Exploiting Context Graph Attention for POI Recommendation in Location-based Social Networks

Siyuan Zhang and Hong Cheng

The Chinese University of Hong Kong, Hong Kong, China
syzhang@se.cuhk.edu.hk, hcheng@se.cuhk.edu.hk

Abstract. The prevalence of mobile devices and the increasing popularity of location-based social networks (LBSNs) generate a large volume of user mobility data. As a result, POI recommendation systems, which play a vital role in connecting users and POIs, have received extensive attention from both research and industry communities in the past few years. The challenges of POI recommendation come from the very sparse user check-in records with only positive feedback and how to integrate heterogeneous information of users and POIs. The state-of-the-art methods usually exploit the social influence from friends and geographical influence from neighboring POIs for recommendation. However, there are two drawbacks that hinder their performance. First, they cannot model the different degree of influence from different friends to a user. Second, they ignore the user check-ins as context information for preference modeling in the collaborative filtering framework.
To address the limitations of existing methods, we propose a *Context Graph Attention* (CGA) model, which can integrate context information encoded in different context graphs with the attention mechanism for POI recommendation. CGA first uses two context-aware attention networks to learn the influence weights of different friends and neighboring POIs respectively. At the same time, it applies a dual attention network, which considers the mutual influence of context POIs for a user and the context users for a POI, to learn the influence weights of different context vertices in the user-POI context graph. A multi-layer perceptron integrates the context vectors of users and POIs for estimating the visiting probability of a user to a POI. To the best of our knowledge, this is the first work that applies the attention mechanism for POI recommendation. Extensive experiments on two public check-in data sets show that CGA can outperform the state-of-the-art methods as well as other attentive collaborative filtering methods substantially.

Keywords: POI recommendation, context graph attention, neural network

1 Introduction

The prevalence of mobile devices stimulates user activities in location-based social networks (LBSNs), which are now becoming part of our daily life. People can

share their visited Points of Interest (POIs) with friends, or search for interesting locations when they travel to a new city on LBSNs. As a result, a large volume of check-in data have been generated in LBSNs, which give rise to the POI recommendation systems that build connections between users and POIs. Different from traditional recommendation systems designed for movies and E-commerce websites, where there are explicit ratings and rich content information for each item, the challenges for POI recommendation come from the very sparse user check-in records with only positive feedback and how to integrate heterogeneous information of users and POIs.

Existing methods for POI recommendation [18,20,11,12,15] exploit the check-ins by friends of a user and visiting patterns of neighboring POIs of a POI to alleviate the data sparsity problem. The state-of-the-art methods [20,15] are based on implicit feedback models such as weighted matrix factorization and ranking-based factorization method. However, the performance of existing methods is not satisfactory for two reasons. First, they cannot model the different degree of influence from different friends to a user. Existing methods either assign equal weights to all friends [20] or determine the influence of friends with fixed heuristic values [11,21]. They ignore the fact that friends have different influence to a user when he/she checks in at different POIs. Second, few models consider the interactions between users and POIs as context information to alleviate the data sparsity problem. If we represent the user check-ins in a context graph (according to the definition in [20], a *context graph* is a graph that describes the affinity relationship between instances), the preference of a user can be modeled by his/her check-in POIs, and the visiting pattern of a POI can be modeled by its visiting users. Incorporating such a context graph between users and items into the collaborative filtering (CF) framework has shown promising improvement on movie recommendation and multimedia recommendation [10,8,3]. The challenge in exploiting the user-POI context graph is how to filter irrelevant context vertices for both users and POIs when estimating the visiting probability of a user at a POI.

To address the limitations of existing methods, we propose a **Context Graph Attention** (**CGA**) model to fuse different context information in LBSNs for POI recommendation. CGA is inspired by the attention mechanism adopted by many neural network based learning tasks [1,17,3,19,7]. The basic assumption of the attention mechanism is that only selective parts of input features are informative for the end machine learning tasks, which is reasonable in modeling context information for POI recommendation. CGA first builds three context graphs: a user friend context graph, a POI neighborhood context graph and a user-POI context graph to encode the heterogeneous information in an LBSN. Then it represents each user and each POI in the three context graphs with attentive latent vectors, which are the weighted sum of the embeddings of relevant context vertices in different context graphs. Two types of attention networks: context-aware attention network and dual attention network are proposed to compute the weights of different context vertices in different context graphs. The final representation for users and POIs consists of their original context-free latent

vectors and the attentive latent vectors from context graphs. A multi-layer perceptron is applied on the final representations of users and POIs to estimate the visiting probabilities.

For a user, to model the influence from his/her friends, CGA computes the attentive weight from each friend using a context-aware attention network, which is a multi-layer neural network that takes characteristic of his/her friends, the target user and a candidate POI as inputs. The learned attentive weights are context aware, in the sense that they depend on the candidate POI. Similarly, for a POI, to model the influence from its neighboring POIs, CGA computes the attentive weights using another context-aware attention network, which further exploits the geographical influence measured by a distance decay function in the weight computation process. From the check-in records, CGA uses a dual attention network, which consists of two context-aware attention networks and some connections between them, to sequentially output the attentive context vectors for users and POIs. The dual attention network enables the mutual influence of two types of context vertices in the weight computation process. Thus it can further filter out irrelevant context vertices in the final attentive vectors.

In summary, we make the following contributions in this paper:

- We propose an integrated framework named Context Graph Attention (CGA) that can model heterogeneous context information with the attention mechanism in LBSNs for POI recommendation. To the best of our knowledge, this is the first attention based POI recommendation model that integrates the heterogeneous user-POI context graph for preference modeling.
- We design a context-aware attention network for the homogeneous context graph (between users and between POIs) and a dual attention network for heterogeneous context graph (between users and POIs), which can model the different and mutual influence of context vertices respectively.
- Through extensive experiments on two public check-in data sets, we show that CGA outperforms the state-of-the-art POI recommendation method as well as other attentive CF based methods for general item recommendations.

The remainder of this paper is organized as follows. We give a formal definition of our problem in Section 2. Section 3 introduces the proposed Context Graph Attention model. Section 4 describes details on the attentive weight computation and model learning. Section 5 reports the experiment results. We discuss related works in Section 6 and conclude this paper in Section 7.

2 Problem Description

Assume we have a user set $U = \{u_1, \ldots, u_N\}$ and a POI set $V = \{v_1, \ldots, v_M\}$. We use a user-POI matrix $Y \in \mathbb{R}^{N \times M}$ to denote the check-in records of all users in U. An entry $y_{ij} = 1$ if user u_i performed check-in at POI v_j. Otherwise, $y_{ij} = 0$. We denote the POIs visited by user u_i as V_i, and the set of users that performed check-in at v_j as U_j. The POI recommendation problem in an LBSN is defined as:

Definition 1. *(POI recommendation) Given a user $u_i \in U$, a POI set V, and the user-POI matrix Y, return a ranked list of POIs $\{v_j \in V \setminus V_i\}$, in descending order of the estimated check-in probability $\widehat{y}_{ij}$ by user u_i.*

3 The Context Graph Attention Model

3.1 Context Graphs in LBSN

We define three context graphs to encode the heterogeneous information in an LBSN. The *user friend context graph* captures the social relationships between users.

Definition 2. *(User friend context graph) The user friend context graph is denoted as $G_{UU} = (U, E_{UU})$, where a vertex $u_i \in U$ represents a user, and an edge $e = (u_i, u_k)$ indicates that u_i and u_k are friends. The context vertices of u_i in G_{UU} are denoted as $F_i = \{u_k | (u_i, u_k) \in E_{UU}\}$.*

The *POI neighborhood context graph* captures the geographical proximity between POIs.

Definition 3. *(POI neighborhood context graph) The POI neighborhood context graph is denoted as $G_{VV} = (V, E_{VV})$, where a vertex $v_i \in V$ represents a POI, and an edge $e = (v_i, v_j)$ indicates that v_j is one of the k-nearest neighbors of v_i. The context vertices of v_i in G_{VV} are denoted as $L_i = \{v_j | (v_i, v_j) \in E_{VV}\}$.*

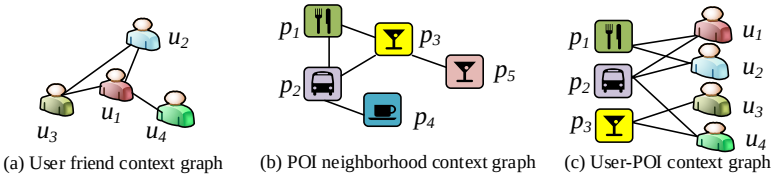
In addition, we construct a *user-POI context graph* from the user-POI matrix Y.

Definition 4. *(User-POI context graph) The user-POI context graph is a bipartite graph denoted as $G_{UV} = (U \cup V, E_{UV})$, where an edge $e = (u_i, v_j) \in E_{UV}$ if $y_{ij} = 1$ in the user-POI matrix Y. The context vertices of u_i in G_{UV} are denoted as $P_i = \{v_k | (u_i, v_k) \in E_{UV}\}$; and the context vertices of v_j in G_{UV} are denoted as $Q_j = \{u_k | (u_k, v_j) \in E_{UV}\}$.*

The first two context graphs are homogeneous, while the user-POI context graph is heterogeneous as the vertices are of different types. To the best of our knowledge, no previous work has considered the user-POI interaction as a context graph for POI recommendation. Examples of the three context graphs are shown in Fig. 1.

3.2 Attentive Representations for Users and POIs in Context Graphs

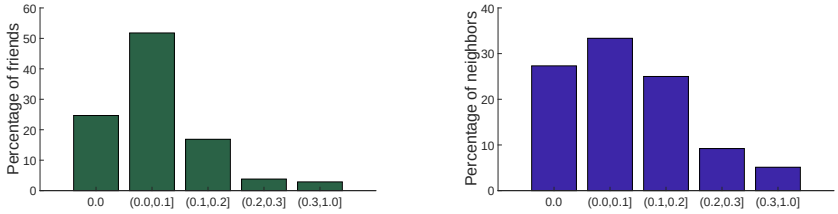
Existing methods [18,20] use graph embedding based methods to model the information in context graphs. However, they assume that vertices with similar context neighbors have similar latent representations which, however, may not hold in the real scenario. To prove this, we perform data analysis on a Gowalla data set, which contains 736,148 check-ins made in California and Nevada [4,5].



(a) User friend context graph (b) POI neighborhood context graph (c) User-POI context graph

Fig. 1. The three context graphs used in our model

In the user friend context graph G_{UU}, for each pair of friends, we calculate the cosine similarity of their check-in POI lists; in the POI neighborhood context graph G_{VV}, for each pair of neighbor POIs, we calculate the cosine similarity of the visiting user lists. We plot the cosine similarity histogram in Fig. 2. We observe that more than 20% of the friend pairs in G_{UU} have 0 cosine similarity, more than 50% have cosine similarity in (0, 0.1], and only less than 10% have cosine similarity larger than 0.2. In Fig. 2(b), more than 25% of neighbor POIs have 0 cosine similarity, more than 30% have cosine similarity in (0, 0.1], and only less than 10% have cosine similarity larger than 0.3. This analytical results show that the assumption in existing methods [18,20] may not hold.



(a) Cosine similarity between friends (b) Cosine similarity between neighbor POIs

Fig. 2. Cosine similarity between vertices in user-friend context graph and POI-neighborhood context graph

The above observation motivates us to re-consider the importance of different context vertices in the representation learning for users and POIs. We propose to learn the context representations for users and POIs in different context graphs using the attention mechanism. The *attention mechanism* was first applied in image/video recommendation [3] and neural machine translation [1], which assumes that only selective parts of the input features are informative for the end machine learning tasks. In the task of POI recommendation, applying the attention mechanism means selecting informative context vertices in the context graphs for learning an attentive representation.

Specifically, to represent u_i in G_{UU} for modeling the influence from his/her friends F_i, we first use friend embeddings $\{\boldsymbol{f}_{i,1}, \boldsymbol{f}_{i,2}, ..., \boldsymbol{f}_{i,|F_i|}\}$ to encode u_i's friend context vertices in G_{UU}. Then the representation of u_i in G_{UU} is defined as an attentive vector:

$$\widetilde{\boldsymbol{f}}_i = \sum_{u_{i,k} \in F_i} \alpha(i,k) \boldsymbol{f}_{i,k}, \tag{1}$$

where $\alpha(i,k)$ is the attentive weight of $\boldsymbol{f}_{i,k}$. We can model the influence of different friends by assigning suitable attentive weights, which are computed by a context-aware attention network. Similarly, after using neighborhood embeddings $\{\boldsymbol{l}_{j,1}, \boldsymbol{l}_{j,2}, ..., \boldsymbol{l}_{j,|L_j|}\}$ to encode context vertices in G_{VV} for v_j, we can compute an attentive vector $\widetilde{\boldsymbol{l}}_j$ to represent v_j in G_{VV} as:

$$\widetilde{\boldsymbol{l}}_j = \sum_{v_{j,k} \in L_j} \beta(j,k) \boldsymbol{l}_{j,k}, \tag{2}$$

where the attentive weight $\beta(j,k)$ is computed by another context-aware attention network.

Different from the neighborhood model in general item recommendation systems [10,8], where the preference of a user is represented by the set of rated items, we not only consider the context vertices (i.e., POIs) of a user in G_{UV}, but also consider the context vertices (i.e., users) of a POI in G_{UV} for POI recommendation. We believe that the dual context setting can further alleviate the data sparsity problem in POI recommendation. To represent u_i and v_j in G_{UV}, we first introduce two sets of context vertex embeddings: user-POI embeddings $\{\boldsymbol{p}_{i,1}, \boldsymbol{p}_{i,2}, ..., \boldsymbol{p}_{i,|P_i|}\}$ which are the embeddings of context vertices for u_i, and POI-user embeddings $\{\boldsymbol{q}_{j,1}, \boldsymbol{q}_{j,2}, ..., \boldsymbol{q}_{j,|Q_j|}\}$ which are the embeddings of context vertices for v_j in G_{UV}. Then we define two attentive vectors $\widetilde{\boldsymbol{p}}_i$ for u_i and $\widetilde{\boldsymbol{q}}_j$ for v_j to represent their context information in G_{UV} as:

$$\widetilde{\boldsymbol{p}}_i = \sum_{v_{i,k} \in P_i} \gamma(i,k) \boldsymbol{p}_{i,k}, \tag{3}$$

$$\widetilde{\boldsymbol{q}}_j = \sum_{u_{j,k} \in Q_j} \gamma(j,k) \boldsymbol{q}_{j,k}, \tag{4}$$

where $\gamma(i,k)$ and $\gamma(j,k)$ are the attentive weights for different context vertices in G_{UV}. As the context vertices of u_i and v_j can have mutual influence on each other through the edges in the context graph G_{UV}, we use a new dual attention network, which builds connections between the context-aware attention networks for two sets of context vertices P_i and Q_j, to compute the attentive weights.

Compared with the existing graph embedding based methods, which give equal weights to all context vertices [20] or using predefined, fixed edge weights [18] for different context vertices, our attentive context representation scheme enables learning the weights of different context vertices automatically to optimize the POI recommendation performance. We will describe how to compute the attentive weights from the attention networks and how to use the attentive vectors for POI recommendation in the following sections.

3.3 Solution Framework

The solution framework of our context graph attention model is shown in Fig. 3. We use the three context graphs G_{UU}, G_{VV}, and G_{UV} as inputs to our model.

To compute the visiting probability of user u_i at POI v_j, we first use the context-free user embedding $\boldsymbol{u}_i$ and POI embedding $\boldsymbol{v}_j$ to represent u_i and v_j in a latent space. Then two context-aware attention networks are applied on G_{UU} and G_{VV} to output an attentive friend vector $\widetilde{\boldsymbol{f}}_i$ for u_i and an attentive neighbor vector $\widetilde{\boldsymbol{l}}_j$ for v_j. A dual attention network is applied on G_{UV} to output an attentive user-POI vector $\widetilde{\boldsymbol{p}}_i$ for u_i and an attentive POI-user vector $\widetilde{\boldsymbol{q}}_j$ for v_j.

The final user vector of u_i is a summation on user embedding and its two attentive context vectors in G_{UU} and G_{UV}. The final POI vector of v_j is a summation on POI embedding and its two attentive context vectors in G_{VV} and G_{UV}. We then merge the user vector and the POI vector through concatenation and feed it into a multi-layer perceptron (MLP). A binary softmax layer is applied on top of the MLP layers to output the estimated visiting probability $\widehat{y}_{ij}$ of u_i at v_j. We use the cross entropy between the estimated $\widehat{y}_{ij}$ and the true entry label y_{ij} as the loss function for training parameters.

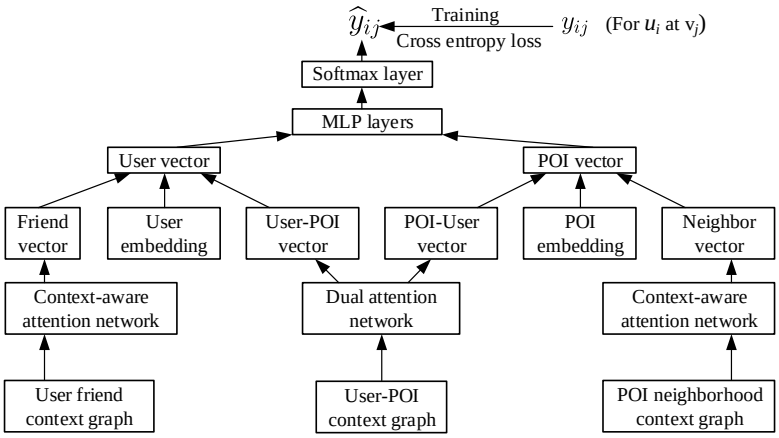


Fig. 3. The context graph attention framework

4 Attention Computation and Model Learning

In this section, we describe how to compute the attentive weights using attention networks, how to train the CGA model, and perform prediction using the model.

4.1 Context-aware Attention for Friends and Neighbors

Compute Attentive Friend Vector As we have discussed, different friends of a user can have different influence on him/her. Furthermore, even the same friend can have different influence to the user at different locations. For example, the check-in behavior of a user around a work place may be very similar to that of his colleagues, but the check-in behavior around his/her home location may be quite different from that of his colleagues. To model such different influences, we propose to use a context-aware attention network to output the friend vector

in the user context graph. Given the user embedding $\boldsymbol{u}_i$, the POI embedding $\boldsymbol{v}_j$ and the friend embeddings $\{\boldsymbol{f}_{i,1}, \boldsymbol{f}_{i,2}, ..., \boldsymbol{f}_{i,|F_i|}\}$, we compute the attentive score $a(i,k)$ for each friend embedding with a two-layer network as:

$$a(i,k) = \boldsymbol{w}_f^T \phi(\boldsymbol{W}_f[\boldsymbol{f}_{i,k}; \boldsymbol{u}_i; \boldsymbol{v}_j] + \boldsymbol{b}_f), \tag{5}$$

where we first concatenate the three embeddings $\boldsymbol{f}_{i,k}$, $\boldsymbol{u}_i$ and $\boldsymbol{v}_j$ to form a new vector. Then we transform it to another vector by multiplying kernel $\boldsymbol{W}_f$ and adding bias $\boldsymbol{b}_f$. $\phi(x) = \max(0, x)$ is the ReLU function for activation of the first layer, $\boldsymbol{w}_f$ is the parameter for the second layer. The final weights of friends are obtained by normalizing the above attentive scores using Softmax as:

$$\alpha(i,k) = \frac{\exp(a(i,k))}{\sum_{u_{i,k'} \in F_i} \exp(a(i,k'))}. \tag{6}$$

Then we compute the final context-aware attentive friend vector $\widetilde{\boldsymbol{f}}_i$ by Eq. (1). Compared with the self attention proposed in [3], where $a(i,k)$ is not related to $\boldsymbol{v}_j$ and $\widetilde{\boldsymbol{f}}_i$ is fixed for different $\boldsymbol{v}_j$, the context-aware attention network introduces more flexibility in modeling influences from friends and candidate POI v_j.

Compute Attentive Neighbor Vector Existing studies on POI recommendation [12,16,15] suggested that incorporating neighborhood information of POIs can improve the recommendation performance. In order to utilize the geographical influence weighted by distance, we compute the attentive score for each neighbor embedding $b(j,k)$ as:

$$b(j,k) = \boldsymbol{w}_l^T \phi(\boldsymbol{W}_l[\boldsymbol{l}_{j,k}; \boldsymbol{v}_j; \boldsymbol{u}_i; g(j,k)] + \boldsymbol{b}_l), \tag{7}$$

where $g(j,k) = 1/(0.5 + d(v_j, v_k))$ is a distance decay function defined in [12], $d(v_j, v_k)$ is the distance between POI v_j and its neighbor v_k. $\boldsymbol{W}_l$, $\boldsymbol{b}_l$ and $\boldsymbol{w}_l$ are the network parameters. After normalization, the context-aware neighbor attention is:

$$\beta(j,k) = \frac{\exp(b(j,k))}{\sum_{v_{j,k'} \in L_j} \exp(b(j,k'))}. \tag{8}$$

Then we compute the context-aware attentive neighbor vector $\widetilde{\boldsymbol{l}}_j$ by Eq. (2).

4.2 Dual Attention for User-POI Context Graph

We show the structure of the proposed dual attention network in Fig. 4. Given the user-POI embeddings $\{\boldsymbol{p}_{i,1}, \boldsymbol{p}_{i,2}, \ldots, \boldsymbol{p}_{i,|P_i|}\}$ and the POI-user embeddings $\{\boldsymbol{q}_{j,1}, \boldsymbol{q}_{j,2}, \ldots, \boldsymbol{q}_{j,|Q_j|}\}$ as inputs, the dual attention network first summarizes the user-POI embeddings with a user-POI summary vector through mean pooling. Then it sequentially outputs an attentive POI-user vector and an attentive user-POI vector through two context-aware attention networks. The structure of the dual attention network is consistent with our two intuitions: (1) a user would like to check in at a POI either because the POI is similar to some of his/her visited POIs or the POI has been visited by some users with similar preference; (2) the attentive weights of context users and context POIs have mutual influence.

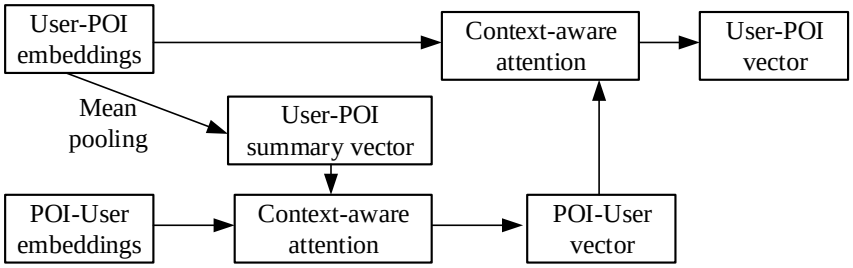


Fig. 4. The dual attention network

Context POIs Guided Context Users Attention As we focus on recommending POIs to users, we first perform mean pooling on the user-POI embeddings and obtain the user-POI summary vector $\overline{\boldsymbol{p}}_i$ as:

$$\overline{\boldsymbol{p}}_i = \frac{1}{|P_i|} \sum_{v_{i,k} \in P_i} \boldsymbol{p}_{i,k}. \tag{9}$$

Then we compute the dual attentive scores for $\boldsymbol{p}_{i,k}$ as:

$$c(j,k) = \boldsymbol{w}_q^T \phi(\boldsymbol{W}_q[\boldsymbol{q}_{j,k}; \boldsymbol{v}_j; \boldsymbol{u}_i; \overline{\boldsymbol{p}}_i] + \boldsymbol{b}_q), \tag{10}$$

where $\boldsymbol{W}_q$ and $\boldsymbol{b}_q$ are the parameters for the first layer neural network and $\boldsymbol{w}_q$ is the parameter for the second layer. Compared with the context-aware attention for friends and neighbors in Section 4.1, adding user-POI summary vector $\overline{\boldsymbol{p}}_i$ in Eq. (10) helps to filter out context users of v_j that have no overlapping with context POIs of u_i in G_{UV}. After normalization, the attention for context users of v_j is:

$$\gamma(j,k) = \frac{\exp(c(j,k))}{\sum_{u_{j,k'} \in Q_j} \exp(c(j,k'))}. \tag{11}$$

The dual attention POI-user context vector $\widetilde{\boldsymbol{q}}_j$ is computed by Eq. (4).

Context Users Guided Context POIs Attention After generating the dual-attention POI-user context vector $\widetilde{\boldsymbol{q}_j}$, we use it to guide the attention of context POIs for u_i as follows:

$$c(i,k) = \boldsymbol{w}_p^T \phi(\boldsymbol{W}_p[\boldsymbol{p}_{i,k}; \boldsymbol{u}_i; \boldsymbol{v}_j; \widetilde{\boldsymbol{q}}_j] + \boldsymbol{b}_p), \tag{12}$$

$$\gamma(i,k) = \frac{\exp(c(i,k))}{\sum_{v_{i,k'} \in P_i} \exp(c(i,k'))}, \tag{13}$$

where $\boldsymbol{W}_p$ and $\boldsymbol{b}_p$ are the first layer parameters, $\boldsymbol{w}_p$ is the second layer parameter. Incorporating attentive POI-user vector $\widetilde{\boldsymbol{q}}_j$ can help to select more informative visited POIs for characterizing u_i's intention when performing check-in at v_j. Since the attentive POI-user vector $\widetilde{\boldsymbol{q}}_j$ is influenced by the user-POI embedding summary vector $\overline{\boldsymbol{p}}_i$, the dual attention layer enables the mutual influence between $\{\boldsymbol{p}_{i,*}\}$ and $\{\boldsymbol{q}_{j,*}\}$. The final dual attention user-POI context vector $\widetilde{\boldsymbol{p}}_i$ is computed by Eq. (3).

4.3 POI Visiting Probability Prediction

Given the user embedding $\boldsymbol{u}_i$, the attentive context vectors $\widetilde{\boldsymbol{f}}_i$ and $\widetilde{\boldsymbol{p}}_i$, the final user vector can be represented through summation: $\boldsymbol{uvec}_i = \boldsymbol{u}_i + \widetilde{\boldsymbol{f}}_i + \widetilde{\boldsymbol{p}}_i$. Similarly, the final POI vector is: $\boldsymbol{vvec}_j = \boldsymbol{v}_j + \widetilde{\boldsymbol{l}}_j + \widetilde{\boldsymbol{q}}_j$. To model the non-linear interactions between $\boldsymbol{uvec}_i$ and $\boldsymbol{vvec}_j$, we first merge $\boldsymbol{uvec}_i$ and $\boldsymbol{vvec}_j$ through concatenation. Then we feed the new vector $[\boldsymbol{uvec}_i; \boldsymbol{vvec}_j]$ into a multi-layer perceptron (MLP) as:

$$\begin{aligned}
&\boldsymbol{z}_1 = \phi(\boldsymbol{W}_1[\boldsymbol{uvec}_i; \boldsymbol{vvec}_j] + \boldsymbol{b}_1),\\
&\boldsymbol{z}_2 = \phi(\boldsymbol{W}_2\boldsymbol{z}_1 + \boldsymbol{b}_2),\\
&.....\\
&\boldsymbol{z}_H = \phi(\boldsymbol{W}_H\boldsymbol{z}_{H-1} + \boldsymbol{b}_H),\\
&\widehat{y}_{ij} = \sigma(\boldsymbol{w}_y^T\boldsymbol{z}_H),
\end{aligned} \tag{14}$$

where $\boldsymbol{W}_h$ and $\boldsymbol{b}_h$ are the kernel and bias for the h-th layer, $\phi(x) = \max(0, x)$ is the ReLU function for activation. After H layers' non-linear transformation, we use the sigmoid function $\sigma(x) = 1/(1 + e^{-x})$ to output the final predicted visiting probability $\widehat{y}_{ij}$, where $\boldsymbol{w}_y$ is the parameter in $\sigma(x)$.

4.4 Model Training

Since the user-POI matrix Y is sparse with only positive feedback $Y^+ = \{y_{ij}|y_{ij} = 1\}$, we randomly sample some unobserved entries from $Y^n = \{y_{ij}|y_{ij} = 0\}$ to form the negative instance set Y^- for training a binary classifier. The number of negative instances $|Y^-|$ is proportional to the positive instance number $|Y^+|$. We use the *binary cross-entropy loss* as the loss function for training,

$$\begin{aligned}
\mathcal{J} &= -\log p(Y^+ \cup Y^-|\Theta)\\
&= -\sum_{y_{ij}\in Y^+} \log \widehat{y}_{ij} - \sum_{y_{ij}\in Y^-} \log(1 - \widehat{y}_{ij})\\
&= -\sum_{y_{ij}\in Y^+\cup Y^-} y_{ij}\log \widehat{y}_{ij} + (1 - y_{ij})\log(1 - \widehat{y}_{ij})
\end{aligned} \tag{15}$$

The parameters of our model are $\Theta = \{\Theta_e, \Theta_a, \Theta_p\}$, where $\Theta_e = \{\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{f}, \boldsymbol{l}, \boldsymbol{p}, \boldsymbol{q}\}$ are the parameters for embeddings, $\Theta_a = \{\boldsymbol{W}_*, \boldsymbol{w}_*, \boldsymbol{b}_*|* = f, l, p, q\}$ are the parameters for attentions, $\Theta_p = \{\boldsymbol{w}_y, \boldsymbol{W}_h, \boldsymbol{b}_h|h = 1, 2, ..., H\}$ are the parameters for MLP. The CGA model can be trained efficiently by stochastic gradient descent (SGD) with minibatch Adam [9].

5 Experiments

In this section, we conduct experiments on two public check-in data sets to evaluate the performance of our proposed POI recommendation method.

5.1 Settings

Data Sets The first is a Gowalla data set that contains 736,148 check-ins made in California and Nevada between Feb. 2009 and Oct. 2010 [4,5]. The second is a Foursquare data set made by 4,163 users living in California [18]. Both data sets contain friendship information between users. After filtering users with less than 15 visited POIs and POIs with less than 10 visited users, we get 3,172 users, 5,665 POIs and 132,634 check-ins in Gowalla. The Foursquare data set contains 1,513 users, 2,686 POIs and 55,554 check-ins after applying the same filtering condition. For each data set, we use the earliest 80% visited POIs of each user for training and the remaining 20% for testing.

Evaluation Metrics Three metrics are used to evaluate the performance of different POI recommendation algorithms: Pre@K (precision), Rec@K (recall) and hit rate. Given the top-K returned POIs for a user, we compute the precision and recall. We do the POI recommendation for a set of test users and report the average precision and recall for all test users. To compute the hit rate, we follow the strategy in [6,20] of mixing each ground truth POI in the test data with 100 random POIs that are not visited by the user and performing ranking with different algorithms. Then we check whether the ground truth POI is in the top-K list and report the average HR@K (hit rate) values for all test POIs.

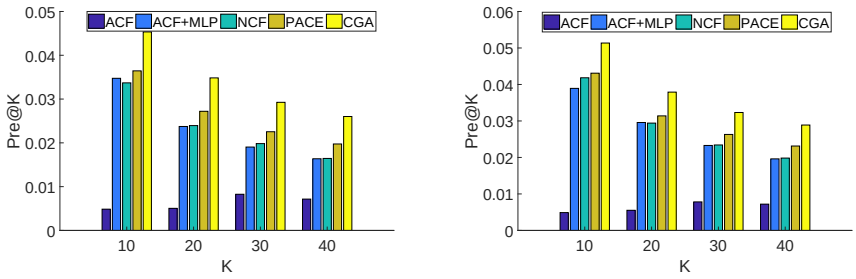
Baseline Methods We denote our method as Context Graph Attention (**CGA**) and compare CGA with the following baselines.

- **PACE** [20]: the state-of-the-art method for POI recommendation. PACE combines MLP and context graph embedding for jointly training user embeddings and POI embeddings. The context graph embedding component is considered as the regularizer for alleviating the data sparsity problem. Since it outperforms many matrix factorization based methods such as **IRenMF** [16], **ASMF** and **ARMF** [11], we do not compare with those methods in experiments.
- **NCF** [6]: it combines MLP and general matrix factorization to model the interaction between users and items in two different latent spaces. It achieves the state-of-the-art performance on movie recommendation and picture recommendation. But it has not been tested on POI recommendation in previous works.
- **ACF** [3]: the first work that applies the attention mechanism on multimedia recommendation for pictures and short videos. We only use the item level attention component in ACF as POIs have no rich component features as pictures and short videos. Since the original ACF uses dot product to compute the ranking score between users and items, we further feed the representations of users and items produced by ACF into an MLP to output the ranking score. We denote the new variant as **ACF+MLP**.

Parameter Settings For all the compared methods, we set the embedding size $D = 16$. The number of layers H in MLP is 3 and the size of MLP layer is $32 \rightarrow 16 \rightarrow 8$. For the attention network in ACF (ACF+MLP) and CGA, the size of attention unit d is the same as the embedding size. We choose $k = 30$ (k-nearest neighbors) in the construction of POI context graph. To train the neural network, we set the batch size to 512 and learning rate to 0.0001. For each positive entry in Y^+, we sample 5 negative instances to form Y^-.

5.2 Overall Performance

We report the Pre@K, Rec@K and HR@K of different methods on the two data sets in Fig. 5, Fig. 6 and Fig. 7, respectively. We observe that: (1) Our method CGA outperforms all the other methods on the three metrics. Compared with the second best method PACE, CGA increases the precision by 28.5%, recall by 28.9% and hit rate by 35.9% on Gowalla data set. The improvements on the Foursquare data set are 21.9% on precision, 26.3% on recall and 28.1% on hit rate. The results demonstrate the benefits of exploiting context graph attention for POI recommendation. (2) PACE has better performance than NCF, which shows the effectiveness of exploiting user and POI context graphs. (3) The performance of ACF+MLP is close to that of NCF and is worse than that of PACE, which shows that only applying the item level attention for POI recommendation is not enough due to data sparsity in check-in records. (4) ACF+MLP outperforms the original ACF, which shows that the MLP can improve the performance of the matrix factorization method with the attention mechanism.

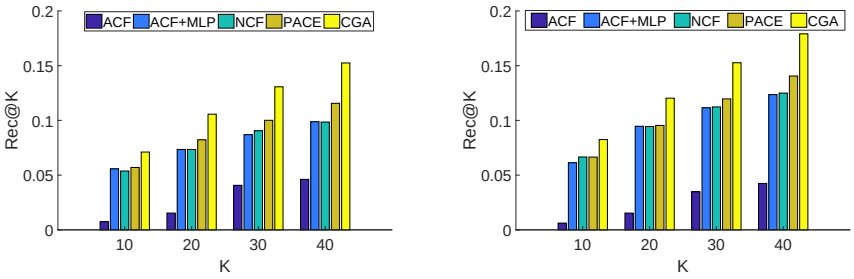


(a) Precision on Gowalla (b) Precision on Foursquare

Fig. 5. Precision on two data sets

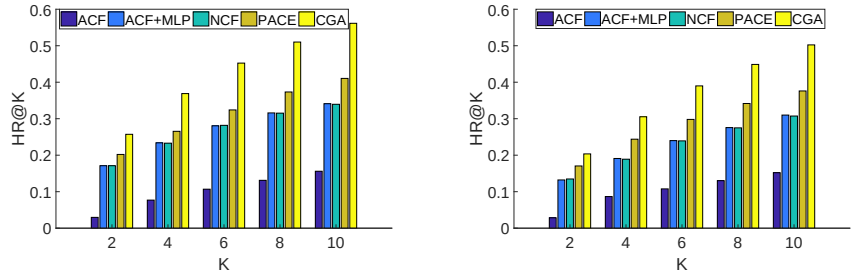
5.3 Effectiveness of Different Attention Networks

To show the effectiveness of different attention networks, we create two variants of CGA as follows:



(a) Recall on Gowalla (b) Recall on Foursquare

Fig. 6. Recall on two data sets



(a) Hit rate on Gowalla (b) Hit rate on Foursquare

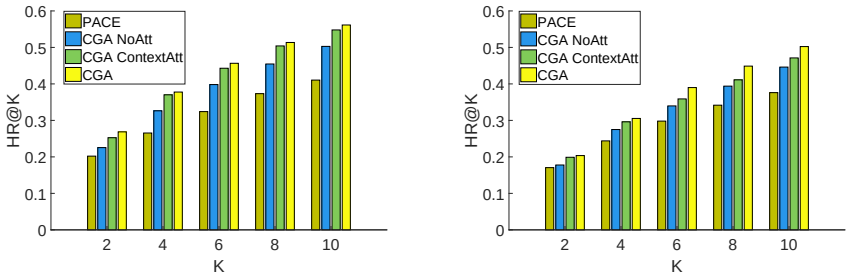
Fig. 7. Hit rate on two data sets

- **CGA_NoAtt**: it performs mean-pooling on all the context vertex embeddings in the three context graphs for both users and POIs. All context vertices in context graphs have equal weights in the final context vectors for both users and POIs, i.e., without learning the attention weights.
- **CGA_ContextAtt**: it performs context-aware attention on all the context vertex embeddings in the three context graphs. Specifically, it ignores $\overline{\boldsymbol{p}}_i$ in Eq. (10) and $\widetilde{\boldsymbol{q}}_j$ in Eq. (12). Thus there is no mutual influence on computing the attentive user-POI vector and POI-user vector.

The results on the two data sets are shown in Fig. 8. Three observations are made: (1) CGA_NoAtt has a higher hit rate than PACE, which shows the benefits of adding the user-POI context graph for POI recommendation; (2) CGA_ContextAtt outperforms CGA_NoAtt, which shows the effectiveness of applying the attention mechanism on modeling context graph information for users and POIs; (3) CGA has the best performance among all variants. This is because the proposed dual attention network can exploit the mutual influence of context vertices for users and POIs in the user-POI context graph.

5.4 Parameter Sensitivity Test

We evaluate the performance of CGA w.r.t. four parameters: the size of the embedding D, the size of the attention units d, the number of layers H in MLP



(a) Hit rate on Gowalla (b) Hit rate on Foursquare

Fig. 8. Hit rate on two data sets for different attention networks

and the number of neighbors k in POI neighborhood graph. The results on HR@10 are reported for both data sets. We make the following observations: (1) Small value of D (e.g., D=8) decreases the performance of CGA, while the HR@10 becomes stable when D is larger than 16 as shown in Table 1(a). (2) The HR@10 increases when d varies from 8 to 64, but the increase is marginal as shown in Table 1(b); (3) Table 1(c) shows that MLP with $H = 3$ has significant performance improvement compared with $H = 0$, which verifies the effectiveness of using MLP to model the interaction between users and POIs; (4) The number of neighbors k has marginal influence on the performance of CGA as shown in Table 1(d). In summary, the performance of CGA depends on the size of embeddings D and the number of layers H, but is not sensitive to the size of attention units d and the number of neighbors k.

Table 1. HR@10 w.r.t. different parameters in CGA

(a) HR@10 w.r.t. D

Data set	D=8	D=16	D=32	D=64
Gowalla	0.557	0.565	**0.591**	0.582
Foursquare	0.465	**0.504**	0.502	0.503

(b) HR@10 w.r.t. d

Data set	d=8	d=16	d=32	d=64
Gowalla	0.562	0.566	**0.574**	**0.574**
Foursquare	0.475	0.482	0.488	**0.506**

(c) HR@10 w.r.t. H

Data set	H=0	H=1	H=2	H=3
Gowalla	0.504	0.530	0.550	**0.581**
Foursquare	0.406	0.457	0.460	**0.492**

(d) HR@10 w.r.t. k

Data set	K=10	K=30	K=50	K=70	K=90
Gowalla	0.567	0.566	0.567	**0.582**	0.581
Foursquare	0.487	0.484	0.479	0.473	**0.492**

6 Related Works

6.1 POI Recommendation

In order to alleviate the data sparsity problem in check-in records and exploit rich context in LBSN, many methods for POI recommendation have been proposed [2,15], most of which are based on a fused model that extends the matrix factorization methods with context embeddings [16,12,11,13,18,14]. In [15], four types of POI recommendation models, including matrix factorization models, poisson factor models, link-based models and hybrid models, are compared by

extensive experiments on three public data sets. The experiment results show that models [16,12,13] based on implicit feedback and incorporating geographical influence have better performance than other types of models. Recently, a neural embedding method [20] that bridges collaborative filtering and semi-supervised learning with context graphs has been proposed, which outperforms the state-of-the-art matrix factorization based methods such as [11,16]. However, these context embedding methods either ignore the influence of different context vertices or set the weights of different context vertices with fixed and heuristic values. Our proposed method exploits the attention mechanism to overcome these limitations and further incorporates the user-POI context graph to improve the POI recommendation performance.

6.2 Attention Mechanism

The attention mechanism [1,17,3,19,7] has been successfully applied in many machine learning tasks such as neural machine translation [1], image/video recommendation [3] and hashtag recommendation [17]. [3] proposes a two-level self attention network to model the item-level and component-level attention for multimedia recommendation. [17] uses a co-attention network to jointly model the mutual influence of image and text information in microblog. Our proposed context graph attention model first extends the self attention network in [3] to context-aware attention network for modeling the different influence from different context vertices. Then we further extend the context-aware attention network to dual attention network inspired by the idea of co-attention network [17]. We also show that incorporating deep neural network structure like MLP can further improve the performance of attention-based collaborative filtering model for POI recommendation.

7 Conclusions

In this paper, we proposed a context graph attention (CGA) model for POI recommendation in LBSNs. CGA first models the different influence of friends and POI neighbors with a context-aware attention network. Then it uses a dual attention network to model the context information in the user-POI context graph. The learned attentive representation for different context graphs can be fed into a multi-layer perceptron for modeling non-linear interaction between users and POIs. Extensive experimental results on two public check-in data sets show that the context attention model can outperform the state-of-the-art method for POI recommendation and other attentive collaborative filtering methods by a large margin.

References

1. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. In: ICLR (2015)
2. Bao, J., Zheng, Y., Wilkie, D., Mokbel, M.: Recommendations in location-based social networks: a survey. GeoInformatica 19(3), 525–565 (2015)

3. Chen, J., Zhang, H., He, X., Nie, L., Liu, W., Chua, T.S.: Attentive collaborative filtering: Multimedia recommendation with item-and component-level attention. In: SIGIR. pp. 335–344 (2017)
4. Cho, E., Myers, S.A., Leskovec, J.: Friendship and mobility: user movement in location-based social networks. In: SIGKDD. pp. 1082–1090 (2011)
5. Feng, S., Li, X., Zeng, Y., Cong, G., Chee, Y.M., Yuan, Q.: Personalized ranking metric embedding for next new poi recommendation. In: IJCAI. pp. 2069–2075 (2015)
6. He, X., Liao, L., Zhang, H., Nie, L., Hu, X., Chua, T.S.: Neural collaborative filtering. In: WWW. pp. 173–182 (2017)
7. Jun Xiao, Hao Ye, X.H.H.Z.F.W.T.S.C.: Attentional factorization machines: Learning the weight of feature interactions via attention networks. In: IJCAI. pp. 3119–3125 (2017)
8. Kabbur, S., Ning, X., Karypis, G.: Fism: factored item similarity models for top-n recommender systems. In: SIGKDD. pp. 659–667 (2013)
9. Kingma, D., Ba, J.: Adam: A method for stochastic optimization. In: ICLR (2015)
10. Koren, Y.: Factorization meets the neighborhood: a multifaceted collaborative filtering model. In: SIGKDD. pp. 426–434 (2008)
11. Li, H., Ge, Y., Hong, R., Zhu, H.: Point-of-interest recommendations: Learning potential check-ins from friends. In: SIGKDD. pp. 975–984 (2016)
12. Li, X., Cong, G., Li, X.L., Pham, T.A.N., Krishnaswamy, S.: Rank-geofm: A ranking based geographical factorization method for point of interest recommendation. In: SIGIR. pp. 433–442 (2015)
13. Lian, D., Zhao, C., Xie, X., Sun, G., Chen, E., Rui, Y.: Geomf: joint geographical modeling and matrix factorization for point-of-interest recommendation. In: SIGKDD. pp. 831–840 (2014)
14. Liu, X., Liu, Y., Li, X.: Exploring the context of locations for personalized location recommendations. In: IJCAI. pp. 1188–1194 (2016)
15. Liu, Y., Pham, T.A.N., Cong, G., Yuan, Q.: An experimental evaluation of point-of-interest recommendation in location-based social networks. VLDB 10(10), 1010–1021 (2017)
16. Liu, Y., Wei, W., Sun, A., Miao, C.: Exploiting geographical neighborhood characteristics for location recommendation. In: CIKM. pp. 739–748 (2014)
17. Qi Zhang, Jiawen Wang, H.H.X.H.Y.G.: Hashtag recommendation for multimodal microblog using co-attention network. In: IJCAI. pp. 3420–3426 (2017)
18. Xie, M., Yin, H., Wang, H., Xu, F., Chen, W., Wang, S.: Learning graph-based poi embedding for location-based recommendation. In: CIKM. pp. 15–24 (2016)
19. Xiong, C., Callan, J., Liu, T.Y.: Word-entity duet representations for document ranking. In: SIGIR. pp. 763–772 (2017)
20. Yang, C., Bai, L., Zhang, C., Yuan, Q., Han, J.: Bridging collaborative filtering and semi-supervised learning: A neural approach for poi recommendation. In: SIGKDD. pp. 1245–1254 (2017)
21. Ye, M., Yin, P., Lee, W.C., Lee, D.L.: Exploiting geographical influence for collaborative point-of-interest recommendation. In: SIGIR. pp. 325–334 (2011)