2019 IEEE 35th International Conference on Data Engineering Workshops (ICDEW)

Context-aware Co-Attention Neural Network for Service Recommendations

Lei Li
Department of Computer Science
Hong Kong Baptist University
Hong Kong, China
csleili@comp.hkbu.edu.hk

Ruihai Dong
Insight Centre for Data Analytics
University College Dublin
Dublin, Ireland
ruihai.dong@ucd.ie

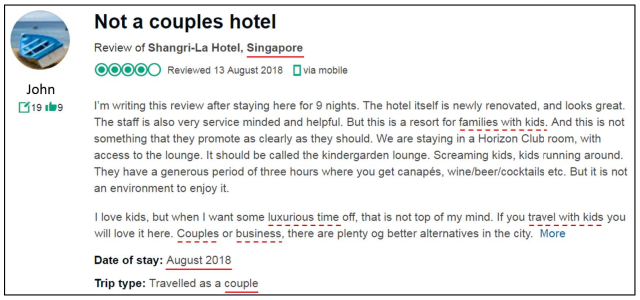
Li Chen
Department of Computer Science
Hong Kong Baptist University
Hong Kong, China
lichen@comp.hkbu.edu.hk

***Abstract*—Context-aware recommender systems are able to produce more accurate recommendations by harnessing contextual information, such as consuming time and location. Further, user reviews as an important information resource, providing valuable information about users' preferences, items' aspects, and implicit contextual features, could be used to enhance the embeddings of users, items, and contexts. However, few works attempt to incorporate these two types of information, i.e., contexts and reviews, into their models. Recent state-of-the-art context-aware methods only characterize relations between two types of entities among users, items and contexts, which may be insufficient, as the final prediction is closely related to all the three types of entities. In this paper, we propose a novel model, named Context-aware Co-Attention Neural Network (CCANN), to dynamically infer relations between contexts and users/items, and subsequently to model the degree of matching between users' contextual preferences and items' context-aware aspects via co-attention mechanism. To better leverage the information from reviews, we propose an embedding method, named Entity2Vec, to jointly learn embeddings of different entities (users, items and contexts) with words in a textual review. Experimental results, on three datasets composed of millions of review records crawled from TripAdvisor, demonstrate that our CCANN significantly outperforms state-of-the-art recommendation methods, and Entity2Vec can further boost the model's performance.**

***Index Terms*—recommender systems, context, co-attention, neural network**

I. Introduction

Nowadays recommender systems (RS) play a significant role in many web applications, such as e-commerce and social network. Recently, contextual information, such as time [1], location [2] and companion, is incorporated into RS for providing better service recommendations. Context in context-aware recommender systems generally refers to "*any information that can be used to characterize the situation of entities*" [3]. It is beneficial for RS to characterize a user's contextual preferences, especially in service recommendation scenarios where customers' behavior can be significantly affected by contextual situations. Intuitively, a user may only care about some specific aspects under a certain contextual situation. For example, when traveling with kids (*context*), a user would probably like to stay in a hotel that provides enough space (*aspect*) for kids to run around; while a user may prefer a quiet environment (*aspect*) if he wants to enjoy luxurious time with his lover (*context*) (See review explanation in Fig. 1). Accordingly, the target item, e.g., a hotel, may be more suitable to some contextual situations because of its corresponding aspects, e.g., room space.


Not a couples hotel
Review of **Shangri-La Hotel, Singapore**
Reviewed 13 August 2018 via mobile
John
19 9

I'm writing this review after staying here for 9 nights. The hotel itself is newly renovated, and looks great. The staff is also very service minded and helpful. But this is a resort for families with kids. And this is not something that they promote as clearly as they should. We are staying in a Horizon Club room, with access to the lounge. It should be called the kindergarden lounge. Screaming kids, kids running around. They have a generous period of three hours where you get canapés, wine/beer/cocktails etc. But it is not an environment to enjoy it.

I love kids, but when I want some luxurious time off, that is not top of my mind. If you travel with kids you will love it here. Couples or business, there are plenty og better alternatives in the city. More

Date of stay: August 2018

Trip type: Travelled as a couple

Fig. 1. A hotel review example from TripAdvisor. Explicit contexts are highlighted with solid lines, and implicit contexts with dash lines.

There are two types of contexts as illustrated in Fig. 1: explicit contexts that explicitly indicate the user's situation (e.g. August 2018), and implicit contexts mentioned in a textual review and mixed with the user's personal experiences (e.g., travel with kids). There are some recommendation algorithms [4], [5] that take both contextual information and user reviews into account when recommending items. However, they require either domain knowledge to extract implicit contexts from user reviews [4], or manual efforts to design feature representations for explicit contexts [5]. As a consequence, the proposed models in [4], [5] can hardly be generalized to other domains.

Recent works on implicit contexts, i.e., review-based models, mainly employ convolutional neural network (CNN) [6]–[9] and recurrent neural network (RNN) [10] as feature extractors to extract feature representations from user reviews for constructing embeddings of users and items (see [11] for another type of feature extractor). However, these methods are computationally expensive, as CNN has hundreds of convolving filters to update [12] and RNN is unable to parallelize for computation. As to context-aware methods that only consider explicit contexts, factorization machines (FM)-based models [13], [14] as a class of general machine learning algorithms perform context-aware recommendation task by regarding context values as sparse features. Consequently, FM-based methods are unable to capture complex relations among users, items and contexts, as these entities are treated as feature IDs in the same way. To solve this problem, Mei et al. [15] recently propose attentive interaction network (AIN) to characterize user-context and item-context interactions, but the user and item representations in AIN do not interact with each other before being passed into the prediction layer, which may be insufficient to model users' preferences for items' aspects.

To address the issues mentioned above, we propose to model interactions between contexts and users/items, and subsequently to estimate the degree of matching between a user's preferences and an item's aspects under each contextual situation via co-attention mechanism [16]. Since the final representations of users and items are mutually learned from users, items and contexts, our approach enables richer modeling of users' contextual preferences and items' context-aware aspects. We name our proposed model Context-aware Co-Attention Neural Network (CCANN). As our model is composed of multi-layer perceptron (MLP) with a few hidden layers, it can be more efficient in terms of computation compared with review-based methods. To leverage implicit contexts in user reviews, we propose a novel embedding method, named Entity2Vec, to jointly learn embeddings of different entities (users, items and contexts) with words in a review. After training Entity2Vec, relations between different users/items/contexts can be captured by their embedding vectors. We use them to initialize our recommendation model CCANN, so that more accurate recommendations can be generated.

The main contributions of our work are summarized as follows:

- We propose a context-aware neural network that leverages co-attention mechanism to characterize the degree of matching between users' contextual preferences and items' context-aware aspects.
- We also propose a novel entity embedding method to jointly learn various entities' embeddings from user generated reviews.
- We conduct extensive experiments on three large datasets to demonstrate the effectiveness of our proposed recommendation model and embedding method.

II. Related Work

There are two lines of research closely related to our work: the first is context-aware recommender systems, and the second utilizes user reviews to enhance recommendations. We present brief summary of the two branches of research work in the following.

Context-aware recommendation algorithms can be broadly classified into three categories according to the phase when contextual information is incorporated: contextual pre-filtering, contextual post-filtering and contextual modeling [17]. Contextual pre-filtering approaches are able to make use of traditional recommendation algorithms, for instance, matrix factorization (MF) [18], where contexts play the role of data filtering, e.g., selecting ratings data for one certain context. However, this approach suffers from severe data sparsity problem, as the data may become sparse after filtering. Existing recommendation algorithms can also be adopted to contextual post-filtering, in which a given context is used to filter out irrelevant recommendations or adjust the recommendation list. Recently, more researchers start to investigate contextual modeling techniques. Beutel et al. [19] incorporate contextual information, such as watch time and device type, into RNN for video recommendation. Baral et al. [5] use manually constructed context features combined with features extracted from user reviews to make point-of-interest (POI) recommendations. To study context-dependent and context-independent preferences of users for service recommendations, Chen and Chen [4] define several contextual variables and context values based on domain knowledge, and design an automatic extraction rule to extract contexts and opinions from user reviews. As it can be seen, these contextual modeling methods are hard to generalize to other problems because they require either domain knowledge or feature engineering techniques. To reduce human efforts, we propose a general context-aware recommendation framework that does not involve manually constructed context features.

The second line of research related to our work is review-based recommender systems. User generated reviews have been widely investigated in recent years for improving recommendation accuracy. Existing methods can be generally categorized into two groups: explicit methods and implicit methods. Explicit methods refer to those algorithms that employ topic modeling or sentiment analysis tools to explicitly analyze review contents [20], while implicit methods primarily extract some latent features from user reviews without analyzing their contents. The aforementioned context-aware method in [4] can be regarded as an explicit method in terms of review modeling, as it extracts opinions and contexts from user reviews for recommendation task. There are many other explicit methods. For example, Zhang et al. [21] propose EFM for explainable recommendation by aligning some typical aspects of items with latent factors of MF. Wang el al. [22] construct a three-way tensor over users, items and aspects using sentiments, and decompose and reconstruct this tensor for achieving both goals of recommendation and explanation. For these explicit methods, one major limitation is that manual preprocessing is usually required for sentiment analysis. Implicit methods, on the other hand, employ a feature extractor to extract feature representations from user reviews for modeling users and items. Given that CNN is capable of extracting representative features from user reviews, some CNN-based recommendation models have been proposed, such as TransNets [6], NARRE [7], ConvMF+ [8], and DeepCoNN [9]. As RNN is able to process sequential data, Lu et al. [10] recently employ bi-directional RNN to construct users' and items' profiles for recommendation by extracting features from word sequences of reviews. Although these methods achieve significant recommendation accuracy improvement, one common limitation is that they are more computationally expensive than traditional recommendation algorithms. To reduce training time, our recommendation model CCANN is constructed with MLP with only a few hidden layers. Furthermore, we propose to learn embeddings of different entities in an automatic way from user reviews using our Entity2Vec, in order to reduce manual efforts on preprocessing contexts, aspects and sentiments.

III. Methodology

In this section, we first present the detail of our recommendation model, Context-aware Co-Attention Neural Network (CCANN), where the co-attention mechanism enables rich interactions between the user's and the item's context-aware embeddings. Further, we introduce our embedding method Entity2Vec that learns representations of different entities from user generated reviews. Using these embedding vectors that carry certain semantic meaning to initialize CCANN, we are able to leverage the knowledge in user reviews to make better recommendations.

A. Context-aware Co-Attention Neural Network (CCANN)

The objective of our recommendation model CCANN is to predict a rating $\hat{r}_{u,i}$ that a user u is likely to comment on an item i when s/he is within certain contextual situations, $c_1, c_2, ..., c_m$, where m denotes the total number of contextual variables and $j \in \{1, 2, ..., m\}$ represents the j-th contextual variable. Specifically, there could be multiple contextual variables, such as *time*, *place* and *companion*, and each contextual variable consists of multiple context values, e.g., *families* and *friends* for the context *companion*. With IDs of a user u, an item i and context values $c_1, c_2, ..., c_m$ as input, we show how to produce a rating score $\hat{r}_{u,i}$ from our context-aware recommendation framework, as shown in Fig. 2.

The user u's low-dimensional representation can be computed via:

$$\mathbf{p}_u = \mathbf{P}^T \mathrm{g}(u) \tag{1}$$

where $\mathbf{P} \in \mathbb{R}^{|\mathcal{U}| \times d}$ denotes the user embedding matrix, $|\mathcal{U}|$ is the total number of users in a dataset, d is the dimension of embedding vectors, and $\mathrm{g}(u) \in \{0, 1\}^{|\mathcal{U}|}$ is a one-hot vector representing which row in the user matrix $\mathbf{P}$ the user u corresponds to. Accordingly, we can obtain the item i's embedding $\mathbf{q}_i$ from the item embedding matrix $\mathbf{Q} \in \mathbb{R}^{|\mathcal{I}| \times d}$, where $|\mathcal{I}|$ is the total number of items. Also, the vector of the j-th contextual variable with the value of l_j, $\mathbf{k}_j^{l_j}$, can be acquired from the j-th context embedding matrix $\mathbf{K}_j \in \mathbb{R}^{|\mathcal{C}_j| \times d}$, where $l_j \in \mathcal{C}_j$ and $|\mathcal{C}_j|$ denotes the total count of values in context j. Notice that, we can initialize embedding matrices $\mathbf{P}, \mathbf{Q}$ and $\mathbf{K}_j$, where $j \in \{1, 2, ..., m\}$, with either random distribution or pre-trained embeddings learned from our Entity2Vec (see next subsection). During the training process, these initial embedding matrices will be fine-tuned by back-propagation, so as to better harmonize our recommendation task.

To characterize the user u's preferences in context j, we pass the user u's embedding $\mathbf{p}_u$ and the corresponding context j's embedding $\mathbf{k}_j^{l_j}$ through multi-layer perceptron (MLP) with one hidden layer

$$\mathbf{p}_{u \to j} = \sigma(\mathbf{W}_j^p [\mathbf{p}_u, \mathbf{k}_j^{l_j}] + \mathbf{b}_j^p) \tag{2}$$

where $[\cdot, \cdot]$ denotes the concatenation of two vectors, $\sigma(\cdot)$ is a nonlinear activation function, and $\mathbf{W}_j^p \in \mathbb{R}^{d \times 2d}$ and $\mathbf{b}_j^p \in \mathbb{R}^d$ are respectively parameter matrix and bias. Similarly, we can compute the user's preferences within other contextual situations, i.e., the same operation is conducted on each contextual variable $j \in \{1, 2, ..., m\}$. Assuming j represents the context *companion*, (2) can be interpreted as that the user's preferences may vary when s/he is with different persons l_j. For instance, when s/he is having a family trip, s/he may prefer a hotel with large bed, while s/he is likely to pay attention to the availability of Wi-Fi when s/he is taking a business trip. As to item i, we can also compute $\mathbf{q}_{i \to j}$ using $\mathbf{q}_i$ and $\mathbf{k}_j^{l_j}$ in a similar way, which can be translated as the suitableness of an item's aspects for context j (e.g., whether a hotel is suitable for the family trip).

Then we measure how much a user's contextual preferences match an item's aspects in context j, by computing an *attention score* between the two vectors, $\mathbf{p}_{u \to j}$ and $\mathbf{q}_{i \to j}$. The attention network is formally defined as

$$\beta_{u,i \to j} = \mathbf{h}^T \sigma(\mathbf{W}_1 [\mathbf{p}_{u \to j}, \mathbf{q}_{i \to j}] + \mathbf{b}_1) \tag{3}$$

in which $\mathbf{W}_1 \in \mathbb{R}^{d \times 2d}$, $\mathbf{b}_1 \in \mathbb{R}^d$ and $\mathbf{h} \in \mathbb{R}^d$ are model parameters. We normalize each element in the attention vector $\boldsymbol{\beta}$ through a softmax function as follows,

$$\alpha_{u,i \to j} = \frac{\exp(\beta_{u,i \to j})}{\sum_{j'=1}^{m} \exp(\beta_{u,i \to j'})} \tag{4}$$

and regard each element $\alpha_{u,i \to j}$ in the resulting vector $\boldsymbol{\alpha}$ as the degree of matching between a user's contextual preferences and an item's associated aspects. Intuitively, the larger a contextual score in attention score vector $\boldsymbol{\alpha}$ is, the more the corresponding context will contribute to the final prediction. As a result, we can obtain an enhanced user vector $\hat{\mathbf{p}}_u$, by computing the weighted sum of the user u's representations over all contexts,

$$\hat{\mathbf{p}}_u = \sum_{j=1}^{m} \alpha_{u,i \to j} \mathbf{p}_{u \to j} \tag{5}$$

which can be interpreted as the contribution of the user u's preferences in each context to her/his final representation. Similarly, we compute the enhanced profile $\hat{\mathbf{q}}_i = \sum_{j=1}^{m} \alpha_{u,i \to j} \mathbf{q}_{i \to j}$ for the item i. The attention score vector $\boldsymbol{\alpha}$ is mutually learned from the user's contextual preferences and the item's context-aware aspects, and then be utilized to enhance user and item representations. We therefore term such an attention mechanism *Co-Attention*.

To make rating prediction, we adopt factorization machines (FM) [13] as our prediction layer, following [9], [23]. To be more specific, we feed the concatenation of the user and the item enhanced representations, $\mathbf{x} = [\hat{\mathbf{p}}_u, \hat{\mathbf{q}}_i]$, into FM as input, and perform a regression task to predict a rating score which measures how much a user u likes an item i,

$$\hat{r}_{u,i} = w_0 + \mathbf{w}_1^T \mathbf{x} + \frac{1}{2} \sum_{f=1}^{k} [(\mathbf{v}_f^T \mathbf{x})^2 - (\mathbf{v}_f^2)^T \mathbf{x}^2] \tag{6}$$

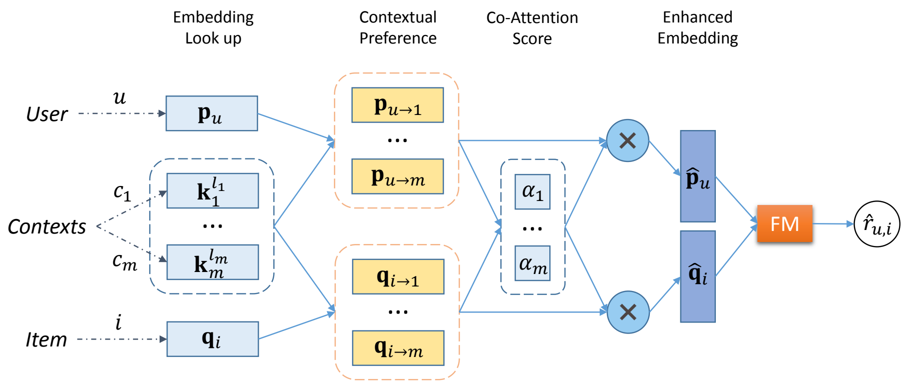


Fig. 2. An overview of our recommendation model Context-aware Co-Attention Neural Network (CCANN).

where $w_0 \in \mathbb{R}$ is the global bias, $\mathbf{w}_1 \in \mathbb{R}^{2d}$ is the weight vector, $\mathbf{v}_f \in \mathbb{R}^{2d}$ is the f-th column of weight matrix $\mathbf{V} \in \mathbb{R}^{2d\times k}$, and k denotes the dimension of factorized parameters. The first two terms in (6) can be regarded as linear regression component, and the third term as the core component of FM represents the second order interactions between features in feature vector $\mathbf{x}$. We employ FM in the form of (6) for implementation because it is computationally efficient, i.e., its time-complexity is $O(kn)$ [13], where $n = 2d$ in our case.

Since rating prediction is essentially a regression problem, we adopt the commonly used mean squared error loss as our objective function,

$$L_r = \sum_{u,i\in\mathcal{T}} (r_{u,i} - \hat{r}_{u,i})^2 \tag{7}$$

where $\mathcal{T}$ is the training set, $r_{u,i}$ denotes the ground truth rating that user u assigned to item i, and $\hat{r}_{u,i}$ is the predicted rating.

As an end-to-end neural network, our model can be easily optimized by stochastic gradient descent (SGD). In our implementation, we adopt an advanced optimizer Adam [24] to minimize the objective function, as it is able to automatically adjust the learning rate during the training phase, which helps neural network to converge faster than vanilla SGD.

To summarize, our CCANN has several advantages. Firstly, as the major contribution of our work, the co-attention mechanism is able to model the user's contextual preferences and the item's context-aware aspects in a richer manner compared with existing context-aware methods [13]–[15]. Secondly, for a context value that a user never experienced before (e.g., a user who used to dine alone now looks for a restaurant for dating), our model can utilize the knowledge learned from other users on this context value to make prediction. Thirdly, since our model is composed of MLP with only a few hidden layers, it can be more efficient in terms of computation than existing review-based models [6]–[10]. Lastly, the co-attention mechanism in our model allows us to identify the most influential contextual factor to the prediction. With selected contexts, we can also explain recommended items to users. For example, we can say "*this hotel is recommended to you because it is suitable for family trip*" to a user after we recommend her/him a hotel. We leave it as the future work.

B. Learning Entity Embeddings from User Reviews

The aim of this component is to learn embedding vectors of different entities from user generated reviews for our recommendation task. We propose a novel embedding method named Entity2Vec, which simultaneously learns embeddings of a variety of entities, including users, items, contexts and words. Intuitively, within a certain contextual situation, a user is more likely to discuss her/his experiences related to this context in the review. Thus, the content of the textual review should also be relevant to the context, in addition to the user and the target item. For example, in the second paragraph of Fig. 1, John (*user*) says that the hotel (*item*) is not suitable for couples or business (*context*). Therefore, it is reasonable to put all of the entities in one model and jointly learn the embeddings of them.

Following PV-DM [25], which learns word vectors and paragraph vectors from a document simultaneously, we average vectors of a user, an item, associated context values, and surrounding words in the review, and subsequently use the resulting vector as features to predict the target word, as shown in Fig. 3. It should be noted that in this subsection we use the term *surrounding words* to denote the target word's nearby words in order to avoid the confusion with *context*.

To reduce training time, we employ CBOW [26] for learning entity embeddings, as our task involves millions of reviews, which could be time-consuming. Formally, given the word sequence of a review, s_1, s_2, ..., $s_{T_{u,i}}$, which a user u wrote for an item i within contextual situations of $c_1, ..., c_m$, we are able to obtain their corresponding embedding vectors from randomly initialized embedding matrices by conducting the same operation as (1). Next, we compute the average of these vectors as follows,

$$\hat{\mathbf{e}}_t = \text{avg}(\mathbf{p}_u, \mathbf{q}_i, \mathbf{k}_1^{l_1}, ..., \mathbf{k}_m^{l_m}, \mathbf{e}_{t-z}, ..., \mathbf{e}_{t+z}) \tag{8}$$

where $\mathbf{p}_u$ and $\mathbf{q}_i$ are the user and the item embeddings respectively, $\mathbf{k}_1^{l_1}, ..., \mathbf{k}_m^{l_m}$ are context embeddings, $\mathbf{e}_{t-z}, ..., \mathbf{e}_{t+z}$ are embeddings of surrounding words, t denotes the target word's

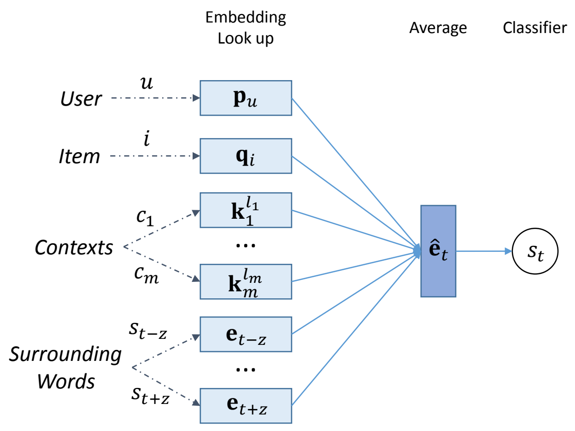


Fig. 3. Illustration of our embedding method Entity2Vec.

position in the review, and z is the size of sliding window on the word sequence. Notice that the target word s_t's embedding $\mathbf{e}_t$ is excluded from input vectors for computing $\hat{\mathbf{e}}_t$.

Then we treat the resulting vector $\hat{\mathbf{e}}_t$ as features to predict the target word s_t by projecting this vector onto the space of vocabulary,

$$\mathbf{y} = \mathbf{W}^e \hat{\mathbf{e}}_t + \mathbf{b}^e \tag{9}$$

where $\mathbf{W}^e \in \mathbb{R}^{|\mathcal{V}| \times d}$ and $\mathbf{b}^e \in \mathbb{R}^{|\mathcal{V}|}$ are respectively parameter matrix and bias, and $\mathcal{V}$ is the vocabulary containing all words in a dataset. We normalize the vector $\mathbf{y} \in \mathbb{R}^{|\mathcal{V}|}$ through the softmax function and estimate the probability of the predicted target being the word s_t as follows:

$$p_t = \frac{\exp(y_t)}{\sum_{t'} \exp(y_{t'})} \tag{10}$$

where p_t denotes the probability that the prediction is word s_t.

Since this task is a multi-class classification problem, we draw on the widely used cross-entropy loss as our objective function and compute the loss for each input-target pair in the training set,

$$L_c = \frac{1}{|\mathcal{T}|} \sum_{u,i \in \mathcal{T}} \frac{1}{T_{u,i}} \sum_{t=z}^{T_{u,i}-z} -\log p_t \tag{11}$$

where $T_{u,i}$ denotes the length of a review that the user u wrote for the item i, z is the window size, and $\mathcal{T}$ consists of reviews in the training set. Similar to our recommendation task, we employ Adam [24] as the optimizer for minimizing the loss function (11).

Obviously, one major advantage of our Entity2Vec is that with the information about user preferences for item aspects in reviews, the semantic meaning of different entities can be well captured by their learned embeddings. For example, the distance between users who share similar preferences is likely to be smaller than those who have different tastes, when computing distance scores for them.

IV. Experiments

A. Dataset Description

To evaluate our proposed model, in September 2018 we collected 10 million hotel reviews in total from three populous cities in TripAdvisor[1], i.e., Hong Kong, New York City and London. We executed the following procedure to each city for constructing three distinct datasets: we first crawled all the review records of each hotel in one city, and subsequently scraped all the historical reviews of users who wrote those reviews from their homepages. We removed non-English reviews because at this stage we are mainly interested in analyzing English text. The statistics of our datasets is shown in Table I, where the values given in brackets correspond to sizes of the datasets without users' past reviews. As it can be seen, each user approximately wrote 1.3 reviews, and each item has hundreds of reviews if historical records are not considered. Notably, we found a large proportion of review records in our datasets are users' historical reviews. For example, the review count in HK dataset is 2,118,108, but only 176,840 reviews (around 8.35%) were written for hotels located in Hong Kong.

Each review record in our datasets contains user ID, item ID, overall star rating in the range of 1 to 5, textual review, and the contexts in which a user was experiencing the item. The contextual information consists of *companion*, *time* and *place*, as depicted in Table II. For the context *time*, we adopt 12 months of a year when a user visited a hotel, instead of the time when a user wrote a review. Similarly, we use the target city, where a hotel locates, as the context value of *place*. As it is impossible to list all values for the context *place* in Table II, we select 10 cities with the largest review counts in HK dataset as examples.

B. Evaluation Metric

To compare the recommendation performance of different methods, we adopt root mean square error (RMSE) as the evaluation metric. RMSE is calculated by estimating the difference between ground-truth rating $r_{u,i}$ and the predicted one $\hat{r}_{u,i}$ in the test set,

$$RMSE = \sqrt{\frac{1}{N} \sum_{u,i} (r_{u,i} - \hat{r}_{u,i})^2} \tag{12}$$

where N indicates the number of instances in the test set.

C. Compared Methods

To evaluate the performance of our CCANN, we compare it with the following state-of-the-art models:

- **PMF**: Probabilistic Matrix Factorization [18]. This is the standard matrix factorization method that characterizes users and items by latent factors inferred from observed ratings. We use alternative least square (ALS) to optimize its objective function for implementation.
- **FM**: Factorization Machines [13]. This is a general machine learning algorithm that uses factorized parameters to model second order interactions between sparse

[1]https://www.tripadvisor.com

TABLE I
OVERVIEW OF OUR TRIPADVISOR HOTEL REVIEW DATASETS.

	# users	*# items*	*# reviews*	*# reviews per user*	*# reviews per item*
HK (Hong Kong)	137,145	247,889 (618)[a]	2,118,108 (176,840)	15.44 (1.29)	8.54 (286.15)
NYC (New York City)	471,243	297,270 (531)	4,572,716 (583,257)	9.70 (1.24)	15.38 (1098.41)
LDN (London)	639,710	354,841 (1,660)	6,382,831 (870,184)	9.98 (1.36)	17.99 (524.21)

[a]The values given in brackets correspond to sizes of the datasets without users' past reviews.

TABLE II
CONTEXTUAL VARIABLES AND CONTEXT VALUES.

Contextual Variables	**Context Values**
Companion	Families, Couples, Solo, Business, Friends
Time	January, February, March, April, May, June, July, August, September, October, November, December
Place[a]	Hong Kong, Bangkok, Singapore, London, New York City, Dubai, Kuala Lumpur, Shanghai, Paris, Sydney

[a]As examples, we list 10 selected cities with the largest review counts in HK dataset.

features. We only use user and item IDs as inputs, and perform experiments using LibFM[2] with SGD learner.

- **ConvMF+**: Convolutional Matrix Factorization [8]. It employs a CNN to exploit textual information from item description for enhancing MF. Here we concatenate user reviews of an item as item description.
- **DeepCoNN**: Deep Cooperative Neural Networks [9]. This model consists of two parallel CNNs for learning feature representations from user reviews for users and items respectively.
- **NFM**: Neural Factorization Machines [14]. It is a more generalized FM built upon neural network for learning high-order feature interactions in a non-linear way for sparse data prediction. We conduct experiments using the source code provided in the paper, and feed IDs of users, items and contexts into this model as input features.
- **AIN**: Attentive Interaction Network [15]. This neural network employs two pathways to model the effects of contexts on users and items. To be fair, we remove the fully connected layers in this model, so that we can focus on the comparison of its attention mechanism with the co-attention mechanism of our CCANN.
- **CCANN**$_{rand}$: It is a variant of our CCANN where entity embeddings are randomly initialized rather than obtained from Entity2Vec.

PMF and FM both are interaction-based models, which only take user and item IDs into account for rating prediction. Therefore, we regard them as context-unaware baselines. ConvMF+ and DeepCoNN are both review-based models that extract features, including contextual information, from user reviews. We hence call ConvMF+ and DeepCoNN as implicit context-aware models. NFM, AIN and our CCANN explicitly use the information of contexts for recommendation, for which reason they are context-aware methods.

D. Experiment Setup

We randomly divide each dataset into training (80%), validation (10%) and test (10%) sets. We also guarantee each user/item has at least one instance in the training set. We repeat the splitting process for 5 times, and report the averaged performance. The validation set is used for hyper-parameters tuning. The early stopping strategy is performed for all models, i.e., we report a model's RMSE on the test set at the epoch where it reaches the best performance on the validation set. We implement ConvMF+, DeepCoNN, AIN and our CCANN[3] in Python using TensorFlow[4]. All neural network based methods, i.e., ConvMF+, DeepCoNN, NFM, AIN, and CCANN, are optimized by Adam [24]. Specifically, we fix the batch size to 128 and conduct grid search for each model's learning rate from [10^{-5}, 10^{-4}, ..., 10^{-1}]. The learning rate of FM is also searched from this range. For MF-based models, i.e., PMF and ConvMF+, we set the dimension of the latent factor to 20, and search tradeoff parameters from [0.1, 1, 10, 100]. For FM-based models, i.e., FM, DeepCoNN, NFM, and CCANN, the dimension of factorized parameters k is set to 10, following [23]. For NFM, AIN and our CCANN, we set the embedding size d to 50, and use $ReLU(\cdot)$ as the activation function. In addition, to fairly compare the models' capability we disable the batch normalization technique in NFM, since other models do not apply this trick for improving rating prediction accuracy. For CNN-based models (ConvMF+ and DeepCoNN), we initialize word embedding layer with pre-trained word vectors on Google News from Word2Vec[5] [26], and the two models are regularized with dropout ratio of 0.2. The maximum document length of concatenated reviews is set to 1,000 words for ConvMF+ and DeepCoNN, following the settings in [6]. We calculate tf-idf score for each word in each dataset, and select top 20,000 distinct words with the largest document frequency (df) from user reviews to construct the vocabulary $\mathcal{V}$. For Entity2Vec, we randomly sample 10% input-target pairs from each user review, in order to reduce the training time.

In our datasets, some users did not explicitly indicate their contexts in their reviews, so we use a special token <*UNK*>

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

[3]Codes are available at https://github.com/Eli1995CS/CCANN-Entity2Vec
[4]https://www.tensorflow.org
[5]https://code.google.com/archive/p/word2vec

TABLE III
PERFORMANCE COMPARISON IN TERMS OF RMSE.

Category	Methods	**HK**	**NYC**	**LDN**
context-unaware	**PMF**	1.1185	1.1926	1.1923
	FM	1.0140	1.0491	1.0437
implicit context-aware	**ConvMF+**	0.9001	0.9975	0.9832
	DeepCoNN	0.8546	0.8815	0.8805
context-aware	**NFM**	0.8481	0.8727	0.8661
	AIN	0.8469	0.8664	0.8606
ours	**CCANN**$_{rand}$	0.8439	0.8658	0.8598
	CCANN	**0.8409***	**0.8652***	**0.8586***

* denotes the statistical significance for $p < 0.001$ given by student's t-test, compared to NFM.

to denote the missing contexts. Moreover, for the context of *place*, we also use this token to represent a city that contains less than 100 reviews.

V. RESULTS AND DISCUSSIONS

A. Comparative Analysis on Overall Performances

The rating prediction results of our recommendation model CCANN and baseline models on three datasets are given in Table III. From the results, we have three observations.

Firstly, context-aware methods (NFM, AIN and CCANN) generally perform better than the other models (PMF, FM, ConvMF+ and DeepCoNN) that do not explicitly consider contexts. This is not surprising, as users' decisions can be affected by contextual factors, especially in service recommendation scenarios. As such, context-aware models can better characterize users' preferences over items' aspects for recommendation.

Secondly, review-based methods (ConvMF+ and DeepCoNN) outperform traditional context-unaware models (PMF and FM), because user reviews contain a lot of information about users and items, including contextual factors, which can be implicitly extracted by feature extractors, such as CNN and RNN, to enhance user and item representations. However, since some users did not indicate their contextual situations in the reviews, which may explain why review-based methods underperform context-aware algorithms.

Thirdly, as shown in Table III, our CCANN consistently outperforms all the baselines, including context-aware methods NFM and AIN. Although NFM is a context-aware approach, it treats all entities (i.e., users, items, and contexts) equally and models their relations in the same way, which may be insufficient for characterizing the effects of contexts on users and items. On the other hand, AIN employs two pathways equipped with attention mechanism to dynamically infer the relations between contexts and users/items, so it can generate better recommendations than NFM. However, the representations of users and items in this model are only interacted in the final prediction layer, which may be not enough for accommodating complex relations between users and items. In comparison, our model leverages co-attention mechanism to automatically adjust the weight of a contextual variable that matches a user's contextual preferences and the target item's related aspects, and uses this weight on the user and item embeddings in turn. Compared with AIN, the co-attention mechanism in our model enables richer interactions between users and items, and thus leads to better performance. In addition, with embeddings learned from user reviews via Entity2Vec, our CCANN's performance is further boosted compared with the randomly initialized CCANN$_{rand}$, which validates the rationale of our embedding method Entity2Vec.

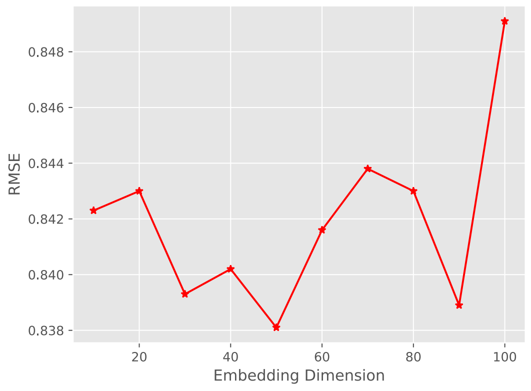


Fig. 4. Performance of CCANN w.r.t different embedding dimensions.

To verify the efficiency of our proposed CCANN, we estimate and compare the computational runtime of different neural network models (ConvMF+, DeepCoNN, NFM, AIN and CCANN). We only show the runtime when these models reach the best performance on HK dataset on an NVIDIA Tesla K80 GPU, as the results are similar on three datasets. Let ρ be the runtime of NFM, the runtime of AIN and our CCANN is around ρ as well. To reach the performance shown in Table III, ConvMF+ and DeepCoNN run approximately at 13ρ and 68ρ, respectively. In a degree, this proves the efficiency of our CCANN.

B. Tuning Hyper-parameters

In this subsection, we show our exploration on how different settings of the hyper-parameters would influence the performance of our proposed CCANN. We only show the results on HK dataset, since they share similar patterns on the other datasets. The examined hyper-parameters include the dimension of embeddings and the initial learning rate. The curves of RMSE for two hyper-parameters on validation set are presented respectively in Fig. 4 and Fig. 5. Although the curve in Fig. 4 fluctuates dramatically, we can see that the model performs poor when the embedding dimension is too large or too small. Therefore, we set the embedding dimension to be 50, as the model performs the best with this value. Fig. 5 shows the similar trend, so we set the learning rate to be 0.001 for our CCANN.

VI. CONCLUSION AND FUTURE WORK

In this paper, we propose a context-aware recommendation method that models relations between contexts and users/items, and subsequently estimates the degree of matching between a user's preferences and an item's aspects for

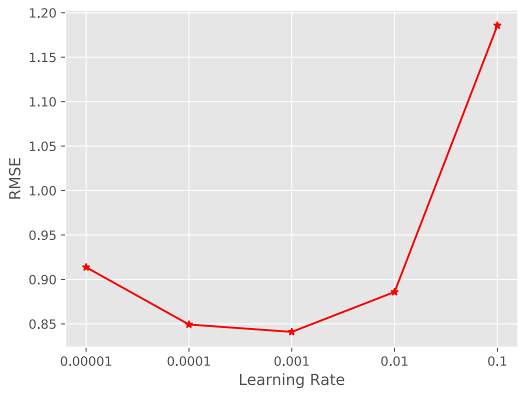


Fig. 5. Performance of CCANN w.r.t different learning rates.

each contextual situation via co-attention mechanism. This mechanism enables rich interactions between users' and items' context-aware representations, since they are mutually learned from users, items and contexts. To further boost the performance, we propose an embedding method Entity2Vec to jointly learn different entities' embeddings from user reviews. Experimental results on three large datasets show that our CCANN not only achieves better performance but also takes acceptable training time, compared with the state-of-the-art recommendation methods.

In the future, we will test our model on other service domains (e.g., restaurant). We also plan to verify the necessity of each component in our neural network, and will revise the architecture accordingly to improve its performance. As we mainly focus on recommendation task, the investigation on co-attention scores has been omitted in this paper. We will conduct a live-user study to investigate the selected contexts for explanation purposes. We are also interested in the techniques of automatic text generation, e.g., review generation [27], as it may allow us to produce customized explanations in the form of human-readable text.

References

[1] H. Yin, B. Cui, L. Chen, Z. Hu, and Z. Huang, "A temporal context-aware model for user behavior modeling in social media systems," in *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*. ACM, 2014, pp. 1543–1554.

[2] H. Yin, Y. Sun, B. Cui, Z. Hu, and L. Chen, "Lcars: a location-content-aware recommender system," in *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2013, pp. 221–229.

[3] G. D. Abowd, A. K. Dey, P. J. Brown, N. Davies, M. Smith, and P. Steggles, "Towards a better understanding of context and context-awareness," in *International symposium on handheld and ubiquitous computing*. Springer, 1999, pp. 304–307.

[4] G. Chen and L. Chen, "Augmenting service recommender systems by incorporating contextual opinions from user reviews," *User Modeling and User-Adapted Interaction*, vol. 25, no. 3, pp. 295–329, 2015.

[5] R. Baral, X. Zhu, S. Iyengar, and T. Li, "Reel: Review aware explanation of location recommendation," in *Proceedings of the 26th Conference on User Modeling, Adaptation and Personalization*. ACM, 2018, pp. 23–32.

[6] R. Catherine and W. Cohen, "Transnets: Learning to transform for recommendation," in *Proceedings of the Eleventh ACM Conference on Recommender Systems*. ACM, 2017, pp. 288–296.

[7] C. Chen, M. Zhang, Y. Liu, and S. Ma, "Neural attentional rating regression with review-level explanations," in *Proceedings of the 2018 World Wide Web Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 2018, pp. 1583–1592.

[8] D. Kim, C. Park, J. Oh, S. Lee, and H. Yu, "Convolutional matrix factorization for document context-aware recommendation," in *Proceedings of the 10th ACM Conference on Recommender Systems*. ACM, 2016, pp. 233–240.

[9] L. Zheng, V. Noroozi, and P. S. Yu, "Joint deep modeling of users and items using reviews for recommendation," in *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*. ACM, 2017, pp. 425–434.

[10] Y. Lu, R. Dong, and B. Smyth, "Coevolutionary recommendation model: Mutual learning between ratings and reviews," in *Proceedings of the 2018 World Wide Web Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 2018, pp. 773–782.

[11] Y. Zhang, H. Yin, Z. Huang, X. Du, G. Yang, and D. Lian, "Discrete deep learning for fast content-aware recommendation," in *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. ACM, 2018, pp. 717–726.

[12] Y. Kim, "Convolutional neural networks for sentence classification," pp. 1746–1751, 2014.

[13] S. Rendle, "Factorization machines," in *10th International Conference on Data Mining (ICDM)*. IEEE, 2010, pp. 995–1000.

[14] X. He and T.-S. Chua, "Neural factorization machines for sparse predictive analytics," in *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval*. ACM, 2017, pp. 355–364.

[15] L. Mei, P. Ren, Z. Chen, L. Nie, J. Ma, and J.-Y. Nie, "An attentive interaction network for context-aware recommendations," in *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. ACM, 2018, pp. 157–166.

[16] B. Hu, C. Shi, W. X. Zhao, and P. S. Yu, "Leveraging meta-path based context for top-n recommendation with a neural co-attention model," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 2018, pp. 1531–1540.

[17] G. Adomavicius and A. Tuzhilin, "Context-aware recommender systems," in *Recommender Systems Handbook*, 2nd ed., B. Shapira, Ed. Springer, 2015, ch. 6, pp. 191–226.

[18] A. Mnih and R. R. Salakhutdinov, "Probabilistic matrix factorization," in *Advances in neural information processing systems*, 2008, pp. 1257–1264.

[19] A. Beutel, P. Covington, S. Jain, C. Xu, J. Li, V. Gatto, and E. H. Chi, "Latent cross: Making use of context in recurrent recommender systems," in *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. ACM, 2018, pp. 46–54.

[20] L. Chen, G. Chen, and F. Wang, "Recommender systems based on user reviews: the state of the art," *User Modeling and User-Adapted Interaction*, vol. 25, no. 2, pp. 99–154, 2015.

[21] Y. Zhang, G. Lai, M. Zhang, Y. Zhang, Y. Liu, and S. Ma, "Explicit factor models for explainable recommendation based on phrase-level sentiment analysis," in *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*. ACM, 2014, pp. 83–92.

[22] N. Wang, H. Wang, Y. Jia, and Y. Yin, "Explainable recommendation via multi-task learning in opinionated text data," in *Proceedings of the 41st international ACM SIGIR conference on Research & development in information retrieval*. ACM, 2018, pp. 165–174.

[23] Y. Tay, L. A. Tuan, and S. C. Hui, "Multi-pointer co-attention networks for recommendation," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 2018.

[24] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014.

[25] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *International Conference on Machine Learning*, 2014, pp. 1188–1196.

[26] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *ICLR Workshop*, 2013.

[27] Y. Lu, R. Dong, and B. Smyth, "Why i like it: Multi-task learning for recommendation and explanation," in *Proceedings of the Eighth ACM Conference on Recommender Systems*. ACM, 2018.