CARS^2: Learning Context-aware Representations for Context-aware Recommendations

Yue Shi[a], Alexandros Karatzoglou[b], Linas Baltrunas[b], Martha Larson[c], Alan Hanjalic[c]
[a]Yahoo Labs, USA; [b]Telefonica Research, Spain; [c]Delft University of Technology, Netherlands
[a]yueshi@acm.org, [b]{alexk, linas}@tid.es,
[c]{m.a.larson, a.hanjalic}@tudelft.nl

ABSTRACT

Rich contextual information is typically available in many recommendation domains allowing recommender systems to model the subtle effects of context on preferences. Most contextual models assume that the context shares the same latent space with the users and items. In this work we propose CARS^2, a novel approach for learning context-aware representations for context-aware recommendations. We show that the context-aware representations can be learned using an appropriate model that aims to represent the type of interactions between context variables, users and items. We adapt the CARS^2 algorithms to explicit feedback data by using a quadratic loss function for rating prediction, and to implicit feedback data by using a pairwise and a listwise ranking loss functions for top-N recommendations. By using stochastic gradient descent for parameter estimation we ensure scalability. Experimental evaluation shows that our CARS^2 models achieve competitive recommendation performance, compared to several state-of-the-art approaches.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Information Filtering*

Keywords

Collaborative filtering; context-aware recommendation; latent factor models; learning representations

1. INTRODUCTION

Context-aware recommender systems (CARS) have been attracting considerable attention in recent years [1, 33]. Compared to conventional recommender systems [3, 13], which aims to provide users with personalized item/product recommendations, a context-aware recommender system also aims to make the recommendations contextualized. Many online applications/services have access to the contextual information of the users when they interact with the products, e.g., the time and place when a user listened to a song. This rich contextual information, beyond the user-item preference data (e.g., the ratings), has made modeling the subtle but powerful effects of context on user preferences possible.

Modeling the effects of context on preferences largely depends on having an accurate representation of both users, items and contextual variables. In this work we argue that it is moreover important to accurately representing the types of interactions between context, users and items. Recent contributions to CARS have build upon the foundation of latent factor models [19, 31, 32], which have been shown to be among the most effective group of approaches for conventional collaborative filtering recommender systems [24]. In a typical latent factor model, the relevance between a user and an item is modeled by the inner product of the latent factors of the user and the latent factors of the item. In analogy to this case, recent work has introduced latent factor models for contextual variables. The relevance between a user and a context is then modeled as the inner product between the user and context latent factors [31]. Similarly the relevance of a user, item, context combination is modeled by a three-mode inner product of the latent factors of the user, the item and the context [19, 32].

However, we argue that such models could be suboptimal; and specifically, two issues remain. First, the latent space of the context can hardly be consistent with that of the user and item. For example, if 10 latent factors are used to represent a user and an item, then the latent factors of the item can be interpreted as 10 latent properties possessed by the item and those of the user can be interpreted as the interest of the user to these 10 properties. However, such an interpretation may not be adequate for the latent factors of a context. Second, it is unclear how to interpret the relevance between a user and a context (or an item and a context). In contrast to the case of relevance between a user and an item, it might not be reasonable to assume that a user (or an item) is relevant to a context. For example, it is not intuitive that a user is relevant to "Friday" rather than "Monday". Similarly, it is also not intuitive if a movie is relevant to "Evening" rather than "Afternoon".

In order to specifically address these issues in CARS, we propose a novel formulation of latent factor models based on context-aware representations of users and items. In this model, the context is defined in its own latent space, which can be independent of that of the user and item. In addition, instead of modeling the relevance (which is unreasonable as explained before) between the user/item and context, our model includes an additional layer of latent space

CIKM'14, November 3–7, 2014, Shanghai, China.
Copyright is held by the owner/author(s). Publication rights licensed to ACM.
ACM 978-1-4503-2598-1/14/11 ...$15.00.
http://dx.doi.org/10.1145/2661829.2662070 .

for the user given a context, and an additional layer of latent space for the item given a context. The latent factors representing the user/item in the new latent space are introduced as the context-aware representation of the user/item. The context-aware representations can be interpreted much more clearly than the relevance between the user/item and the context. The context-aware user representation can be interpreted as the hidden properties (e.g., tense of work, time flexibility, happiness) that may influence the user under the context (e.g., Friday). Similarly, the context-aware item (e.g., movie) representation can be interpreted as the hidden properties (e.g., suitability for family, affectiveness, level of action) of the item under the context (e.g., Sunday afternoon with family).

Based on the proposed context-aware representations, we present three new latent factor models, i.e., one for rating prediction in the case of explicit feedback data, and two for top-N recommendation in the case of implicit feedback data (usage data, purchases). Through extensive experiments on two datasets with different types of user feedback data, we demonstrate the effectiveness and the properties of the new models for context-aware recommendation.

Our main contributions are summarized as below:

- We contribute a new concept of context-aware representations, which essentially captures the underlying properties/features of users and items conditioned on a given context.

- We contribute a new set of latent factor models based on the context-aware representations for context-aware recommender systems, which are shown to be competitive to existing state-of-the-art context-aware recommendation approaches.

The rest of the paper is organized as follows. in Section 2 we discuss the most relevant work and position our paper with respect to it. We formally define the research problem and terminology in Section 3. In Section 4, we present the detail of context-aware representations and specific models for rating prediction and top-N recommendation. Our experimental evaluation is reported in Section 5. Finally, Section 6 concludes our work in this paper and highlights a few future directions.

2. RELATED WORK

The work in this paper closely relates to three research areas: Collaborative filtering, Context-aware recommendation, and learning representations. We present the most relevant previous work in each of them.

2.1 Collaborative Filtering

Collaborative filtering (CF) has been the most popular technique for recommender systems [3, 13]. Conventionally, CF approaches can be classified as memory-based and model-based, while a particular group of methods, referred to as latent factor models, have become dominant in the field [24]. The performance of latent factor models for the rating prediction problem has been tested in recent competitions, such as the Netflix Prize Contest [23] and the KDD CUP 2011 [12]. Latent factor models have been further extended to incorporate content/metadata information [4, 5], so that the rich side information of users and items beyond the user-item relations can be exploited for improving recommendation. In addition, latent factor models have also been developed for the top-N recommendation problem in domains with implicit feedback data [18]. In these scenarios, the latent factor models are trained by optimizing a pairwise ranking criterion, such as Bayesian Personalized Ranking (BPR) [30], or a listwise ranking criterion, such as tensor factorization for mean average precision (TFMAP) [32]. The work in this paper also builds upon latent factor models. The proposed CARS^2-based models can be seen as a two-layer latent factor model, with one layer corresponding to the latent space of users and items (which is the same as in previous work), and the other corresponding to the latent space of users and items dependent on the contexts (the proposed context-aware representations). The CARS^2-based models can be specialized to extend the rating prediction latent factor model, BPR, TFMAP, respectively, by incorporating context-aware representations for context-aware recommendation.

2.2 Context-aware Recommender Systems

Researchers have devoted a lot of efforts to context-aware recommender systems (CARS). Early work in CARS exploited contextual information for pre-processing or post-processing, where context is used to guide the selection of the training data or the recommendation results [2, 6]. Previous work has also attempted to extend topic models and graph-based methods for CARS [14, 27]. As discussed in Section 1, a significant portion of recent work has focused on incorporating context variables into the latent factor models. Due to the success of matrix factorization for modeling the user-item relations, one major group of approaches exploited the tensor factorization techniques [21] for modeling the three-way user-item-context relations [15, 19, 32, 37]. A tensor in this case is a generalization of matrix from 2-dimension to n-dimension. Another signification contribution in this direction are the Factorization Machines [31, 29] (FM), which models the interactions between each pair of entities in terms of their latent factors, such as user-user, user-item, user-context interactions. Although the two types of approaches are both shown effective for context-aware recommendation, they also suffer from the limitations that first, the contexts are restricted to the latent space of users and items, and second, the hidden properties of users and items conditioned to the given context are overlooked. The work in this paper specifically addresses these limitations, resulting in improved recommendation performance for context-aware recommendation, as demonstrated in the experiments.

2.3 Learning Representations

The CARS^2 models in this work can be seen as part of the literature on Learning Representations [8]. While factors models can be seen as methods that learn the representations of users and items in a latent factor space, CARS^2 differs from the standard factor models in that it aims at also learning the type of relationships between users, items and context. The relations of the items and the users with the context variables are models through the parameters of two tensors which relate the representations of the users and items with the context variables. Related work in the area includes the Neural Tensor Networks [34] that models the relations between two entities, and tensor factorization with Bayesian clustering that learns relational structures [36].

Similar methods for learning representations from a knowledge base were introduced in [10] and [9].

3. PROBLEM AND TERMINOLOGY

The research problem we study in this paper can be defined as follows: *Given the users' feedback (either explicit or implicit) to the items and the contextual information related to the feedback data, provide item recommendations as relevant as possible to each user under a given context.*

In the following, we denote the feedback data from M users to N items under K types of context as a tensor $Y \in \mathcal{R}^{M \times N \times K}$. In the case of explicit feedback data, such as ratings, we use $y_{ijk} = r$ to denote a rating r assigned by user i to item j under context k, while in the case of implicit feedback data, such as clicks, we use $y_{ijk} = 1$ to denote user i has clicked item j (inferred as item j relevant to user i) under context k. Note that $y_{ijk} = 0$ indicates the preference of user i to item j under context type k is unknown. We denote the number of elements in a set S as $|S|$, and the Frobenuis norm of a matrix X as $\|X\|_F$.

Following the notation of latent factor models, we use u_i to denote a D-dimensional column vector, which represents the latent factors for user i. Similarly, v_i denotes the D-dimensional latent factors of item j. Since we allow a different latent space for contexts in our models, we denote the dimensionality of latent factors of context k as D_c, i.e., c_k is a D_c-dimensional column vector.

4. CARS^2

In this section, we present the main technical contributions of this paper: (1) the context-aware representations of users and items; and (2) based on that the latent factor models and the corresponding learning algorithms for the recommendation scenarios with explicit and implicit feedback data.

4.1 Context-aware Representations

An overview of our CARS^2-based models is presented in Fig. 1. As can be seen, there are two layers of latent factors that contribute to the relevance between a user (e.g., user i) and an item (e.g., item j) under a context (e.g., context k). In the first layer, the user and item share the same latent space, and the interaction of their latent factors (i.e., u_i and v_j) is used to model the relevance between the user and item independent of the context. In the second layer, another set of latent factors p_{ik}, i.e., the context-aware user representation, are learned for user i under context k, by projecting the interaction between user i and context k through a tensor W. Likewise, the context-aware item representation q_{jk} is learned by projecting the interaction between item j and context k through a tensor Z. The latent factors a_j for item j, defined corresponding to the latent space of p_{ik}, is introduced, so that the relevance of item j for user i under context k is modeled by the interaction of the latent factors of a_j and p_{ik}. Similarly, we model the interest of user i in item j when she is under context k by the interaction of the latent factors of the user, i.e., b_i, and the context-aware item representation, i.e., q_{jk}.

Specifically, we formulate the context-aware representation p_{ik} for user i under context k as below:

$$p_{ik} = u_i^T W c_k \tag{1}$$

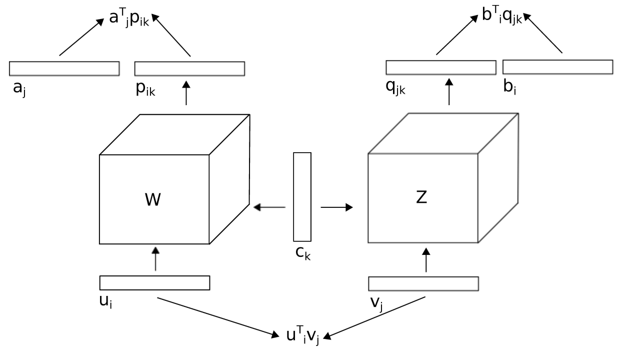


Figure 1: Illustration of the model. The cubes represent the tensors that represent the learned relationships, and the latent factor vectors are represented as elongated rectangles.

in which $W \in \mathcal{R}^{D \times D_p \times D_c}$ is a three-way matrix (or called a tensor) parameterized to project the interaction between user i and context k into D_p-dimensional latent space. In other words, p_{ik} is a D_p-dimensional vector that represents user i specifically under context k. Similarly, the representation of item j under context k is formulated as a D_q-dimensional vector q_{jk}:

$$q_{jk} = v_j^T Z c_k \tag{2}$$

The relevance between user i and item j under context k is then modeled as below:

$$\hat{y}_{ijk} = u_i^T v_j + a_j^T p_{ik} + b_i^T q_{jk} \tag{3}$$

in which a_j is a D_p-dimensional column vector of latent factors for item j, and b_i is a D_q-dimensional column vector of latent factors for user i. The latent factors for all the users in the D_q-dimensional latent space are denoted as matrix B (one user per column), and those for all the items in the D_p-dimensional latent space are denoted as matrix A (one item per column).

4.2 Recommendation with Explicit Feedback

In the scenarios with explicit feedback data, such as ratings, the latent factor model can be learned by minimizing the regularized squared error across all the training examples and the corresponding predictions. Following this convention, we formulate CARS^2-Rating, a model with context-aware representations as shown in Eq.(3) for rating prediction. Specifically, the loss function of CARS^2-Rating is expressed below:

$$L(\Theta) = \frac{1}{2} \sum_{(i,j,k) \in S} (y_{ijk} - \hat{y}_{ijk})^2 + \Omega(\Theta) \tag{4}$$

in which $\Theta = \{U, V, C, W, Z, A, B\}$ denotes the set of parameters that need to be estimated. S denotes the set of training data, i.e., $S = \{(i, j, k)|y_{ijk} > 0\}$. $\Omega(\Theta)$ is the regularization term that serves to control the magnitude of parameters, as shown below:

$$\begin{aligned}\Omega(\Theta) =& \frac{\lambda_1}{2}(\|U\|_F^2 + \|V\|_F^2 + \|C\|_F^2) + \frac{\lambda_2}{2}(\|W\|_F^2 + \|Z\|_F^2) \\ &+ \frac{\lambda_3}{2}(\|A\|_F^2 + \|B\|_F^2)\end{aligned} \tag{5}$$

in which λ_1, λ_2 and λ_3 are regularization parameters for the three different types of model parameters in CARS^2-Rating, respectively. Note that the model is equivalent to the standard matrix factorization when the context-related terms are neglected. We apply stochastic gradient descent (SGD) for learning the parameters in Θ. For notation convenience, we use: $\xi_{ijk} := \hat{y}_{ijk} - y_{ijk}$. The gradients of $L(\Theta)$ with respect to each variable can be computed as below:

$$\frac{\partial L}{\partial u_i} = \xi_{ijk}(v_j + Wc_k a_j) + \lambda_1 u_i \tag{6}$$

$$\frac{\partial L}{\partial v_j} = \xi_{ijk}(u_i + Zc_k b_i) + \lambda_1 v_j \tag{7}$$

$$\frac{\partial L}{\partial c_k} = \xi_{ijk}\left((u_i^T W)^T a_j + (v_j^T Z)^T b_i\right) + \lambda_1 c_k \tag{8}$$

$$\frac{\partial L}{\partial W_{:l:}} = \xi_{ijk} a_{jl} u_i c_k^T + \lambda_2 W_{:l:} \tag{9}$$

$$\frac{\partial L}{\partial Z_{:m:}} = \xi_{ijk} b_{im} v_j c_k^T + \lambda_2 Z_{:m:} \tag{10}$$

$$\frac{\partial L}{\partial a_j} = \xi_{ijk} p_{ik} + \lambda_3 a_j \tag{11}$$

$$\frac{\partial L}{\partial b_i} = \xi_{ijk} q_{jk} + \lambda_3 b_i \tag{12}$$

Note that $W_{:l:}$ $(l = 1, 2, \ldots, D_p)$ denotes the lth $D \times D_c$ matrix in the second dimension of W, and likewise, $Z_{:m:}$ $(m = 1, 2, \ldots, D_q)$ denotes the mth $D \times D_c$ matrix in the second dimension of Z.

Based on the learned parameters $\hat{\Theta}$, the rating predictions can then be computed by using Eq.(3).

4.3 Recommendation with Implicit Feedback

For the use scenarios with implicit feedback data, our proposed context-aware representations can be used with pairwise and listwise learning to rank approaches [20, 26], which are demonstrated to be effective for recommendation in such scenarios. In the following, we present two models, with respect to a state-of-the-art pairwise ranking approach and a state-of-the-art listwise ranking approach, respectively, by learning context-aware representations.

4.3.1 Pairwise Ranking

As mentioned in Section 2, BPR has been recognized as a state-of-the-art pairwise ranking method for collaborative filtering with implicit feedback data. We propose a new model, CARS^2-Pair, which is built on the same optimization criterion of BPR with context-aware representations for context-aware recommendation. We use Ψ to denote a set of examples of users' positive and (assumed) negative feedback under each context. Specifically, Ψ can be defined as below:

$$\Psi = \{(i, j, s, k) | y_{ijk} > 0 \quad and \quad y_{isk} = 0\} \tag{13}$$

According to the criterion of BPR, we formulate the loss function of our CARS^2-Pair model as below:

$$\begin{aligned} L(\Theta) &= -\sum_{(i,j,s,k)\in\Psi} \log g(\hat{y}_{ijk} - \hat{y}_{isk}) + \Omega(\Theta) \qquad (14) \\ &= -\sum_{(i,j,s,k)\in\Psi} \log g(\alpha_{ijsk}) + \Omega(\Theta) \end{aligned}$$

in which $g(x)$ is the logistic function, i.e., $g(x) = 1/(1 + e^{-x})$, and α_{ijsk} is used as a substitution as shown below for notation convenience:

$$\alpha_{ijsk} := u_i^T(v_j - v_s) + (a_j^T - a_s^T)p_{ik} + b_i^T(q_{jk} - q_{sk}) \tag{15}$$

Note that this loss function returns to the original BPR loss function if the terms involving contextual variable C are neglected.

As we use SGD for learning the model parameters, we compute the gradients of $L(\Theta)$ with respect to each variable as below. Note that we use a trick for simplifying the computation: $-(\log g(x))' = -g'(x)/g(x) = g(x) - 1$. A further substitution is made as: $\eta := g(\alpha_{ijsk}) - 1$.

$$\frac{\partial L}{\partial u_i} = \eta\left(v_j - v_s + Wc_k(a_j^T - a_s^T)\right) + \lambda_1 u_i \tag{16}$$

$$\frac{\partial L}{\partial v_j} = \eta(u_i + Zc_k b_i) + \lambda_1 v_j \tag{17}$$

$$\frac{\partial L}{\partial v_s} = \eta(-u_i - Zc_k b_i) + \lambda_1 v_s \tag{18}$$

$$\frac{\partial L}{\partial c_k} = \eta\left((u_i^T W)^T(a_j - a_s) + (v_j^T Z - v_s^T Z)^T b_i\right) + \lambda_1 c_k \tag{19}$$

$$\frac{\partial L}{\partial W_{:l:}} = \eta(a_{jl} - a_{sl}) u_i c_k^T + \lambda_2 W_{:l:} \tag{20}$$

$$\frac{\partial L}{\partial Z_{:m:}} = \eta b_{jm}(v_j c_k^T - v_s c_k^T) + \lambda_2 Z_{:m:} \tag{21}$$

$$\frac{\partial L}{\partial a_j} = \eta p_{ik} + \lambda_3 a_j \tag{22}$$

$$\frac{\partial L}{\partial a_s} = -\eta p_{ik} + \lambda_3 a_s \tag{23}$$

$$\frac{\partial L}{\partial b_i} = \eta(q_{jk} - q_{sk}) + \lambda_3 b_i \tag{24}$$

Note that as in the original BPR model, we randomly select one negative/irrelevant item for each relevant item in the training data. This choice is made so as to have a balanced set for training the model, and also keep the complexity of the learning algorithm linear rather than quadratic, as shown in Section 4.4.

4.3.2 Listwise Ranking

In [32], we introduced a listwise ranking approach, TFMAP, that optimizes the smoothed version of mean average precision (MAP) with tensor factorization. We use the listwise ranking optimization from TFMAP, resulting in a new context-aware recommendation model, referred to as CARS^2-List. Specifically, according to [32], given user i and context k, the smoothed average precision (AP) of a recommendation list is defined as below:

$$AP_{ik} = \frac{1}{n_{ik}^+} \sum_{j=1}^{N} y_{ijk} g(\hat{y}_{ijk}) \sum_{s=1}^{N} y_{isk} g(\hat{y}_{isk} - \hat{y}_{ijk}) \tag{25}$$

in which n_{ik}^+ is the number of relevant items for user i under context k. For later convenience, we neglect the constant coefficient and rewrite the objective function taking into account all the users and contexts (note that we minimize the negative of MAP as above (which is equivalent to maximize MAP), and include the regularization terms), as shown below:

$$L(\Theta) = -\sum_{(i,j,s,k)\in\Phi} g(\hat{y}_{ijk})g(\hat{y}_{isk} - \hat{y}_{ijk}) + \Omega(\Theta) \quad (26)$$
$$= -\sum_{(i,j,s,k)\in\Phi} g(\hat{y}_{ijk})g(\alpha_{isjk}) + \Omega(\Theta)$$

in which α_{isjk} is defined in the same way as shown in Eq.(15), and Φ denotes a set of training examples of users' positive feedback over two items under each context. Specifically, Φ can be defined as below:

$$\Phi = \{(i,j,s,k) | y_{ijk} > 0 \quad and \quad y_{isk} > 0\} \quad (27)$$

We introduce another two substitutions for notation convenience in the gradients as below:

$$\beta_1 := -g'(\hat{y}_{ijk})g(\alpha_{isjk}), \quad \beta_2 := -g(\hat{y}_{ijk})g'(\alpha_{isjk}) \quad (28)$$

The gradients by applying SGD can be then computed as below:

$$\frac{\partial L}{\partial u_i} = \beta_1(v_j + Wc_k a_i) + \beta_2(v_s - v_j + Wc_k(a_s - a_j)) + \lambda_1 u_i \quad (29)$$

$$\frac{\partial L}{\partial v_j} = (\beta_1 - \beta_2)(u_i + Zc_k b_i) + \lambda_1 v_j \quad (30)$$

$$\frac{\partial L}{\partial v_s} = \beta_2(u_i + Zc_k b_i) + \lambda_1 v_s \quad (31)$$

$$\frac{\partial L}{\partial c_k} = (\beta_1 - \beta_2)\left((u_i^T W)^T a_j + (v_j^T Z)^T b_i\right) + \beta_2\left((u_i^T W)^T a_s + (v_s^T Z)^T b_i\right) + \lambda_1 c_k \quad (32)$$

$$\frac{\partial L}{\partial W_{:l:}} = ((\beta_1 - \beta_2)a_{jl} + \beta_2 a_{sl})\, u_i c_k^T + \lambda_2 W_{:l:} \quad (33)$$

$$\frac{\partial L}{\partial Z_{:m:}} = (\beta_1 - \beta_2)b_{im} v_j c_k^T + \beta_2 b_{jm} v_s c_k^T + \lambda_2 Z_{:m:} \quad (34)$$

$$\frac{\partial L}{\partial a_j} = (\beta_1 - \beta_2)p_{ik} + \lambda_3 a_j \quad (35)$$

$$\frac{\partial L}{\partial a_s} = \beta_2 p_{ik} + \lambda_3 a_s \quad (36)$$

$$\frac{\partial L}{\partial b_i} = (\beta_1 - \beta_2)q_{jk} + \beta_2 q_{sk} + \lambda_3 b_i \quad (37)$$

Note that given a relevant item for a user under a context, we randomly select one additional relevant item from the training data to form the set Φ. This choice is made with the same purpose as in the case of CARS^2-Pair (and also in BPR), which is to keep the complexity of the learning algorithm linear rather than quadratic.

Table 1: Complexity of updating parameters in the learning algorithms for CARS^2-based models

Parameter	Complexity
U	$O(D \times D_p \times D_c \times \lvert S\rvert)$
V	$O(D \times D_q \times D_c \times \lvert S\rvert)$
C	$O(D \times (D_p + D_q) \times D_c \times \lvert S\rvert)$
W	$O(D \times D_p \times D_c \times \lvert S\rvert)$
Z	$O(D \times D_q \times D_c \times \lvert S\rvert)$
A	$O(D \times D_p \times D_c \times \lvert S\rvert)$
B	$O(D \times D_q \times D_c \times \lvert S\rvert)$

4.4 Complexity Analysis

Although CARS^2-Rating, CARS^2-Pair and CARS^2-List are developed with different objectives and for two different use cases, their learning algorithms have the same complexity regarding the update of each type of parameters. They also share the same overall complexity, which is $O(D\times(D_p + D_q) \times D_c \times |S|)$, linear in the number of training examples and in the size of each latent dimension. The complexity of updating each type of parameters is summarized in Table 1. Note that in the case that low dimensionality is used for the latent spaces of users, items, contexts and context-aware representations (which is typically the case for recommender systems), we have $D \times (D_p + D_q) \times D_c << |S|$. As a result, the overall complexity of CARS^2-based models can be seen as linear in the size of the training data, which provides a reasonable scalability for real applications.

4.5 Characteristics

We summarize the characteristics of CARS^2-based models into the following three aspects: First, on the basis of context-aware representations, the CARS^2-based models are able to learn the user interests and the item properties given a specific context, which provide additional support for context-aware recommendation beyond the global interest of the user and the general properties of the item. Second, the CARS^2-based models allow the latent space of contexts to be independent of that of the users and items. This property makes our models better reflect the realistic situations, since the nature of the interaction between a user (or an item) and a context is essentially different from that between a user and an item. Third, our models can be adapted to the use cases with either explicit or implicit feedback data, and the model parameters can be learned in linear time. In the next section, we will experimentally demonstrate the impact of these characteristics.

5. EXPERIMENTAL EVALUATION

In this section we present a collection of experiments that evaluate the proposed CARS^2-based latent factor models for context-aware recommendation. We first describe the datasets and setup that are used in the experiments. Then, we investigate the impact of the latent dimensionality of contexts and the impact of the latent dimensionality of context-aware representations. Finally, we evaluate the recommendation performance of CARS^2-based models in both explicit and implicit feedback scenarios, compared to several baseline approaches.

The experiments were designed to address the following research questions:

1. Are the property of the CARS^2-based models, that enable the latent space of the contexts to be independent

of that of the users and items, useful for improving the context-aware recommendation performance?

2. Do the context-aware representations used in the CARS^2-based models make a significant contribution to the improving context-aware recommendation performance?

3. Are the CARS^2-based models competitive for context-aware recommendation, compared to other state-of-the-art and baseline approaches in both explicit and implicit feedback domains?

4. Can the CARS^2-based models scale to large datasets?

5.1 Datasets

The experiments are conducted mainly using two datasets, one with explicit feedback data and the other with implicit feedback data. The first one is the *Food* dataset [28], which has also been used in the previous work on context-aware recommendation [19, 31]. This dataset contains ca. 6.3K 5-scale ratings from 212 users on 20 menus/items, and each rating is associated with 2 contextual factors, *i.e.* one factor about whether the user's feeling about hunger is real or virtual (2 values: 1=real, 2=virtual) when she rated a menu, and the other factor about the user's hunger degree (3 values: 1=normal, 2=hungry and 3=full). By taking into account all the combinations of the two contextual factors, in total we have 6 different contexts, i.e., $K = 6$ in the Food dataset.

The second dataset is the *Frappe* dataset, which is collected from Frappe[1], a context-aware personalized recommender of mobile apps that we developed in 2013. It runs on Android phones and recommends the most relevant apps for the user based on his/her current situation (context) and usage patterns. Frappe uses a standard client-server architecture. The client side consists of an Android mobile app that displays recommendations and a background service, which is used to collect app usage data and other contextual information. This background service in the client samples the current state of the phone once per minute and gathers (1) the last known readings of various sensors and (2) information on which applications are currently in use. The dataset used in our experiments contains ca. 95K interactions (implicitly as positive feedback) from 953 users and 4073 items. Three contextual factors are involved in this dataset, i.e., day time (with 7 possible values, e.g., morning, night), day type (with 2 possible values, weekday and weekend), and location (with 3 possible values, work, home and other place). Thus, in total we have 42 ($K = 42$) contexts in the Frappe dataset.

Note that conventional CF benchmark datasets, *e.g.*, the Netflix dataset and the KDD CUP 2011 dataset, are not enriched with contextual information. The *Frappe* dataset is not as large as these benchmark datasets. However, we emphasize that this dataset is much larger than the datasets used in the previous work of context-aware recommendation [19, 31]. We will make the Frappe dataset publicly available for research purposes.

In addition to the two datasets, we use another dataset collected from Tuenti[2], a social network in Spain, for validating the scalability of CARS^2-based models. We choose this dataset since its size is much larger than the two datasets used in our main experiments. This dataset contains about 3.4M interactions (as implicit feedback from user-video clicks) from ca. 99.9K users and 99.6K videos. Although the Tuenti dataset is not enriched with contextual features as in the other two datasets, we choose to use the video category of the previously watched video as the context for the current user-video interaction. This choice is made to allow us to run CARS^2-based models, rather than to test the context-aware recommendation performance. In total, we have 4 video categories, i.e. $K = 4$, in the Tuenti dataset. The experiment regarding the scalability using this dataset is presented in Section 5.6.

[1]http://frappe.cc/

[2]https://www.tuenti.com/

5.2 Experimental Setup

For each of the two datasets, we randomly separate it into three parts: a training set, a validation set, and a test set. The training set consists of 75% data, while the test set contains 20% data. We use the remaining 5% data as the validation set for tuning parameters, such as the latent dimensionality of contexts and the latent dimensionality of context-aware representations. The training set is used to learn the recommendation model, *i.e.*, Ω in each CARS^2-based model, which is then used to predict ratings (in the Food dataset) or generate recommendation lists (in the Frappe dataset) for each user under each context in the test set.

For the Food dataset, we measure the recommendation performance by Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE), which are conventional metrics for quantifying the rating prediction error. Specifically, the definitions of RMSE and MAE are given below:

$$RMSE = \sqrt{\frac{1}{|S_{test}|} \sum_{(i,j,k) \in S_{test}} (y_{ijk} - \hat{y}_{ijk})^2} \tag{38}$$

$$MAE = \frac{1}{|S_{test}|} \sum_{(i,j,k) \in S_{test}} \|y_{ijk} - \hat{y}_{ijk}\| \tag{39}$$

in which S_{test} denotes the test set, and y and $\hat{y}$ are the ground-truth ratings and the predictions, respectively. The lower RMSE and MAE, the better recommendation performance.

For the Frappe dataset, we first generate a recommendation list for each user under each context, and then use the Mean Average Precision (MAP) metric according to the ground truth in the test set. We choose MAP, since it is a standard metric for evaluating the quality of ranked lists in information retrieval, and its top-bias property is particularly important for top-N recommendation. The definition of MAP is shown below:

$$MAP = \frac{1}{MK} \sum_{i=1}^{M} \sum_{k=1}^{K} \frac{\sum_{j=1}^{N} \frac{y_{ijk}}{R_{ijk}} \sum_{s=1}^{N} y_{isk} \mathbb{I}(R_{isk} \leq R_{ijk})}{\sum_{j=1}^{N} y_{ijk}} \tag{40}$$

in which R_{ijk} is the rank position of item j in the recommendation list for user i under context k, and y_{ijk} is the binary relevance value between user i and item j under context k in the test set. $\mathbb{I}(\cdot)$ is an indicator function, which is equal to 1 if the condition is true, and otherwise 0. We also measure the performance of recommendation lists by precision and recall at top positions, such as precision at 10 (P@10) and recall at 10 (R@10). P@N measures the ratio of relevant item recommendations in the top-N recommendation list, while R@N measures the ratio of relevant items covered in the top-N list against the total number of relevant items. The higher values of MAP, P@N and R@N, the better recommendation performance. Note that we cannot treat all the items that have no feedback in the test set as irrelevant/negative ones, in which case the recommendation performance could be severely underestimated. For this reason, we adopt a conventionally widely-used evaluation strategy [11, 22], in which we randomly select 1000 items that have no feedback as irrelevant ones for each user under each context in the test set. The performance is then measured based on the recommendation list that contains only these 1000 items together with relevant items in the test set.

Finally, note that we empirically tune the conventional parameters according to the performance in the validation set. In both dataset the regularization parameters are set as $\lambda_1 = \lambda_2 = \lambda_3 = 0.001$, which are sufficient for alleviating overfitting. We also fixed the latent dimensionality of users and items as D=10, in order to observe the impact of the latent spaces of the contexts and the context-aware representations. The learning rate is set to 0.005 for the Food dataset and 0.01 for the Frappe dataset. In the following, we focus our investigation and discussion on the parameters of the latent dimensionality of contexts and the latent dimensionality of context-aware representations, which are essentially the main contribution of our work.

5.3 Impact of Latent Space of Contexts

As discussed before, one of the characteristics of CARS^2-based models is that they allow the flexibility of the latent space of contexts. Given the fixed dimensionality of latent factors of users and items, i.e., $D = 10$, we investigate the impact of the latent space of contexts by varying its dimensionality D_c for CARS^2-Rating, CARS^2-Pair, and CARS^2-List, respectively. Note that initially we set $D_c = D_p = D_q = 6$ and gradually tune each of them as presented in this and next subsections. As can seen in Fig. 2, the best D_c in terms of RMSE achieved for CARS^2-Rating on the validation set of the Food dataset is 4, which is much smaller than the dimensionality of the latent space for the users and items, i.e., $D = 10$. This observation indicates that the latent space of contexts has an impact on the recommendation performance, and the proposed CARS^2-Rating model allows the flexibility in the latent dimensionality of contexts so as to attain better recommendation performance. Similar observations can also be found in Fig. 2 the CARS^2-Pair and CARS^2-List models, which achieved the best performance at $D_c = 4$ and $D_c = 12$, respectively, in terms of MAP in the validation set of the Frappe dataset. The results of this experiment demonstrate the contribution of our CARS^2-based models for their ability to represent contexts independent of users and items, based on which we can give a positive answer to our first research question.

5.4 Impact of Context-aware Representations

With the best values of D_c in each of the CARS^2-based models as obtained from previous experiment, we further investigate the impact of context-aware representations by varying their dimensionality, i.e., D_p and D_q. We first still keep the $D_q = 6$ (the dimensionality of context-aware item representations) fixed, while investigating the impact of the dimensionality of context-aware user representations D_p. The results are shown in Fig. 3, from which we can see that the best value of D_p for CARS^2-Rating on the Food dataset is 4, 2 for CARS^2-Pair on the Frappe dataset, and 6 for CARS^2-List on the Frappe dataset. The results indicate that the context-aware user representations exploited in our models have a significant impact on the recommendation performance, and the best performance can be attained by tuning the dimensionality of their latent space. Similarly, with the best values of D_c and D_p in each case, we show in Fig. 4 the impact of the dimensionality of context-aware item representations D_q on the recommendation performance. As observed, the best performance is achieved when D_q is 8 for CARS^2-Rating on the Food dataset, 4 for both CARS^2-Pair and CARS^2-List on the Frappe dataset. It is evident that the context-aware item representations in our models also have a significant impact on the recommendation performance. Summarizing, our CARS^2-based models benefit from exploiting the context-aware representations for improving context-aware recommendation performance, which allows us to have an affirmative answer to our second research question.

5.5 Performance Comparison

We compare the performance of CARS^2-based models with that of baseline approaches on the two datasets, respectively. The results are measured on the test set of each dataset, using the best parameter settings as observed from the validation set. Specifically, on the Food dataset, we set $D_c = 4, D_p = 4, D_q = 8$ for CARS^2-Rating, and compare its performance with the baseline approaches as listed below:

- **MF** [24]. The standard matrix factorization approach, which learns the latent factors of users and items based on their ratings. The regularization parameter is tuned to 0.001, according to the performance in the validation set.

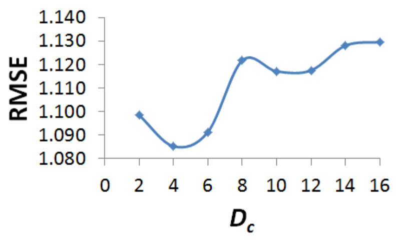
- **FM** [31]. A state-of-the-art context-aware approach for the domains with explicit feedback data using Factorization Machines [29], which models the pair-wise interactions between each pair of users, items and contexts in terms of their latent factors. The implementation of FM is done with the publicly available software libFM[3], and the learning algorithm is chosen to be SGD, which is the same as we use in CARS^2-Rating.

On the Frappe dataset, we compare the performance of CARS^2-Pair (with the best parameter setting $D_c = 4, D_p = 2, D_q = 4$) and CARS^2-List ((with the best parameter setting $D_c = 12, D_p = 6, D_q = 4$)) with that of the following baseline approaches:

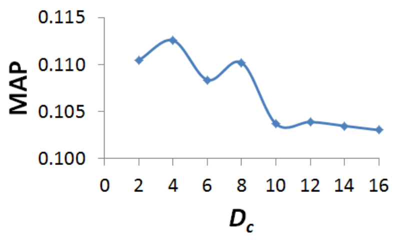
- **Pop**. A naive approach that generates non-personalized recommendation based on the item popularity, i.e., the number of users interacted with the item.

- **BPR** [30]. Bayesian personalized ranking (BPR) represents a state-of-the-art ranking criterion for implicit feedback data in recommender systems. As presented in Section 4.3.1, our CARS^2-Pair model builds on BPR with context-aware representations. Note that both Pop and BPR are context-free approaches.

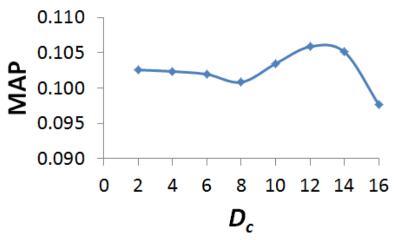
[3]http://www.libfm.org/



(a) CARS^2-Rating

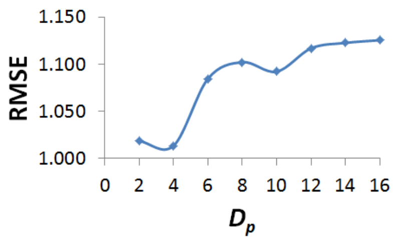


(b) CARS^2-Pair

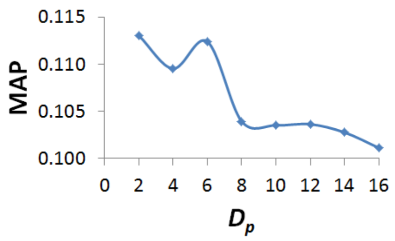


(c) CARS^2-List

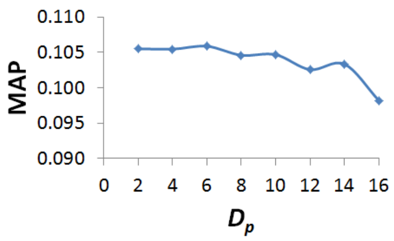
Figure 2: The impact of D_c on the recommendation performance: (a) RMSE of CARS^2-Rating (with $D = 10, D_p = D_q = 6$) in the Food dataset, (b) MAP of CARS^2-Pair (with $D = 10, D_p = D_q = 6$) in the Frappe dataset and (c) MAP of CARS^2-List (with $D = 10, D_p = D_q = 6$) in the Frappe dataset.



(a) CARS^2-Rating

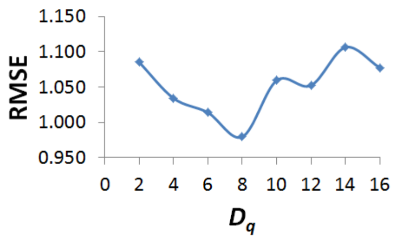


(b) CARS^2-Pair

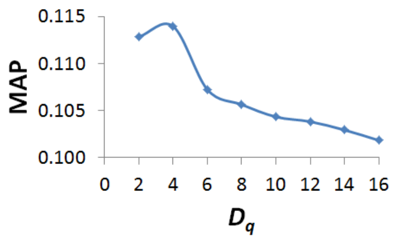


(c) CARS^2-List

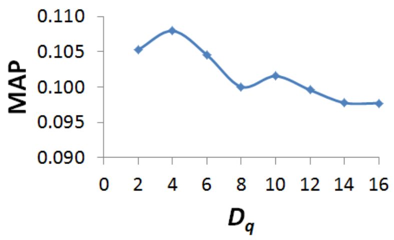
Figure 3: The impact of D_p on the recommendation performance: (a) RMSE of CARS^2-Rating (with $D = 10, D_c = 4, D_q = 6$) in the Food dataset, (b) MAP of CARS^2-Pair (with $D = 10, D_c = 4, D_q = 6$) in the Frappe dataset and (c) MAP of CARS^2-List (with $D = 10, D_c = 12, D_q = 6$) in the Frappe dataset.



(a) CARS^2-Rating



(b) CARS^2-Pair



(c) CARS^2-List

Figure 4: The impact of D_q on the recommendation performance: (a) RMSE of CARS^2-Rating (with $D = 10, D_c = 4, D_p = 4$) in the Food dataset, (b) MAP of CARS^2-Pair (with $D = 10, D_c = 4, D_p = 2$) in the Frappe dataset and (c) MAP of CARS^2-List (with $D = 10, D_c = 12, D_p = 6$) in the Frappe dataset.

Table 2: **Performance comparison of CARS^2-Rating and the baselines on Food dataset**

	RMSE	MAE
MF	1.061	0.850
FM	0.988	0.777
CARS^2-Rating	**0.970***	**0.770***

Table 3: **Performance comparison of CARS^2-Pair, CARS^2-List and the baselines on Frappe dataset**

	MAP	P@5	R@5	P@10	R@10
Pop	0.076	0.043	0.121	0.028	0.153
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***
CARS^2-List	0.129	0.062	0.165	0.040	0.202

- **TFMAP** [32]. Our previous work on context-aware recommendation for implicit feedback domains. This approach optimizes a smoothed version of MAP through tensor factorization, while it requires that the contexts share the same latent space as the users and items. Our CARS^2-List model builds on the same optimization criterion with two layer latent spaces involving context-aware representations.

Note that for all the baselines involving latent factors, we set the dimensionality of the latent factors for users and items to be 10, which is the same as we use in the CARS^2-based models. This setting allows us to make a fair comparative evaluation. Other parameters are tuned according to the performance observed on the validation set.

The results of performance comparison on each of the two datasets are shown in Table 2 and 3, from which we have three main observations. First, the CARS^2-based models outperform context-free baselines in both cases. CARS^2-Rating improves over MF by 8.5% in RMSE and 9.4% in MAE on the Food dataset. On the Frappe dataset, CARS^2-Pair and CARS^2-List achieved a large improvement over Pop, and a significant improvement over BPR, e.g., CARS^2-Pair improving over BPR by 15% in MAP. Note that * in two tables indicates the significant improvement over all the baselines, according to Wilcoxon signed rank significance test with $p<0.05$. This observation indicates that CARS^2-based models are effective for exploiting contextual information for improving recommendation performance. Second, the CARS^2-based models also outperform context-aware baseline approaches in both cases. CARS^2-Rating achieves RMSE 1.8% lower than FM and MAE 1% lower than FM on the Food dataset. On the Frappe dataset, CARS^2-List performs slightly better than TFMAP, while CARS^2-Pair performs substantially better than TFMAP, e.g., 10% improvement in MAP, 12.8% improvement in P@10 and 15.9% in R@10. This observation indicates that our CARS^2-based models succeed in exploiting the context-aware representations and improving the state-of-the-art approaches on context-aware recommendation. Finally, we notice that CARS^2-Pair performs significantly better than CARS^2-List on the Frappe dataset. For example, MAP of CARS^2-Pair is 8.5% better than CARS^2-List. This observation may be interpreted as that the negative examples, which are exploited by CARS^2-Pair but not CARS^2-List, result in the contribution to the additional gain achieved in CARS^2-Pair. Note that even without using negative examples, CARS^2-List still outperforms TFMAP based on its exploitation of context-aware representations. Summarizing, the experimental results of performance comparison lead to a positive answer to our third research question.

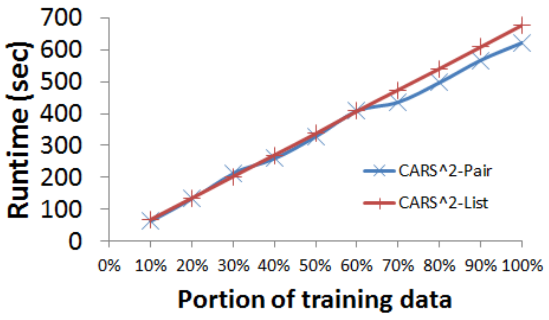


Figure 5: Scalability analysis of CARS^2-based models on the Tuenti dataset.

5.6 Scalability

In the last experiment, we investigate the scalability of CARS^2-based models. As mentioned in Section 5.1, we use the Tuenti dataset, which is much larger than the Food and Frappe datasets, in this investigation. Since it is an implicit feedback dataset, CARS^2-Pair and CARS^2-List are applicable. Note that we set $D_c = D_p = D_q = 4$ in both models without tuning them for the performance, since the purpose of this experiment is only to validate the scalability of the models. We vary from 10% to 100% of all the data for training the models and measure the corresponding runtime per iteration. The results are shown in Fig. 5. As can be seen, for both CARS^2-Pair and CARS^2-List, the runtime increases linearly as the size of training data increases. This result empirically confirms the scalability of CARS^2-based models as analyzed in Section 4.4, allowing us to give a positive answer to our last research question.

6. CONCLUSIONS AND FUTURE WORK

We have presented CARS^2-based models that build on two layer latent factor spaces and learn context-aware representations for context-aware recommender systems. Our models can be adapted to the use cases with either explicit or implicit feedback data, and learned in linear time with respect to the size of training data. Our experiments on an explicit feedback dataset and an implicit feedback dataset demonstrate the effectiveness of context-aware representations and the improved performance for context-aware recommendation compared to several baseline approaches.

Our future work may further investigate the following directions. First, we would like to exploit deep learning techniques [7, 16, 17, 35] for building multi-layer latent factor models for context-aware recommender systems and recommender systems in general. Second, it may also be interesting to investigate the possibilities of embedding non-linear interactions [25] in the CARS^2-based models. Third, we will look into the possibilities of more efficiently determining the dimensionality of the latent space of context-aware representations in our models.

7. ACKNOWLEDGEMENTS

This work was done when Yue Shi was at Delft University of Technology, Netherlands. The work leading to these results has received funding from the European Union's Seventh Frame work Programme (FP7/2007-2013) under grant agreement no. 610594 (CrowdRec).

8. REFERENCES

[1] G. Adomavicius, B. Mobasher, F. Ricci, and A. Tuzhilin. Context-aware recommender systems. *AI Magazine*, 32(3):67–80, 2011.

[2] G. Adomavicius, R. Sankaranarayanan, S. Sen, and A. Tuzhilin. Incorporating contextual information in recommender systems using a multidimensional approach. *ACM Trans. Inf. Syst.*, 23:103–145, January 2005.

[3] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Kowledge and Data Engineering*, 17(6):734–749, 2005.

[4] D. Agarwal and B.-C. Chen. Regression-based latent factor models. KDD '09, pages 19–28, New York, NY, USA, 2009. ACM.

[5] D. Agarwal and B.-C. Chen. fLDA: Matrix factorization through latent dirichlet allocation. WSDM '10, pages 91–100, New York, NY, USA, 2010. ACM.

[6] L. Baltrunas and F. Ricci. Context-based splitting of item ratings in collaborative filtering. RecSys '09, pages 245–248, New York, NY, USA, 2009. ACM.

[7] Y. Bengio. Learning deep architectures for ai. *Foundations and trends® in Machine Learning*, 2(1):1–127, 2009.

[8] Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828, 2013.

[9] A. Bordes, X. Glorot, J. Weston, and Y. Bengio. Joint learning of words and meaning representations for open-text semantic parsing. In *International Conference on Artificial Intelligence and Statistics, AISTATS*, pages 127–135, 2012.

[10] A. Bordes, J. Weston, R. Collobert, Y. Bengio, et al. Learning structured embeddings of knowledge bases. In *AAAI*, 2011.

[11] P. Cremonesi, Y. Koren, and R. Turrin. Performance of recommender algorithms on top-n recommendation tasks. RecSys '10, pages 39–46, New York, NY, USA, 2010. ACM.

[12] G. Dror, N. Koenigstein, Y. Koren, and M. Weimer. The Yahoo! Music Dataset and KDD-Cup'11. *Journal of Machine Learning Research-Proceedings Track*, 18:8–18, 2012.

[13] M. D. Ekstrand, J. T. Riedl, and J. A. Konstan. Collaborative filtering recommender systems. *Foundations and Trends in Human-Computer Interaction*, 4(2):81–173, 2011.

[14] N. Hariri, B. Mobasher, and R. Burke. Context-aware music recommendation based on latent topic sequential patterns. In *Proceedings of the Sixth ACM Conference on Recommender Systems*, RecSys '12, pages 131–138, New York, NY, USA, 2012. ACM.

[15] B. Hidasi and D. Tikk. Fast als-based tensor factorization for context-aware recommendation from implicit feedback. In *Machine Learning and Knowledge Discovery in Databases*, pages 67–82. Springer, 2012.

[16] G. E. Hinton. Learning multiple layers of representation. *Trends in cognitive sciences*, 11(10):428–434, 2007.

[17] G. E. Hinton, S. Osindero, and Y.-W. Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

[18] Y. Hu, Y. Koren, and C. Volinsky. Collaborative filtering for implicit feedback datasets. In *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, pages 263–272, Washington, DC, USA, 2008. IEEE Computer Society.

[19] A. Karatzoglou, X. Amatriain, L. Baltrunas, and N. Oliver. Multiverse recommendation: N-dimensional tensor factorization for context-aware collaborative filtering. RecSys '10, pages 79–86, New York, NY, USA, 2010. ACM.

[20] A. Karatzoglou, L. Baltrunas, and Y. Shi. Learning to rank for recommender systems. In *Proceedings of the 7th ACM Conference on Recommender Systems*, RecSys '13, pages 493–494, New York, NY, USA, 2013. ACM.

[21] T. G. Kolda and B. W. Bader. Tensor decompositions and applications. *SIAM Rev.*, 51(3):455–500, Aug. 2009.

[22] Y. Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. KDD '08, pages 426–434, New York, NY, USA, 2008. ACM.

[23] Y. Koren. Collaborative filtering with temporal dynamics. *Commun. ACM*, 53:89–97, April 2010.

[24] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42:30–37, August 2009.

[25] N. D. Lawrence and R. Urtasun. Non-linear matrix factorization with gaussian processes. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 601–608. ACM, 2009.

[26] T.-Y. Liu. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, 3(3):225–331, 2009.

[27] N. Natarajan, D. Shin, and I. S. Dhillon. Which app will you use next?: collaborative filtering with interactional context. In *Proceedings of the 7th ACM conference on Recommender systems*, pages 201–208. ACM, 2013.

[28] 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. UMAP '09, pages 102–113, Berlin, Heidelberg, 2009. Springer-Verlag.

[29] S. Rendle. Factorization machines with libFM. *ACM Trans. Intell. Syst. Technol.*, 3(3):57:1–57:22, May 2012.

[30] S. Rendle, C. Freudenthaler, Z. Gantner, and S.-T. Lars. BPR: Bayesian personalized ranking from implicit feedback. UAI '09, pages 452–461, Arlington, Virginia, United States, 2009. AUAI Press.

[31] S. Rendle, Z. Gantner, C. Freudenthaler, and L. Schmidt-Thieme. Fast context-aware recommendations with factorization machines. SIGIR '11, pages 635–644, New York, NY, USA, 2011. ACM.

[32] Y. Shi, A. Karatzoglou, L. Baltrunas, M. Larson, A. Hanjalic, and N. Oliver. TFMAP: Optimizing MAP for top-n context-aware recommendation. SIGIR '12, pages 155–164, New York, NY, USA, 2012. ACM.

[33] Y. Shi, M. Larson, and A. Hanjalic. Collaborative filtering beyond the user-item matrix: A survey of the state of the art and future challenges. *ACM Comput. Surv.*, 47(1):3:1–3:45, May 2014.

[34] R. Socher, D. Chen, C. D. Manning, and A. Ng. Reasoning with neural tensor networks for knowledge base completion. In *Advances in Neural Information Processing Systems*, pages 926–934, 2013.

[35] N. Srivastava and R. Salakhutdinov. Multimodal learning with deep Boltzmann machines. In *Advances in Neural Information Processing Systems 25*, pages 2231–2239, 2012.

[36] I. Sutskever, R. Salakhutdinov, and J. B. Tenenbaum. Modelling relational data using bayesian clustered tensor factorization. In *NIPS*, pages 1821–1828, 2009.

[37] L. Xiong, X. Chen, T.-K. Huang, J. Schneider, and J. G. Carbonell. Temporal collaborative filtering with Bayesian probabilistic tensor factorization. In *SDM'10*, pages 211–222, 2010.