Context-Aware Recommendations Based on Deep Learning Frameworks

MOSHE UNGER and ALEXANDER TUZHILIN, New York University
AMIT LIVNE, Ben-Gurion University of the Negev

In this article, we suggest a novel deep learning recommendation framework that incorporates contextual information into neural collaborative filtering recommendation approaches. Since context is often represented by dynamic and high-dimensional feature space in multiple applications and services, we suggest to model contextual information in various ways for multiple purposes, such as rating prediction, generating top-k recommendations, and classification of users' feedback. Specifically, based on the suggested framework, we propose three deep context-aware recommendation models based on explicit, unstructured, and structured latent representations of contextual data derived from various contextual dimensions (e.g., time, location, user activity). Offline evaluation on three context-aware datasets confirms that our proposed deep context-aware models surpass state-of-the-art context-aware methods. We also show that utilizing structured latent contexts in the proposed deep recommendation framework achieves significantly better performance than the other context-aware models on all datasets.

CCS Concepts: • **Computing methodologies** → **Learning latent representations**; **Neural networks**; • **Information systems** → **Recommender systems**;

Additional Key Words and Phrases: Context, deep learning, latent, context-aware recommendation, neural networks

ACM Reference format:
Moshe Unger, Alexander Tuzhilin, and Amit Livne. 2020. Context-Aware Recommendations Based on Deep Learning Frameworks. *ACM Trans. Manage. Inf. Syst.* 11, 2, Article 8 (May 2020), 15 pages.
https://doi.org/10.1145/3386243

1 INTRODUCTION

Over the past decade, there has been extensive and growing interest in the recommender systems (RSes) community in studying how to add contextual information, such as the activity of the user, time, location, and weather to RSes [Adomavicius and Tuzhilin 2005]. Traditional context-aware recommender systems (CARSs) [Adomavicius and Tuzhilin 2015] mostly use predefined explicit contextual information for the recommendation process. The specific contexts describe the circumstances of the information collection, such as weather conditions (sunny, cloudy, raining, etc.) or time conditions (weekday, weekend, etc.). The main advantage of these explicit contexts is their

Authors' addresses: M. Unger and A. Tuzhilin, Department of Technology, Operations and Statistics, Stern School of Business, NYU, 44 West Fourth Street, New York, NY 10012; emails: {munger, atuzhili}@stern.nyu.edu; A. Livne, Ben-Gurion University of the Negev, Israel; email: livneam@post.bgu.ac.il.

2158-656X/2020/05-ART8 $15.00
https://doi.org/10.1145/3386243
Corrected Version of Record v.1.1. Published June 26, 2020

lower dimensionality encompassing only few contextual variables for a specific application domain. However, explicit CARSs have the following limitations: (1) the selection of specific contexts for the service is a resource-demanding task since it is performed manually by domain experts; (2) predefined contexts may not represent the most effective and all-encompassing set of contextual features for the recommendation application; and (3) using explicit contexts, such as location of the customer, may raise privacy issues [Lane et al. 2010] since the exact context is known to the service.

These limitations with explicit contexts may cause serious problems in several practical applications, such as smart health and well-being [Chen et al. 2012], mobile sensing, and Internet-of-Things (IoT) [Lane et al. 2015], where the contextual feature space is complex and dynamic. For example, by exploiting numerous sensors derived from smartphones, such as accelerometer, magnetic field, GPS, and light, high-dimensional contextual information (up to 600 contextual variables) can be collected automatically to infer users' behaviors and contexts [Lane et al. 2010]; the accelerometer sensor can be used to infer the activity of the user (e.g., walking or sitting), whereas other features from the GPS sensor can be used to infer her location (e.g., at work, at home). Identifying all the explicit contexts that are relevant for the service can be very challenging, especially in big data environments [Chen et al. 2012], and therefore automatic latent modeling of contextual information can be useful to capture the richness of the contextual information and can *implicitly* identify various types of contexts.

To resolve these limitations and model contextual information efficiently, recent studies [Mei et al. 2018; Smirnova and Vasile 2017; Unger et al. 2016; Wu et al. 2017] suggest reducing dimensionality of the contextual space by using latent contextual embeddings modeled as an unstructured numeric vector and propose integrating this *latent* information into the recommendation process. An alternative way to model latent contextual information is by extracting structured latent contexts from a hierarchical model, composed of groups of similar latent context vectors, called *contextual situations* [Unger and Tuzhilin 2019]. Modeling latent contextual embeddings in a structured manner can also take into consideration the structure of latent contextual variables and semantically meaningful interrelationships among them.

Most research efforts in the area of CARSs have been devoted to the enhancement of matrix factorization (MF) [Baltrunas et al. 2011; Unger et al. 2016], which is a common approach for latent factor model-based recommendation [Koren 2008]. Despite the effectiveness of MF for collaborative filtering (CF), the main limitations of MF are its use of a fixed linear interaction function to estimate the complex structure of user-item interactions and its use of squared regression loss for optimization [Rendle et al. 2009].

In this article, we propose a novel deep learning (DL) recommendation framework that takes into consideration contextual information in multiple forms, as illustrated in Figure 1 and further explained in Section 3. In the proposed framework, we suggest enhancing two neural-CF models with several context representations. Specifically, we extend the neural-CF (NCF) and neural-MF (NeuMF) models [He et al. 2017], which suggest utilizing deep neural networks (DNNs) to learn a nonlinear function of user-item interactions using DNNs as opposed to the fixed linear inner product used in MF. In our approach, we describe how to extend the neural recommendation models with three types of context representations: explicit, unstructured, and structured latent information, to capture complex interactions between users, items, and contexts. Based on the proposed context-aware recommendation framework, we present three diverse tasks: (1) rating prediction, (2) top-N recommendation, and (3) classification of user feedback ("like," "dislike," and "check-in".

The main contributions of this article are as follows. First, we propose a framework of how to incorporate contextual information into DL-based CF approaches. Second, we propose three specific deep context-aware recommendation models that fit in the proposed framework: (1) a deep explicit context-aware model (ECAM) that incorporates non-hierarchical explicit contexts

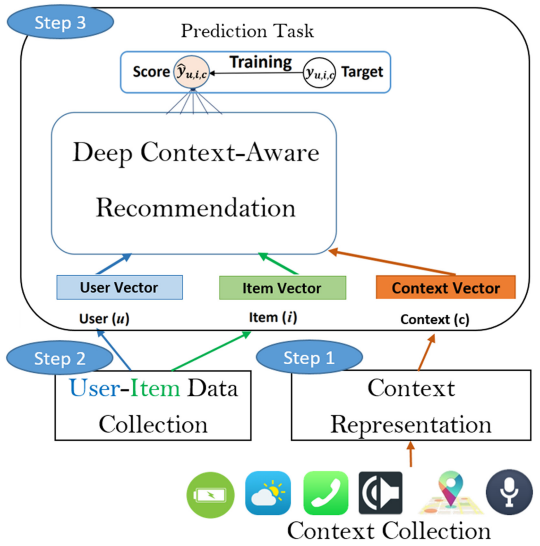

Fig. 1. Deep context-aware recommendation framework.

in a neural model, (2) an unstructured (latent) context-aware model (UCAM) that incorporates unstructured latent context embeddings derived by an auto-encoder (AE) network, and (3) a hierarchical (structured) context-aware model (HCAM) that uses hierarchical representations of latent contextual embeddings. Third, we compare the three proposed approaches with two DL-based CF approaches (NCF and NeuMF) and show that the structured context-aware approach is significantly better than the others. Finally, we show that the DL-based MF method dominated the non-DL MF method when context is added. Section 2 describes related work, and Section 3 describes the deep context-aware recommendation framework. Section 4 presents the datasets, the evaluation protocols and the baselines, and in Section 5 we discuss the results. Finally, in Section 6 we conclude and outline our plans for future work.

2 RELATED WORK

CARSs deal with modeling and predicting users' tastes and preferences by incorporating available contextual information into the recommendation process [Adomavicius et al. 2011; Adomavicius and Tuzhilin 2015]. The representation of contextual information can be one of the following forms: explicit or implicit (latent). Explicit context is specified by a well-defined group of known labeled contextual dimensions (e.g., time, weather, location). In contrast, latent context is modeled as a numeric vector and represents the hidden contextual information of the user. In existing CARSs, which utilize explicit contextual information in the recommendation process [Baltrunas and Ricci 2009, 2014; Li et al. 2013; Mei et al. 2018; Yin et al. 2017; Zheng et al. 2014], a reduced set of contexts is usually selected manually by domain experts to handle the expansion of the model's dimensionality (i.e., users, items, and contexts). Latent CARSs [Unger et al. 2016; Wu et al. 2017] suggest to automatically model contextual information for the recommendation in an unsupervised and compressed manner, called *latent context*. As shown in the work of Shi et al. [2014], the latent space of contexts should be independent of users and items and can be constructed for the prediction of multiple tasks, such as rating prediction and top-N recommendation.

There are multiple ways for modeling and constructing latent contextual information. Rawat and Kankanhalli [2016] suggested to learn contextual features from images that are processed by a

feed-forward neural network for tagging images, and Kim et al. [2016] extracted latent information from documents using a convolutional neural network. Unger et al. [2016] suggested to construct latent contextual embeddings by an AE [Hayat et al. 2014] (Figure 2(a)) or by principal component analysis (PCA) [Wold et al. 1987] for the rating prediction and top-N recommendation purposes. Latent contextual embeddings can be also modeled in a hierarchical manner to capture structures and recurrent patterns within the latent contextual vector space [Unger and Tuzhilin 2019], as shown in Figure 2(b). Wu et al. [2017] proposed a bias tensor factorization model that uses encoded contextual features that are extracted from a regression tree. Although in the work of Wu et al. [2017] the contextual features are learned in a hierarchical manner, it has the following limitations. First, since tensor factorization can handle a small number of contextual features, they represented contextual features by considering only one cluster, although more complex context patterns can be modeled. Second, clustering was applied on all available contextual variables with low-dimensional datasets (4–12 variables). However, in various applications with rich and high-dimensional context data [Chen et al. 2012], clustering becomes inefficient and challenging [Kriegel et al. 2009], and therefore the context dimension should be reduced due to computational complexity.

Recently, DL has revolutionized recommendation architectures [Batmaz et al. 2019; Zhang et al. 2019]. In particular, Jhamb et al. [2018] aimed to learn contextual latent features that reflect a user's preferences over all candidate items and considers explicit contextual features. Twardowski [2016] incorporated contextual information of user activity within a single session, defined as a sequence of events. Smirnova and Vasile [2017] proposed a context-aware session-based recommendation utilizing conditional recurrent neural networks (RNNs), which inject contextual information into input and output layers and modify the behavior of the RNN by combining context embedding with item embedding. Although this work strengthens the hypothesis that contextual information improves recommendation accuracy, they only integrated three types of explicit contextual features: time, time difference since the last event, and event type.

In the IS community, several context-aware recommendation methods have been studied [Panniello et al. 2016; Sahoo et al. 2010]. Sahoo et al. [2010] proposed a hidden Markov model for representing changes in user preferences, and Panniello et al. [2016] measured the effects of contextual information on purchasing behavior and trust of customers and proposed a context-aware recommendation with two contextual attributes: the intent of a purchase and the customer's mood. In both works, most of the changes in user preferences were modeled by a small set of predefined explicit contextual factors.

In our work, we define context as any user situation (user activity, mobile state, environmental information, etc.) that might affect user preferences. We use many contextual dimensions (e.g., weather, sound, light, location) to extract hidden contextual patterns, called *latent contextual embeddings*, that are not predefined with a limited list of activities. We assume that there are many different patterns of latent contexts, and therefore we suggest modeling latent contexts in a structured manner.

In our research, we extend the neural-CF models presented by He et al. [2017], proposing a deep context-aware recommendation framework that uses contextual information in an explicit, latent, or hierarchical latent manner. We show that by automatically recognizing the the user's contextual situations, we can improve recommendation performance. Specifically, by adding hierarchical latent contexts in the deep recommendation model, we can significantly improve recommendation accuracy. The hierarchical representation is obtained from unstructured latent contexts using a hierarchical tree and is capable of modeling the recurrent patterns of the latent context space, and more accurately captures complex situations of the user. A detailed description of our extended framework is provided in Section 3.

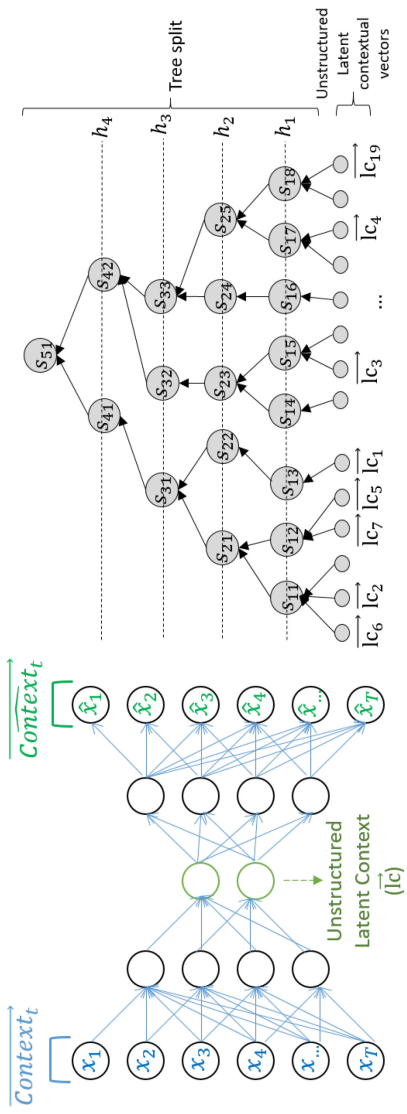

Fig. 2. Latent context representations. (a) Unstructured latent context [Unger et al. 2016]. (b) Hierarchical latent context [Unger and Tuzhilin 2019].

3 METHOD

We aim to design a deep context-aware recommendation framework for multiple recommendation purposes that learns the relations between users, items, and contexts. Specifically, we present three novel deep CF extensions for the CARS domain that are used for multiple prediction tasks: rating prediction, generating top-N recommendation, and classification of users' feedback. The extension models include the following: the first is a simple deep context-aware model that adds a numeric vector of all available explicit contexts to the model. The second adds compressed latent contextual embeddings extracted from an AE, and the third model utilizes hierarchical contextual information in a structured and latent manner. We further describe the overall framework that is composed of three main steps as illustrated in Figure 1. Step 1 is responsible for extracting explicit, unstructured, or hierarchical latent contextual representation (denoted by "Context (c)" in Figure 1) from the collected contextual feature space. Step 2 is responsible for capturing user-item interactions by representing "User (u)" and "Item (i)" one-hot vectors by their index ID, as shown in Figure 1. Step 3 constitutes a general CF recommendation model that is used for a particular prediction task—that is, prediction of the rating score or classification of users' feedback, considering the user, item, and context representations. A description of each of the steps is provided next.

3.1 Context Representation

The goal of this step is to extract a context representation for the recommendation process. We represent context in three different ways: explicit, latent unstructured, and latent structured. For each of the context representations, we first normalize the contextual feature values to a scale of 0 to 1 and transform nominal features to binary features, which results in a contextual vector $\overrightarrow{context} = [x_1, x_2, ..., x_T]$. Extracting explicit contextual representation includes using all of the available (raw) T contextual features, whereas the unstructured latent contextual information is a compact representation containing L latent values ($L < T$) extracted from the hidden layer of an AE [Unger et al. 2016], as shown in Figure 2(a). This latent contextual information is defined in its own latent space [Shi et al. 2014] and is used for different tasks, such as rating prediction, top-N recommendation, and classification. The auto-encoding process reconstructs the original input values by passing them through a low-dimensional "bottleneck" layer, with the aim of obtaining a compact feature representation [Wang et al. 2016]. To automatically learn the structure of latent contextual variables and the semantically meaningful interrelationships among them, we also suggest to utilize a structured latent contextual representation for the recommendation process [Unger and Tuzhilin 2019], which includes a set of contextual situations (clusters IDs) at different granularity levels extracted from a hierarchical tree, as shown in Figure 2(b).

The process of constructing a hierarchical model is done automatically by grouping the set of unstructured latent contextual vectors into a finite set of clusters, with each cluster representing an implicit contextual situation s_i. Specifically, to extract structured latent contextual vectors, we (1) extract compressed latent contextual vectors from the bottleneck layer of an AE, and (2) build a hierarchical tree from the unstructured latent contextual vectors (see $\overrightarrow{lc_i}$ in Figure 2(b)) in a bottom-up approach, to reveal structured contextual situations at different granularity levels. We apply agglomerative hierarchical clustering (AHC) [Davidson and Ravi 2005] to automatically estimate the number of possible contextual situations (clusters) and apply the k-means algorithm to group similar latent contextual vectors to a particular contextual situation. A hierarchical latent contextual vector is defined as the path (i.e., a set of contextual situations) of the unstructured latent contextual vector from its leaf to the top of the tree. For example, the extracted hierarchical latent contextual vector of $\overrightarrow{lc_{19}}$ is $[s_{18}, s_{25}, s_{33}, s_{42}, s_{51}]$.

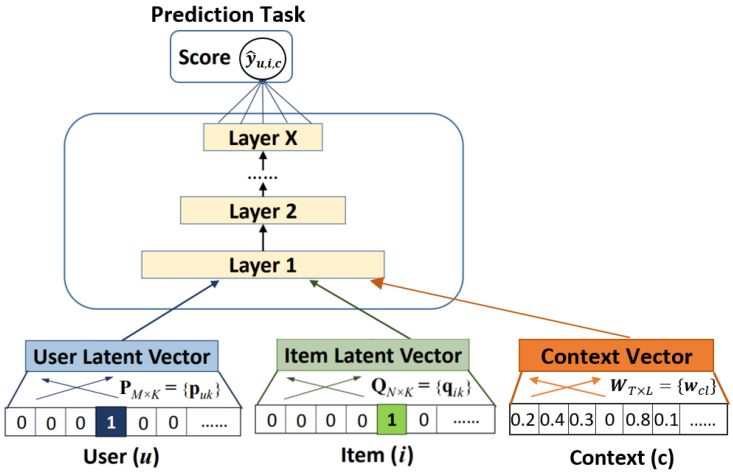

Fig. 3. Neural-CF (NCF) extension with contextual information.

The three contextual representations (explicit, unstructured, and hierarchical latent contexts) can be interpreted as follows. With regard to explicit properties, for example, for the Frappe dataset (as described in Section 4.1), the explicit contextual representation will contain all available contextual features (22 in total) describing the time of the day, day of the week, location, and weather. With regard to latent (hidden) unstructured properties, for the Frappe dataset, the unstructured latent context vector will contain 11 numerical values representing nonlinear correlations between the original contextual features in a compressed and low-dimensional manner (e.g., Sunday afternoon at location X, Friday morning at location Y). With regard to latent hierarchical properties, for the Frappe dataset, the structured representation of latent contexts will contain eight contextual situations derived from the hierarchical tree, representing different patterns of the latent unstructured properties. Some of the contextual situations in the low level of the tree might represent situations such as "at home" or "at work," as those situations are patterns and correlations of the original contextual features (time of the day, day of the week, and location). Some contextual situations in the middle level of the tree can represent higher-level situations, such as time spent at work and time flexibility, and the highest level of the tree might represent users at their home city on weekdays and users outside their home city on weekends.

3.2 User-Item Data Collection

The second step, as shown in Figure 1, is a standard process in RSes, in which users' feedbacks or ratings are collected for each of the items the system wishes to recommend. We represent users and items in a standard way as one-hot vectors by their index ID (denoted by "User (u)" and "Item (i)" in Figure 1).

3.3 Deep Context-Aware Recommendation

The third step refers to the training of an offline context-aware recommendation model with the information collected about users, items, and contexts. The recommendation algorithm can be any CF recommendation model that takes into account user, item, and context representations for the prediction task. Specifically, we extend each of the neural-CF models (NCF and NeuMF, as shown in Figure 3 and Figure 4, respectively) with each of the extracted contextual representations from step 1 (as described in Section 3.1).

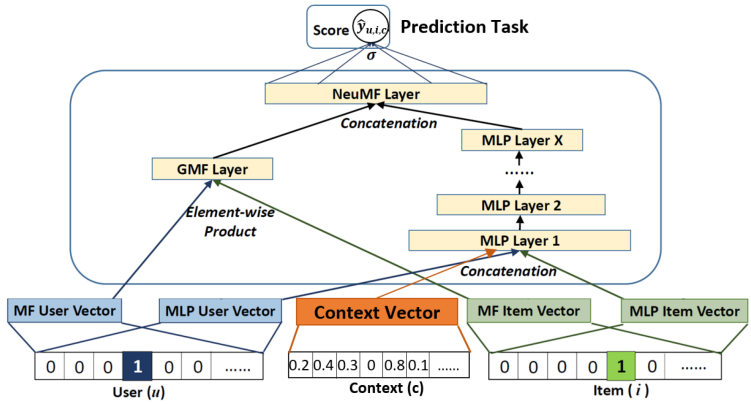

Fig. 4. Neural-MF (NeuMF) extension with contextual information.

In our extension of the neural-CF recommendation models, we propose adding a new component of contextual information: a contextual vector (denoted by "Context (c)" in Figure 3 and Figure 4) that is modeled as a set of explicit, unstructured, or structured latent contextual features. The first extension is ECAM, which incorporates all available contextual features in each of the neural-CF models. The second model is UCAM, which incorporates unstructured latent contexts, and the third is HCAM, which incorporates hierarchical latent contexts in each of the neural-CF models. In both extensions of the neural-CF recommendation models, we concatenate the contextual vector c to the user u and item i embeddings to learn a new nonlinear function between all three components (users, items, and contexts). In this way, the dimension of context is considered within the neural framework to automatically learn its influence on the overall predicted target. In the case of rating prediction task (as shown as an example in Figure 1), the target is a predicted rating score $\hat{y}_{u,i,c}$. In case of the classification task, we apply the softmax activation function that provides the probability of each class label (categorical value), and then we predict the class with the highest probability.

The NeuMF model that is shown in Figure 4 is a generalization of the NCF model (as presented in Figure 3) with new learned interactions between user, item, and context features. It consists of two major components: generalized matrix factorization (GMF) and multilayer perceptron (MLP). The GMF (denoted by "GMF Layer" in Figure 4) uses a linear function that is learned by embeddings of one-hot vectors of users and items. The second component is the MLP, which can endow the model with a large amount of flexibility and nonlinearity to learn the interactions between user, item, and contextual vector representations (explicit, structured, and unstructured latent contexts).

3.4 Method Complexity

Our method is an extension of the deep collaborative models (NeuMF and NCF) that are trained with stochastic gradient descent, where in each step a single (user, item) entry from the users' feedback (e.g., rating matrix) is randomly sampled to perform a gradient update. The integration of various contextual representations to the framework is done by representing explicit contextual vector with all available contextual factors (no training) or by extraction of latent context. The process of extracting latent contextual vectors is done in two steps. The first step is training an AE with all contextual vectors for compressing the original contextual features to an unstructured latent contextual space. The training of an AE is an iterative process that is performed with stochastic gradient descent through back propagation. The second step is extracting hierarchical latent

Table 1. Description of Context-Aware Datasets

Dataset	Users	Items	Contextual Features	Interactions	Scale
CARS	98	1,918	247	38,900	1 (dislike), 3 (like), 5 (check-in)
Frappe	957	4,082	22	96,203	0–4.4.6
Yelp	451,341	80,796	9	3,383,536	1–5

contexts from the unstructured latent contextual vectors. This process includes the construction of hierarchical tree by applying AHC with Ward's variance method [Murtagh and Legendre 2014], which is $O(n^2)$ computational complexity for the clustering of n observations.

In this section, we presented a deep context-aware recommendation framework that integrates various contextual representations into the recommendation model. We proposed three different variations of contextual representations aiming to capture multiplicity of user contextual situations and discussed the method complexity. In the next section, we evaluate the proposed models with three context-aware datasets and report our results.

4 EVALUATION

4.1 Datasets

We conducted our experiments on three context-aware datasets with different contextual dimensions. For each of the datasets, we normalized the contextual features to a scale of 0 to 1 and transformed nominal features to binary features. The overall number of contextual features (after normalization) and the characteristics of the three datasets are summarized in Table 1.

CARS is a dataset that contains three types of user feedback and their corresponding ratings (i.e., dislike, 1; like, 3; and check-in, 5) regarding points of interest, such as restaurants and pubs [Unger et al. 2017]. Overall, there are 38,900 explicit ratings (11,484 likes, 27,058 dislikes, and 358 check-ins). For each user interaction, the dataset contains 247 contextual features (e.g., location, noise level, part of the week, time of day) derived by 15 types of mobile sensors, such as GPS, battery, and light. From mobile sensors that produce raw data in the form of a 3D axis, such as accelerometer, magnetic field, orientation, and gravity, we extracted statistical features such as the average, median, standard deviation, correlation, minimum and maximum values, range, first and third quarters, root mean square, and entropy for each axis. The dataset is used for multiple tasks: (1) rating prediction (based on a rating estimation of 1, 3, or 5), (2) top-N recommendations, and (3) classification of users' feedback (like, dislike, and check-in).

Frappe is an implicit feedback dataset that is collected from a context-aware personalized recommender of mobile apps [Baltrunas et al. 2015]. The app monitoring mobile application usage was installed by 957 users, who used 4,082 different apps in total. This data captures the usage frequencies of an app by each user within 2 months. To estimate the usage frequencies of each user, we employ a log transformation on the raw frequency numbers represents the application usage rating scale of 0 to 4.46. We used 22 contextual features derived by four contextual dimensions, including time of the day, day of the week, location, and weather. Overall, the data contained 96,203 ratings. The dataset does not include the specific timestamp of each user interaction with the system. The dataset is used for the application usage prediction task.

Yelp is a publicly available dataset that contains 451,341 users, 80,796 items, and 3,383,536 interactions [Yelp 2018]. We use the following contextual features: year, month, day of the week, week number, longitude, and latitude. We extracted further contextual features from the date and location features: season, isHoliday, and isWeekend. Overall, we used nine contextual features associated with this dataset. The dataset is used for the rating prediction task.

4.2 Setup

We divided the data into 10 subsets and randomly split each subset into three portions: 80% for training, 10% for validation, and 10% for testing. The validation set is used for tuning hyperparameters for the recommendation model training, and the final performance comparison is conducted on the test set. The final results are the mean of the 10 experiments. We also applied a time-based splitting, where historical data is split into two temporal parts; the prediction model is trained on the first part of the historical data, and model performance is measured on the second part of the data. In our case, the splitting ratio was 80:20 for the training and testing purposes. We applied the time-based splitting on the CARS and Yelp datasets that contained the timestamp feature. We evaluated the accuracy of the recommendation models by assessing two measures of prediction accuracy: RMSE and MAE. For the CARS dataset, we also measure the classification accuracy of three types of user feedback (like, dislike, and check-in) with Precision and Recall metrics.

To generate latent contextual vectors for the different tasks (rating prediction and classification tasks), we trained an AE network that contained three layers (input, compressed, and output), with (247, 40, 247) units in each layer in the CARS dataset, (22, 8, 22) units in each layer in the Frappe dataset, and (9, 3, 9) units in each layer in the Yelp dataset, respectively. The number of units in the compressed layer, which represents the unstructured latent contextual vector size, was determined by cross validation (CV) in a separate calibration process for each dataset. We randomly initialized AE parameters with a Gaussian distribution (with a mean of 0 and standard deviation of 0.01), optimizing the model with the Adam algorithm. We tested the batch size of [128, 256, 512, 1024] and the learning rate of [0.0001 ,0.0005, 0.001, 0.005]. For constructing a hierarchical tree from latent context vectors, we used the Euclidian distance with Ward's method [Murtagh and Legendre 2014].

4.3 Recommendation Models

We propose the following extensions of the neural-CF models (NeuMF and NCF) for the CARS domain, as described in Section 3.3:

- *Explicit context-aware model* (*ECAM*): This model considers explicit contextual information, which is modeled by a numeric vector of the available contextual features. $ECAM_{NCF}$ is the extension of the NCF model, and $ECAM_{NeuMF}$ is the extension of the NeuMF model.
- *Unstructured context-aware model* (*UCAM*): This model considers unstructured latent contextual vectors extracted from an AE. $UCAM_{NCF}$ and $UCAM_{NeuMF}$ are the extensions of the NCF and NeuMF models, respectively.
- *Hierarchical context-aware model* (*HCAM*): This model considers structured latent contexts extracted from a hierarchical tree. $HCAM_{NCF}$ and $HCAM_{NeuMF}$ are the extensions of the NCF and NeuMF models, respectively.

We compared our models to the following recommendation baselines:

- *Traditional matrix factorization* (*MF*) [Koren 2008]: We use MF as a non-contextual baseline method to show performance differences between context-aware and context-unaware methods.
- *Factorization machines* (*FM*) [Rendle et al. 2011]: A generic approach that combines the flexibility of feature engineering with factorization models. We use LibFM to implement the method.
- *Explicit context model* (*ECM*) [Baltrunas et al. 2011]: An extension of the traditional MF by learning the rating bias under different explicit context conditions for each item.
- *Latent context model* (*LCM*) [Unger et al. 2016]: An extension of the traditional MF by learning for each item the rating bias under different latent contextual conditions.

Table 2. Prediction Results

		CARS Dataset (Rating Prediction)				Yelp Dataset (Rating Prediction)				Frappe Dataset (App Usage Prediction)	
		10-Fold CV		Time Based		10-Fold CV		Time Based		10-Fold CV	
Model Type	Model	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE
Non-neural baselines	*MF*	0.614	0.5	0.487	0.397	1.16	0.89	2.19	1.698	0.786	0.624
	FM	0.59	0.48	0.468	0.356	1.141	0.881	1.9	1.446	0.71	0.55
	ECM	0.602	0.488	0.481	0.319	1.146	0.883	1.81	1.332	0.69	0.523
	LCM	0.51	0.426	0.395	0.301	1.132	0.871	1.71	1.285	0.689	0.522
Neural-CF baselines	*NCF*	0.431	0.362	0.372	0.315	1.078	0.8	1.54	1.25	0.692	0.528
	NeuMF	0.416	0.338	0.356	0.28	1.026	0.762	1.471	1.19	0.685	0.52
Deep context-aware models	$ECAM_{NCF}$	0.419	0.357	0.361	0.306	0.971	0.73	1.43	1.16	0.679	0.5
	$UCAM_{NCF}$	0.407	0.346	0.353	0.292	0.959	0.69	1.42	1.13	0.668	0.43
	$HCAM_{NCF}$	0.395*	0.335*	0.341*	0.284*	0.881*	0.6*	1.35*	1.09*	0.591*	0.355*
	$ECAM_{NeuMF}$	0.41	0.332	0.354	0.29	0.95	0.713	1.394	1.12	0.654	0.42
	$UCAM_{NeuMF}$	0.392	0.319	0.343	0.278	0.931	0.679	1.379	1.1	0.633	0.396
	$HCAM_{NeuMF}$	**0.381***	**0.308***	**0.33***	**0.243***	**0.862***	**0.587***	**1.311***	**1.077***	**0.578***	***0.349**

- *Neural CF* (*NCF*) [He et al. 2017]: A neural-CF model for learning the biases of users and items, without taking into account contextual information.
- *Neural MF* (*NeuMF*) [He et al. 2017]: A neural-MF model for learning latent features of users and items, and their biases, without taking into account contextual information.

5 RESULTS

We present the rating prediction and classification results of our proposed deep context-aware models for the CARS dataset, the application usage prediction for the Frappe dataset, and the rating prediction results for the Yelp dataset. We apply two splitting strategies: 10-fold CV and a time-based splitting strategy, as described in Section 4.2.

Table 2 illustrates the prediction performance with respect to RMSE and MAE on the three context-aware datasets. The best results in each column are denoted in bold, whereas results that are statistically significant are denoted by an asterisk ($*$) ($p < 0.05$).

We can make the following conclusions from Table 2. First, neural-CF baselines (NCF, NeuMF) and our suggested deep context-aware recommendation models (ECAM, UCAM, and HCAM) outperform non-neural recommendation models (MF, FM, ECM, and LCM). This results are similar to those reported by He et al. [2017], as the NeuMF model achieved the best results across the baselines and improved the non-neural-MF model in terms of all measures on both datasets.

Second, we can notice that in all settings, our deep context-aware models (ECAM, UCAM, and HCAM) outperformed all other baseline models for all of the measures, as observed in Table 2. These results indicate that adding contextual information to the neural models improves recommendation accuracy.

Third, we can observe that the structured representation of latent contexts ($HCAM_{NeuMF}$ and $HCAM_{NCF}$) significantly improves the recommendation accuracy and outperforms all other suggested deep context-aware models ($ECAM_{NeuMF}$, $UCAM_{NeuMF}$, $ECAM_{NCF}$, and $UCAM_{NCF}$), as demonstrated in Table 2. This may be because HCAM handled complex situations at different hierarchy levels, whereas ECAM and UCAM utilized only the current context of the user in an explicit and latent manner without learning a meaningful contextual representation. More specifically, HCAM improves over the best neural-CF baseline NeuMF by 9.2% and 9.7% on the CARS dataset in terms of in RMSE and MAE, respectively, in the 10-fold CV setting, and 7.9% and 15.2% in RMSE and

Table 3. Ranking and Classification Results with the CARS Dataset

		Ranking Metrics			Classification Metrics	
Model Type	Model	Hit@1	Hit@3	Hit@5	Precision	Recall
Non-neural baselines	*MF*	0.117	0.364	0.507	0.687	0.695
	FM	0.159	0.38	0.528	0.694	0.701
	ECM	0.163	0.381	0.522	0.71	0.705
	LCM	0.17	0.386	0.525	0.719	0.728
Neural-CF baselines	*NCF*	0.321	0.54	0.62	0.739	0.765
	NeuMF	0.343	0.588	0.69	0.744	0.77
Deep context-aware models	$ECAM_{NeuMF}$	0.35	0.591	0.694	0.761	0.784
	$UCAM_{NeuMF}$	0.355	0.592	0.696	0.823	0.822
	$HCAM_{NeuMF}$	**0.393***	**0.635***	**0.719***	**0.841***	**0.834***

MAE, respectively, in the time-based setting. On the Yelp dataset, a similar phenomenon was observed, as HCAM performed the best in terms of both the RMSE (0.862) and MAE (0.587) in the 10-fold CV setting, with 19% and 29.8% improvements over the best baseline (NeuMF), respectively. For the task of application usage prediction with the Frappe dataset, HCAM performed the best in terms of both the RMSE (0.578) and MAE (0.349) in the 10-fold CV setting, with 18.5% and 49% improvements over the best baseline (NeuMF), respectively.

Since in the CARS dataset the recommendations were presented to users as a top-10 list, we also measured the ranking quality and performed an evaluation using various hit levels (1, 3, 5), as presented in Table 3. It can be seen that for every value of k, the HCAM model outperforms the other models. In particular, the relative improvement of the HCAM model over the baseline models is 14.6%, 8%, and 4.2% for k values of 1, 3, and 5, respectively. As RSes recommend only a few items at a time, the relevant item should be among the first few items on the list. As we can observe from the table, HCAM obtained major improvements when k is set to a small size (e.g., 1 or 3), as it is significantly better than the best baseline NeuMF and achieves 0.393 in terms of Hit@1. These results confirm that the suggested deep context-aware models can be effectively exploited not only for the rating prediction task but also for generating top-k recommendations.

We further analyzed the classification of three types of user interactions in the CARS dataset: like, dislike, and check-in. We measured the accuracy of classification in terms of Precision and Recall, as presented in Table 3. As can be observed in the table, the classification performance of HCAM is significantly better than the other deep context-aware models, as the Precision (0.841) and Recall (0.834) were improved by 13% and 8.3% over the best baseline NeuMF. This further confirms that utilizing multiple contextual representations in a deep context-aware model significantly improves multiple tasks, such as classification of user feedback, rating prediction, and application usage prediction.

5.1 Context Dimensions

Each context-aware dataset contained different types of contextual dimensions: the CARS dataset contained 247 high-dimensional contextual features obtained from mobile phones; the Frappe dataset contained 22 contextual features collected from time, date, location, and weather factors; and the Yelp dataset contained 9 low-dimensional contextual features obtained from time and location factors. In our experiments, we modeled three types of contextual representations: explicit, unstructured latent contexts, and structured latent contexts (as described in Section 3.1). Explicit modeling considers all available contextual features, whereas the unstructured latent contextual vector size is determined by a separate calibration process for each dataset. As latent context is

represented by a limited number of contextual dimensions, choosing the size of the latent context vector is a fundamental parameter for latent context-based recommendation models.

During the calibration process, we noticed that the deep context-aware method utilizing unstructured latent contexts (UCAM) obtained the best RMSE and MAE results when the latent contextual vector size is compressed from 247 to 40 in the CARS dataset, from 22 to 11 in the Frappe dataset, and from 9 to 3 in the Yelp dataset. For the construction of structured latent contexts, we applied AHC to automatically determine the dimension of the structured latent contextual vector (number of possible contextual situations). We observed that the number of contextual situations in the CARS dataset is 26, in the Frappe dataset is 8, and in the Yelp dataset is 8. Hence, for the high-dimensional dataset CARS, a relatively small size of contextual situations can produce satisfactory results, as HCAM utilized more compact hierarchical latent vectors with the dimension of 26, whereas UCAM used 40 dimensions of the unstructured latent contexts. For the medium-dimensional dataset Frappe, the same phenomenon of a small size of contextual situations produced satisfactory results, as the number of dimensions in the hierarchical latent context vectors is 8, and the number of dimensions in the unstructured latent contexts vectors is 11. However, in the Yelp dataset, which contains low-dimensional contextual space, better recommendation results are obtained when expanding the context size, as HCAM used hierarchical latent vectors with the size of 8 and UCAM used unstructured latent contexts vectors with the size of 3. Therefore, when the dimension of context is low, it is preferable to expand the number of contextual situations to produce better recommendations.

6 CONCLUSION

In this article, we presented a novel deep recommendation framework that incorporates contextual information into the recommendation process. We proposed three deep context-aware models that incorporate three types of contextual representations: explicit, latent, and hierarchical latent information. Experimental results on three context-aware datasets showed that the proposed models, ECAM, UCAM, and HCAM, significantly outperformed the state-of-the-art context-aware methods used for multiple tasks, such as rating prediction, generating top-k recommendations, and classification of users' feedback. We also showed that utilizing structured latent contexts in the proposed deep recommendation framework achieves significantly better performance than the other context-aware models across all tasks. We demonstrated that for high-dimensional contexts, a relatively small size of contextual situations is what one actually needs for the recommendation process, whereas for low-dimensional contexts, it is preferable to expand the number of possible contexts for the recommendation process.

REFERENCES

Gediminas Adomavicius, Bamshad Mobasher, Francesco Ricci, and Alexander Tuzhilin. 2011. Context-aware recommender systems. *AI Magazine* 32, 3 (2011), 67–80. http://www.aaai.org/ojs/index.php/aimagazine/article/view/2364

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

Gediminas Adomavicius and Alexander Tuzhilin. 2015. Context-aware recommender systems. In *Recommender Systems Handbook*. Springer, 191–226.

Linas Baltrunas, Karen Church, Alexandros Karatzoglou, and Nuria Oliver. 2015. Frappe: Understanding the usage and perception of mobile app recommendations in-the-wild. arXiv:1505.03014.

Linas Baltrunas, Bernd Ludwig, and Francesco Ricci. 2011. Matrix factorization techniques for context aware recommendation. In *Proceedings of the 5th ACM Conference on Recommender Systems*. ACM, New York, NY, 301–304.

Linas Baltrunas and Francesco Ricci. 2009. Context-based splitting of item ratings in collaborative filtering. In *Proceedings of the 3rd ACM Conference on Recommender Systems*. ACM, New York, NY, 245–248.

Linas Baltrunas and Francesco Ricci. 2014. Experimental evaluation of context-dependent collaborative filtering using item splitting. *User Modeling and User-Adapted Interaction* 24, 1–2 (2014), 7–34.

Zeynep Batmaz, Ali Yurekli, Alper Bilge, and Cihan Kaleli. 2019. A review on deep learning for recommender systems: Challenges and remedies. *Artificial Intelligence Review* 52, 1 (2019), 1–37.

Hsinchun Chen, Roger H. L. Chiang, and Veda C. Storey. 2012. Business intelligence and analytics: From big data to big impact. *MIS Quarterly* 36, 4 (2012), 1165–1188.

Ian Davidson and S. S. Ravi. 2005. Agglomerative hierarchical clustering with constraints: Theoretical and empirical results. In *Proceedings of the European Conference on Principles of Data Mining and Knowledge Discovery*. 59–70.

Munawar Hayat, Mohammed Bennamoun, and Senjian An. 2014. Learning non-linear reconstruction models for image set classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 1907–1914.

Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings of the 26th International Conference on World Wide Web*. 173–182.

Yogesh Jhamb, Travis Ebesu, and Yi Fang. 2018. Attentive contextual denoising autoencoder for recommendation. In *Proceedings of the 2018 ACM SIGIR International Conference on Theory of Information Retrieval*. ACM, New York, NY, 27–34.

Donghyun Kim, Chanyoung Park, Jinoh Oh, Sungyoung Lee, and Hwanjo Yu. 2016. Convolutional matrix factorization for document context-aware recommendation. In *Proceedings of the 10th ACM Conference on Recommender Systems*. ACM, New York, NY, 233–240.

Yehuda Koren. 2008. Factorization meets the neighborhood: A multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, New York, NY, 426–434.

Hans-Peter Kriegel, Peer Kröger, and Arthur Zimek. 2009. Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering. *ACM Transactions on Knowledge Discovery from Data* 3, 1 (2009), 1.

Nicholas D. Lane, Sourav Bhattacharya, Petko Georgiev, Claudio Forlivesi, and Fahim Kawsar. 2015. An early resource characterization of deep learning on wearables, smartphones and Internet-of-Things devices. In *Proceedings of the 2015 International Workshop on Internet of Things Towards Applications*. ACM, New York, NY, 7–12.

Nicholas D. Lane, Emiliano Miluzzo, Hong Lu, Daniel Peebles, Tanzeem Choudhury, and Andrew T. Campbell. 2010. A survey of mobile phone sensing. *IEEE Communications Magazine* 48, 9 (2010), 140–150.

Jiyun Li, Pengcheng Feng, and Juntao Lv. 2013. ICAMF: Improved context-aware matrix factorization for collaborative filtering. In *Proceedings of the IEEE 25th International Conference on Tools with Artificial Intelligence (ICTAI'13)*. IEEE, Los Alamitos, CA, 63–70.

Lei Mei, Pengjie Ren, Zhumin Chen, Liqiang Nie, Jun Ma, and Jian-Yun Nie. 2018. An attentive interaction network for context-aware recommendations. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. ACM, New York, NY, 157–166.

Fionn Murtagh and Pierre Legendre. 2014. Ward's hierarchical agglomerative clustering method: Which algorithms implement Ward's criterion? *Journal of Classification* 31, 3 (2014), 274–295.

Umberto Panniello, Michele Gorgoglione, and Alexander Tuzhilin. 2016. In CARSs we trust: How context-aware recommendations affect customers' trust and other business performance measures of recommender systems. *Information Systems Research* 27, 1 (2016), 182–196.

Yogesh Singh Rawat and Mohan S. Kankanhalli. 2016. ConTagNet: Exploiting user context for image tag recommendation. In *Proceedings of the 24th ACM International Conference on Multimedia*. ACM, New York, NY, 1102–1106.

Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence*. 452–461.

Steffen Rendle, Zeno Gantner, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2011. Fast context-aware recommendations with factorization machines. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, New York, NY, 635–644.

Nachiketa Sahoo, Param Vir Singh, and Tridas Mukhopadhyay. 2010. A hidden Markov model for collaborative filtering. *MIS Quarterly* 36, 4 (2010), 1329–1356.

Yue Shi, Alexandros Karatzoglou, Linas Baltrunas, Martha Larson, and Alan Hanjalic. 2014. Cars2: Learning context-aware representations for context-aware recommendations. In *Proceedings of the 23rd ACM International Conference on Information and Knowledge Management*. ACM, New York, NY, 291–300.

Elena Smirnova and Flavian Vasile. 2017. Contextual sequence modeling for recommendation with recurrent neural networks. arXiv:1706.07684.

Bartłomiej Twardowski. 2016. Modelling contextual information in session-aware recommender systems with neural networks. In *Proceedings of the 10th ACM Conference on Recommender Systems*. ACM, New York, NY, 273–276.

Moshe Unger, Ariel Bar, Bracha Shapira, and Lior Rokach. 2016. Towards latent context-aware recommendation systems. *Knowledge-Based Systems* 104 (2016), 165–178.

Moshe Unger, Bracha Shapira, Lior Rokach, and Ariel Bar. 2017. Inferring contextual preferences using deep auto-encoding. In *Proceedings of the 25th Conference on User Modeling, Adaptation, and Personalization*. ACM, New York, NY, 221–229.

Moshe Unger and Alexander Tuzhilin. 2019. *Hierarchical Latent Context Representation for CARS*. Stern School of Business, New York University.

Yasi Wang, Hongxun Yao, and Sicheng Zhao. 2016. Auto-encoder based dimensionality reduction. *Neurocomputing* 184 (2016), 232–242.

Svante Wold, Kim Esbensen, and Paul Geladi. 1987. Principal component analysis. *Chemometrics and Intelligent Laboratory Systems* 2, 1-3 (1987), 37–52.

Wenmin Wu, Jianli Zhao, Chunsheng Zhang, Fang Meng, Zeli Zhang, Yang Zhang, and Qiuxia Sun. 2017. Improving performance of tensor-based context-aware recommenders using bias tensor factorization with context feature auto-encoding. *Knowledge-Based Systems* 128 (2017), 71–77.

Yelp. 2018. Challenge Yelp. 2018. Retrieved April 18, 2020 from https://www.yelp.com/dataset/.

Hongzhi Yin, Weiqing Wang, Hao Wang, Ling Chen, and Xiaofang Zhou. 2017. Spatial-aware hierarchical collaborative deep learning for POI recommendation. *IEEE Transactions on Knowledge and Data Engineering* 29, 11 (2017), 2537–2551.

Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. 2019. Deep learning based recommender system: A survey and new perspectives. *ACM Computing Surveys* 52, 1 (2019), 5.

Yong Zheng, Bamshad Mobasher, and Robin Burke. 2014. CSLIM: Contextual SLIM recommendation algorithms. In *Proceedings of the 8th ACM Conference on Recommender Systems*. ACM, New York, NY, 301–304.

Received June 2019; revised January 2020; accepted February 2020