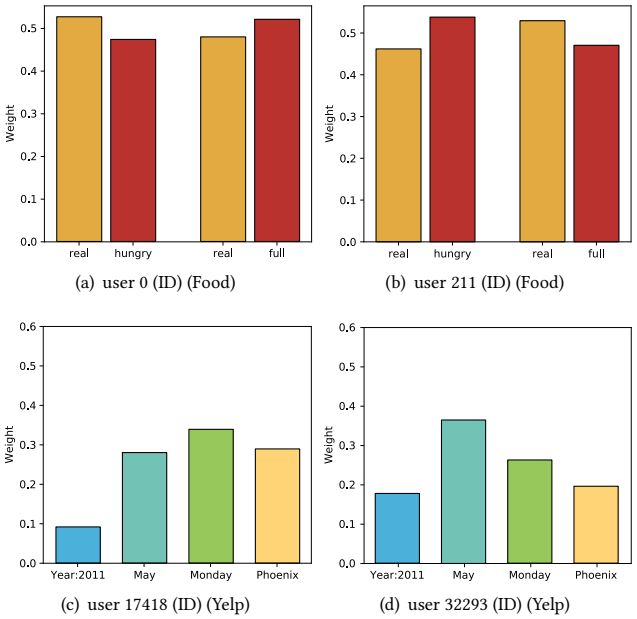
To demonstrate the utility of attention mechanism for AIN, we compare the performance of two versions of AIN – with and without attention mechanism. For AIN without attention mechanism, we use element-wise sum to aggregate the effects of interactions. As shown in Table 8, AIN with attention mechanism obtains considerable improvements over the one without attention. This result verifies our assumption that different contextual factors have different influence on users and items, and thus their respective impact should be properly incorporated in context-aware recommendation tasks.

To confirm the necessity of adaptively modeling context weights ($\beta(u, c_m)$ in Eq. 8), we show the distribution of learned weights of



(a) Monday (b) February

Figure 4: Weight distributions of Monday and February on Yelp dataset.



(a) user 0 (ID) (Food) (b) user 211 (ID) (Food)

(c) user 17418 (ID) (Yelp) (d) user 32293 (ID) (Yelp)

Figure 5: Visualization of attention weights learned by AIN.

context "Monday" and "February" on Yelp dataset in Figure 4(a) and 4(b) respectively. The X-axis indicates the weight of a given context with interval of 0.01. The Y-axis indicates the number of instances with weight falling in an interval. As can be seen from Figure 4(a), although for a large number of instances, the weight of "Monday" falls in the interval [0.33,0.34], there are also quite some number of instances for which the weight scatters into other intervals. Similar observation can be noted from Figure 4(b). This indicates that the weights of context effects vary with different users and different context combinations. What's more, it further highlights the necessity of capturing the adaptive effects of contexts.

Besides the improved performance, the proposed attention mechanism makes the context effects more explainable through interpreting the attention weights. To get a better understanding of this, we select some real cases from Food and Yelp datasets, and visualize the attention weights learned by AIN, as shown in Figure 5. Figure 5(a) shows the learned attention weights for user 0 under context combinations (real, hungry) and (real, full) respectively. 5(b) shows the learned attention weights for user 211 under context combinations (real, hungry) and (real, full) respectively. 5(c) and 5(d) show the learned weights for user 17418 and user 32293

Table 8: Performance of AIN with and without attention mechanism at effect level. EWS represents the element-wise sum and ATT represents the attention mechanism.

Model	Aggregation Type	Food		Yelp		Frappe				
		RMSE	MAE	RMSE	MAE	MAP@10	P@5	R@5	P@10	R@10
AIN	EWS	1.010	0.800	1.098	0.858	0.145	0.072	0.197	0.047	0.244
	ATT	**0.998**	**0.791**	**1.097**	**0.852**	**0.167**	**0.086**	**0.230**	**0.057**	**0.288**

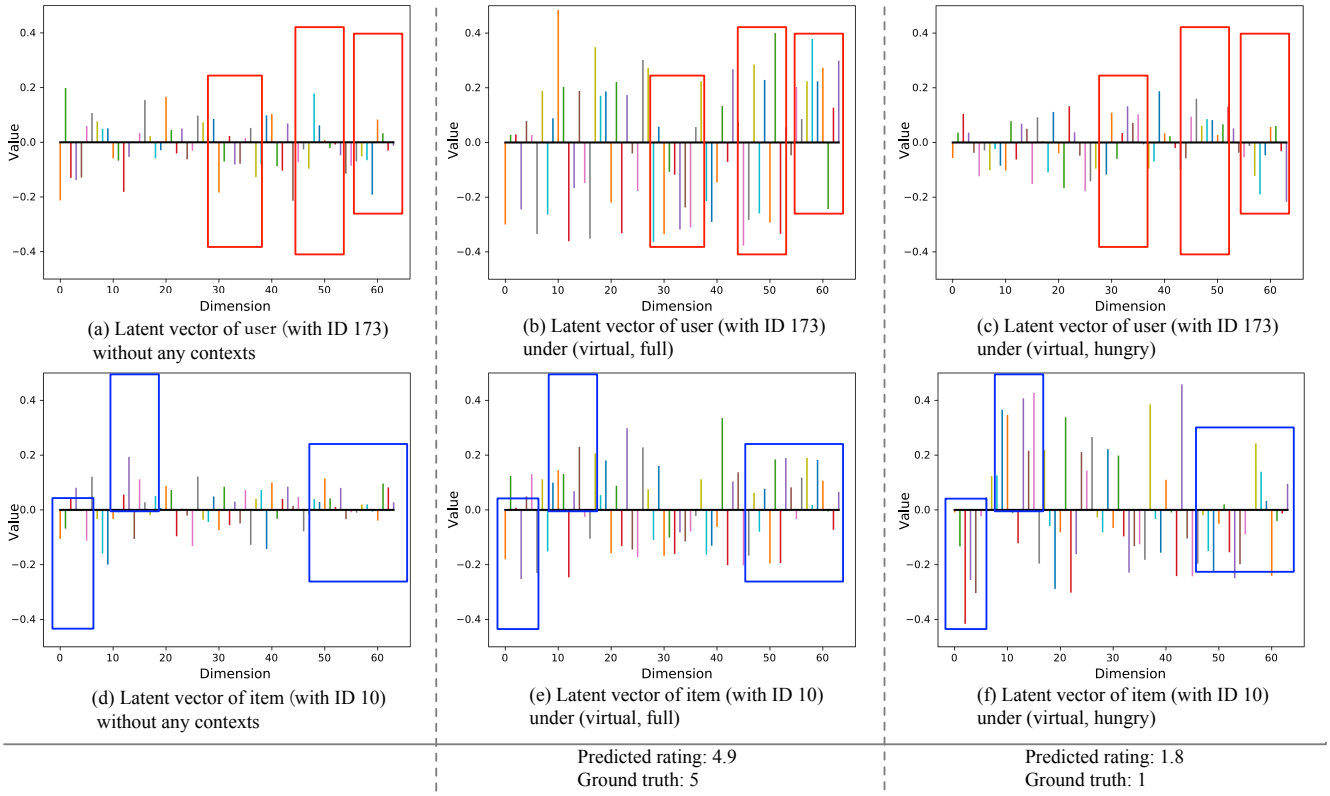


Figure 6: Examples of user and item latent vectors under different context combinations. The differences between latent vectors of the same user under different context combinations are marked in red rectangles, and the differences between latent vectors of the same item under different context combinations are marked in blue rectangles.

under (Year: 2011, May, Monday, Phoenix) respectively. We have the following observations.

First, as shown in Figure 5(a), when user 0 rates one kind of food under context combination (real, hungry), the effect of "real" is more important than that of "hungry". Under (real, full), the effect of "real" on user 0 becomes less important than the effect of "full" on the user. Similar observations can also be obtained from Figure 5(b) This reflects the benefit of attention mechanism in effectively discriminating the importance of different context effects. Besides, we can obtain another interestingfi nding from Figure 5(c) and 5(d). When user 17418 rates a business under context combination (Year: 2011, May, Monday, Phoenix), the effect of "Monday" has the highest attention weight, while the effect of "Year: 2011" has a much smaller weight. However, for user 32293, the attention weight of "May" is higher than that of the other three under the same context combination. Similar observations can be made by comparing the two leftmost (or rightmost) bars in Figure 5(a) and Figure 5(b). This demonstrates that *Effect-Level Attention* is able to attribute different weights to contexts depending on users.

7.4 Case study

To analyze the context effects on learned user and item representations, we select some real cases from Food dataset, shown in Figure 6. Figure 6(a) and 6(d) refer to the learned user and item representations without any contexts respectively. Figure 6(b) and 6(e) refer to the learned user and item representations under context combination (virtual, hungry). Figure 6(c) and 6(f) refer to the learned user and item representations under context combination (virtual, hungry). From Figure 6, we can make two observations. On the one hand, it is shown that the given user's rating over the same item varies greatly in different context combinations, as reflected by the rating scores under Figure 6(e) and 6(f). The rating given by the user under (virtual, full) is the full mark 5. However, the user gives the lowest rating of 1 under (virtual, hungry). This

confirms that contexts have greatly influence on user preferences. On the other hand, as can be seen from the red rectangles in Figure 6(a), 6(b) and 6(c), there is a large difference between the user representations under contexts and the user representation without any contexts. Besides, the user representation under (virtual, full) is also quite different from the user representation under (virtual, full). Similar observations can be made in the examples of the given item from Figure 6(d), 6(e), and 6(f) (especially in blue rectangles). These observations indicate that AIN can effectively capture the dynamic changes of user interests and item properties under different context combinations.

8 CONCLUSION AND FUTURE WORK

In this paper, we propose AIN, which provides a novel deep learning framework for modeling the effects of contextual information. AIN explicitly models the adaptive effects of interactions between contexts and users/items, and allows these effects to further influence the latent representations of users/items. Experimental results on three datasets show that AIN outperforms state-of-the-art context-aware methods on both rating prediction tasks and personalized ranking tasks. Besides, we show that AIN can improve the interpretability of recommendations through identifying the most important contexts.

We believe our work can be further extended in several directions. First, we can incorporate recurrent neural networks into AIN for context-aware sequential recommendations, e.g., session-based recommendations [13, 37]. Second, we can also improve AIN by jointly considering explicit contexts as explored in this paper and implicit contexts, such as the contextual information implied in user reviews for an item [21].

ACKNOWLEDGMENTS

The authors thank the anonymous reviewers for their helpful comments. This work is supported by the National Natural Science Foundation of China (Grant No. 61672322, 61672324).

REFERENCES

[1] G. Adomavicius and A. Tuzhilin. Context-aware recommender systems. In *Recommender Systems Handbook*, pages 217–253. Springer, 2011.
[2] G. Adomavicius, R. Sankaranarayanan, S. Sen, and A. Tuzhilin. Incorporating contextual information in recommender systems using a multidimensional approach. *ACM Transactions on Information Systems (TOIS)*, 23(1):103–145, 2005.
[3] L. Baltrunas and X. Amatriain. Towards time-dependant recommendation based on implicit feedback. In *RecSys Workshop*, 2009.
[4] L. Baltrunas and F. Ricci. Context-based splitting of item ratings in collaborative filtering. In *RecSys*, pages 245–248, 2009.
[5] L. Baltrunas and F. Ricci. Experimental evaluation of context-dependent collaborativefi ltering using item splitting. *User Modeling and User-Adapted Interaction*, 24(1-2):7–34, 2014.
[6] L. Baltrunas, B. Ludwig, and F. Ricci. Matrix factorization techniques for context aware recommendation. In *RecSys*, pages 301–304, 2011.
[7] P. W. Battaglia, R. Pascanu, M. Lai, D. J. Rezende, and K. Kavukcuoglu. Interaction networks for learning about objects, relations and physics. In *NIPS*, pages 4502–4510, 2016.
[8] J. Chen, H. Zhang, X. He, L. Nie, W. Liu, and T.-S. Chua. Attentive collaborative filtering: Multimedia recommendation with item- and component-level attention. In *SIGIR*, pages 335–344, 2017.
[9] T. Ebesu and Y. Fang. Neural citation network for context-aware citation recommendation. In *SIGIR*, pages 1093–1096, 2017.
[10] X. Glorot, A. Bordes, and Y. Bengio. Deep sparse rectifier neural networks. In *AISTATS*, pages 315–323, 2011.
[11] X. He and T.-S. Chua. Neural factorization machines for sparse predictive analytics. In *SIGIR*, pages 355–364, 2017.
[12] B. Hidasi and D. Tikk. Fast als-based tensor factorization for context-aware recommendation from implicit feedback. In *ECML PKDD*, pages 67–82, 2012.
[13] B. Hidasi, M. Quadrana, A. Karatzoglou, and D. Tikk. Parallel recurrent neural network architectures for feature-rich session-based recommendations. In *RecSys*, pages 241–248, 2016.
[14] W. Huang, Z. Wu, C. Liang, P. Mitra, and C. L. Giles. A neural probabilistic model for context based citation recommendation. In *AAAI*, pages 2404–2410, 2015.
[15] A. Karatzoglou, X. Amatriain, L. Baltrunas, and N. Oliver. Multiverse recommendation: N-dimensional tensor factorization for context-aware collaborative filtering. In *RecSys*, pages 79–86, 2010.
[16] D. Kim, C. Park, J. Oh, S. Lee, and H. Yu. Convolutional matrix factorization for document context-aware recommendation. In *RecSys*, pages 233–240, 2016.
[17] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICLR*, pages 1–15, 2015.
[18] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.
[19] J. Li, P. Feng, and J. Lv. Icamf: Improved context-aware matrix factorization for collaborativefi ltering. In *ICTAIN*, pages 63–70, 2013.
[20] J. Li, P. Ren, Z. Chen, Z. Ren, T. Lian, and J. Ma. Neural attentive session-based recommendation. In *CIKM*, pages 1419–1428, 2017.
[21] Y. Li, J. Nie, Y. Zhang, B. Wang, B. Yan, and F. Weng. Contextual recommendation based on text mining. In *COLING*, pages 692–700, 2010.
[22] Q. Liu, S. Wu, and L. Wang. Collaborative prediction for multi-entity interaction with hierarchical representation. In *CIKM*, pages 613–622, 2015.
[23] Q. Liu, S. Wu, and L. Wang. Cot: Contextual operating tensor for context-aware recommender systems. In *AAAI*, pages 203–209, 2015.
[24] Q. Liu, S. Wu, D. Wang, Z. Li, and L. Wang. Context-aware sequential recommendation. In *ICDM*, pages 1053–1058, 2016.
[25] T. V. Nguyen, A. Karatzoglou, and L. Baltrunas. Gaussian process factorization machines for context-aware recommendations. In *SIGIR*, pages 63–72, 2014.
[26] C. Ono, Y. Takishima, Y. Motomura, and H. Asoh. Context-aware preference model based on a study of difference between real and supposed situation data. In *UMAP*, pages 102–113, 2009.
[27] U. Panniello, A. Tuzhilin, M. Gorgoglione, C. Palmisano, and A. Pedone. Experimental comparison of pre- vs. post-filtering approaches in context-aware recommender systems. In *RecSys*, pages 265–268, 2009.
[28] Y. S. Rawat and M. S. Kankanhalli. Contagnet: Exploiting user context for image tag recommendation. In *MM*, pages 1102–1106, 2016.
[29] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *UAI*, pages 452–461, 2009.
[30] S. Rendle, Z. Gantner, C. Freudenthaler, and L. Schmidt-Thieme. Fast context-aware recommendations with factorization machines. In *SIGIR*, pages 635–644, 2011.
[31] Y. Shi, A. Karatzoglou, L. Baltrunas, M. Larson, A. Hanjalic, and N. Oliver. Tfmap: Optimizing map for top-n context-aware recommendation. In *SIGIR*, pages 155–164, 2012.
[32] Y. Shi, A. Karatzoglou, L. Baltrunas, M. Larson, and A. Hanjalic. Cars2: Learning context-aware representations for context-aware recommendations. In *CIKM*, pages 291–300, 2014.
[33] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.
[34] L. R. Tucker. Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31(3):279–311, 1966.
[35] M. Unger. Latent context-aware recommender systems. In *RecSys*, pages 383–386, 2015.
[36] M. Unger, A. Bar, B. Shapira, and L. Rokach. Towards latent context-aware recommendation systems. *Knowledge-Based Systems*, 104:165–178, 2016.
[37] C.-Y. Wu, A. Ahmed, A. Beutel, A. J. Smola, and H. Jing. Recurrent recommender networks. In *WSDM*, pages 495–503, 2017.
[38] S. Wu, Q. Liu, L. Wang, and T. Tan. Contextual operation for recommender systems. *IEEE Transactions on Knowledge and Data Engineering*, 28(8):2000–2012, 2016.
[39] J. Xiao, H. Ye, X. He, H. Zhang, F. Wu, and T. S. Chua. Attentional factorization machines: Learning the weight of feature interactions via attention networks. In *IJCAI*, pages 3119–3125, 2017.
[40] L. Xiong, X. Chen, T.-K. Huang, J. Schneider, and J. G. Carbonell. Temporal collaborativefi ltering with bayesian probabilistic tensor factorization. In *SDM*, pages 211–222, 2010.
[41] Y. Zheng, R. Burke, and B. Mobasher. Splitting approaches for context-aware recommendation: An empirical study. In *SAC*, pages 274–279, 2014.
[42] Y. Zheng, B. Mobasher, and R. Burke. Cslim: Contextual slim recommendation algorithms. In *RecSys*, pages 301–304, 2014.
[43] Y. Zheng, B. Mobasher, and R. Burke. Deviation-based contextual slim recommenders. In *CIKM*, pages 271–280, 2014.