Hindawi
Complexity
Volume 2019, Article ID 2926749, 11 pages
https://doi.org/10.1155/2019/2926749

Research Article

A Joint Deep Recommendation Framework for Location-Based Social Networks

Omer Tal and Yang Liu

Department of Physics and Computer Science, Wilfrid Laurier University, Waterloo, Ontario, Canada

Correspondence should be addressed to Yang Liu; yangliu@wlu.ca

Received 1 December 2018; Revised 11 February 2019; Accepted 5 March 2019; Published 19 March 2019

Guest Editor: Jiajie Xu

Location-based social networks, such as Yelp and Tripadvisor, which allow users to share experiences about visited locations with their friends, have gained increasing popularity in recent years. However, as more locations become available, the need for accurate systems able to present personalized suggestions arises. By providing such service, point-of-interest recommender systems have attracted much interest from different societies, leading to improved methods and techniques. Deep learning provides an exciting opportunity to further enhance these systems, by utilizing additional data to understand users' preferences better. In this work we propose *Textual and Contextual Embedding-based Neural Recommender* (TCENR), a deep framework that employs contextual data, such as users' social networks and locations' geo-spatial data, along with textual reviews. To make best use of these inputs, we utilize multiple types of deep neural networks that are best suited for each type of data. TCENR adopts the popular multilayer perceptrons to analyze historical activities in the system, while the learning of textual reviews is achieved using two variations of the suggested framework. One is based on convolutional neural networks to extract meaningful data from textual reviews, and the other employs recurrent neural networks. Our proposed network is evaluated over the Yelp dataset and found to outperform multiple state-of-the-art baselines in terms of accuracy, mean squared error, precision, and recall. In addition, we provide further insight into the design selections and hyperparameters of our recommender system, hoping to shed light on the benefit of deep learning for location-based social network recommendation.

1. Introduction

In today's age of information, it has become a prerequisite to have reliable data prior to making any decision. Preferably, we expect to obtain reviews from like-minded users before investing our time and money for any product or service. Location-based social networks (LBSN), such as Yelp, TripAdvisor, and Foursquare, provide such information about potential locations, while allowing users to connect with their friends and other users that have similar tastes. And indeed, these networks are gaining popularity. Yelp recently (https://www.yelp.ca/factsheet) reported having 72 million mobile active users every month, while TripAdvisor achieved no less than 390 million monthly users (https://www.tripadvisor.com/TripAdvisorInsights/w828). However, as LBSNs have more users with various preferences and additional locations are being added on a daily basis, it becomes time consuming to browse through all possibilities before finding an appropriate restaurant or venue to visit.

Point-of-interest (POI) recommendation, a subfield of recommender systems (RS), attempts to provide LBSN users with personalized suggestions. Properly exploited, it can save time and effort for the end user and encourages her to make future use in the LBSN both as a consumer and as a content provider. Collaborative filtering (CF) is probably the most prominent RS paradigm. It is based on the assumption that users who had resembling past activities will have similar future preferences.

While POI recommendation methods usually attempt to solve a similar problem as the standard recommender system, they are required to overcome additional challenges. First, data sparsity, commonly referred to as the cold start problem, is based on the fact that most users will only interact with a small fraction of the possible locations and vice-versa. When adopting methods of CF, it becomes harder to represent users and locations based on similar past activities, when the variation is high. This issue is worsened for users and

locations that have few or no past interactions known to the system, as available data is insufficient to fully capture their features. Second, in contrast to other recommendation scenarios, the decision whether to visit a location depends not only on the target user's preferences but on that of her friends [1]. They might share different interests that are unknown to the system, resulting in a more complex decision process for the RS to learn. Furthermore, attributes that relate to the location itself and are unknown to the system may have impact on the decision. For example, the availability of parking or convenient public transport, or the popularity of the area itself, can be the decisive factor for a user. These challenges proved the classic CF methods to be insufficient and prone to overfit the data as reported by previous works [2, 3].

A common approach to address sparsity while acknowledging the user's deriving factors is by utilizing contextual data, such as social networks [2, 4], geographical locations [5, 6], categories [1], and textual reviews [7, 8]. Deep neural networks have been enthusiastically adopted in recent years [2, 9, 10] to employ such complex contextual inputs, while being able to process the vast amount of available data. Different neural network architectures have been introduced to improve the recommendation performance, such as multilayer perceptrons (MLP) [3, 11, 12], convolutional neural networks (CNN) [13–15], and recurrent neural networks (RNN) [16–18]. These techniques were shown to highly benefit from the addition of contextual attributes. As more data becomes available, incorporating these features presents a promising opportunity to improve the personalized POI recommendation process, as will be demonstrated in this paper.

Although numerous works established the potential of recommender systems based on deep learning, most have focused on only a single type of neural network that best suited their given task. However, in this work, we intend to incorporate multiple types of neural networks, i.e., MLP, CNN, and RNN, to provide POI recommendation using various types of inputs. First we will describe our proposed method, *Textual and Contextual Embedding-based Neural Recommender* (TCENR), a framework that takes users' social network, locations' geographical data, and textual reviews along with historical activities, to provide personalized top-k POI recommendations. By optimizing MLP and CNN jointly, the proposed model will learn different aspects of the same user-location interaction in conjunction and therefore will be better suited to capture the underlying factors in the user's selection. We will then present an extension of TCENR, denoted as $\text{TCENR}_{\text{seq}}$, where we replace the CNN component with that of an RNN and attempt to learn user preferences and location attributes by treating the written reviews as a sequential input, rather than by focusing on their most important words. Although the proposed solution has been developed to provide recommendations for specific types of inputs, we claim it can be easily generalized to a framework able to support additional features.

The main contributions of our work are as follows:

(1) We present TCENR, a framework that jointly trains MLP and CNN to provide POI recommendations, as well as a variation, $\text{TCENR}_{\text{seq}}$, that performs the same task while adopting RNN instead of the CNN component.

(2) To the best of our knowledge, no work has been done in jointly training MLP and CNN for the task of POI recommendation using social networks, geographical locations, and natural language reviews as inputs.

(3) Evaluated over the Yelp dataset, our proposed frameworks were found to outperform seven state-of-the-art baselines in terms of accuracy, MSE, precision, and recall.

(4) By comparing the two alternatives to our suggested model, we provide insight into the impact gained by analyzing textual reviews as a secondary input to the common past interactions, as well as a comparison of CNN and RNN for the task of sentiment analysis in the same experimental settings.

(5) We further present comprehensive analysis over the most important hyper-parameters and design selections of our proposed networks, shedding some light over the different components of deep neural networks in the task of POI recommendation.

In the following section, we first lay the background and introduce related works to our model. In Section 3 we develop our proposed frameworks, which are evaluated and analyzed in Section 4. Finally, we summarize our work and introduce future work in Section 5.

2. Related Work

Incorporating additional data about users and items is a common strategy to provide meaningful recommendations and to mitigate the cold-start problem. In the area of POI recommender systems, where the data is highly sparse, such practices are essential.

2.1. Context-Based Recommender Systems. Location-based social networks are usually rich in contextual inputs which present various opportunities for data enrichment in RS. Such features include time [9], spatial location [19], user's social network [4], item's meta-data [1], photos [20], and demographics [11].

Contextual data is usually incorporated into RS either as part of the input or as a regularizing factor. Reference [11, 21] exploits the strengths of MLP-based networks in modeling complex relationships by concatenating multiple feature embeddings to the input before feeding it to a series of nonlinear layers. The final layer's output is then a representation of a user-item interaction, adjusted to the given context. Sequentiality is very often introduced to recommender systems by treating sequences of past user and item interactions in a timely manner. In the case of POI recommendations, it consists of feeding sequences of check-ins to RNN-based layers such as long-short term memory (LSTM) [22] or the more concise gated recurrent units (GRU) [23].

The use of spatial data is often done by dividing the input space into roles and regions. Assuming users' behavior varies when traveling far from home, previous works [5, 6] generated two profiles for each user, one to be used in her home region while another in more distant locations. A recent approach [24–26] attempts to divide the input space into geographical regions before incorporated into the model, often by hierarchical structures. Although methods based on regions and roles are able to better distinguish user behaviors in varied locations, they do not provide a personalized user representation and can ignore potential shifts of preferences from one region to another. For example, a user might prefer to visit a Starbucks location in different regions, close or far from home. Enriching geographical features with additional data is demonstrated in [25, 27] where the next location prediction is partially determined by past sequences of check-ins. However, these methods are not generic and cannot be extended to include additional features, derived from social networks or textual data.

Furthermore, in case of tasks in highly sparse environments, such as POI recommendation, adding user, or item specific inputs may diminish the model's ability to generalize. However, applying the same data as a regulating factor can enhance the model's performance and reduce over-fitting. Such has been done in [4], where the similarity between connected users in the social network was used to constrain a matrix factorization (MF) model. Reference [2] utilized social networks and geographical distances to enforce similar embeddings for users and locations, thus improving the model's ability to generalize for users and locations with few historical records.

2.2. Text-Based Recommendation. Since many websites encourage users to provide a written explanation to their numeric ratings, textual reviews are one of the most popular types of data to be integrated into RS. Previous works had adopted probabilistic-based approaches to alleviate the data sparsity problem using textual input [28–31]. By expressing each review as a bag of words, LDA-based models are able to extract topics which can be used to represent users' interests and locations' characteristics [5, 6, 25]. These probabilistic methods are usually successful in handling issues that standard CF approaches struggle with, such as out-of-town recommendations where similar users lack sufficient historical data. However, as demonstrated in recent works [17, 32], failing to preserve the original order of words and ignoring their semantic meaning prevent the successful modeling of a given review. On the other hand, an emerging trend of adopting neural networks over reviews allows such learning without the loss of data. These implementations can generally be categorized into RNN-based [17, 18] and CNN-based [14, 15] models.

RNN-based recommender systems usually rely on the sequential structure of sentences to learn their meaning. In [18], the words describing a target item are fed to a bidirectional RNN layer. Following the GRU architecture, the model utilizes an accumulated context from successive words to provide better representation of each word. To preserve and update the context of words, the recurrent model has to manage a large number of parameters. The problem becomes more prominent when adopting the popular LSTM paradigm as done in [17].

Following its success in the field of computer vision [33], CNN-based models are gaining popularity in other areas, such as textual modeling [34]. By employing a sliding window over a given document, such networks are able to represent different features found within the text by identifying relevant subsets of words. Reference [15] follows the standard CNN structure, comprised of an embedding to represent the semantic meaning of each word, a convolution layer for generating local features, and a max pooling operation to identify the most relevant factors. In [14], two CNNs are developed to represent the target user and item based on their reviews. The resulting vectors are regarded as the user and item representations, which are then fed to a nonlinear layer to learn their corresponding rating.

We claim that by jointly learning contextual and textual based deep models our proposed method will better exploit the strengths of collaborative filtering, while being more resilient to its shortcomings in sparse scenarios. This will be achieved by learning users' and locations' representations as similarities in direct interactions, along with the correlation in underlying features extracted from their written reviews. The notable work of [35] proposed JRL, a framework that similarly attempts to jointly optimize multiple models, where each is responsible for learning a unique perspective of the same task by focusing on different inputs. However, while JRL is a general framework, focused on extendability to many types of input, our proposed method is tailored for the task of POI recommendation.

3. The Neural Recommender

3.1. Neural Network Architecture. The following recommender system aims to improve the POI recommendation task by learning user-location interactions using two parallel neural networks, as shown in Figure 1. The context-based network, presented in the left part of the figure, is designed to model the user-POI preferences using social and geographical attributes and based on a multilayer perceptron structure [2, 3]. Shown in the right side of Figure 1, the convolutional neural network is responsible for the textual modeling unit [14]. It attempts to learn the same preference by analyzing the underlying meaning in users' and locations' reviews. Each of the two networks is based on modeling the user/POI input individually with regard to their shared interaction, defined in the merge layers.

3.1.1. Context-Based Network Layers. To better capture the complex relations between users and locations in LBSN, we chose to adopt the multilayer perceptron architecture. By stacking multiple layers of nonlinearities, MLP is capable of learning relevant latent factors of its inputs. It is first fed with user and location vectors of sizes N and M, where each input tuple $\langle u, p \rangle$ is transformed into sparse one-hot encoding representations. The two fully connected embedding layers, found on top of the input layer, project the sparse representations of users and locations into smaller and denser

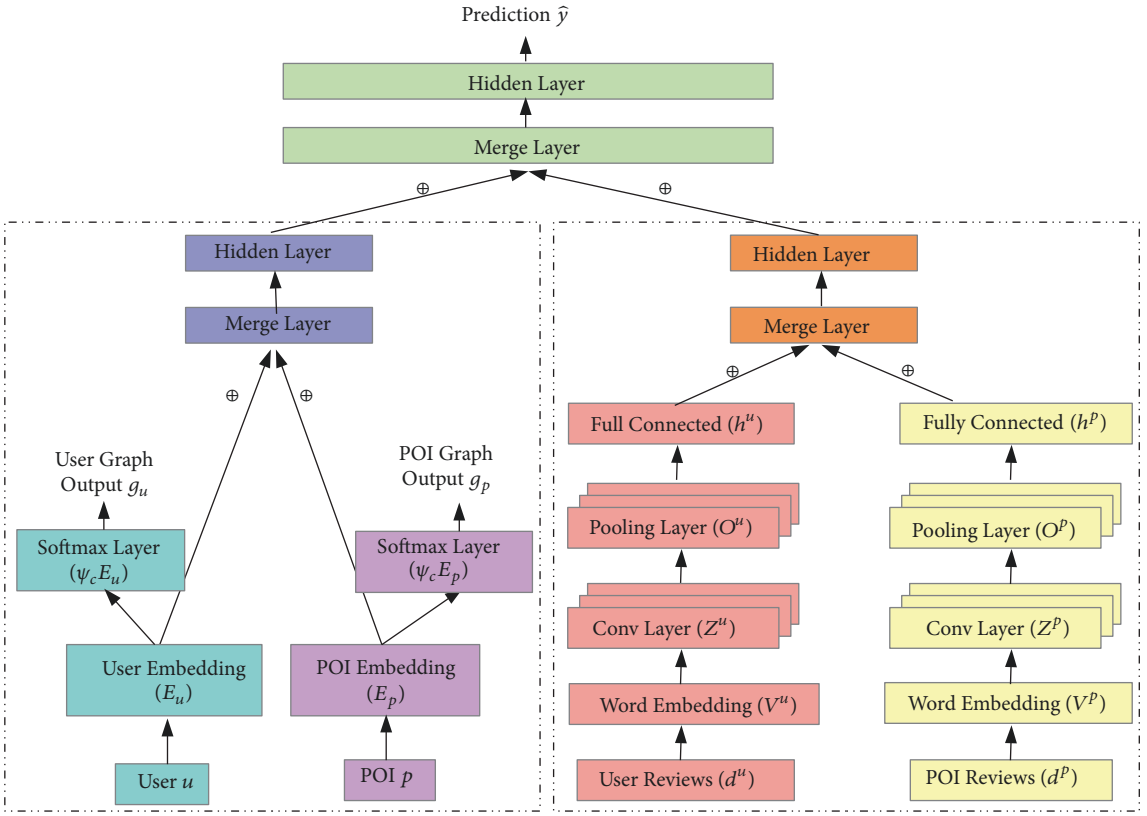


Figure 1: TCENR Framework.

vectors. For u and p, the respective embedding matrices are $E_u \in \mathbb{R}^{k_u \times N}$ and $E_p \in \mathbb{R}^{k_p \times M}$, where k_u and k_p are the corresponding dimensions.

We exploit the users' social networks along with the locations' geographical graphs to constrain the learned embeddings. Two softmax layers take the representations E_u and E_p as input and transform them back to N and M sized vectors, respectively. The user output layer, $\psi_c E_u \in \mathbb{R}^N$, describes the similarity of a user's embedding to all N users and can be formally denoted as

$$\psi_c E_u = a\left(W_c^u \times E_u + b_c^u\right), \tag{1}$$

where W_c^u and b_c^u are the layer's weight matrix and bias vector and a is a nonlinear activation function. Due to the similarity between the user and POI specific layers, the location output layer, $\psi_c E_p$, will not be developed in this section. Enforcing $\psi_c E_u$ and $\psi_c E_p$ to resemble user u's social graph, g_u, and location p's spatial graph, g_p, respectively, results in a smoothing factor that limits the amount by which connected entities' embeddings differ.

The user and location embeddings are projected to a merge layer and combined by a concatenation operator. Using concatenation instead of dot-product allows varied embedding structures, which in turn improves the generated representations [3]. As the input layer for the following neural network, the merge layer can be represented as $h^{(0)}(x)$, where

$$x = \left[E_u, E_p\right]. \tag{2}$$

Since simple concatenation of the user-location embedded vectors does not allow for interactions to be modeled, hidden layers are added to learn these connections. The popular Rectified Linear Units (ReLU) is employed as the activation function for these layers. More formally, the q-th hidden layer can be defined as

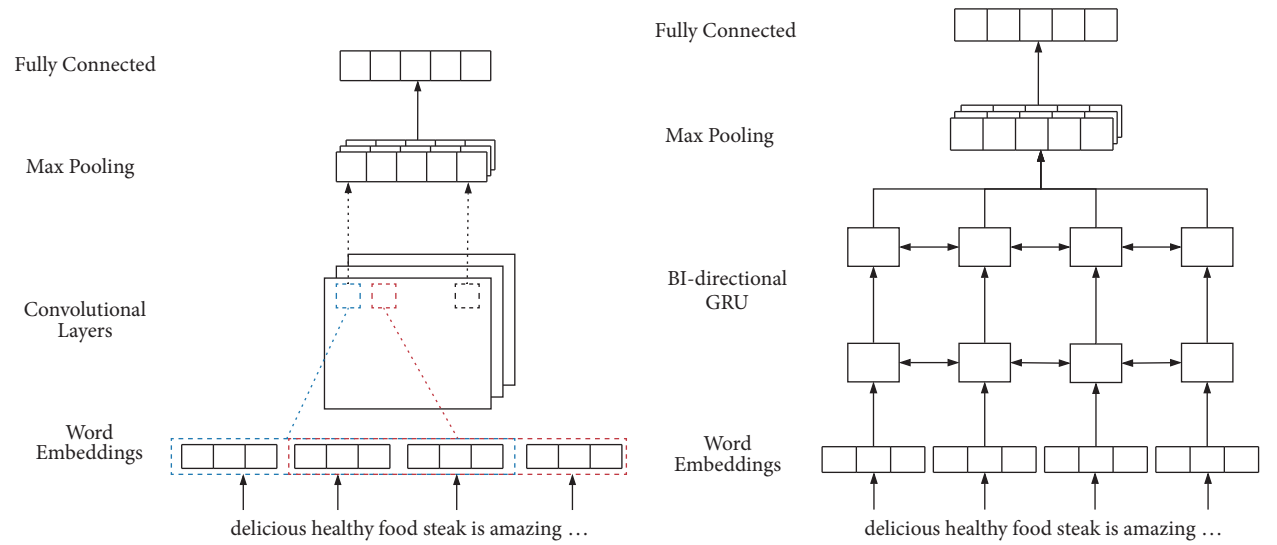
$$h_{context}^q(x) = ReLU\left(W^q h_{context}^{q-1}(x) + b^q\right), \tag{3}$$

where W^q and b^q are the q-th layer parameters.

3.1.2. Textual Modeling Network Layers. To improve the model's coverage, a textual-based network is introduced. It simultaneously learns the same interaction as the contextual-based network, but with a natural language input. Two additional vectors d^u and d^p, representing user u's and location p's textual reviews, respectively, are applied as inputs for this network. Each vector is comprised of all n words written by the user or about the location merged together, kept in their original order. These words are then mapped to c-dimensional vectors defining their semantic meaning in the following embedding layers. The output of the user embedding layer is the representation of all words used by a user u in the form of a matrix, and can be denoted as

$$V^u = \left[\phi\left(d_1^u\right), \ldots, \phi\left(d_l^u\right), \ldots, \phi\left(d_n^u\right)\right], \tag{4}$$

where $[\phi_1, \phi_2]$ denotes the concatenation of two vectors, d_l^u is user u's l-th word, and $\phi : D \longrightarrow \mathbb{R}^{k_w}$ is a lookup function to



(a) Textual modeling component using CNN

(b) Textual modeling component using RNN

FIGURE 2: Proposed alternatives to learn user and location representations from textual reviews. 2(a) is a CNN-based solution employed in TCENR, while 2(b) illustrates the suggested extension using RNN.

a pre-trained textual embedding layer [36, 37] that represents each word in vocabulary D as a vector in size k_w. Similarly, V^p denotes the word embedding matrix for location p.

Due to the large amount of parameters required to train the aforementioned contextual model, the textual network is implemented using a CNN-based architecture which is usually more computationally efficient than RNN. The semantic representations of users' and locations' reviews are fed to convolution layers, to detect parts of the text that best capture the review's meaning. These layers produce t feature maps over the embedded word vector, using a window size of ws and filter $K \in \mathbb{R}^{k_w \times t}$. As suggested by [14], ReLU is used as an activation function for this layer:

$$\begin{aligned} z_{jl}^u &= ReLU\left(V_{l:l+ws}^u * K_j^u + b_j^u\right), \\ z_j^u &= \left[z_{j1}^u, \ldots, z_{jl}^u, \ldots, z_{jn}^u\right], \end{aligned} \tag{5}$$

where V_l^u is user u's l-th input word embedding and z_j^u the j-th feature, extracted from the complete text.

Based on the standard CNN structure, feature maps produced by the convolution layers are reduced by a pooling layer:

$$\begin{aligned} o_j^u &= \max\left(z_j^u\right), \\ O^u &= \left[o_1^u, \ldots, o_j^u, \ldots, o_t^u\right], \end{aligned} \tag{6}$$

where max-pooling is selected to identify the most important words and their latent values. O^u is the collection of all concise features extracted from user u's textual input. These are followed by fully connected layers to jointly model the different features and result in the latent textual representations for user u and location p, respectively, denoted as h^u and h^p:

$$h^u = ReLU\left(W_1^u \times O^u + b_1^u\right). \tag{7}$$

To combine the outputs of the users and locations fully connected layers to the same feature space, a shared layer is utilized. It concatenates its two inputs and learns their interaction by employing an additional hidden layer:

$$h_{reviews} = ReLU\left(W_2 \times \left[h^u, h^p\right] + b_2\right). \tag{8}$$

The two neural networks are then finally merged to produce a prediction $\hat{y}_{up} \in [0, 1]$. The last layers of the two networks, each representing a different view of the user-location interaction, are concatenated and fed to yet another hidden layer, responsible to blend the learning:

$$\hat{y}_{up} = \sigma\left(W_3 \times \left[h_{context}, h_{reviews}\right] + b_3\right), \tag{9}$$

where the sigmoid function was selected to transform the hidden layer output to the desired range of $[0, 1]$.

3.2. Sequential Textual Modeling. To further investigate the gain achieved by integrating a textual modeling component over reviews in TCENR, we suggest an extension, denoted as $\text{TCENR}_{\text{seq}}$. Following its success in previous language modeling tasks [17, 18] and its ability to capture sentences' sequential nature, we employ an RNN component to learn latent features from reviews. An illustration of the proposed extension is presented in Figure 2(b), while the CNN method used in the vanilla TCENR is shown in Figure 2(a), to provide a convenient base for comparisons. More specifically we follow the findings of previous works [18, 23] and implement our recurrent network using GRU, an architecture that achieves competitive performance compared to LSTM, but with fewer parameters, making it more efficient:

$$\begin{aligned} f_l &= \sigma\left(W_f V_l^u + R_f h_{l-1} + b_f\right) \\ s_l &= \sigma\left(W_s V_l^u + R_s h_{l-1} + b_s\right) \end{aligned}$$

$$\begin{aligned}\widetilde{c}_l &= \tanh\left(W_{\widetilde{c}} V_l^u + f_l \odot R_{\widetilde{c}} h_{l-1} + b_{\widetilde{c}}\right)\\ h_l &= \left(1 - s_l\right) \odot h_{l-1} + s_l \odot \widetilde{c}_l,\end{aligned} \tag{10}$$

where f_l is the forget gate for input word l, s_l is the output gate, $\widetilde{c}_l$ is the new candidate state combining the current word embedding, V_l^u, with the previous hidden state, and h_l is current state for word l modeled by the output gate. $\odot$ denotes the element-wise product, and W_f, R_f, W_s, R_s, $W_{\widetilde{c}}$, and $R_{\widetilde{c}}$ are the GRU weight matrices while b_f, b_s, and $b_{\widetilde{c}}$ are the bias vectors.

Since the context of a word can be determined by other preceding and successive words or sentences, our proposed method employs a bidirectional GRU over the user embedding, V^u, and the location, V^p. Each word l's hidden state is learned by forward and backward GRU layers, denoted as $\overrightarrow{h_l^1}$ and $\overleftarrow{h_l^1}$, respectively. To learn a more concise and combined representation of a word while taking into account the context of all surrounding words, we feed the concatenation of $\overrightarrow{h_l^1}$ and $\overleftarrow{h_l^1}$ to an additional bidirectional GRU layer, such that its input for every word l is $e_l^2 = [\overrightarrow{h_l^1}, \overleftarrow{h_l^1}]$. The second recurrent unit will output n latent vectors, each is a sequentially infused representation of a word written by the target user or about the candidate item. To allow the method of textual modeling to be the only variant between TCENR and $\text{TCENR}_{\text{seq}}$ and to further reduce the number of learned parameters, all modified word vectors will be fed to the pooling and fully connected layers, originally presented in (6) and (7), respectively. This will allow us to directly determine the effect RNN has on textual modeling for POI recommender systems compared to CNN, as well as enabling the model to learn a more concise user and location representations. As in TCENR, the resulting vectors will be merged in order to learn the user-location interaction.

3.3. Training the Network. To train the recommendation models, we adopt a pointwise loss objective function, as done in [2, 3, 14], where the difference between the prediction $\widehat{y}_{up}$ and the actual value y_{up} is minimized. To address the implicit feedback nature of LBSNs, we sample a set of negative samples from the dataset, denoted as Y^-.

Due to the implicit feedback nature of the recommendation task, the algorithm's output can be considered as a binary classification problem. As the sigmoid activation function is being used over the last hidden layer, the output probability can be defined as

$$\begin{aligned}&p\left(Y, Y^- \mid E_u, E_p, V^u, V^p, \Theta_f\right)\\ &\quad= \prod_{(u,p)\in Y} \widehat{y}_{up} \prod_{u,p'\in Y^-} \left(1 - \widehat{y}_{up'}\right),\end{aligned} \tag{11}$$

where E_u and E_p are the embedding layers for users and locations, respectively. Similarly, V^u and V^p are the textual reviews embedding layers and Θ_f represents the model parameters. Taking the negative log-likelihood of p results in the binary cross-entropy loss function for the prediction portion of the model:

$$\begin{aligned}L_{pred} = &-\sum_{(u,p)\in Y\cup Y^-} y_{up} \log \widehat{y}_{up}\\ &+ \left(1 - y_{up}\right) \log\left(1 - \widehat{y}_{up}\right).\end{aligned} \tag{12}$$

As there are two more outputs in the model, the users' social network $\psi_c E_u$ and the locations' distance graph, $\psi_c E_p$, two additional loss functions are required to train the network. We follow the process done in [2], assuming two users who share the same context should have similar embeddings. This is achieved by minimizing the log-loss of the context given the instance embedding:

$$\begin{aligned}&L_{u\_context}\\ &\quad= -\sum_{(u,u_c)} \log\left(\psi_c E_u - \log \sum_{u_c'\in C_u} \exp\left(\psi_{c'} E_u\right)\right),\end{aligned} \tag{13}$$

where $\psi_c E_u$ is as defined in (1). Taking the binary class label into account prompts the following loss function, corresponding with minimizing the cross-entropy loss of user i and context c with respect to the y class label:

$$\begin{aligned}L_{u\_context} = &-I\left(y \in Y\right) \log \sigma\left(\psi_c E_u\right)\\ &- I\left(y \in Y^-\right) \log \sigma\left(-\psi_c E_u\right),\end{aligned} \tag{14}$$

where I is a function that returns 1 if y is in the given set, and 0 otherwise. The same logic is used to formulate the loss function for the POI context and will not be provided due to space limitations.

We simultaneously minimize the three loss functions L_{pred}, $L_{u\_context}$, and $L_{p\_context}$. The joint optimization improves the recommendation accuracy while enforcing similar representations for locations in close proximity and users connected in the social network. The loss functions are combined using two hyper-parameters, to weight the contextual contribution:

$$L = L_{pred} + \lambda_1 L_{u\_context} + \lambda_2 L_{p\_context}. \tag{15}$$

To optimize the combined loss function, a method of gradient descent can be adopted, and more specifically we utilize the Adaptive Moment Estimation (Adam) [38]. This optimizer automatically adjusts the learning rate and yields faster convergence than the standard gradient descent in addition to making the learning rate optimization process more efficient. In order to avoid additional overfitting when training the model, an early stopping criteria is integrated. The model parameters are initialized with Gaussian distribution, while the output layer's parameters are set to follow uniform distribution.

4. Experiments and Evaluation

4.1. Experimental Setup. To evaluate our proposed algorithm, we use Yelp's real-world dataset (https://www.yelp.com/dataset/challenge). It includes a subset of textual reviews along

with the users' friends and the businesses locations. Due to the limited resources used in the model evaluation, we chose to filter the dataset and keep only a concise subset, where all users and locations with less than 100 written reviews or less than 10 friends are removed. The filtered dataset includes 141,028 reviews and 98.08% sparsity for the rating matrix. The social and geographical graphs were constructed by random walks. 10% of the original vertices were sampled as base nodes, while 20 and 30 vertices were connected to each base node for users and locations, respectively, with a window size of 3. To build the POI graph, two locations are considered directly connected if they are up to 1 km apart.

To test our models' performance, the original data was split to training-validation-test sets by random sampling, with the respective ratios of 56%-24%-20%, resulting in 78,899 training instances. In addition, the input data was negatively sampled with 4 negative locations for every positive one.

To effectively compare our proposal with other alternatives, we adopt the same settings as applied in [2, 3]. The MLP input vectors are represented with an embedding size of 10, while two hidden layers are added on top of the merged result. Following the tower architecture, where the size of each layer is half the size of its predecessor, the numbers of hidden units are 32 and 16 for the first and second layers, respectively.

In the CNN component each word is represented by a pretrained embedding layer with 50 units, while the convolutional layer is constructed with a window size of 10 and a stride of 3. It results in 3 feature maps that are flattened after performing the max-pooling operation with a pool size of 2. The results are further modeled by a hidden layer with 32 units. Following the merge of the two hidden units, their interaction is learned using another hidden layer with 8 units. To combine the three loss functions as described in (15), we follow the results of [2] and set the hyperparameters $\lambda_1 = \lambda_2 = 0.1$. For the training phase of the model, a learning rate of 0.005 was used over 50 maximum epochs and a batch size of 512 samples.

4.2. Baselines. To evaluate our algorithm, we chose to compare it to these seven, empirically proven, frameworks:

(i) HPF [39]. Hierarchical Poisson matrix Factorization. A Bayesian framework for modeling implicit data using Poisson Factorization.

(ii) NMF [40]: Nonnegative Matrix Factorization, a CF method that takes only the rating matrix as input.

(iii) Geo-SAGE [5]: A generative method that predicts user check-ins in LBSNs using geographical data and crowd behaviors.

(iv) LCARS [6]: Location Content Aware Recommender System. A probabilistic model that exploits local preferences in LBSN and content information about POIs.

(v) NeuMF [3]. Neural Matrix Factorization. A state-of-the-art model combining MF with MLP on implicit ratings.

Table 1: Performance comparison over the Yelp dataset. Improvement of TCENR compared to each method is shown in brackets.

Model	Accuracy	MSE	Pre@10	Rec@10
HPF	0.8141	0.1800	0.5526	0.3699
	(1.69%)	(34.94%)	(18.51%)	(40.98%)
NMF	0.8222	0.1189	*0.7851*	0.3517
	(0.69%)	(1.51%)	*(-16.58%)*	(48.28%)
Geo-SAGE	0.7995	0.1807	0.2912	0.4145
	(3.55%)	(35.19%)	(124.89%)	(25.81%)
LCARS	0.8142	0.1612	0.6408	0.5127
	(1.68%)	(27.36%)	(2.2%)	(1.72%)
NeuMF	0.8273	0.1421	0.6488	*0.5586*
	(0.07%)	(17.59%)	(0.94%)	*(-6.64%)*
Pace	0.8239	0.1186	0.6406	0.5049
	(0.49%)	(1.26%)	(2.23%)	(3.29%)
DeepCoNN	0.8037	0.1454	0.5385	0.323
	(3.01%)	(19.46%)	(21.62%)	(64.46%)
TCENR	*0.8279*	0.1171	0.6549	*0.5215*
$\text{TCENR}_{\text{seq}}$	0.8273	*0.1161*	*0.6655*	0.4738
	(0.07%)	*(-0.86%)*	*(-1.59%)*	(10.07%)

(vi) PACE [2]. Preference and Context Embedding. A MLP-based framework with the addition of contextual graphs' smoothing for POI recommendation.

(vii) DeepCoNN [14]. Deep Cooperative Neural Networks. A CNN-based method that jointly learns an explicit prediction by exploiting users' and locations' natural language reviews.

For the task of evaluating our model and the baselines, we chose to apply Accuracy and Mean Square Error (MSE) over all n test samples, as well as Precision (Pre@10) and Recall (Rec@10) for the average top 10 predictions per user.

The proposed models were implemented using Keras (https://keras.io) on top of TensorFlow (https://www.tensorflow.org) backend. All experiments were conducted using Nivida GTX 1070 GPU.

4.3. Performance Evaluation. The performance of our proposed algorithms and the seven baselines is reported in Table 1, along with the improvement ratio of TCENR over each method in brackets. The presented results are based on the average of three individual executions.

As can be witnessed from the results, the proposed model, TCENR, achieves the best results overall compared to all baselines. Furthermore, it was found to significantly outperform HPF, NMF, Geo-SAGE, LCARS, Pace, and DeepCoNN for $p < 0.05$ based on a one-sided unpaired t-test in terms of accuracy and MSE. The contrasting results in terms of precision and recall compared to NeuMF suggest that TCENR offers less,but more relevant recommendations to the user. While NMF provides the best precision score compared to all methods, it underperforms in all other measures, making it a less desirable model. Taking a closer look shows that, surprisingly, NeuMF outperforms PACE in accuracy, precision and recall. This may be due to the less

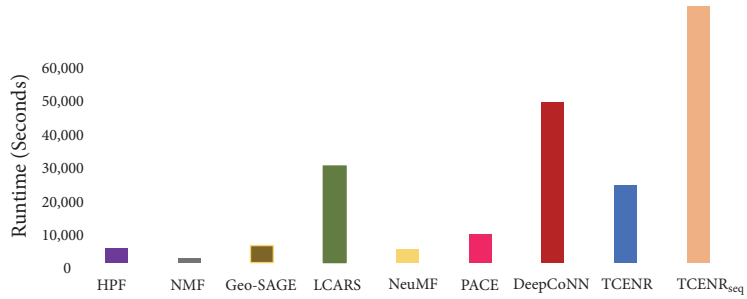


Figure 3: Runtime (seconds) of all models on the Yelp dataset.

sparse dataset tested, which does not allow the contextual regularization to be fully harvested. In addition, the use of only the first 500 words to represent the textual input for each user and location may explain the relatively low scores of the DeepCoNN model on the dataset, while the performance of Geo-SAGE and LCARS demonstrates that relying solely on geographical data does not allow such models to fully capture users' preferences in LBSNs.

Comparing TCENR and its proposed extension TCENR$_{seq}$ provides contrasting results. By employing RNN instead of CNN to extract user and location features from textual reviews, TCENR$_{seq}$ achieves lower error rate and improved precision score, while accuracy and recall are worsened. It may be considered that, by accurately capturing different aspects from user reviews, the model is able to reinforce its hypotheses and therefore reduce the uncertainty in some cases. However, when faced with a contrast between textual aspects and the ground truth, it might choose the wrong class label. Nonetheless, the results demonstrate the importance of adopting the most suitable techniques and measures to learn different data types, rather than employing a single method over all inputs. Moreover, it shows the positive impact of using textual data in conjunction to historical activities. The reported performance further suggests additional insight towards the selection of CNN and RNN for the task of language modeling in future recommendation tasks.

To further evaluate our suggested frameworks and the seven baselines in terms of runtime, the average time required to fully train each method is presented in Figure 3. As demonstrated by the results, TCENR is competitive with most baselines, and found to be more efficient than DeepCoNN and LCARS. The reported runtime of TCENR$_{seq}$ further demonstrates the relative efficiency of CNN-based solutions for textual modeling tasks. As the number of trainable parameters is increased due to the use of recurrent layers, our RNN-based extension takes 329% longer to train compared to TCENR, while achieving comparative results.

4.4. Model Design Analysis. In this section we discuss the effect of several design selections over the suggested model's performance.

4.4.1. Merge Layer. The importance of the model's final layers, responsible for combining the dense output of both the MLP and convolutional networks, requires a close attention, as it effects the networks' ability to jointly learn and the prediction itself. To properly select the fusion operator, the following methods had been considered:

(i) Combining the last hidden layers of the two models using concatenation. A model using this method will be denoted as $TCENR_{con}$ and described in (9).

(ii) Merging the last hidden layers using dot product, resulting in a model named $TCENR_{dot}$ that can be defined as

$$\hat{y}_{up} = h_{context} \cdot h_{reviews}. \tag{16}$$

(iii) Combining the two previously described methods, where the two representations will be jointly learned by concatenation and dot product. The resulted model will be denoted as $TCENR_{dot\_con}$ and can be developed by combining (9) and (16) using addition and translating the result to a range of $[0, 1]$ with the sigmoid function:

$$\hat{y}_{up} = \sigma\left(\sigma\left(W_4 \times \left[h_{context}, h_{reviews}\right] + b_4\right) + h_{context} \cdot h_{reviews}\right). \tag{17}$$

(iv) Adopting a weighted average for the prediction result of the two networks. Denoted as $TCENR_{weight}$, this model can be defined as

$$\hat{y}_{up} = \lambda_1 \sigma\left(W_5 \times h_{context} + b_5\right) + \lambda_2 \sigma\left(W_6 \times h_{reviews} + b_6\right). \tag{18}$$

As shown in Figure 4, adopting the more simple methods of weighted average and dot product leads to an inferior performance of TCENR, demonstrating the added value of utilizing the latent features learned by each subnetwork jointly. When combined with the underperforming method of dot product in $TCENR_{dot\_con}$, the use of concatenation improves over dot product alone. However, since the two methods are integrated using a simple average, employing only concatenation as done in $TCENR_{con}$ produces the best results, and therefore integrated into the final model.

4.4.2. MLP Layer Design. Although it was found by [3] that adding more layers and units to the MLP-based recommender has a positive effect, the use of CNN and the additional hidden layer suggests it is a subject worth investigating.

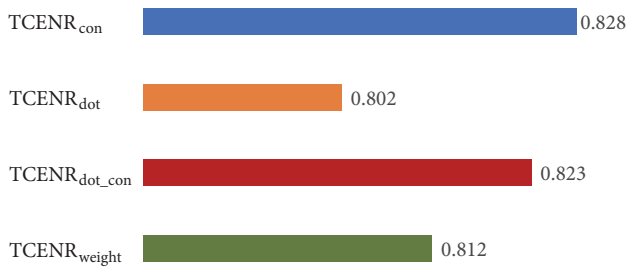


Figure 4: Comparison of merging methods in terms of accuracy.

Table 2: Model's accuracy with different layers.

1st layer	H=1	H=2	H=3	H=4
16	0.824	0.827	-	-
32	0.823	*0.837*	0.825	-
64	0.822	0.829	0.83	0.827
128	0.823	0.828	0.829	0.827

To this end, we test the proposed algorithm with 1-4 hidden layers used to learn the user-item interaction with contextual regularization in varying sizes from 8 to 128 hidden units. The results in terms of test set's accuracy are presented in Table 2, where the number of hidden layers is defined as columns and the size of the first unit is presented as rows. Unlike previous results, we find that two hidden layers with 32 and 16 hidden units result in the best performance for our dataset.

4.4.3. Number of Words. The use of written reviews in their original order allows the strengths of CNN and RNN to be exploited by finding the best representation for every few words and eventually for the whole text. Our final dataset, however, is composed of very long reviews, where to fully learn a single user or location, more than 20,000 words are required, making it computationally expensive to extract relevant representations. To benefit from the sequential nature of the written reviews while keeping the solution feasible, the number of words was limited to a range of 500-6000. As can be witnessed from Figure 5, there is a slight improvement in accuracy as the number of words increase up to 3000, while additional words result in an increased bias towards users and locations with longer reviews and in turn reduce the model's learning capabilities.

5. Conclusion and Future Work

In this paper, we developed a neural POI recommender system called TCENR. The model exploits data about users, locations, spatial data, social networks, and textual reviews to predict the implicit preference of users regarding POIs. TCENR models two types of user-location interactions: native check-ins regularized by contextual information and the words used to describe the users' experiences. We further extended our proposed method and presented $TCENR_{seq}$, where textual data was modeled using RNN instead of CNN. Evaluated over the Yelp dataset, the proposed algorithms consistently achieved superior results compared to seven state-of-the-art baselines in terms of accuracy and MSE.

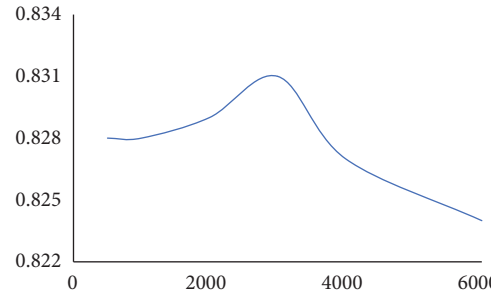


Figure 5: Number of words comparison in terms of accuracy.

For future work, we intend to extend our models' evaluation over additional LBSN datasets. In addition we plan to investigate the proposed frameworks' contribution to the cold-start problem by analyzing its performance on additional data, while taking new users and locations with few reviews into account.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China Grant (61572289) and NSERC Discovery Grants.

References

[1] H. Li, Y. Ge, R. Hong, and H. Zhu, "Point-of-interest recommendations: learning potential check-ins from friends," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery And Data Mining*, pp. 975–984, ACM, San Francisco, California, USA, August 2016.

[2] C. Yang, L. Bai, C. Zhang, Q. Yuan, and J. Han, "Bridging collaborative filtering and semi-supervised learning: a neural approach for poi recommendation," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1245–1254, ACM, Halifax, NS, Canada, August 2017.

[3] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," in *Proceedings of the 26th International Conference on World Wide Web*, International World Wide Web Conferences Steering Committee, pp. 173–182, Perth, Australia, April 2017.

[4] H. Ma, D. Zhou, C. Liu, M. R. Lyu, and I. King, "Recommender systems with social regularization," in *Proceedings of the 4th ACM International Conference on Web Search and Data Mining*, pp. 287–296, ACM, February 2011.

[5] W. Wang, H. Yin, L. Chen, Y. Sun, S. Sadiq, and X. Zhou, "Geo-sage: a geographical sparse additive generative model for spatial item recommendation," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1255–1264, ACM, Sydney, NSW, Australia, August 2015.

[6] 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*, pp. 221–229, ACM, Chicago, Illinois, USA, August 2013.

[7] C. Wang and D. M. Blei, "Collaborative topic modeling for recommending scientific articles," in *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '11)*, pp. 448–456, ACM, August 2011.

[8] H. Wang, N. Wang, and D.-Y. Yeung, "Collaborative deep learning for recommender systems," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1235–1244, ACM, Sydney, NSW, Australia, August 2015.

[9] J. Manotumruksa, C. Macdonald, and I. Ounis, "A deep recurrent collaborative filtering framework for venue recommendation," in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pp. 1429–1438, ACM, Singapore, Singapore, November 2017.

[10] Q. Liu, S. Wu, D. Wang, Z. Li, and L. Wang, "Context-aware sequential recommendation," in *Proceedings of the 2016 IEEE 16th International Conference on Data Mining (ICDM)*, pp. 1053–1058, IEEE, Barcelona, Spain, December 2016.

[11] H.-T. Cheng, L. Koc, J. Harmsen et al., "Wide & deep learning for recommender systems," in *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*, pp. 7–10, ACM, 2016.

[12] Y. Yu, L. Zhang, C. Wang, R. Gao, W. Zhao, and J. Jiang, "Neural personalized ranking via poisson factor model for item recommendation," *Complexity*, vol. 2019, Article ID 3563674, 16 pages, 2019.

[13] A. Van Den Oord, S. Dieleman, and B. Schrauwen, "Deep content-based music recommendation," in *Proceedings of the 26th International Conference on Neural Information Processing Systems*, Advances in neural information processing systems, pp. 2643–2651, 2013.

[14] 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*, pp. 425–434, ACM, Cambridge, UK, Feburary 2017.

[15] 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*, pp. 233–240, ACM, 2016.

[16] B. Hidasi, A. Karatzoglou, L. Baltrunas, and D. Tikk, "Session-based recommendations with recurrent neural networks," 2015, https://arxiv.org/abs/1511.06939.

[17] A. Almahairi, K. Kastner, K. Cho, and A. Courville, "Learning distributed representations from reviews for collaborative filtering," in *Proceedings of the 9th ACM Conference on Recommender Systems*, pp. 147–154, ACM, Vienna, Austria, September 2015.

[18] T. Bansal, D. Belanger, and A. McCallum, "Ask the gru: multi-task learning for deep text recommendations," in *Proceedings of the 10th ACM Conference on Recommender Systems*, pp. 107–114, ACM, 2016.

[19] J. Chen, W. Zhang, P. Zhang, P. Ying, K. Niu, and M. Zou, "Exploiting spatial and temporal for point of interest recommendation," *Complexity*, vol. 2018, Article ID 6928605, 16 pages, 2018.

[20] P. Zhao, X. Xu, Y. Liu, V. S. Sheng, K. Zheng, and H. Xiong, "Photo2trip: exploiting visual contents in geo-tagged photos for personalized tour recommendation," in *Proceedings of the 2017 ACM on Multimedia Conference - MM 17*, pp. 916–924, ACM Press, Mountain View, California, USA, October 2017.

[21] P. Covington, J. Adams, and E. Sargin, "Deep neural networks for youtube recommendations," in *Proceedings of the 10th ACM Conference on Recommender Systems*, pp. 191–198, ACM, 2016.

[22] C.-Y. Wu, A. Ahmed, A. Beutel, A. J. Smola, and H. Jing, "Recurrent recommender networks," in *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, pp. 495–503, ACM, 2017.

[23] A. Beutel, P. Covington, S. Jain et al., "Latent cross: making use of context in recurrent recommender systems," in *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pp. 46–54, ACM, Marina Del Rey, CA, USA, 2018.

[24] H. Yin, W. Wang, H. Wang, L. Chen, and X. Zhou, "Spatial-aware hierarchical collaborative deep learning for POI recommendation," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 11, pp. 2537–2551, 2017.

[25] H. Yin, X. Zhou, Y. Shao, H. Wang, and S. Sadiq, "Joint modeling of user check-in behaviors for point-of-interest recommendation," in *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pp. 1631–1640, ACM, Melbourne, Australia, October 2015.

[26] P. Zhao, X. Xu, Y. Liu et al., "Exploiting hierarchical structures for POI recommendation," in *Proceedings of the 2017 IEEE International Conference on Data Mining (ICDM)*, IEEE, New Orleans, LA, USA, November 2017.

[27] P. Zhao, H. Zhu, Y. Liu et al., "Where to go next: a spatio-temporal gated network for next poi recommendation," in *Proceedings of the 33rd AAAI Conference on Artificial Intelligence (AAAI 2019)*, 2019.

[28] H. Wang, F. Zhang, X. Xie, and M. Guo, "Dkn: deep knowledge-aware network for news recommendation," in *Proceedings of the 2018 World Wide Web Conference*, pp. 1835–1844, Lyon, France, April 2018.

[29] Y. Gong and Q. Zhang, "Hashtag recommendation using attention-based convolutional neural network," in *Proceedings of the 25th International Joint Conference on Artificial Intelligence, IJCAI 2016*, pp. 2782–2788, NY, USA, July 2016.

[30] Y. Tay, A. T. Luu, 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*, pp. 2309–2318, London, UK, August 2018.

[31] Z. Cheng, Y. Ding, L. Zhu, and M. Kankanhalli, "Aspect-aware latent factor model: rating prediction with ratings and reviews," in *Proceedings of the 2018 World Wide Web Conference*, pp. 639–648, Lyon, France, April 2018.

[32] D. Tang, B. Qin, T. Liu, and Y. Yang, "User modeling with neural network for review rating prediction," in *Proceedings of the 24th International Conference on Artificial Intelligence, IJCAI 2015*, pp. 1340–1346, Argentina, July 2015.

[33] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[34] Y. Kim, "Convolutional neural networks for sentence classification," in *Proceedings of the 2014 Conference on Empirical*

Methods in Natural Language Processing (EMNLP), pp. 1746–1751, Association for Computational Linguistics, Doha, Qatar, 2014, https://aclanthology.info/papers/D14-1181/d14-1181.

[35] Y. Zhang, Q. Ai, X. Chen, and W. B. Croft, "Joint representation learning for top-N recommendation with heterogeneous information sources," in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pp. 1449–1458, ACM, Singapore, Singapore, November 2017.

[36] J. Pennington, R. Socher, and C. Manning, "GloVe: global vectors for word representation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543, 2014.

[37] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proceedings of the 26th International Conference on Neural Information Processing Systems*, Advances in neural information processing systems, pp. 3111–3119, Lake Tahoe, Nevada, 2013, https://dl.acm.org/citation.cfm?id=2999959.

[38] D. P. Kingma and J. Ba, "Adam: a method for stochastic optimization," 2014, https://arxiv.org/abs/1412.6980.

[39] P. Gopalan, J. M. Hofman, and D. M. Blei, "Scalable recommendation with hierarchical Poisson factorization," in *Proceedings of the 31st Conference on Uncertainty in Artificial Intelligence, UAI 2015*, pp. 326–335, Netherlands, July 2015.

[40] X. Luo, M. Zhou, Y. Xia, and Q. Zhu, "An efficient non-negative matrix-factorization-based approach to collaborative filtering for recommender systems," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 2, pp. 1273–1284, 2014.