Semantic-aware Spatio-temporal App Usage Representation via Graph Convolutional Network

YUE YU*, TONG XIA*, HUANDONG WANG, JIE FENG, and YONG LI, Beijing National Research Center for Information Science and Technology, Department of Electronic Engineering, Tsinghua University, China

Recent years have witnessed a rapid proliferation of personalized mobile Apps, which poses a pressing need for user experience improvement. A promising solution is to model App usage by learning semantic-aware App usage representations which can capture the relation among time, locations and Apps. However, it is non-trivial due to the complexity, dynamics, and heterogeneity characteristics of App usage. To smooth over these obstacles and achieve the goal, we propose SA-GCN, a novel representation learning model to map Apps, location, and time units into dense embedding vectors considering spatio-temporal characteristics and unit properties simultaneously. To handle *complexity* and *dynamics*, we build an App usage graph by regarding App, time, and location units as nodes and their co-occurrence relations as edges. For *heterogeneity*, we develop a Graph Convolutional Network with meta path-based objective function to combine the structure of the graph and the attribute of units into the semantic-aware representations. We evaluate the performance of SA-GCN via a large-scale real-world dataset. In-depth analysis shows that SA-GCN can model the complex relationships among different units and recover meaningful spatio-temporal patterns. Moreover, we make use of the learned representations in App usage prediction task without post-training and user profile prediction task with a standard classifier, which achieves notable performance gain of 8.3% compared with state-of-the-art baselines.

CCS Concepts: • **Information systems** → **Spatial-temporal systems**; **Data mining**; • **Human-centered computing** → *Empirical studies in ubiquitous and mobile computing*; • **Computing methodologies** → Machine learning.

Additional Key Words and Phrases: Graph Convolutional Network, Representation Learning, App Usage Modeling

ACM Reference Format:
Yue Yu, Tong Xia, Huandong Wang, Jie Feng, and Yong Li. 2020. Semantic-aware Spatio-temporal App Usage Representation via Graph Convolutional Network. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 3, 1, Article 30 (September 2020), 24 pages. https://doi.org/0000001.0000001

1 INTRODUCTION

With the prevalence of smartphones as well as the Internet services, the past few years have witnessed the tremendous expansion of the mobile application (*Abbr.* Apps). For the first quarter of 2019, there are around 2.1 million Apps in Google's Android market and 1.8 million available Apps in Apple's App Store[1]. With such

*Both authors contributed equally to this research.
[1]https://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/

This work was supported in part by The National Key Research and Development Program of China under grant 2018YFB1800804, the National Nature Science Foundation of China under U1936217, 61971267, 61972223, 61941117, 61861136003, Beijing Natural Science Foundation under L182038, Beijing National Research Center for Information Science and Technology under 20031887521, and research fund of Tsinghua University - Tencent Joint Laboratory for Internet Innovation Technology.
Authors' address: Yue Yu; Tong Xia; Huandong Wang; Jie Feng; Yong Li, liyong07@tsinghua.edu.cn, Beijing National Research Center for Information Science and Technology, Department of Electronic Engineering, Tsinghua University, Beijing, 100084, China.

2474-9567/2020/9-ART30
https://doi.org/0000001.0000001

a huge number of widely-used Apps, it becomes increasingly important to model users' spatio-temporal App usage behaviors, i.e., when and where to use which Apps [35], for both App developers and service providers to improve user experience by context-aware recommendations [62], personalized resource allocation [35], etc. Under this circumstance, we aim to learn semantic-aware App usage representation, which can reveal intrinsic relationships among App, location and time, to facilitate extensive downstream application tasks.

Although learning semantic-aware representation for App usage modeling is crucial, it is not trivial due to the following challenges. First, App usage is *dynamic* and *complex*, as users may use various Apps in different time and locations. For example, a civil servant is more likely to use productivity Apps during the daytime of weekdays in the office but prefers entertaining Apps during the night of weekdays and daytime of weekends at home. Modeling such dynamics requests us to take spatio-temporal context into consideration. Second, spatio-temporal App usage information is *heterogeneous* as different kinds of units (e.g. Apps, times and locations) have different attributes. Apps can be divided into several categories (e.g., *Entertainment, Office*) and they are generally used in different spatio-temporal scenarios. Meanwhile, time units would be working time or non-working time, and location units would have different functions with unique PoI (Points of Interest) distribution. These all make App usage diverse, while existing works, which learn the representation by end-to-end prediction tasks [19, 31, 61] or extract the relation among each two kinds of units separately [5], fail to deal with the complexity, dynamics, and heterogeneity of App usage simultaneously.

To solve these challenges, we propose SA-GCN[2] to project App, location, and time units into the same and low-dimensional embedding vectors where their semantic similarities remain. For the *complexity* and *dynamics*, we build an App usage graph by regarding App, time and location units as nodes and their co-occurrence relations as edges. Based on this graph, we learn the representation for each node. Specifically, to tackle the second challenge of *heterogeneity*, we adopt Graph Convolutional Network (GCN) [16] to generate the node representation. Different from traditional graph embedding methods [4, 8, 12, 38], GCN is able to generate the embedding of one node from its neighbors with its attributes. It allows to model graph structure and node property simultaneously. To learn the parameter of GCN, we design a non-task specific objective function, which is presented by the likelihood function of different units co-occurring in the App-location-time meta-path. By maximizing this function, we are about to obtain semantic-aware App, location, and time embeddings automatically. With the learned representations, we carry out in-depth analysis to showcase how the embedding vectors preserve the semantics in App, location and time domain. In addition, we adopt these embeddings in App usage prediction tasks to further demonstrate their superiority on downstream applications. In summary, our paper makes the following contributions:

(1) To the best of our knowledge, we are the first to build a heterogeneous graph with App, location, and time units as nodes with their attributes as node features for App usage representation. By combining the attributes and the co-occurrence relation of different units, we are able to learn semantic-aware embedding vectors for App, location, and time, respectively and simultaneously, which achieve better result.
(2) We develop a novel GCN-based model SA-GCN with non-task specific objective function to learn the representation, which captures the signal in spatio-temporal App usage patterns and the semantic information of Apps and locations, and can facilitate a wide range of downstream applications.
(3) We evaluate our proposed SA-GCN via a large-scale real-world App usage dataset. Through the in-depth analysis, we showcase the proposed model reveals intrinsic complex relationships among App, location and time units. Moreover, we adopt the learned embedding vectors in App usage prediction tasks, through which, we demonstrate the superiority of the proposed model in terms of achieving significant performance gain of 8.3% against the state-of-the-art baselines.

[2]short for **S**emantic-**A**ware representation learning model based on **G**raph **C**onvolutional **N**etwork.

2 PRELIMINARIES

2.1 Background

Our work is motivated by the recent advancement of App representation learning. Till now, there are several works along this line [5, 19, 22, 61]. However, most of them [19, 22, 61] are mainly based on App usage sequences without considering rich contextual information. As demonstrated in [25, 53], such information would provide complementary semantics to improve the quality of the prediction. Although Chen et al. [5] proposed CAP model to learn context-aware representation for App usage prediction, they use three bipartite graphs to encode App-location, App-time, and App-App co-occurrence relations separately and then design an embedding learning method to jointly optimize on three graphs. However, we find that such method cannot aggregate the contextual information well due to the following reasons. First, it only model similarities between different node pairs and fail to model spatio-temporal co-occurrence simultaneously, which leads to suboptimal results. Second, they are unable to consider the node attributes, i.e., App categories and location functions. Node attributes are another kind of features indicating specific usage patterns and are different from the edge attributes, i.e., co-occurrence. Neglecting such attribute information will reduce the semantics of these presentations and worsen the performance for downstream tasks.

Enlightened by the recent method on graph representation learning [16], we aim to model spatio-temporal correlations simultaneously for better App usage presentation by using graph convolutional networks (GCN). Graph convolutional networks have been adopted in location-based social network (LBSN) analysis [47] and semantic recommendation [46] with promising results, which justify the strong ability for GCN in combining the rich structural information and node attributes. However, these methods mainly consider user-location [47] or user-item relationships [46], but in App representation learning, there are more types of units which makes it more challenging. To address this, we designed a meta-path guided learning scheme to maximize the co-occurrence probability from the observation, which can guide the model to learn more semantic-aware relations from raw records. The detailed description of the methodology can be found in section 3.3.

2.2 Problem Definition

Inspired by the phenomenal success of representation learning, we are dedicated to designing a novel embedding algorithm to learn semantic-aware App usage representation. In our paper, *representations* are dense embedding vectors for location, time and App units. *Semantics* are the patterns that reflect the attribute information for different units on App usage [32, 59]. Specifically, location attribute refers to *PoI distribution*, i.e., the proportion of each Point of Interest category within the location unit. Time attribute indicates whether it is a working or non-working day. For App unit, the attribute is its App category. A qualified App usage representation is of low-dimension but can provide insightful knowledge about the intrinsic relation among App, location and time. Consequently, these semantic-aware App usage representations can promote a wide range of downstream applications such as prediction and recommendation.

Motivated by this, we formulate the investigated problem as follows: Let $\mathcal{D} = \{(u_i, t_i, l_i, a_i)\}_{i=1}^{|\mathcal{D}|}$ be a corpus of the App usage records where: (1) u_i is the user ID, (2) t_i is the timestamp in hour, (3) l_i is the location ID with its PoI distribution, and (4) a_i is App ID with its category. Our problem is: given $\mathcal{D}$, we aim to generate embedding vectors for those App, time, and location units in a latent space to preserve their co-occurrence relations as well as attribute information.

3 METHOD

To deal with the dynamic and complex App usage behaviors, we transform App usage with side information into a heterogeneous graph. Based on the graph, we adopt GCN to learn semantic-aware representations, and these learned embedding vectors can be used in downstream applications such as App usage prediction directly.

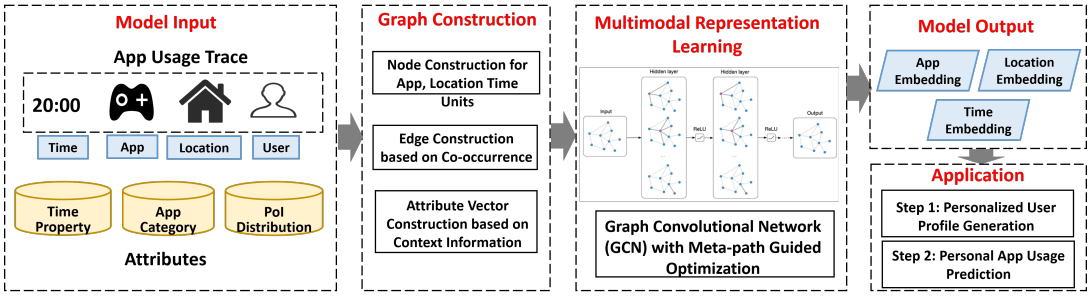


Fig. 1. The overall framework of our method. The input of the model includes App usage traces and contextual information described in section 3.2. Then, the heterogeneous graph is constructed based on the method in section 3.1. Then we use GCN to learn the representations for differnt units (section 3.3) to generate embeddings for different units. Then we use the embeddings for App usage prediction as downstream application (section 3.4).

The framework of our model is shown in Fig. 1. With the input of App usage traces and contextual information, the *graph construction* module builds the heterogeneous graph based on the App usage records. For each node in the graph, we connect them to its neighbors based on App usage records and set the weight of edge based on the co-occurrence frequency, which is able to preserve the co-occurrence relationship within the records. Then, the *feature extraction* module assigns a feature vector for each node, which enables machine learning tools to fully characterize the semantic-rich side information (e.g., App categories and PoI distributions) for nodes with different modalities and ensure the richness of information for training. After constructing the attributed graph, a GCN-based method is proposed to map the units from different modalities into the same latent space by utilizing both the co-occurrence relationships and semantic attributes. To learn the model parameters, we define a non-task specific objective function to present the co-occurrence of each existing App-location-time meta-path. Finally, in *Application* module, we leverage the learned embedding vectors in App usage prediction, which shows the superiority of our model to benefit downstream applications.

3.1 Graph Construction

In order to map the location, time, and App units into a low-dimension latent space and figure out the interior relation between them, we construct a graph to encode the connections between these units. We define the App usage graph as $\mathcal{G} = (V, E, X)$, where $V = \{v_1, v_2, \ldots\}$ and $E = \{e_1, e_2, \ldots\}(e_i \in V \times V)$ stand for the set of nodes and edges respectively and $X = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_{|V|})$ represents the feature vector of each nodes. We construct it as the three steps:

Node Construction. In the graph, each node stands for one location, time or App unit. We denote the set of apps as $\mathcal{A} = \{a_1, a_2, ..., a_M\}$, the set of locations as $\mathcal{L} = \{l_1, l_2, ..., l_L\}$ and the set of time slots as $\mathcal{T} = \{t_1, t_2, ..., t_{2N}\}$[3]. Then, the total number of nodes $|V| = M + 2N + L$ and each node v has a type with 'app', 'location' or 'time'.

Edge Construction. As each record consists of one location unit, one time unit, and one App unit, the co-occurrence relationship between different units can be induced as *three* edge types in the graph: (1) *Time-Location* edge. (2) *App-Time* edge; (3) *App-Location* edge. Specifically, to preserve the direct co-occurrence relationship among time unit t, location unit l and App unit a in the same record (u, t, l, a), there will be three edges between (l, t), (a, t) and (l, a) in the graph respectively.

Edge Weight Normalization. To effectively model the strength of different connections among units, each edge is assigned with a weight. Here we employ min-max normalization to calculate weights between nodes. We first denote the co-occurrence matrix for node i and j as C_{ij}. For each type of edges, we denote the maximum weight for *Time-Location* edge as C_m^{tl}, *App-Time* edge as C_m^{at}, and *App-Location* edge as C_m^{al} respectively. Then, the

[3]We describe how to discretize the location and time slots in section 4.1.1.

normalized weight of edge between node i and j is defined as:

$$A_{ij} = \begin{cases} C_{ij}/C_m^{tl} & (i,j) \text{ is } \textit{Time-Location} \text{ edge}, \\ C_{ij}/C_m^{at} & (i,j) \text{ is } \textit{App-Time} \text{ edge}, \\ C_{ij}/C_m^{al} & (i,j) \text{ is } \textit{App-Location} \text{ edge}, \\ 1 & i = j, \\ 0 & \text{otherwise}. \end{cases} \tag{1}$$

3.2 Feature Extraction

To capture the *semantic-rich attribute information*, each node $v \in V$ is assigned with feature vector $\boldsymbol{x}$. The feature vector $\boldsymbol{x}$ consists of three types of vector: App feature vector $\boldsymbol{a}$, location feature vector $\boldsymbol{l}$, and time feature vector $\boldsymbol{t}$. For *Apps*, the feature information is indicated by the type of the app, as apps within the same type tend to share similar usage patterns [5]. For *locations*, since each location covers the base station and its neighborhood, the PoI distribution within this region reflects its socioeconomic function [56], thus representing the attributes of this region. For *times*, we categorize the time slots as working days and non-working days since people's lifestyles differ a lot between these two types of days. Then we discretize the time in one day into N units for better modeling. Since each node possesses only one type (time/App/location), for types which are different from the node type, the corresponding feature is set to be zero vectors. The derivation of feature vectors is described as follows.

For node v_i with type *App*, let the number of App types be M. The feature vector $\boldsymbol{a}_i \in \mathbb{R}^{1\times M}$ is expressed as:

$$a_{ij} = \begin{cases} 1 & \text{if } v_i.type == App \text{ and } C_i^a == j, \\ 0 & \text{else}, \end{cases} \tag{2}$$

where C_i^a is the category of that App which node i belongs to. For nodes that are not App units, the App feature vector is just zero vector with the same size ($1 \times M$). For a node with type *Location*, we consider the category and density of nearby POIs of that location. Suppose the number of PoI types is P. Due to some types of POIs (e.g. restaurants) are more popular than others (e.g. tourist spots), we first normalize the PoI distribution to eliminate the influence of the population in different regions via computing the term frequency-inverse document frequency weights (i.e., *TF-IDF*) [23], a classical approach to reflect how important a word is to a given document for the PoI in each location. We denote the count for a location i as $\boldsymbol{q}_i = [q_{i1}, q_{i2}, \ldots, q_{iP}]$ with $\hat{q}_{ip} = \frac{q_{ip}}{\sum_{p=1}^{l} q_{ip}} \times \log \frac{|\{\boldsymbol{q}_i\}|}{|\{\boldsymbol{q}_i : q_{ip}>0\}|}$, $(\forall i = 1, 2, ..., L, p = 1, 2, ..., P)$. For each location, we also take the spatial neighborhood relationship into account, since two near locations should be considered as correlated rather than independent. To build the spatial adjacency matrix W, we compute the pairwise connection between locations and the weight is calculated using thresholded Gaussian kernel [36] as:

$$W_{ij} = \begin{cases} \exp\left(-\frac{\text{dist}(v_i,v_j)^2}{\sigma^2}\right) & \text{if } \text{dist}\left(v_i, v_j\right) \le \kappa\,, \\ 0 \quad \text{else}\,. \end{cases} \tag{3}$$

Then, for location i, the neighborhood PoI distribution $\boldsymbol{r}_i$ is calculated as $\boldsymbol{r}_i = \frac{\sum_{j=1}^{L} W_{ij} \cdot \hat{\boldsymbol{q}}_i}{\sum_{j=1}^{L} W_{ij}}$.

To sum up, for node v_i, the location attribute vector $\boldsymbol{l}_i \in \mathbb{R}^{1\times(2P)}$ is ultimately derived as:

$$\boldsymbol{l}_i = \begin{cases} \text{CONCAT } [\hat{\boldsymbol{q}}_j, \boldsymbol{r}_j] & \text{if } v.type == Location \text{ and } C_i^l == j, \\ 0 & \text{else}, \end{cases} \tag{4}$$

where C_i^l is the ID of the location which node i belongs to. For node with type *Time*, since we classify them to the working and non-working day, we use index $[1, N]$ for the former and $[N+1, 2N]$ for the latter. Then, for node

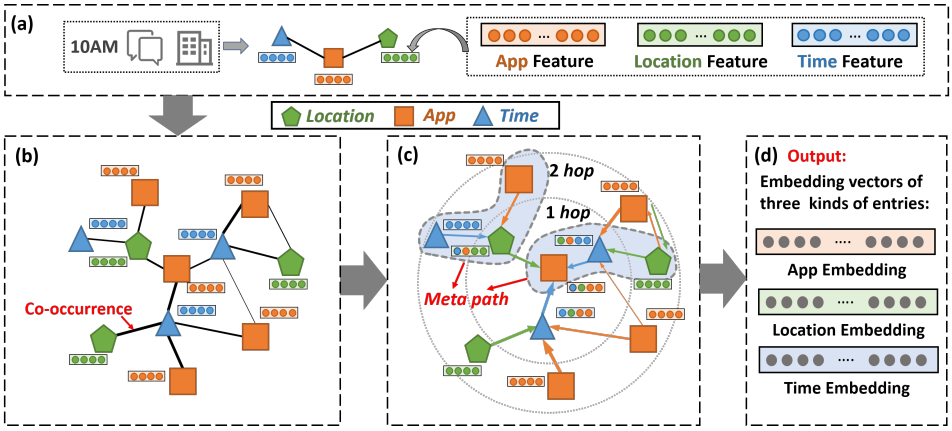


Fig. 2. Diagram of our representation learning model SA-GCN, where (a) is the input to our model, (b) is the co-occurrence graph constructed after Section 3.1, (c) is the representation learning process, where each node's embedding is updated by the weighted sum of itself and its neighbors, and (d) is the output of our model, which will be adopted to further tasks.

v_i, the feature vector $\boldsymbol{t}_i \in \mathbb{R}^{1\times(2N)}$ is a one-hot vector that represents the time slot information, which can be expressed as

$$t_i = \begin{cases} 1 & \text{if } v.type == Time \text{ and } C_i^t == j, \\ 0 & \text{else,} \end{cases} \tag{5}$$

where C_i^t is the time slot the node i belongs to. To ensure all nodes have the same dimension of the feature vector, we concatenate Eqn. 2, 4, 5 and obtain the overall feature vector $\boldsymbol{x} \in \mathbb{R}^{1\times D_i}$ for each node as Eqn. 6 shows, where $D_i = M + 2 \times P + 2 \times N$.

$$\boldsymbol{x} = \text{CONCAT } [\boldsymbol{a}, \boldsymbol{l}, \boldsymbol{t}]. \tag{6}$$

Take all nodes in $\mathcal{G}$ for consideration, the feature matrix X in our model has the shape of $(M + L + 2N) \times D_i$.

3.3 Representation Learning

Based on the built graph, the representation learning method is confronted with several challenges: (*a*) How to *preserve the semantic relations among multimodal units* i.e. time, location and App units. For example, there is a record with a time unit t, a location unit l and an App unit a, their co-occurrences would imply the intrinsic relationship among them. (*b*) How to *capture the similarities and differences for units with the same type.* For example, apps with the same category tend to have a similar embedding since their usage time and location are likely to be similar, while apps that belong to different categories tend to have different usage patterns. To this end, we proposed our SA-GCN model, short for **S**emantic-**A**ware representation learning via **G**raph **C**onvolutional **N**etwork, which is illustrated in Fig. 2.

3.3.1 Propagation via Graph Convolutional Network. Graph Convolutional Network (GCN) [16] is a kind of deep learning models over the graph data, which is based on a first-order Approximation of spectral convolutions on graphs. Let $\Theta^{(k)} = (\boldsymbol{\theta}_1^{(k)}, \boldsymbol{\theta}_2^{(k)}, \ldots, \boldsymbol{\theta}_{|V|}^{(k)})$ be the matrix of all node embedding vectors at step k, the aggregation function is calculated as:

$$\Theta^{(k)} = \sigma(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}\Theta^{(k-1)}W^{(k)}), \tag{7}$$

where $\tilde{A} = A + I_N$, and A is the adjacency matrix with I_N the identity matrix. D is a diagonal matrix with $D_{ii} = \sum_j \tilde{A}_{ij}$, σ represents the non-linear activation function. Input node features X_u serves as the initial embedding $\Theta_u^{(0)}$. $W^{(k)}$ is the trainable parameters in the k-th layer. In this way, for each node, the embedding is

updated to the weighted average of itself and its neighbors in the graph. In our problem, since the neighboorhood relationship in the graph indicates co-occurrence in records, from fig. 2 we can find that GCN model can address the challenge (*b*) above via aggregating the high-order neighborhood information for each node, which is able to utilize the contextualized information for each unit in records.

Similar to the setting in [16] where two layers of convolution are adopted, we set $k = 2$ in our SA-GCN model. In this way, the output embedding Θ for all units can be written as:

$$\Theta = \hat{A}\sigma(\hat{A}XW^{(1)})W^{(2)}, \tag{8}$$

where $W^{(1)} \in \mathbb{R}^{D_i \times D_o}, W^{(2)} \in \mathbb{R}^{D_o \times D_o}$ are parameters for training, $\hat{A} = \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$ is the normalized adjacency matrix, D_i is the dimension of feature vector and D_o is the dimension of output embeddings.[4]

3.3.2 Objective Function. Unlike existing GCN models which are trained for a regression or classification task [16], our problem is not a typical supervised task. Under this circumstance, inspired by [38] which adopted unsupervised training approaches to well preserve *the second-order proximity*, we design a learning method to encode the *co-occurrence relationship* for units on graphs with different modalities to address the challenge (*a*). Specifically, we design a *meta path-guided learning strategy* to preserve the similarity of multi-modal units. Meta path is a kind of path consisting of a sequence of relations defined between different object types [37], which is a common method to enhance the representations of different nodes in heterogeneous graphs and capture the structural and intrinsic semantic relation between different units. In our problem, we consider the meta path "*Location - Time - App*", which is a simple aggregation of all types of units. Then, for each meta path, if the elements (i.e. Location l, Time t and App a) co-occur in a record $r \in \mathcal{D}$, we maximize the likelihood if the occurrence probability of each unit e given its context $c_{-e} = \{o | o \in r, o \notin \{e, u\}\}$ as:

$$p\left(e|c_{-e}\right) = \frac{\exp\left(s(e, c_{-e})\right)}{\sum_{o \in \mathcal{X}} \exp\left(s(o, c_{-e})\right)}, \tag{9}$$

where $\mathcal{X}$ is the entire set of candidate units, and $s(\cdot)$ is a score function reflecting the similarity between the unit e and its context c_{-e} as $s\left(e, c_{-e}\right) = \boldsymbol{\theta}_e \cdot \left(\frac{1}{|c_{-e}|}\sum_{o \in c_{-e}} \boldsymbol{\theta}_o\right)$ with $\boldsymbol{\theta}_e \in \mathbb{R}_o^D$ denoting D_o-dimensional embedding of unit e. In this way, the meta path can reflect the intrinsic similarities between different information units within the same record. For example, if there are many users using camera apps at a scenery spot in the morning of weekends, we can infer that these units have similar semantics, and their embeddings would be close with our meta path-guided learning method.

Ultimately, the objective function of our learning model is to maximize the log-likelihood of observing all the units e given their contexts c_e in all sampled meta paths $\mathcal{L}$ in training set $\mathcal{D}$:

$$O = -\sum_{r \in R}\sum_{e \in r} \log P\left(e|c_{-e}\right). \tag{10}$$

Optimizing the objective function in Eqn. 10 requires a summation over the entire set of units X, which leads to high computational complexity. In order to improve the computation efficiency, we adopt the negative sampling approach [21] which is a standard approach for both recommendation systems and GCNs [7, 54] to sample multiple negative pairs from a noise distribution to estimate one true pair. In this way, we can learn the similarity between units in the true pair and the discrepancy between units in the negative pairs . In practice, for each unit e, we sample K negative units which share the same type with e but do not appear in the record r. Hence the loss function can be approximated as:

$$L = -\log\sigma\left(s\left(e, c_{-e}\right)\right) - \sum_{k=1}^{K} E_{o_k \sim P_n(o)}\left[\log\sigma\left(-s\left(o_k, c_{-e}\right)\right)\right], \tag{11}$$

[4]For detailed derivation of propagation rules for Graph Convolution Networks, please refer to Appendix I for more information.

where $\sigma(\cdot)$ is the sigmoid function, o_k is the k-th negative sample and $P_n(o)$ is the noise distribution and is set to $P_n(o) \propto d_o^{3/4}$ as [21] suggested, where d_o is the degree for node o in graph $\mathcal{G}$. We use stochastic gradient descent (SGD) to optimize Eqn. 11 and the updating rule for parameters in Graph Convolutional Networks can be easily obtained via autograd mechanisms in advanced deep learning tools [26].

3.3.3 Complexity Analysis. In SA-GCN algorithm, the layer-wise propagation is the main operation. In our problem, recall that D_i, D_o represent the dimension of feature vector and output embeddings respectively. Since we adopt a 2-layer GCN model, the sparse matrix multiplication has computational complexity as $O(|A^+|D^iD^o)$ and $O(|A^+|D^oD^o)$ for two layers respectively, where $|A^+|$ denotes the number of nonzero entries in the normalized adjacency matrix $\hat{A}$ and it is linear to the total number of edges in the multimodal graph $\mathcal{G}$. For optimization, the computation of Eqn. 11 only involves dot product and the time complexity is $O((K+1)D^o)$, where K is the number of negative samples. To sum up, the overall complexity for SA-GCN is $O(|A^+|D^iD^o + |A^+|D^oD^o + (K+1)D^o)$.

3.4 Applications

To show the effectiveness of our learned representations, we apply the embedding vectors in App usage prediction directly, i.e., to predict the App a user will use give a further timestamp. To achieve this goal, we need to consider both the embedding information from different modalities and their past App usage traces. Therefore, we first generate an embedding for each user to encode the above information, and then compare this with App embedding vectors for prediction. This procedure can be divided into two steps as,

Step 1: User embedding generation. To model the feature of App and locations with the consideration of the dynamic usage pattern for users, we follow the generation method in [5] described as follows: given a time τ and a user u, we first extract her previous App usage records (i.e., all records of user u with $T < \tau$). The set for these records is denoted as $\mathcal{D}_u = \{(u_i, t_i, l_i, a_i)\}_{i=1}^{|\mathcal{D}_u|}$, where (u, t_i, l_i, a_i) is the tuple for i-th record in $\mathcal{D}_u$. We denote the profile of user at time τ as

$$\boldsymbol{u}_\tau = \beta \sum_{(u,t_i,l_i,a_i)\in\mathcal{D}_u} e^{-(\tau-\tau_i)/T}\boldsymbol{\theta}(l_i) + (1-\beta) \sum_{(u,t_i,l_i,a_i)\in\mathcal{D}_u} e^{-(\tau-\tau_i)/T}\boldsymbol{\theta}(a_i), \tag{12}$$

where $\boldsymbol{\theta}(a_i)$ and $\boldsymbol{\theta}(l_i)$ denote the embedding for App a_i and location l_i derived from section 3.3 respectively. $e^{-(\tau-\tau_i)}$ models the time decay influence as we believe that the influence of older records are smaller. $\beta \in [0, 1]$ balances the influence of the usage trajectory information and the App information.

Step 2: Prediction. We compute the scores for different mobile Apps a_j as[5]:

$$s_j = \frac{\boldsymbol{u}_\tau^a \cdot \boldsymbol{a}_j}{||\boldsymbol{u}_\tau^a|| \cdot ||\boldsymbol{a}_j||}. \tag{13}$$

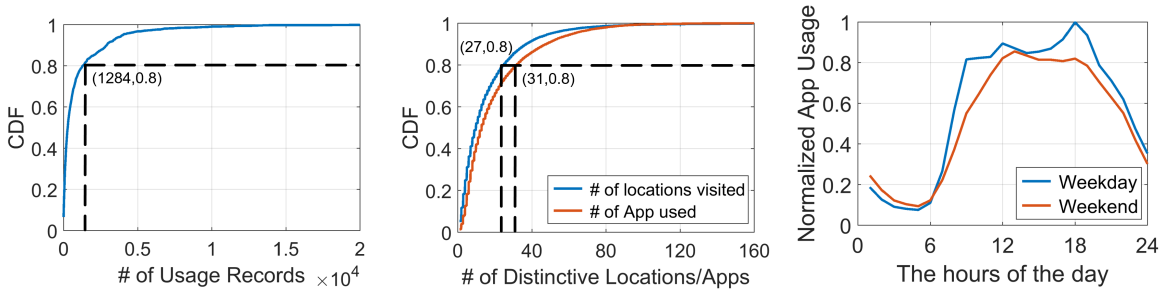
This score not only captures the App usage preference, but also captures the user's history trajectory. In practice, we take the Apps with maximum scores as prediction.

4 EXPERIMENTS

4.1 Dataset

4.1.1 Dataset collection. We leverage a large-scale real-world mobility trace released by a recent study [40]. Specifically, it was collected by one of the largest Internet service provider (ISP) in Shanghai, China, from April 20th to 26th, 2016. It contains more than ten thousand mobile users' accessing logs to the cellular network. Through deep packet inspection, each access record is characterized by an anonymized user ID, timestamp, cellular base station with GPS location and the metadata of the networking communication. For App information,

[5]Here we use the normalized score to reduce the effect of vector norm.



(a) CDF of the number of records for each App.

(b) CDF of locations and Apps.

(c) Normalized count of App usage.

Fig. 3. The statistics of dataset.

we identify more than 2000 Apps from the networking metadata by applying SAMPLES [55]. For location information, we take the cellular base stations as the basic location units and utilize the widely used Voronoi diagrams [1] to partition the city based on stations' coordinates. Then for each location, it can be mapped to a unique cellular base station, and the corresponding ID are used to represent the location of App usage. For time information, we discretize the time in a day into 24 units and use these 24 units to record the App usage time.

4.1.2 Data Preprocessing and Cleaning. Since the usage of the most frequent Apps can easily occur hundreds of millions of times. Such Apps provide less information value than rare Apps. To counter the imbalance between the rare and frequent Apps, we used a simple sub-sampling Approach [21]: for each record with App $a \in A$ in the dataset, it is discarded with probability of $P(a) = \max\left(1 - \sqrt{f_{th}/f_a}, 0\right)$, where A is set of all Apps, f_a is the frequency of App a, f_{th} is the chosen threshold, which is set as $2/|A|$ by default. This Approach sub-samples Apps whose frequency is greater than f_{th} while preserving the ranking of the frequencies. After sub-sampling, we filter the user with less than 10 records, App with less than 5 records, location with less than 5 records as they do not have enough records to reveal meaningful temporal or spatial patterns. After preprocessing, there are 11,170 users, 1792 Apps, and 9,330 locations used for the following experiments.

4.1.3 Ethics. We are very aware of the privacy implications of using ISP dataset for research and have taken active steps to protect mobile users privacy. First, the App usage traces do not contain any personally identifiable information. The userID has been anonymized (as a bit string) by the ISP, and we never have access to the true userID. Second, all the researchers are regulated by a strict non-disclosure agreement. This work has received approval from both the ISP and our local institution. Third, we store all the data in a secure off-line server, and only the authorized core researchers can access the data.

4.1.4 Basic Statistics. To provide a comprehensive understanding on the utilized dataset, we display several distributions in Fig. 3. The cumulative distribution of the number of records for each user is shown in Fig. 3(a). It shows that more than 20% of the App are frequently used with than more 1,200 records during the week. The distribution for the number of accessing different Apps and locations of mobile users is shown in Fig. 3(b), which shows that 20% of the users are recorded in more than 27 locations, and 20% of the users use more than 31 unique Apps, indicting a diversified user preference w.r.t. Apps and locations. In is also worth noting that the average number of used Apps per user is 23.16. In addition, we explore how App usages vary per hour for weekdays and the weekends in Fig. 3(c). From the results, we observe that users' activities are at the highest peak during the daytime while reduces over the night period. When at about 4 AM, the number of Application usages is the smallest since most people are sleeping and their devices are inactive. Also, during the daytime, the App usage frequency on weekdays is higher than weekends.

Table 1. App categories and example Apps.

No.	Category	Example Apps	No.	Category	Example Apps
1	*Utilities*	Safari, Clock	11	*Education*	YouDaoDict, ICiBa
2	*Games*	QQGame, SimpleGame	12	*Health/Fitness*	GuDong, Keep
3	*Entertainment*	iMovie, Livestream	13	*Infant&Mom*	BeiBeiWang, MuYingHome
4	*News*	QQNews, SohuNews	14	*Navigation*	AppleMap, GaoDeMap
5	*Social/Networking*	Wechat, Linkedin	15	*Weather*	MoJiTianQi, 360TianQi
6	*Shopping*	Taobao, JD	16	*Music*	QQMusic, NetEase Music
7	*Finance*	AliPay, MobileBanking	17	*References*	Baidu, Wikipedia
8	*Business*	QQMail, 163mail	18	*Books*	BaiduRead, iBooks
9	*Travel*	QuNaEr, XieChengTravel	19	*Photo&Video*	BaiduVideo, YouKu
10	*Lifestyle*	Dianping, Meituan	20	*Sports*	SinaSports, HuPu

Considering that Apps in the same category work as similar function, we obtain this side information from Apple Store and Google Play. There are totally 20 App categories. The detailed divisions with examples are given in Table 1.

For locations, we leverage the Point of Interests (PoI) to describe the specific urban and economic functions of a base station [56]. We collect the PoI information of Shanghai from BaiduMap[6]. There are 17 categories of PoIs in total including *Food, Hotel, Shopping, Life Service, Beauty, Tourism, Entertainment, Sports, Education, Media, Medical Care, Automotive Service, Traffic Facilities, Finance, Real Estate, Company* and *Government*. For each base station, we could collect the PoI distribution within that region to represent its semantics information.

4.2 Experiment Setup

4.2.1 Baseline Methods. We compare our SA-GCN with the following latest representation learning models, among which **Grarep**, **Node2vec**, **Metapath2vec**, **ReconEmbed**, and **GCN** are based on the same graph with our model constructed in Section 3.1, while **GraphEmbed** and **CAP** build their own graphs.

- **Grarep** [4]: It is a matrix factorization-based method, which learns node representations on weighted graphs via integrating global structural into the learning process.
- **Node2vec** [12]: It first samples random-walks from the graph. Then, it treats the random-walk as a sentence, and thus learns node embedding by skip-gram.
- **Metapath2vec** [8]: It samples meta paths as random walks and leverages a heterogeneous skip-gram model to obtain node embedding. We use the meta path as '*App-location-time*' to consider the type information of nodes in the graph and ensure three types of units appear alternately in the path.
- **ReconEmbed** [59]: It samples random-walks from the graph and takes Eqn. 9-11 as objective function directly to learn embedding vectors by gradient descent.
- **GraphEmbed** [59]: It adds spatial and temporal neighborhood edges to the co-occurrence graph, and learns low-dimensional representations by preserving first and second-order proximities in the graph.
- **CAP** [5], short for Context-Aware App Prediction, is the state-of-the-art representation learning-based method for App usage prediction. It builds three bipartite graphs by considering three types of relationships (*App-location*, *App-time*, and *App-App type*) and proposes a heterogeneous graph embedding algorithm to map them into a common latent space jointly.
- **GCN** [16]: It is a variation of our method which applies GCN on the graph without node attributes. Instead, the feature for each node is a one-hot vector to encode its information.

[6]https://map.baidu.com/

In particular, **Grarep** and **Node2vec** are unaware of the node type. Even though **Metapath2vec** samples meta path, its objective function is weak to capture spatio-temporal characters. Compared with **ReconEmbed, GraphEmbed, CAP** which also consider to reinforce co-occurrence, SA-GCN can better model the complex relationships among different units via deep node attribute propagation in Graph Convolution layers.

Besides these representation learning-based baselines, we also implement six classic methods to serve as benchmarks for the App prediction tasks described as follows:

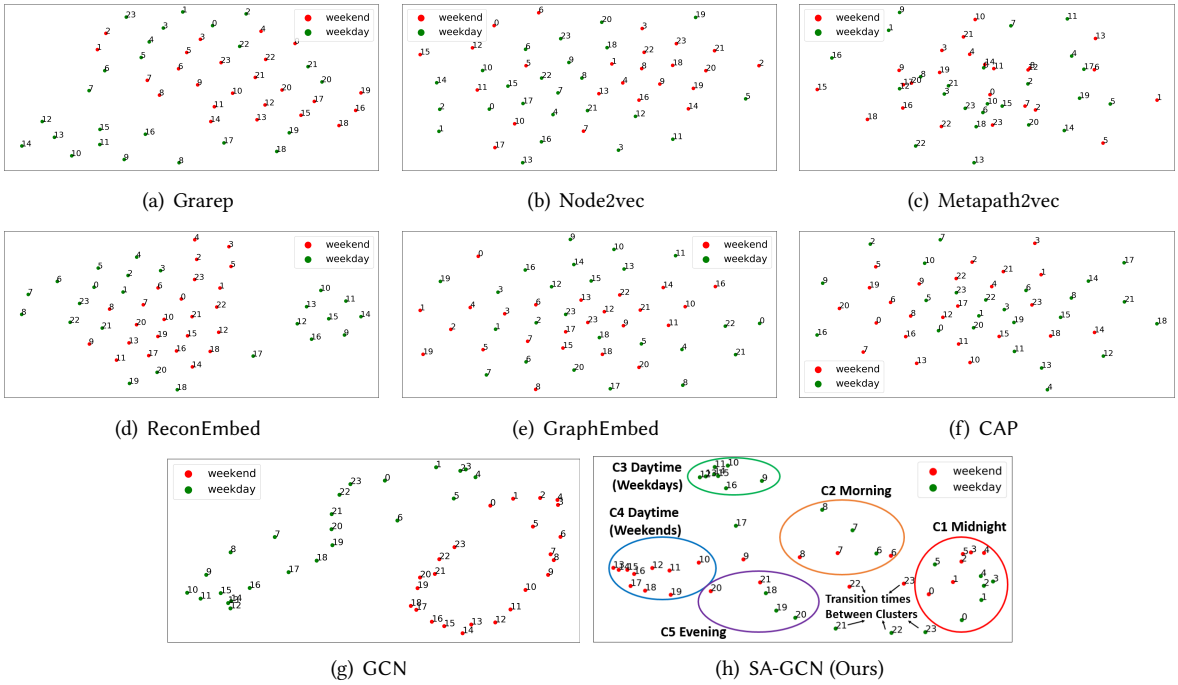
- **MRU** [35]: This method takes the **m**ost **r**ecently **u**sed Apps, i.e., the Apps used in last timestamp, as prediction. It assumes that most App are used across several time slots continually.
- **MFU** [35]: It counts the users' App usage history and selects the **m**ost **f**requently **u**sed Apps. This is the straightforward method for prediction, which does not use time and location context.
- **Falcon** [53]: It uses contexts such as user location and temporal access patterns to predict app launches before they occur.
- **APPM** [25]: Similar to [53], this method integrate spatio-temporal information as contexts. Moreover, it also adapts to usage dynamics to predict personal app usage.
- **MFApp** [52]: It incorporates robust similarity estimation between users, and incorporates this measure into prediction with a relatively simple voting scheme.
- **Bayesian** [13]: It designs a Bayesian network to use the contextual information, such as time, location, user profile, and latest used App, to predict the mobile App usage.

4.2.2 Evaluation Protocol. To learn the representation of App, location and time unit, we first need to build the graph as in Section 3.1. While in order to evaluate the performance of App usage prediction without post-learning process, we hold out some data for testing. Specifically, we sort the data by time, then use the first 80% of it on both weekdays and weekends of each user to build the graph, and use the last 20% as the testing set for prediction. For time, we discretize one day into 24 time slots with each hour as a time unit. As such, we build the App usage graph with 11,170 nodes and 1,014,904 edges in total. Since there are $M = 20$ App categories, $P = 17 (L = 34)$ PoI categories and $2N = 48$ temporal units, the feature vector for each unit is a 1×102 ($D_i = M + L + 2N = 102$) dimension vector.

Our evaluation for the learning representations can be divided into two aspects: On the one hand, we conduct in-depth analysis on the learned embedding vectors to gain some insights of App usage from the empirical data. On the other hand, we present some quantitative comparisons by adopting the embedding vectors in App usage prediction. Specifically, there are 332,255 records held out for testing, with 2.99 records per user on average. For each record, we predict the App used by ranking all Apps according to the score as Eqn. 13. It is worth noting that all 2000 Apps are candidates because we don't know which App is pre-installed and users may install new Apps.

For the quantitative evaluation, we adopt two metrics, *Accuracy* and *MRR*. For App prediction, $Accuracy@k$ is the statistical result of all test predictions, which is calculated with $hit@k$. It is calculated as the average over all test cases: $Accuracy@k = \frac{\#hit@k}{|R_{\text{test}}|}$, where $\#hit@k$ and $|R_{\text{test}}|$ represent the number of hits in the whole test set and the number of test sets respectively. *MRR*, short for *(Mean Reciprocal Rank)*, complements $Accuracy@k$ by assigning higher scores to the hits at higher positions of the ranking list. It is calculated as $MRR = \frac{1}{|R_{\text{test}}|}\left(\sum_{i=1}^{|R_{\text{test}}|}\frac{1}{r_i}\right)$, where r_i represents the reciprocal of the rank for the i-th prediction. For both the two metrics, the higher, the better of the performance.

4.2.3 Implementation Details and Parameter Settings. We implement all the representation-based models in Pytorch. To be more specific, we train them using Adam optimizer [15] for 200 epochs. For each method, we set the size of mini-batch to 256. Then we tune its learning rate μ in [0.0001, 0.0002, 0.0005, 0.001, 0.002, 0.005, 0.01, 0.02, 0.05, 0.1] and L2-regularization term λ in [0.0001, 0.0002, 0.0005, 0.001, 0.002, 0.005, 0.1, 0.2, 0.5, 1] and report the best performance. Moreover, for Node2vec and Metapath2vec model, we use Gensim to learn the embedding



(a) Grarep (b) Node2vec (c) Metapath2vec

(d) ReconEmbed (e) GraphEmbed (f) CAP

(g) GCN (h) SA-GCN (Ours)

Fig. 4. Visualization for embedding of time units, where the number near each point stands for the time in hour.

for nodes after gathering random walks. We tune the length of the walk l in [40, 60, 80, 100] as well as search bias p, q in [0.5, 1, 2, 4] to report the best performance. For Grarep model, we train it with sklearn toolkit [27] after obtaining the weighted adjacency matrix. Also, there are some crucial settings in the experiments such as the threshold σ and κ in feature vector generation and the dimension of embedding D_o for different units. In our method, we set σ to 0.5km, κ to 1km, which is reasonable in urban settings [18]. Also, we fix the dimension D_o to 64 and the number of negative samples K to 5 for fair comparison over different models.

4.3 Result Analysis

To look into the quality of learned representation directly, we map the obtained embedding vectors into 2-dimensional space with their closeness in high-dimensional space preserved by t-SNE [20], and conduct in-depth analysis from time, App, location perspectives, respectively and simultaneously.

4.3.1 Time Representation. Fig. 4 shows the overall representation of 48 time units. Firstly we can observe that compared with the methods of Node2vec, Metapath2vec, GraphEmbed and CAP, both GCN and our SA-GCN model can well differentiate the time units in both weekdays and weekends, which indicates that the App usage patterns are different between the weekends and weekdays. Then, taking a closer look at the time unit within the weekday, we find that time units are embedded into several clusters with a smaller radius by SA-GCN: midnight (12PM-4AM), morning (6AM-8AM), daytime (9AM-4PM) and evening (6PM-8PM). Moreover, units belong to different clusters are far from each other implying that within these clusters, the App usages of the corresponding time units are similar. While for time units in different clusters, the App usage are quite different. On the other hand, baselines including Node2vec, Metapath2vec, GraphEmbed and CAP fail to characterize such temporal regularities for App usage. For time units in weekends, the App usage patterns can also be grouped

Table 2. Characteristics on App Usage for Different Time Clustering Units.

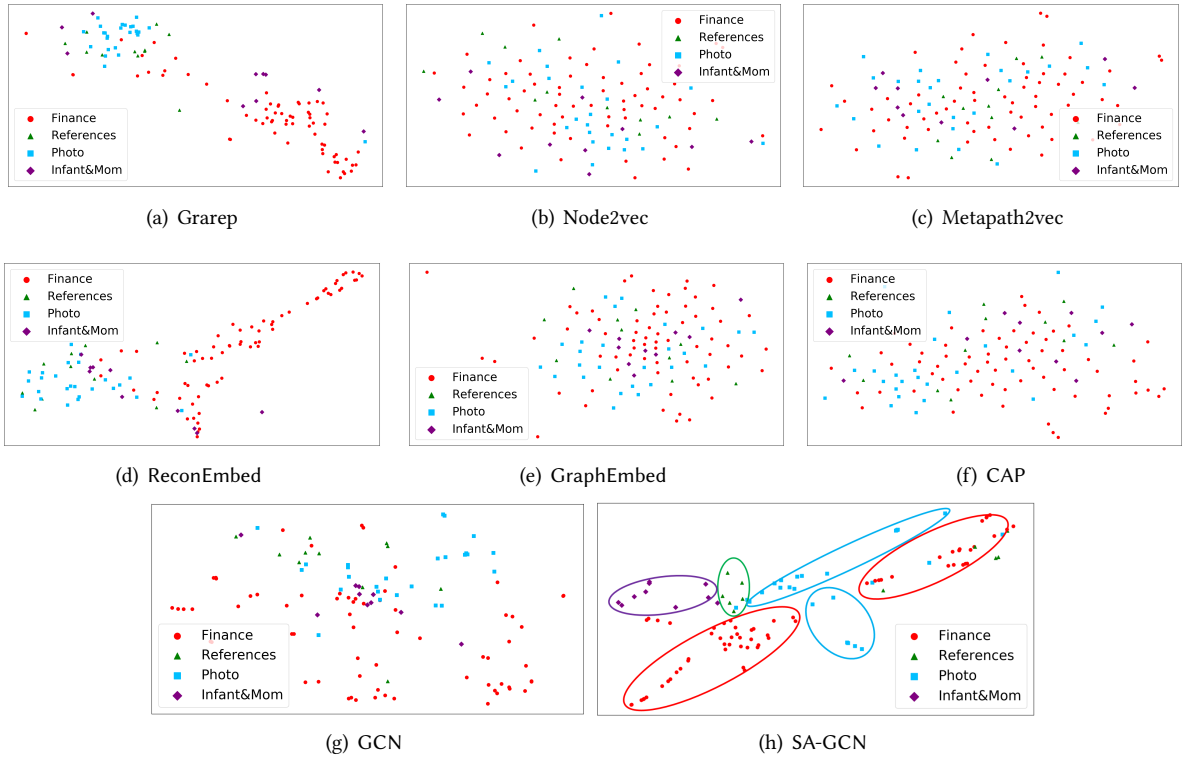
Cluster	Descriptions	Time Units	Average App Usage Counts Per Hour	Characteristic App Usage Categories
C1	Midnight	12AM-5AM (Both Weekdays and Weekends)	357.74	Social/Networking (13.16%), Lifestyle (11.33%), Entertainment (9.53%), Games (9.04%), Music (8.24%)
C2	Morning	6AM-8AM (Both Weekdays and Weekends)	1475.35	Lifestyle (13.50%), Music (12.48%), Social/ Networking (11.70%), News (10.83%), Navigation (10.27%)
C3	Daytime (Weekdays)	9AM-4PM (Weekdays)	3380.61	Social/Networking (13.47%), Finance (12.48%), Lifestyle (11.78%), Navigation (10.11%), News (8.18%)
C4	Daytime (Weekends)	10AM-7PM (Weekends)	2278.37	Lifestyle (14.75%), Social/Networking (13.12%), Navigation (12.82%), Music (9.11%), Entertainment (7.27%)
C5	Evening	6PM-8PM (Weekdays) 8PM-9PM (Weekends)	1991.72	Social/Networking (14.44%), Lifestyle (13.22%) Music (10.11%), Navigation (9.78%), News (8.28%)

into several clusters by SA-GCN as midnight (12PM-5AM), morning (6AM-8AM), daytime (10AM-7PM) and evening (8PM-9PM). Comparing the clusters with those for weekdays, we find that the embedding of daytime cluster for weekdays and weekends are well separated, indicating that the App usage patterns are diverse during that time. This is interpretable since office-related Apps might be more frequently used during weekdays while recreational/social Apps are more popular on weekends. Moreover, for other time units like morning and midnight, the weekday and the weekend share similar embeddings, which indicates that App usage patterns in those time are similar between weekdays and weekends. All these results along with the analysis clearly show that our proposed SA-GCN can understand temporal patterns of App usages thoroughly while the baselines fail to appropriately model the similarities and differences of the patterns between different time units on weekdays and weekends.

To delve deeper into temporal App usage, we manually cluster these units into five-time parts as shown in Fig. 4(h), where we use an ellipse to outline the shape of the cluster. It is worth noting that we omit some time units, as we consider those time slots (e.g., 10 PM on Weekday) as the transition between different time clusters. Besides, we calculate several statistics of the usage of App in Table 2 for those clusters, from which we can obtain several interesting observations:

(1) **Different temporal clusters have diverse App usage frequency.** From the Average App Usage Counts Per Hour, we can observe that during the daytime on weekdays, the App usage per hour is largest, which indicates that Apps are used more frequently during working hours, as there are numerous Apps are related to the work. However, for the time slots in midnight, the App usage is much smaller, which is only about 10% of the daytime usage.
(2) **Some categories of App are frequently used in all of the times** such as Social/Networking and Lifestyle Apps. We think it is also reasonable since App would have different functions at different times. For *social/Networking* Apps as an example, they are used for working collaboration during the working time and family communication during the weekends and nights.
(3) **The App Usage can reveal the rhythm of life within these clusters.** For instance, the usage proportion for *Entertainment* Apps reaches the maximum at midnight and weekends. Moreover,*Finance* Apps are mainly used during the daytime on weekdays, when is the working time for Banks and Stock Markets in China.

To sum up, embedding vectors for time units learned by our proposed SA-GCN can well represent the temporal characters of App usage.

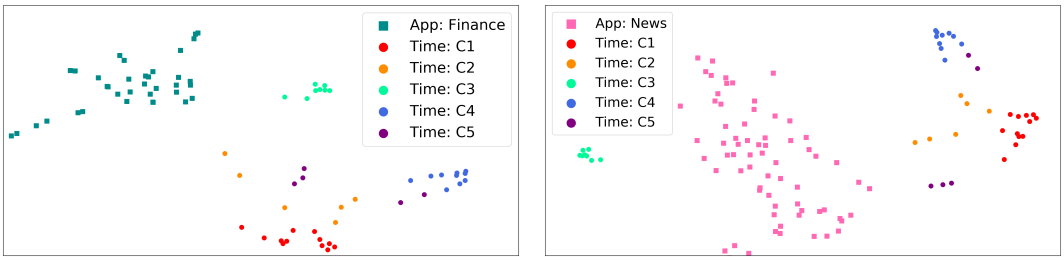


(a) Grarep (b) Node2vec (c) Metapath2vec

(d) ReconEmbed (e) GraphEmbed (f) CAP

(g) GCN (h) SA-GCN

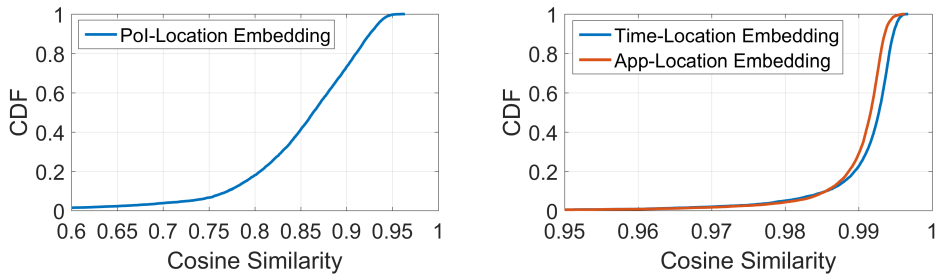
Fig. 5. Visualization for embedding of App units, where different color stands for different categories.



(a) Embedding of *Finance* apps with time units (b) Embedding of *News* apps with time units

Fig. 6. Visualization for embedding of App units and time units.

4.3.2 App Representation. To analyze the representation of Apps, we use the category information to check whether the learned embedding vector remaining such properties. Specifically, we show the visualized embedding vectors of Apps of the most representative App categories in Fig. 5. As we can observe from Fig. 5(h), four categories of Apps are well separated as each ellipse encircles one category. On the contrary, those baselines cannot sufficiently distinguish different App categories via representation learning, as the embedding vectors of these units are mixed. It is mainly due to our SA-GCN model takes the attribute information of apps into consideration. Compared with CAP which also considers the App side information, SA-GCN is capable of preserving the attribute information during the representation learning.



(a) CDF of the correlation between location representation and PoI distribution.

(b) CDF of the correlation between location representation and App/time representation.

Fig. 7. The statistical correlation for location representation.

In addition, we investigate the relation of App representation with temporal units. We visualize the embedding vectors of Apps with the category of *Finance* and *News* together with embedding vectors of clustered time units in Table 2. From Fig. 6(a) and 6(b), we can find out that embedding of *Finance* Apps are concentrated and close to the embedding of time units in C3, i.e., hours at weekday, which coincides with their semantics in our prior knowledge that economic events happen most at office hours. What's more, embedding vectors of *News* Apps are close to that of times in C1, C3, and C5, which is consistent with the statistics shown in Fig. 2, i.e., the *News* is one of the most popular App categories during those periods.

4.3.3 Location Representation. Now we inspect the representations for location units. To demonstrate that these embedding vectors contain semantic information, we analyze the statistical correlation location representation and its PoI distribution as well as spatial App usage [39, 56].

Firstly, we find that representation and PoI distribution are correlated significantly. In specific, embedding similarity vector $\boldsymbol{P}_i = \{p_{ij}\}$ denotes the similarity between location i and others, with p_{ij} representing cosine similarity between the embedding of location i and j. So as the PoI similarity vector $\boldsymbol{Q}_i = \{q_{ij}\}$, where q_{ij} representing cosine similarity between the PoI distribution vector of location i and j calculated by Eqn. 4. To further quantify the relationship between location embeddings and PoIs, we still use Cosine Similarity to compute the correlation between embedding similarity vector P_i and PoI similarity vector Q_i for user i. The correlation $\boldsymbol{C}_i^p$ of $\boldsymbol{P}_i$ and $\boldsymbol{Q}_i$ is computed as $\boldsymbol{C}_i^p = \cos(\boldsymbol{P}_i, \boldsymbol{Q}_i)$, with N representing the total number of locations. Fig. 7(a) shows the Cumulative Distribution Function (CDF) of the correlation $C^p = \{C_i^p\}$. From the result, we can observe that for nearly all the location (i.e. above 80%), the correlation between the location embedding and PoI distribution is strong (more than 0.8). Therefore, our learned representations can reflect PoI semantic for locations.

Moreover, to reveal the correlation between location units and App/time units, we compute the correlation between location embedding similarity vector $\boldsymbol{P}_i$ and App similarity vector $\boldsymbol{S}_i$ as well as time similarity vector $\boldsymbol{T}_i$ respectively. For location i, we denote the App usage vector $\boldsymbol{u}_i$ as the weighted sum of the App embedding. Similarly, the time usage vector $\boldsymbol{v}_i$ is also the weighted sum of the time embedding. Here the weight is the total time of App/time units and location unit i appear simultaneously in a same record. Then, we calculate the correlation for location embedding and time embedding with cosine similarity [56]. Fig. 7(b) shows the corresponding CDF, from which we find that for more than 90% of the location, the correlation between location and both time and App representation are more than 0.98. This demonstrates that the learned location representations have a strong relationship with time and App representation, which indicates that the representation of location units contain rich semantics of App usage patterns.

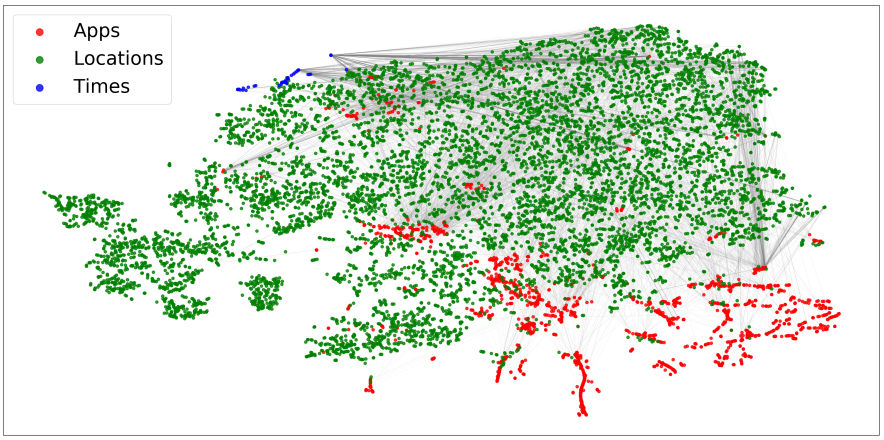


Fig. 8. Representation visualization for all units, where edge thickness indicates the normalized weight of the edge. We only plot edges with highest weights for the clarity.

4.3.4 Representation of All Units. After analyzing the embedding vectors for each type of the units, we consider them together and visualize them in Fig. 8. To consider the co-occurrence relationship of different nodes, we also plot several edges on the graph with the highest weight, where edge thickness indicates the normalized weight of the edge and can be regarded as the strength of the co-occurrence relationship. We can observe that the embedding vectors of App, location, time are generally separated, as most of the time units are located in the top left corner of the figure and most of the App units are located in the lower right corner, which indicates that our model is able to identify different type of units. Moreover, we find that those nodes with high weight edges tend to have a closer representations to their neighbors. For example, the time units with many edges connected to location units in Fig. 8 have a embedding closer to that of the locations. Also, the Apps with many edges connected to location units are embedded into the cluster for the embedding of locations. These results suggest that the representations learned via SA-GCN can well capture co-occurrence relation.

To sum up, our SA-GCN can address spatio-temporal context information as well as cross-modal correlations simultaneously, thus being capable of capturing the semantic patterns within each type as well as modeling the relationships between different type of units.

4.4 Application Performance

Table 3 shows the overall performance compared with baselines when adopting the learned embedding in App usage prediction task. From the experiment results, we have the following observations and conclusions:

1) The majority of the representation learning-based methods, including Metapath2vec, Node2vec, and GraphEmbed, consistently achieve notable performance gains of 22.4% in terms of Accuracy over counting-based benchmarks, which suggests the importance of learning expressive representations. On the other hand, MFU outperforms MRU by 33.6% and 37.4% in term of Accuracy@1 and MRR. The reason is that there are some Apps which user interact with more frequently than others, such as WeChat. But the App usage is rather dynamic, and thus the most recently engaged Apps could be no longer used.
2) Our proposed SA-GCN model significantly outperforms all baseline models. Compared to MFU, i.e., the best counting-based baseline, it provides relative performance gain of 32.6%, 8.3% in terms of Accuracy@1, and MRR, respectively. Besides, it also significantly outperforms the state-of-art representation learning method, GraphEmbed, by 8.3% in Accuracy@1. Moreover, compared with the state-of-the art graph-based

Table 3. App usage prediction performance comparison.

Metrics	Accuracy@1		Accuracy@10		MRR	
Method \ Value	Result	Relative Gain	Result	Relative Gain	Result	Relative Gain
MRU	0.110	77.27%	0.539	11.50%	0.211	44.70%
MFU	0.147	32.65%	0.590	1.86%	0.290	8.27%
Falcon	0.171	14.03%	0.566	6.18%	0.281	10.32%
APPM	0.179	8.93%	0.597	0.67%	0.298	4.02%
MFApp	0.175	11.43%	0.594	1.17%	0.299	3.67%
Bayesian	0.161	21.87%	0.523	14.91%	0.266	17.42%
Grarep	0.112	74.10%	0.274	119.34%	0.167	88.02%
Node2vec	0.153	27.45%	0.505	19.00%	0.261	20.30%
Metapath2vec	0.164	18.90%	0.518	16.02%	0.275	14.18%
ReconEmbed	0.125	56.00%	0.354	69.77%	0.173	81.50%
GraphEmbed	0.180	8.33%	0.545	10.27%	0.272	15.44%
CAP	0.127	53.54%	0.363	65.56%	0.181	73.48%
GCN	0.168	16.07%	0.555	8.29%	0.286	9.79%
SA-GCN	**0.195**	-	**0.601**	-	**0.314**	-

App prediction baseline CAP, our method achieves a huge gain (53.54% - 73.48%) on three metrics. These results show that SA-GCN model consistently outperforms against state-of-art baselines and demonstrate the advantage of our GCN-based model.

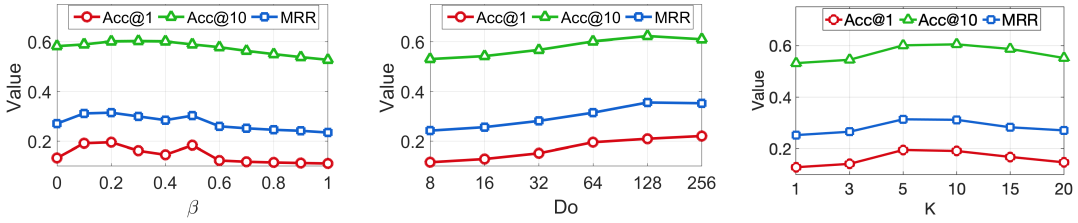
3) By comparing the performance of the SA-GCN model with the App prediction algorithms, we can find that though most of these non-graph prediction methods can outperform the basic counting-based methods MRU and MFU, as they are designed for context-aware App prediction. But their performance gains are rather marginal, which indicates that such methods are insufficient in modeling complex spatial-temporal relationship for App usage. In contrast, our SA-GCN model can outperform all of them by 8.93%-21.87% for Accuracy@1 and 3.67%-17.42% for MRR, which further justifies the merit of our representation-learning approach can better model the relationships among different units. Moreover, at prediction stage, some non-graph prediction methods (e.g. [52]) need to calculate the similarity among different users and features, which is time-consuming. While for our method, we can predict the App usage directly from our pre-trained embeddings, which do not require extra computation compared with other baselines.
4) By comparing the performance of the SA-GCN model with its degraded version GCN, we can observe the consistent performance gains, since the attribute information like PoI distribution and App category are integrated into the framework. These comparisons indicate that the function of location and App play their role in improving the App usage prediction.

To conclude, our SA-GCN model consistently achieves preferable results compared with benchmarks and the state-of-art graph embedding methods. In addition, incorporating attribute information leads to significant performance gain. These results justify SA-GCN's effectiveness in simultaneously capturing the feature of App usage patterns and semantic similarity, therefore it is promising to be applied for downstream applications.

4.5 Parameter Study

For SA-GCN method, we examine several key hyper-parameter including balance coefficient β, embedding dimension D_o, negative sampling size K and the radius of Gaussian kernel σ for location feature and present the impact by the prediction performance.

The influence of β. Fig. 9(a) illustrates how β in Eqn. 12 affects our algorithm performance. All metrics except Acc@10 reach a maximum when $\beta = 0.2$ and Acc@10 (short for Accuracy@10) reach a maximum when $\beta = 0.3$, which means that combining the App embedding with the location embedding to generate users' profile is a



(a) Impact on weight β. (b) Impact on embedding dimension D_o. (c) Impact on negative sample size K.

Fig. 9. The performance of prediction with different parameter settings.

reasonable way to promote predictive performance. Moreover, it implies that our SA-GCN model captures the correlations between spatial and App units.

The influence of D_o. Fig. 9(b) explores how D_o affects the overall prediction performance. We find that for dimension in [8, 16, 32, 64, 128], the prediction result increases as the embedding dim increases and the prediction result of 64-dimensional embedding outperforms baseline by 6.00% - 29.14%. However, for dimension greater than 64, sometimes the result even becomes worse with the increase of the dimension (e.g. the MRR value for dimension 256 is worse than that of 128). Therefore, considering the trade-off between prediction accuracy and computing complexity, we adopt 64 embedding dimensions as the default setting.

The influence of K. Fig. 9(c) explore the effect of the negative sample size K. We find that with the increase of K, the overall performance first increases then decreases. We explain the result in two folds: when K is small, then improving the size of the negative examples will encourage the model to better learn the decision boundary during training. However, when K is large, then the increase of the K will make the model tend to predict negative samples since the size of negative samples in training set is much more than positive samples.

The influence of TF-IDF reweighting. Fig. 10(a) shows the result comparison of App prediction of our model and our model without TF-IDF reweighting. From the result, we find that the predictive result declines by 10.98% to 27.69% on three metrics when we remove TF-IDF reweighting. This is mainly due to that simply using PoI occurrence as location features will lead to a bias of feature towards those popular PoI types, which will lead to inaccurate predictions. By adding TF-IDF, our model will be more balanced for modeling different types of PoI. As a result, the location features are more representative, which facilitate the performance of prediction.

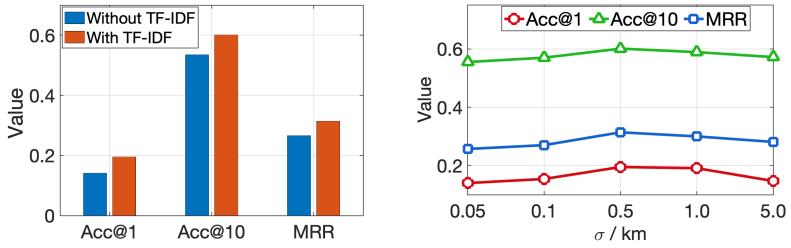
The influence of radius σ. Fig. 10(b) illustrates the effect of the radius σ in Eq. 3. It is clear that the result first increases with σ then decreases. Here we argue that with a low σ (i.e. σ = 0.05km), then for the neighbor locations, the weight $W = \exp(-dist^2/\sigma^2)$ in Eq. 3 will be close to zero, which will eliminate the effect of neighborhood PoIs. As the result, the prediction performance drops, since the characteristic of each location is also relevant to its neighbor PoIs. For high σ (i.e. σ = 5km), then since $\kappa \ll \sigma$, the weight W will be close to constant 1, which indicates that the for each location, its PoI features will be close to the mean of its neighborhood PoI distribution regardless of the distance factor. In this case, it will be less powerful to model the neighbor information. From the result, we can see σ = 0.5km reaches best performance, therefore we set σ = 0.5km as default.

5 RELATED WORK AND DISCUSSION

5.1 Related Work

In this paper, we focus on leveraging the mobility data to learn App usage representation, which is a crucial task to dissect spatio-temporal App usage behaviours and has wide-range application scenarios at the same time. Now, we review the most relevant related works, and summarize them into the following three aspects.

5.1.1 App Usage Behaviour Modeling. With the increasing popularity of mobile network and mobile Apps, recent years have witnessed a large amount of researches on App usage pattern analysis, App prediction as



(a) Impact on TF-IDF reweighting. (b) Impact on radius σ.

Fig. 10. The performance of prediction with different parameter settings.

well as App recommendation [6, 10, 13, 17, 25, 31, 44, 52, 53]. Most of them highlight the importance of utilizing spatio-temporal context for App usage modeling. [14, 45] utilized location as an implicit feature to improve App recommendation. As for App usage prediction, Qiao et al. [28] demonstrated the combination of different spatial and temporal features achieves high App forecast accuracy. Xu et al. [52] developed a multi-faceted approach to predict App usage patterns with time and location as context. Yan et al. [53] proposed FALCON to perform context-aware usage prediction with user location and temporal access patterns. Besides, Parate et al. [25] further considered the user dynamics and combine it with the spatio-temporal contexts to improve the prediction performance, and Huang et al. [13] designed a Bayesian network to jointly model different type of context information together.

Different from existing work mentioned above, we build a heterogeneous graph with App, location, and time units as nodes and their co-occurrence relationship as edges to encode spatio-temporal information simultaneously for better App usage presentation. App usage is spatio-temporal co-correlated so that learning the spatial and temporal context in one heterogeneous graph simultaneously is better than treating them separately as previous work, i.e., learning representation from location-App and time-App bipartite graphs [5].

5.1.2 Semantics-aware Representation Learning for Human Spatio-temporal Behaviours. Recently, representation learning methods have been applied to discover knowledge from spatial and temporal units. For example, Cao et al. proposed habit2vec [3] to represent user trajectories and recognize typical living patterns in metropolis, [59] modeled the embedding of spatio-temporal units based on their co-occurrence with texts in social media check-ins.Moreover, there are various applications with the semantic embeddings of spatio-temporal points, as Shi et al. [32, 34] adopted Hidden Markov Model to capture patterns of human mobility based on the embeddings of different user, location, time and activity units and [58] used the embedding for online local event detection. Different from these works, we introduce a graph neural network to learn representation for App usage behaviors, which resembles location representation learning but have not been explicitly tackled previously. Consequently, we extend the idea of semantic-aware representation from footprint (i.e., the location, POI, etc.) to fingerprint (i.e., online App usage), which deepens our understanding of human activity in both physical and cyber-space.

5.1.3 Graph Representative Learning Methods. Graph is a powerful tool to represent the complex relationship among different entities, thus graph representation learning has received growing attention. As a result, many techniques have been proposed to learn low-dimensional representations of different elements including Node2vec [12], Grarep [4], LINE [38] and Metapath2vec [8]. Recently, massive advanced models including Graph Convolutional Networks (GCN) [16], Graph Attentional Networks (GAT) [41] have been proposed for graph representation learning, which is able to aggregate the information on graphs and achieve competitive results. Moreover, graph neural networks are able to combine both the structure of the graph and the attributes of the nodes, and have been widely adopted in various domains including traffic flow prediction [18, 33], knowledge completion [57] and recommender systems [46, 50]. In our study, we seek to utilize the embedding techniques in

a different semantic-rich dataset, i.e., App usage datasets, of which the hidden semantics have not been fully revealed before. Compare with the existing methods which learn the embedding based on the App usage sequences [19] or App-install sequences [29] , our approach is novel in two folds: *First*, We propose attribute-aware GCN for semantics modeling, which can model co-occurrence relation and node attributes simultaneously. Apart from the co-occurrence relation in the graph, where spatio-temporal context can be modeled by existing graph embedding methods [8, 12], we equip each node with their attributes, i.e., App categories and location functions. Node attributes are another kind of semantics indicating specific usage and are different from the edge attributes, i.e., co-occurrence. In this way, we make GCN more powerful to learn semantic representations as it is both graph structure-aware and node attribute-aware. *Second*, we propose mate-path guided learning scheme to allow GCN to capture multi-modal relationship. Although existing method [22] have adopted GCN to learn the embeddings of mobile Apps, they do not identify the app-location or app-time relationships, which is not adequate to model the relation among App, location and time units. In our problem, we further consider the triple-unit, i.e., App-location-time co-occurrence, therefore we use meta path-guided learning, which is training strategy to sample such triple-units from raw records to learn model parameters. Such method can better learn the relationships among App, location and time units and generate a new set of embedding vectors.

5.2 Discussion

5.2.1 Applications. Our non-task specific App usage representation learning algorithm can promote extensive applications. *First*, for network operators, it is crucial to analyze and predict the App usage, which could help them understand the geographical context of mobile traffic patterns [43] as well as model the traffic dynamics of cellular devices efficiently [30], thus optimizing the allocation of existing network resources. *Second*, for App developers and App platform administrators, our model could incorporate location and time context to build a more comprehensive model for mobile App usage modeling, which could be further utilized for context-aware App usage prediction [31, 42, 61] and App recommendation [14, 56], which can probably elevate the App usage and improve user satisfaction. *Last but not the least*, our work can be applied to various urban planning and city management problems. Our learned multimodal embeddings can be adopted to recover urban rhythms [48, 49], understand the living patterns of citizens [2, 51], and analyze crowd mobilities [6, 60], which will definitely benefit the city planners to make better land-use planning. To sum up, our work sheds light on various areas in real life, which contributes to solving many social issues.

5.2.2 Limitations. This study is the first step to utilize the Graph Convolutional Network to describe the App usage behaviors. Our work has the following limitations. *First*, the length of our App usage dataset is only one week, and we aggregate it to one working day and one non-working day. Thus, we only exhibit the regular dynamics of working and non-working day. *Second*, similar to [56], we assume that in this paper, "App usage" means that the App exists on a user's phone, and is running. However, this definition does not preclude the condition when the App is running in the background and make network requests automatically, which indicates that we are unable to tell whether the user is explicitly interacting with the app, thus the dataset may be noisy and inaccurate. *Third*, since we extracted App usage records through deep packet inspection, those apps which make no network requests were not captured. *Last but not the least*, we do not include users as nodes when constructing the App usage graph due to the complexity of GCN model and the limitation of computational resource. Although we can generate user embedding from the learned App, time, and location embedding, this non end-to-end method may degrade the personalized character of a user.

5.2.3 Future Work. To tackle these problems mentioned above, we plan to do the following work in the future. *First*, we are going to find App usage datasets with a longer period to analyze the long-term App usage patterns, and investigate evolution of App usage. *Second*, we will apply the App trace data with traffic flow information (i.e.

number of packets or overall packet size) generated from the App for better overall performance. *Moreover*, there are various work adopted advanced techniques for privacy protection (such as federated learning [9], differential privacy [11], blockchain techniques [24]), which illustrates that it is plausible to protect the privacy during learning process. One important future work is to adopt these schemes to our model, which will ameliorate the risk of information leakage.

6 CONCLUSION

In this paper, we study the problem of using large-scale App usage records to learn representation for multimodal units including time, location and App. Towards this end, we propose SA-GCN, a novel representation learning model to embed Apps, location, and time units into the same low-dimensional latent space. We build an App usage graph by regarding app, time, and location units as nodes, their property as node features, and their co-occurrence relation as edges. Based on this graph, we develop a graph convolutional network with a meta path-guided objective function to learn semantic-aware representation. We also apply these learned embedding vectors to the personal App usage prediction task and achieve more than 8% performance gain compared with baselines. This study deepens our understanding of time and location-varied App usage, and paves the way for extensive downstream applications including App usage prediction, App recommendation, and App services optimization.

APPENDIX I: THE FUNDAMENTALS FOR GRAPH NEURAL NETWORKS

In this part we will give a brief summary for the definition and the propagation rule of Graph Neural Network. For a given graph $\mathcal{G} = (\boldsymbol{V}, \boldsymbol{E}, \boldsymbol{X})$, where $\boldsymbol{V}$, $\boldsymbol{E}$ stand for the set of vertices and edges respectively and $\boldsymbol{X}$ represents the feature vector of each vertices. Graph Neural Networks (GNNs) is a class of deep learning models over the graph data G. A GNN based model calculates an embedding vector $\boldsymbol{\theta}_u$ of each node $u \in V$ via iteratively aggregating information of itself and its neighbours as $\boldsymbol{\theta}_u^{(k)} = f\big(\boldsymbol{x}_u, \boldsymbol{\theta}_u^{(k-1)}, \{\boldsymbol{x}_v, \boldsymbol{\theta}_v^{(k-1)}\}_{v\in\mathcal{N}(u)}\big)$, where $\mathcal{N}(u)$ denotes the neighbours of node u in the graph, and node features $\boldsymbol{x}_u$ serves as the initial embedding $\boldsymbol{\theta}_u^{(0)}$.

Graph Convolutional Network (GCN) [16] is one of the most well-known GNN model, which is based on a first-order approximation of spectral convolutions on graphs. The propagation rule for GCN on each node can be written as:

$$\boldsymbol{\theta}_u^{(k)} = f\big(\boldsymbol{\theta}_u^{(k-1)}, \{\boldsymbol{\theta}_v^{(k-1)}\}_{v\in\mathcal{N}(u)}\big) = \sigma\left(\sum_{v\in\mathcal{N}(u)} \frac{1}{\sqrt{d_u d_v}} \boldsymbol{\theta}_v^{(k-1)} W^k\right), \tag{14}$$

where d_u, d_v is the degree for node u and v respectively, and $W^{(k)}$ is the trainable parameters in the k-th layer.

To put Eqn. 14 into vectorized form, let $\Theta^{(k)} = (\boldsymbol{\theta}_1^{(k)}, \boldsymbol{\theta}_2^{(k)}, \ldots, \boldsymbol{\theta}_{|V|}^{(k)})$ be the matrix of all node embedding vectors at step k, the aggregation function is calculated as:

$$\Theta^{(k)} = \sigma\big(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}\Theta^{(k-1)}W^{(k)}\big), \tag{15}$$

where $\tilde{A} = A + I_N$, and A is the adjacency matrix with I_N the identity matrix. D is a diagonal matrix with $D_{ii} = \sum_j \tilde{A}_{ij}$, σ represents the non-linear activation function. $W^{(k)}$ is the trainable parameters in the k-th layer.

REFERENCES

[1] Franz Aurenhammer. 1991. Voronoi diagrams-a survey of a fundamental geometric data structure. *ACM Computing Surveys (CSUR)* 23, 3 (1991), 345–405.

[2] Hancheng Cao, Zhilong Chen, Fengli Xu, Yong Li, and Vassilis Kostakos. 2018. Revisitation in Urban Space vs. Online: A Comparison across POIs, Websites, and Smartphone Apps. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 2, 4, Article 156 (Dec. 2018), 24 pages.

[3] Hancheng Cao, Fengli Xu, Jagan Sankaranarayanan, Yong Li, and Hanan Samet. 2020. Habit2vec: Trajectory Semantic Embedding for Living Pattern Recognition in Population. *IEEE Transactions on Mobile Computing* 19, 5 (2020), 1096–1108.

[4] Shaosheng Cao, Wei Lu, and Qiongkai Xu. 2015. Grarep: Learning graph representations with global structural information. In *Proceedings of the 24th ACM international on conference on information and knowledge management (CIKM'15)*. ACM, 891–900.

[5] Xinlei Chen, Yu Wang, Jiayou He, Shijia Pan, Yong Li, and Pei Zhang. 2019. CAP: Context-Aware App Usage Prediction with Heterogeneous Graph Embedding. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 3, 1, Article 4 (March 2019), 25 pages.

[6] Marco De Nadai, Angelo Cardoso, Antonio Lima, Bruno Lepri, and Nuria Oliver. 2019. Strategies and limitations in app usage and human mobility. *Scientific reports* 9, 1 (2019), 10935.

[7] Jingtao Ding, Fuli Feng, Xiangnan He, Guanghui Yu, Yong Li, and Depeng Jin. 2018. An improved sampler for bayesian personalized ranking by leveraging view data. In *Companion Proceedings of the The Web Conference 2018*. 13–14.

[8] Yuxiao Dong, Nitesh V Chawla, and Ananthram Swami. 2017. metapath2vec: Scalable representation learning for heterogeneous networks. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining (KDD'17)*. ACM, 135–144.

[9] Jie Feng, Can Rong, Funing Sun, Diansheng Guo, and Yong Li. 2020. PMF: A Privacy-Preserving Human Mobility Prediction Framework via Federated Learning. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 4, 1, Article 10 (March 2020), 21 pages.

[10] Denzil Ferreira, Jorge Goncalves, Vassilis Kostakos, Louise Barkhuus, and Anind K Dey. 2014. Contextual experience sampling of mobile application micro-usage. In *Proceedings of the 16th international conference on Human-computer interaction with mobile devices & services*. ACM, 91–100.

[11] Chen Gao, Chao Huang, Yue Yu, Huandong Wang, Yong Li, and Depeng Jin. 2019. Privacy-Preserving Cross-Domain Location Recommendation. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 3, 1, Article 11 (March 2019), 21 pages.

[12] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining (KDD'16)*. ACM, 855–864.

[13] Ke Huang, Chunhui Zhang, Xiaoxiao Ma, and Guanling Chen. 2012. Predicting mobile application usage using contextual information. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing (UbiComp '12)*. ACM, 1059–1065.

[14] Alexandros Karatzoglou, Linas Baltrunas, Karen Church, and Matthias Böhmer. 2012. Climbing the app wall: enabling mobile app discovery through context-aware recommendations. In *Proceedings of the 21st ACM international conference on Information and knowledge management (CIKM'12)*. ACM, 2527–2530.

[15] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[16] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations (ICLR)*.

[17] Tong Li, Mingyang Zhang, Hancheng Cao, Yong Li, Sasu Tarkoma, and Pan Hui. 2020. " What Apps Did You Use?": Understanding the Long-term Evolution of Mobile App Usage. In *Proceedings of The Web Conference 2020*. 66–76.

[18] Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. 2018. Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting. In *International Conference on Learning Representations*.

[19] Qiang Ma, S Muthukrishnan, and Wil Simpson. 2016. App2vec: Vector modeling of mobile apps and applications. In *2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*. IEEE, 599–606.

[20] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of machine learning research* 9, Nov (2008), 2579–2605.

[21] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. 3111–3119.

[22] Keiichi Ochiai, Naoki Yamamoto, Takashi Hamatani, Yusuke Fukazawa, and Takayasu Yamaguchi. 2019. Exploiting Graph Convolutional Networks for Representation Learning of Mobile App Usage. In *2019 IEEE International Conference on Big Data (Big Data)*. IEEE, 5379–5383.

[23] Jiaul H. Paik. 2013. A Novel TF-IDF Weighting Scheme for Effective Ranking. In *Proceedings of the 36th International ACM Conference on Research and Development in Information Retrieval (SIGIR'13)*. ACM, 343–352.

[24] Zheyi Pan, Jie Bao, Weinan Zhang, Yong Yu, and Yu Zheng. 2019. TrajGuard: A Comprehensive Trajectory Copyright Protection Scheme. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD '19)*. ACM, 3060–3070.

[25] Abhinav Parate, Matthias Böhmer, David Chu, Deepak Ganesan, and Benjamin M Marlin. 2013. Practical prediction and prefetch for faster access to applications on mobile phones. In *Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing (UbiComp'13)*. 275–284.

[26] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems*. 8026–8037.

[27] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in Python. *Journal of machine learning research* 12, Oct (2011), 2825–2830.

[28] Yuanyuan Qiao, Xiaoxing Zhao, Jie Yang, and Jiajia Liu. 2016. Mobile big-data-driven rating framework: Measuring the relationship between human mobility and app usage behavior. *IEEE Network* 30, 3 (2016), 14–21.

[29] Vladan Radosavljevic, Mihajlo Grbovic, Nemanja Djuric, Narayan Bhamidipati, Daneo Zhang, Jack Wang, Jiankai Dang, Haiying Huang, Ananth Nagarajan, and Peiji Chen. 2016. Smartphone app categorization for interest targeting in advertising marketplace. In *Proceedings of the 25th International Conference Companion on World Wide Web*. ACM, 93–94.

[30] M Zubair Shafiq, Lusheng Ji, Alex X Liu, and Jia Wang. 2011. Characterizing and modeling internet traffic dynamics of cellular devices. *ACM SIGMETRICS Performance Evaluation Review* 39, 1 (2011), 265–276.

[31] Zhihao Shen, Kang Yang, Wan Du, Xi Zhao, and Jianhua Zou. 2019. DeepAPP: a deep reinforcement learning framework for mobile application usage prediction. In *Proceedings of the 17th Conference on Embedded Networked Sensor Systems (SenSys '19)*. 153–165.

[32] Hongzhi Shi, Hancheng Cao, Xiangxin Zhou, Yong Li, Chao Zhang, Vassilis Kostakos, Funing Sun, and Fanchao Meng. 2019. Semantics-Aware Hidden Markov Model for Human Mobility. In *Proceedings of the 2019 SIAM International Conference on Data Mining*. SIAM, 774–782.

[33] Hongzhi Shi, Quanming Yao, Qi Guo, Yaguang Li, Lingyu Zhang, Jieping Ye, Yong Li, and Yan Liu. 2020. Predicting Origin-Destination Flow via Multi-Perspective Graph Convolutional Network. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*. IEEE, 1818–1821.

[34] Hongzhi Shi, Chao Zhang, Quanming Yao, Yong Li, Funing Sun, and Depeng Jin. 2019. State-sharing sparse hidden markov models for personalized sequences. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD '19)*. 1549–1559.

[35] Choonsung Shin, Jin-Hyuk Hong, and Anind K Dey. 2012. Understanding and prediction of mobile application usage for smart phones. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing (UbiComp '12)*. ACM, 173–182.

[36] David Shuman, Sunil Narang, Pascal Frossard, Antonio Ortega, and Pierre Vandergheynst. 2013. The Emerging Field of Signal Processing on Graphs: Extending High-Dimensional Data Analysis to Networks and Other Irregular Domains. *IEEE Signal Processing Magazine* 3, 30 (2013), 83–98.

[37] Yizhou Sun, Jiawei Han, Xifeng Yan, Philip S Yu, and Tianyi Wu. 2011. Pathsim: Meta path-based top-k similarity search in heterogeneous information networks. *Proceedings of the VLDB Endowment* 4, 11 (2011), 992–1003.

[38] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. Line: Large-scale information network embedding. In *Proceedings of the 24th international conference on world wide web (WWW'15)*. ACM, 1067–1077.

[39] Zhen Tu, Yali Fan, Yong Li, Xiang Chen, Li Su, and Depeng Jin. 2019. From Fingerprint to Footprint: Cold-Start Location Recommendation by Learning User Interest from App Data. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 3, 1, Article 26 (March 2019), 22 pages.

[40] Zhen Tu, Runtong Li, Yong Li, Gang Wang, Di Wu, Pan Hui, Li Su, and Depeng Jin. 2018. Your Apps Give You Away: Distinguishing Mobile Users by Their App Usage Fingerprints. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 2, 3, Article 138 (Sept. 2018), 23 pages.

[41] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2018. Graph attention networks. In *International Conference on Learning Representations*.

[42] Huandong Wang, Yong Li, Sihan Zeng, Gang Wang, Pengyu Zhang, Pan Hui, and Depeng Jin. 2019. Modeling Spatio-Temporal App Usage for a Large User Population. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 3, 1, Article 27 (March 2019), 23 pages.

[43] Huandong Wang, Fengli Xu, Yong Li, Pengyu Zhang, and Depeng Jin. 2015. Understanding mobile traffic patterns of large scale cellular towers in urban environment. In *Proceedings of the 2015 Internet Measurement Conference (IMC'15)*. ACM, 225–238.

[44] Pascal Welke, Ionut Andone, Konrad Blaszkiewicz, and Alexander Markowetz. 2016. Differentiating smartphone users by app usage. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp '16)*. ACM, 519–523.

[45] Wolfgang Woerndl, Christian Schueller, and Rolf Wojtech. 2007. A hybrid recommender system for context-aware recommendations of mobile applications. In *2007 IEEE 23rd International Conference on Data Engineering Workshop*. IEEE, 871–878.

[46] Le Wu, Peijie Sun, Yanjie Fu, Richang Hong, Xiting Wang, and Meng Wang. 2019. A neural influence diffusion model for social recommendation. In *Proceedings of the 42nd ACM conference on research and development in information retrieval (SIGIR'19)*. 235–244.

[47] Yongji Wu, Defu Lian, Shuowei Jin, and Enhong Chen. 2019. Graph Convolutional Networks on User Mobility Heterogeneous Graphs for Social Relationship Inference. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI-19)*. 3898–3904.

[48] Tong Xia and Yong Li. 2019. Revealing Urban Dynamics by Learning Online and Offline Behaviours Together. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 3, 1, Article 30 (March 2019), 25 pages.

[49] Tong Xia, Yue Yu, Fengli Xu, Funing Sun, Diansheng Guo, Depeng Jin, and Yong Li. 2019. Understanding Urban Dynamics via State-sharing Hidden Markov Model. In *The World Wide Web Conference (WWW '19)*. ACM, 3363–3369.

[50] Fengli Xu, Jianxun Lian, Zhenyu Han, Yong Li, Yujian Xu, and Xing Xie. 2019. Relation-Aware Graph Convolutional Networks for Agent-Initiated Social E-Commerce Recommendation. In *Proceedings of the 28th ACM International Conference on Information and*

Knowledge Management (CIKM '19). ACM, 529–538.
[51] Fengli Xu, Tong Xia, Hancheng Cao, Yong Li, Funing Sun, and Fanchao Meng. 2018. Detecting Popular Temporal Modes in Population-Scale Unlabelled Trajectory Data. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 2, 1, Article 46 (March 2018), 25 pages.
[52] Ye Xu, Mu Lin, Hong Lu, Giuseppe Cardone, Nicholas Lane, Zhenyu Chen, Andrew Campbell, and Tanzeem Choudhury. 2013. Preference, context and communities: a multi-faceted approach to predicting smartphone app usage patterns. In *Proceedings of the 2013 International Symposium on Wearable Computers (ISWC'13)*. ACM, 69–76.
[53] Tingxin Yan, David Chu, Deepak Ganesan, Aman Kansal, and Jie Liu. 2012. Fast app launching for mobile devices using predictive user context. In *Proceedings of the 10th international conference on Mobile systems, applications, and services (MobiSys '12)*. 113–126.
[54] Zhen Yang, Ming Ding, Chang Zhou, Hongxia Yang, Jingren Zhou, and Jie Tang. 2020. Understanding Negative Sampling in Graph Representation Learning. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '20)*. ACM.
[55] Hongyi Yao, Gyan Ranjan, Alok Tongaonkar, Yong Liao, and Zhuoqing Morley Mao. 2015. Samples: Self adaptive mining of persistent lexical snippets for classifying mobile application traffic. In *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking (MobiCom '15)*. ACM, 439–451.
[56] Donghan Yu, Yong Li, Fengli Xu, Pengyu Zhang, and Vassilis Kostakos. 2018. Smartphone App Usage Prediction Using Points of Interest. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 1, 4, Article 174 (Jan. 2018), 21 pages.
[57] Yue Yu, Yinghao Li, Jiaming Shen, Hao Feng, Jimeng Sun, and Chao Zhang. 2020. STEAM: Self-Supervised Taxonomy Expansion with Mini-Paths. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '20)*. ACM.
[58] Chao Zhang, Liyuan Liu, Dongming Lei, Quan Yuan, Honglei Zhuang, Tim Hanratty, and Jiawei Han. 2017. Triovecevent: Embedding-based online local event detection in geo-tagged tweet streams. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'17)*. ACM, 595–604.
[59] Chao Zhang, Keyang Zhang, Quan Yuan, Haoruo Peng, Yu Zheng, Tim Hanratty, Shaowen Wang, and Jiawei Han. 2017. Regions, Periods, Activities: Uncovering Urban Dynamics via Cross-Modal Representation Learning. In *Proceedings of the 26th International Conference on World Wide Web (WWW '17)*. 361–370.
[60] Chao Zhang, Keyang Zhang, Quan Yuan, Luming Zhang, Tim Hanratty, and Jiawei Han. 2016. Gmove: Group-level mobility modeling using geo-tagged social media. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16)*. ACM, 1305–1314.
[61] Sha Zhao, Zhiling Luo, Ziwen Jiang, Haiyan Wang, Feng Xu, Shijian Li, Jianwei Yin, and Gang Pan. 2019. AppUsage2Vec: Modeling Smartphone App Usage for Prediction. In *Proceedings of the 35th IEEE International Conference on Data Engineering*. IEEE, 1322–1333.
[62] Hengshu Zhu, Enhong Chen, Kuifei Yu, Huanhuan Cao, Hui Xiong, and Jilei Tian. 2012. Mining personal context-aware preferences for mobile users. In *2012 IEEE 12th International Conference on Data Mining*. IEEE, 1212–1217.

Received February 2020; revised May 2020; accepted July 2020