future internet

MDPI

Article

Graph-based Method for App Usage Prediction with Attributed Heterogeneous Network Embedding

Yifei Zhou [1], Shaoyong Li [1,*] and Yaping Liu [2,*]

[1] School of Computer Science and Engineering, Central South University, Changsha 410083, China; zhouyifei@csu.edu.cn
[2] Cyberspace Institute of Advanced Technology, Guangzhou University, Guangzhou 510006, China
* Correspondence: lishao378@sohu.com (S.L.); ypliu@gzhu.edu.cn (Y.L.)

Received: 12 February 2020; Accepted: 18 March 2020; Published: 20 March 2020

Abstract: Smartphones and applications have become widespread more and more. Thus, using the hardware and software of users' mobile phones, we can get a large amount of personal data, in which a large part is about the user's application usage patterns. By transforming and extracting these data, we can get user preferences, and provide personalized services and improve the experience for users. In a detailed way, studying application usage pattern benefits a variety of advantages such as precise bandwidth allocation, App launch acceleration, etc. However, the first thing to achieve the above advantages is to predict the next application accurately. In this paper, we propose AHNEAP, a novel network embedding based framework for predicting the next App to be used by characterizing the context information before one specific App being launched. AHNEAP transforms the historical App usage records in physical spaces to a large attributed heterogeneous network which contains three node types, three edges, and several attributes like App type, the day of the week. Then, the representation learning process is conducted. Finally, the App usage prediction problem was defined as a link prediction problem, realized by a simple neural network. Experiments on the LiveLab project dataset demonstrate the effectiveness of our framework which outperforms the three baseline methods for each tested user.

Keywords: app usage; prediction; attributed heterogeneous network; link prediction

1. Introduction

Mobile Apps have blossomed in popularity and ubiquity in the recent decade. According to the statistic by *buildfire* (https://buildfire.com/app-statistics/), there are over 2.7 billion smartphone users across the world and 90% of mobile time is spent on Apps. The Apple App Store has 2.2 million Apps available for downloading and 2.8 million on the Google Play Store. With so many Apps out there, lots of data about application usage patterns can be collected from smartphones in physical spaces and user preferences can be extracted from these data to improve user experience. At the same time, system support that can improve our daily App interaction experience is poised to be widely beneficial. However, smartphone users install an average of 80–90 Apps per person on their device under investigation and use not more than 30 Apps each month even including newly installed Apps. Therefore, it has been widely studied which applications will be used and which do not need to be used.

App usage prediction refers to the task of predicting the next App that will be used for a given user and at a given time [1]. An accurate prediction can not only improve the user experience but also can be used to optimize startup acceleration, energy consumption, bandwidth allocation, etc. For instance, smartphones can pre-load the possible Apps into the memory for reducing the startup time of Apps if we can predict the exact next App the user is the most likely to launch. The general

approach is to transform the data of the physical world into the cyberspaces to build models. In recent years, researchers have used various methods to build models for prediction with information collected from mobile terminals and combined online and offline learning to improve prediction accuracy, such as taking advantage of temporal information [2,3] or mining application usage patterns [4] to predict. However, there is apparent diversity of application-related features on smart devices. Furthermore, both the number of Apps users only use seldom even just once and the amount of data containing user portrait, temporal information, location information, mobile phone brand, and other related information are growing. Machine learning methods like LSTM (Long Short-Term Memory) do not perform very well in this complex context.

Many previous studies have tackled the App prediction problem by means of network embedding, which can better mine the relationship between different types of applications and other information like time, location, etc. Tan et al. [5] put forward extracting features and relationships from user-App bipartite. This bipartite not only describes the correlation between user and App but also expresses conjunctions of Apps used by a typical user or users who used the same application. However, limited information was exploited in the method because there is no contextual information. Previous endeavors have revealed that place and temporal information or the latest used App all benefit App prediction [6]. Chen et al. [7] proposed one method named CAP, which constructs one network representing App usage records and contains different types of nodes and edges with all users' records. Then, build users' profiles for users' personalized predictions. However, CAP does not use attributed information, so it is not friendly with the prediction of new nodes.

This paper proposes a framework named AHNEAP based on attributed network embedding for predicting the next App to be used for each user, by characterizing the context information before one specific App being launched. At present, we use data of the single user for experiments about our framework, but AHNEAP is also suitable for training multi-user data and predicting. Thus, data of each user will be uploaded to the cloud firstly. Then, our proposed framework transforms the historical App usage records in physical spaces to a large attributed heterogeneous network and the representation learning process is conducted, which can reduce the number of feature dimensions while making full use of various information to mine potential features. Eventually, the App usage prediction is formulated as a link prediction problem based on a Neural Network. More concretely, our main contributions in this paper to App usage prediction are three-pronged:

- First, we introduce network embedding to App prediction problem. By converting application, location, and temporal information into different types of nodes in a heterogeneous network and mapping related attribute information into the same potential space, we can capture the relationship between Apps and time or location or the previous App.
- Second, we propose a novel App usage prediction framework named AHNEAP based on representation learning on the attributed heterogeneous network. The framework contains three major components: data pre-processing, representation learning, and link prediction. The data pre-processing generates an attributed heterogeneous network from historical App usage records. Then, representation learning produces universal embedding features for elements related to the App usage e.g., App name and timestamp. Finally, the link prediction process completes the fusion of embedding of the time, location, and the previous application obtained from the previous step firstly, then constructs a neural network to decide the importance of the three types to predict the most likely used App within a certain period of time.
- Finally, we conduct extensive experiment evaluations with the *LiveLab* App usage dataset, and demonstrate and analyze the prediction performance about the top-k candidate applications. Experiments show that AHNEAP outperforms the baseline methods: Most Frequently Used (MFU), Most Recently Used (MRU), and traditional Bayes.

The rest of the paper is organized as follows: Section 2 provides a coarse introduction of related work. Section 3 shows our graph-based App usage prediction framework. In Section 4, extensive

experiments to evaluate the performance of the new model are conducted and analyzed. Finally, we summarize and conclude this paper in Section 5.

2. Related Work

In this section, we will give a brief introduction of the existing works related to this paper, including research on analysis of App usage prediction and the guide to network embedding.

2.1. App Usage Prediction

There have been many prior research works about predicting smartphone App usage, most of which are based on contextual information. An early representative type of methods are based on probabilistic graphical model, such as CPD (conditional probability distribution) model [8], Markov model [9,10], and Bayesian Network [6,11,12]. Given the dependency of consecutive Apps and their immediate previous Apps in the App usage sequence, it is often assumed that Markovian property stands. For instance, Zou et al. [10] proposed using Markovian models to learn the App usage sequences, and these models included first and second-order Markov models, along with a weighted linear combination of these two models. Considering more than one type of context like time and location and so on, some works take advantage of Bayesian Network to describe the relationship between the App running and these contexts. Baeza et al. [11] propose an effective personalized classification method to solve the App prediction that takes full advantage of human-engineered features and automatically derived features. In [12], Shin et al. developed a dynamic home screen App (dynamic home) on Android and adopted a Naive Bayes classification model to predict App usage behaviors by considering possible sensors in a smart phone.

In general, context-based recommendation technology requires the fusion of multi-party information, and the fusion process of heterogeneous data from multiple sources is complicated. Therefore, there are studies to directly design algorithms based on the context information of the application to predict the next application to be launched. Parate et al. [13] applied the text compression algorithm named PPM to handling with the application startup sequence, and proposed the M algorithm. Rahnamoun et al. [14] establish an online learning mechanism based on the transition probability between applications, predict the applications to be used and cache them in advance to reduce startup time and save equipment resource consumption. Chen et al. [7] propose a heterogeneous graph embedding algorithm to map the context information and calculate each App's probability to be used.

In addition, with the development of Neural Networks in recent years, some works began to adopt this method. Xu et al. [15] proposed a LSTM based multi-label classification model for App usage prediction, which explores the temporal-sequence dependency and contextual information as features for prediction. Xu et al. [16] revealed that the similarity of users' preferences is related to the similarity in their App usage context patterns. Thus, they propose a neural network that models both a user's preference over apps and the user's app usage context pattern to learn the embeddings of both users and Apps and then predicts a user's preference for a given App.

In addition, methods based on frequent sequence pattern mining are also used in some works for App usage prediction [17,18]. In [5], the authors construct the User-App Bipartite network based on the network footprint data and propose the App Usage prediction method based on link prediction. However, the prediction period was set to one day, which is far from practical, and the deep learning based on representation learning was also not involved.

2.2. Network Embedding

Network embedding [19], or network representation learning, is a method to project nodes in a network to a low dimensional continuous space while preserving network structure and inherent properties. Works in the network embedding mainly consist of two categories, graph embedding (GE) and graph neural network (GNN). Representative works for GE include DeepWalk [20],

which generates a corpus on graphs by random walk and then trains a skip-gram model on the corpus. Node2vec [21] designs a biased random walk procedure to efficiently explore diverse neighborhoods. NetMF [22] is a unified matrix factorization framework for theoretically understanding and improving DeepWalk and LINE. For popular works in GNN, GCN [23] integrated neighbors' feature representations into the node feature representation using convolutional operations. GraphSAGE [24] provides an inductive approach to combine structural information with node features. It learns functional representations instead of direct embeddings for each node, which helps it work inductively on unobserved nodes during training.

Due to the ubiquitous network in the real world, network embedding has been paid more and more attention in recent years and used to represent all areas of information. David et al. formalize the question of new interactions among its members in a social network as the network embedding problem, and developed approaches to link prediction based on measures for analyzing the proximity of nodes in a network [25]. Lin et al. proposed an augmented relation embedding to map the image space into a semantic manifold, transforming learning a semantic manifold into solving a constrained optimization problem [26]. A Social Network Embedding algorithm achieved the prediction of friend relation and citation relation by extracting the feature representation of a network node through the fusion of node ID and attribute vector [27]. BIGCLAM is proposed as a covering of the community discovery algorithm for the detection of overlapping societies, which generates a probability of each edge in the network model [28].

There has also been lots of effort to devote to the study of recommendation. Zhao et al. proposed to use the k-partite adoption graph to characterize various kinds of information in recommendation tasks for addressing recommendation tasks by utilizing the network representation learning techniques [29]. Previous work proposed a new product graph embedding model, which used the network representation learning technology to capture the sequential impact of products by converting historical purchase records into product diagrams [30]. A new collaborative user network embedding method is proposed to extract implicit and reliable social information from user feedback such as ratings or purchases, which transforms feedback into a user-item bipartite graph [31].

From this perspective of network embedding, we can transform App, time, location information into attributed nodes when predicting the next App in the next hour, and construct a graph using these nodes. In the graph, a link represents the relationship of adjacent nodes. For instance, if there is a link between a time node and an App node, it indicates that the user used a certain App at the time. In that way, our problem is actually turned into predicting whether the time node or other node is connected to the App node in the future and sorting the possibilities of establishing a connection. However, although a graph is constructed in the training process, we still utilize some new nodes like new temporal nodes isolated from other vertices in the graph to predict actually. Methods based on the random walk can describe the relationship between App nodes and time nodes or location nodes separately, but they can't solve the problem of new nodes while attributed network embedding can. Because attributed network embedding aims to seek lower-dimensional representations of vector for nodes in a network, such that the original network topological structure and node attribute proximity can be preserved in such representations. Cen et al. revealed the problem of embedding learning for the attributed multiplex heterogeneous network and proposed a unified framework to address this problem of new nodes, which supports both transductive and inductive learning [32]. GATNE-I model proposed in [32] solves the problem of the coupling of new nodes and existing nodes.

However, both random walk and attributed network embedding can't solve the problem of integration of various types of edges. Therefore, in this paper, we'll adopt a GATNE-I model as our network embedding method to develop our framework, and integrate the contextual information such as time and location to study the relationship between different nodes on this basis.

3. Graph-Based App Usage Prediction

App Prediction Problem is formally defined as follows: Given a list of installed Apps $apps_u = \{a_1^u, a_2^u, ..., a_n^u\}$ by a user u on his/her phone and the user's context C including spatiotemporal information and smartphone's state like battery level. The problem of App usage prediction is to find an App a_i^u that has the largest probability of being used under C. Specifically, we aim to solve the problem:

$$\max_{a_i^u} P(a_i^u|C, u), \forall i, 1 \leq i \leq n. \quad (1)$$

Based on the above prediction mechanism, our prediction task will resolve around predicting App usage of a typical user. To accurately predict App usage, it is crucial to be able to characterize the features before the App being used by utilizing relevant usage records to the observer. We propose AHNEAP, a framework based on network embedding which firstly transforms the historical App usage records into a large attributed heterogeneous network to obtain every node's embedding, and then trains a neural network to integrate various types of context. Figure 1 illustrates the proposed framework, which contains three major components: data pre-processing, representation learning, and link prediction.

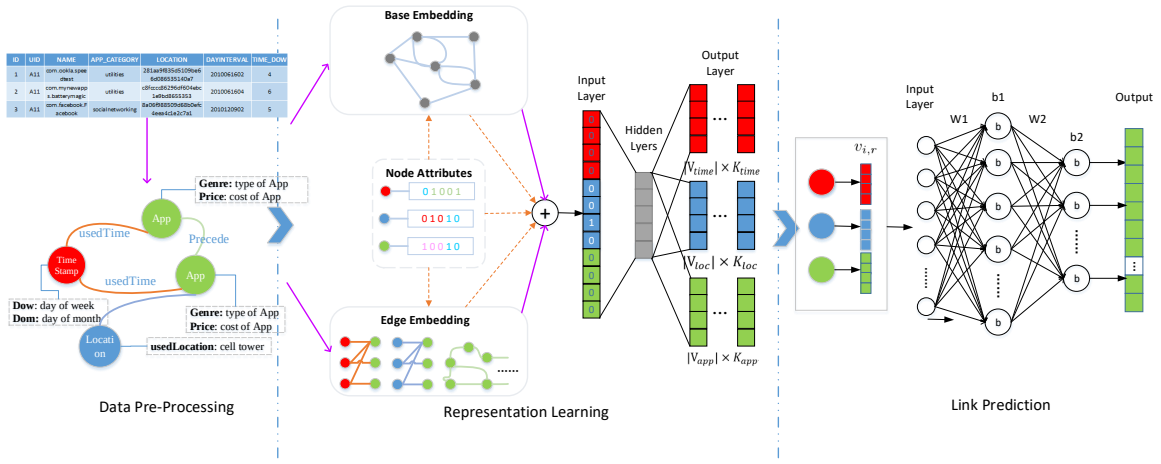


Figure 1. Network embedding based framework for App usage prediction.

3.1. Data Pre-Processing

The aim of data pre-processing is to generate an attributed multiplex heterogeneous network from historical App usage records. An attributed multiplex heterogeneous network is a network $G = (V, E, A)$ associated with a node type mapping function $\phi : V \rightarrow O$ and an edge type mapping function $\psi : E \rightarrow R$, where O and R represent the set of all node types and the set of all edge types, respectively, satisfying $|O| + |R| > 2$ [32]. Each node $v \in V$ belongs to a particular node type and is associated with some types of feature vectors. $A = \{x_i|v_i \in V\}$ is the set of node features for all nodes with attributes, where x_i is the associated node feature of node v_i. Each edge is categorized into a specific edge type.

In our scenario of App usage, typical usage records include basic information like an anonymous user ID, a connected cellular base station ID which we regard as location information, a timestamp, and applications with their names and categories. For a typical user, there are three node types in the heterogeneous network: time, location, and application. Node pairs are divided into three different groups: *"usedTime"* relationship between time and App, *"usedLocation"* relationship between location and App, *"precede"* relationship between the previous App and App. Every type of relationship can be regarded as one subgraph. Obviously, *"time"* and *"App"* nodes have intrinsically different properties and shall not be treated equally. Moreover, different interactions between node pairs imply different

levels of features and should be treated differently. Otherwise, the system cannot precisely capture the user's behavioral patterns and preferences and would be insufficient for practical use. In addition, the time node and App node also take some useful attributes.

Figure 2 shows an example of a part of an attributed heterogeneous network extracted from the historical App usage records. It should be noted that the *day of week*, *day of month*, and one period of one-day *interval of day*, and so on when using Apps are treated as attributes of time nodes if they are not be used to represent time nodes. For example, from 12:00 a.m. to 6:00 a.m., the value of *interval of day* is 1. The App type and price are attributes of the App nodes. Finally, we choose just cellular base station information to represent locations temporarily.

There will be a more detailed description of the dataset in Section 4.1.

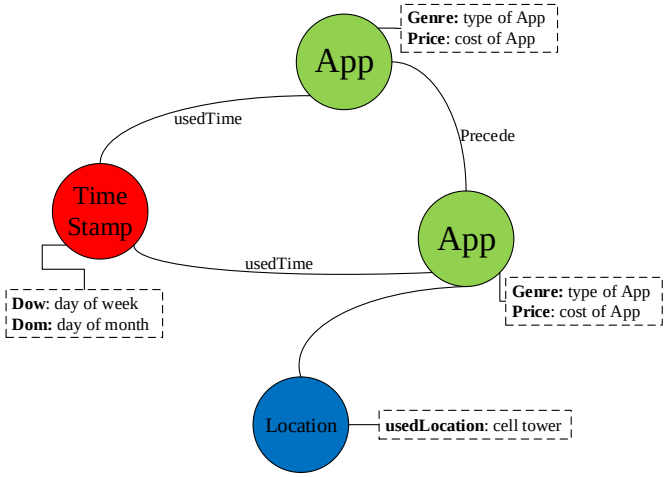


Figure 2. An example attributed heterogeneous network for App usage prediction.

3.2. Representation Learning

The application prediction issue is related to future factors such as time and location, which will generate new temporal nodes constantly. Therefore, we use the *GATNE-I* model proposed in [32] to capture both topological structures from different types of node, edge, and attributed information on the large attributed heterogeneous network for the representation learning. This model can handle nodes that are not seen during training that is suitable for App usage prediction because the future time is definitely new nodes never seen before.

In the *GATNE-I* model, the overall embedding of a certain node v_i on each edge type r is:

$$v_{i,r} = h_z(x_i) + \alpha_r M_r^T U_i a_{i,r} + \beta_r D_z^T x_i. \quad (2)$$

x_i is defined as an attribute vector of the node v_i and $h_z(x_i)$ is a transformation function, whose role is to add its own characteristics to the final result and its base embedding for node v_i. $a_{i,r}$ is computed according to the self-attention mechanism [33] as the coefficients of a linear combination of vectors. D_z is a feature transformation matrix on v_i's corresponding node type z, designed for the purpose of new nodes isolated from all nodes in the existing graph. U_i is defined as a matrix, which integrates three types of edges' embedding of one node respectively in three sub-graphs so that every node's embedding in different sub-graphs can be calculated for each edge type during the model training process. The embedding of the three edges of each node is represented by the mean value of the attributes of a certain number of neighbor nodes corresponding to the three graphs.

In this paper, there are three types of edges: application and time, application and location, application and the previous application, where we use edges separately in "usedTime" sub-graph, "usedLocation" sub-graph, "precede" sub-graph to express three types of relationships. $u_{i,t}, u_{i,l}, u_{i,p}$

express edge embedding of one node respectively in "usedTime" sub-graph, "usedLocation" sub-graph, "precede" sub-graph, and we use the mean of a fixed number of neighbors' embedding to represent edge embedding of one node in one type of sub-graph. In the future, the integration of all neighbor embedding or the multi-hop neighbor embedding can be used as one part of the embedding of the nodes for the propose of comparison. Thus, in Equation (2), $U_i = (u_{i,t}, u_{i,l}, u_{i,p})$.

Taking "usedTime" sub-graph as an example, due to only a connection between application and time, there is no link between application nodes or time nodes themself. Thus, the k-th level edge embedding $u_{i,t}^{(k)} \in R^s, (1 \leq k \leq K)$ of time node v_i in the "usedTime" sub-graph is aggregated from neighbors' edge embeddings as:

$$u_{i,t}^{(k)} = \sigma(W^{(k)} \cdot mean(\{u_{j,t}^{(k-1)}, \forall v_j \in N_{i,t}\})), \tag{3}$$

where $N_{i,t}$ is the neighbors of time node v_i in the "usedTime" sub-graph and neighbors are application nodes. Attributes of time nodes reflect all applications used in a specific time. Similarly, the neighbors of App node in the "usedTime" sub-graph are application nodes. Attributes of App nodes reflect all the execution time of one application. Then, we define $v_{i,t}, v_{i,l}, v_{i,p}$ as embedding of time, location and the previous App nodes in the $i-th$ record. Through network embedding, three types of embedding of node v_i can be obtained:

- $v_{i,t} = \{v_{i,t}^1, v_{i,t}^2, ..., v_{i,t}^n\}$, in "usedTime" sub-graph
- $v_{i,l} = \{v_{i,l}^1, v_{i,l}^2, ..., v_{i,l}^n\}$, in "usedLocation" sub-graph
- $v_{i,p} = \{v_{i,p}^1, v_{i,p}^2, ..., v_{i,p}^n\}$, in "precede" sub-graph

where $v_{i,t}^n$ represents the $n-th$ element in the vector $v_{i,t}$, and $v_{i,l}^n$, $v_{i,p}^n$ do so. However, it was noted that each type of embedding vector can only be used in the corresponding sub-graph, even though there are three embeddings generated with one node finally. For instance, for temporal nodes, only embedding in "usedTime" sub-graph is working, while embedding in "usedLocation" or "precede" sub-graph is invalid. However, it is different for App nodes. Every App node will have three effective embeddings because the other three types of the node are involved in App nodes.

3.3. Link Prediction

Through the GATNE-I model, we transform App usage records into a latent space to construct a network and obtain embedding of each node with representation learning. We will calculate the similarities of embedding vectors of two nodes using Adjusted Cosine Similarity so that we can infer whether there is a linking between two nodes. If two vectors have a high degree of similarity, then the probability that they will be connected in the future is high. As an example, given a spatial context l for a specific user u, linking between l and a means u used a at the location l and not-linking means the user has not used this App in the past. If a new location l' is given and is similar to l, there is a certain possibility that user u will use a at l'. In other words, there may be a linking between l' and a. Here, the App prediction problem is reformulated as link prediction problem in the complex network, that is, linking or not between two nodes.

There are three types of sub-graphs in the new latent space, which contain the relationship between application and time or location or the previous App respectively if only the GATNE-I model is used. On this basis, we can predict the next application in the light of information about time or location or the previous App, but just only one relationship can be made use of per prediction. Nevertheless, there are some works revealing that information about time or location or user profile like the phone in silent mode [6] all have an impact on the next App prediction in the future duration. Hence, for the sake of improvement of accuracy, we need to integrate all the three types of information and decide the importance of three types in the process of fusion.

Above all, one record r includes four nodes: time, location, the previous App, and the current App. Through representation learning in Section 3.2, $v_{r,t}, v_{r,l}, v_{r,p}$ can be obtained, which indicate

embedding of time node, location node, and the previous App node in the $r-th$ record. In AHNEAP, we adopt a neural network with a single hidden layer to handle integration. In the hidden layer, we choose the sigmoid function as the activation function. We concatenate the three embeddings of time, location, and the previous App nodes together for each record as input:

$$v_r = (v_{r,t}, v_{r,l}, v_{r,p}) \tag{4}$$

where r is a serial number indicating a certain record. Then, take the one-hot value of each application as the output. In that way, the probabilities of all application installed at the context $v_{r,t}$, $v_{r,l}$ and $v_{r,p}$ are:

$$q_r = W^{(2)} \cdot sigmoid(v_r \cdot W^{(1)} + b_1) + b_2 \tag{5}$$

where q_r is a probability vector of all applications calculated in the $r-th$ case, $W^{(1)}$ and $W^{(2)}$ are transformation matrices corresponding to layers in the neural network. In addition, we definite p_i as the actual observed distribution, which actually is the one-hot value corresponding to the App user used.

Before minimizing the distance between the probability distributions and really observed distributions, we define the loss function to calculate divergence:

$$H = -\sum_{j=1}^{n} p_r^j log(q_r^j) \tag{6}$$

where n is the number of applications installed, p_r^j is the probability of the $j-th$ application from real distributions, and q_r^j is from the result of the output layer.

The above processes are comprised of representation learning and neural network. The embedding captures both topological structures of App usage interaction from different types of nodes and attributed information, and the neural network integrates all types of relationships. Based on the probability ranking of different App nodes, we can speculate several Apps, one of which will be launched in the future.

4. Experimental Evaluation

This section presents an experimental evaluation of the performance of the proposed network embedding based prediction approach based on comprehensive experiments on the App usage dataset of the LiveLab project [34].

4.1. Evaluation Setup

4.1.1. Dataset

LiveLab (http://livelab.recg.rice.edu/traces.html) dataset is generated by a number of iPhone 3GS users including 24 Rice University students from February 2010 to February 2011 and 10 Houston Community College students from September 2010 to February 2011. The dataset consists of several SQL files, which describe applications run by users, periodic output from the modem regarding the cellular base station the phone is connected to, phone calls made or received by users, charging state of the phone so on.

In our experiments, we choose two traces: App usage trace which records applications run by users and corresponding launching time, cellular base station trace which records the cell tower id that the phone is connected and time the event occurred. For each user, we extract three types of nodes, time, location and application, and links between application and time or location or the previous App, respectively, from this information.

In the pre-processing, we first remove all the cellphone's basic Apps including *SpringBoard* (Desktop), *com.apple.mobileSMS* (Short Messaging Service) and *com.apple.mobilephone* (Phone Call).

These three basic applications take up more than 900,000 of the total 1.3 million usage records. Then, we selected five users with the maximum usage records: A07, A04, A12, B04, B02, each of which has more than 15,000 records. Next, the attributed heterogeneous networks were constructed for each user at two temporal scales as time nodes: *hour of day* and *interval of day*. Each day was divided into four intervals, denoted by 1 (from 12:00 a.m. to 5:59 a.m.), 2 (from 6:00 a.m. to 11:59 a.m.), 3 (from 12:00 p.m. to 5:59 p.m.), and 4 (from 6:00 p.m. to 11:59 p.m.). As attributes of time nodes, *day of month*, *day of week*, and *interval of day* all participate in training. We will perform a comparison experiment for the two scales. Then, for each user, we sort the application usage records by ascending order of time respectively for the sake of finding the previous App. Finally, we extract cellular base station information as the location nodes. Because the cellular base station is recorded separately from application usage information, we select the cellular information during five minutes before the application start and five minutes after the end of the application for every App usage record and decide to select the result that is closest to the middle time of the application. At present, the one-hot value of each cellular base station is the attribute corresponding to each location node.

In this paper, we do not highlight how to select attributes as nodes' features and how to set parameters in our experiment, which is the future work. We mainly compare AHNEAP with baselines.

4.1.2. Performance Metrics

We use the *Accuracy* to evaluate the effect of the proposed network embedding framework. In addition, we also use *F*1 score that combines *Precision* and *Recall* to evaluate the overall performance of AHNEAP since we consider both *Precision* and *Recall* are important for any prediction mechanism. The two measurements *Precision* and *Recall* are adopted for measuring the proportion of actual positive samples in the positive examples calculated by AHNEAP, and the proportion of positive samples determined by our framework AHNEAP in all examples:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}, \tag{7}$$

$$Precision = \frac{TP}{TP + FP}, \tag{8}$$

$$Recall = \frac{TP}{TP + FN}, \tag{9}$$

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall}, \tag{10}$$

where TP, TN, FP and FN represent *TruePositive*, *TrueNegative*, *FalsePositive*, and *FalseNegative*, respectively.

In this section, we mainly measure the accuracy of various methods and AHNEAP, and all evaluations calculate the hit ratio of Apps corresponding to the $top - k$ scores. As long as the score of real used App is within the predicted $top - k$ scores, we count it as one hit. Then, *Accuracy*@*k* [7] indicates the accuracy when predicting the $k - th$ candidate applications. We mostly choose *Accuracy*@4 for comparison.

4.2. Comparison Methods

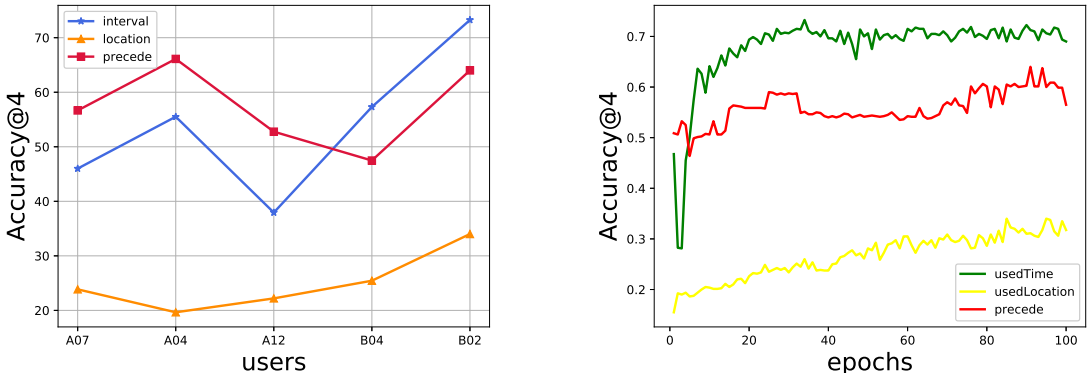
Regarding the problem of App prediction about the LiveLab dataset, there are not many related experiments. Liu et al. [35] mainly use data to study user application usage patterns, such as frequency of application usage, without focusing on App prediction. The main research point of the article [4] is the order about application startup. Lu et al. [36] use a tree to describe the relationship between App and time and space, while AHNEAP uses an attributed multiplex heterogeneous network. Previous work generated nine candidate applications when predicting, which is not necessary for the purpose of loading the application in advance [37]. Therefore, we choose some basic baselines based on some methods [6,10,12,17]. Shin et al. [12] and Eric et al. [17] adopt MFU and MRU, and Ke Huang et al. [6]

adopt the Bayes network at the same time. In order to illustrate the advantages of AHNEAP, we will compare with the following commonly used baselines:

- Most Frequently Used (MFU): The MFU method counts the users' history of mobile App usage and selects the most frequently used ones.
- Most Recently Used (MRU): The MRU method counts the users' history of mobile App usage and selects the most recently used ones.
- Bayes Network Model [6]: In the traditional Bayes model, the input feature is a tuple $(f_l, f_h, f_d, f_p, f_u)$ or $(f_l, f_i, f_d, f_p, f_u) f_l$: the location when using Apps, f_h: the hour of day when using Apps, f_i: the interval of day when using Apps, f_d: the day of day when using Apps, f_p: the latest used App, f_u: a flag whether the battery is being charged when using Apps as user profile extracted from the historical records. This model is improved based on the traditional Bayes model with LivaLab dataset.
- Graph Embedding (GE): We'll predict the next application through only one of the time, location, and the previous application, purely using the GATNE-I model. In this method, we calculate the similarity of three types of relationships according to Adjusted Cosine Similarity separately and calculate the accuracy. Then, we choose the maximum as the final rate of each application.

4.3. Performance Analysis

Performance of GE. Figure 3 shows the comparison of prediction accuracy for the GATNE-I model alone, where we use the *interval of day* scale as time node. We regarded the maximum accuracy during all epochs as the result. The evaluation calculates *Accuracy*@4 respectively in three types of sub-graphs. The figure shows accuracy when predicting four candidate Apps with five users. According to the above experiments of five users, we can preliminarily infer that the influence of various factors on the prediction is related to the user's usage habits, and each user is more or less different from others. Take B02 as an example first. As Figure 3 shows, obviously time and the latest App have more impact on prediction than location. The accuracy of B02 is finally stabilized at about 70% with "usedTime" type, while about 60% with "precede" type. Although accuracy with "usedLocation" type is on the rise, the rate is just 33%, far from 70%. Furthermore, the magnitude of the upward tendency is slow.



(**a**) For users: A07, A04, A11, B04, B02 (**b**) For user B02 in all epochs

Figure 3. The comparison of prediction *Accuracy*@4 for the GATNE-I model alone, GE: App and time, App and location, App and the previous App at *interval of day* scale.

In general, using time information to predict Apps is better than location information or previous Apps with user B04 and B02; however, results of users A07, A04, and A12 are different.

Performance of our framework. We'll perform the *Mean* method as contrast experiments, which is based on the GATNE-I model. The *Mean* method firstly gets the similarity of each App node and

time node, location node, and the latest App node, respectively, from three sub-graphs. Then, calculate the mean of three similarities with the same App as the final score of each application, and find the top-k application based on the similarity ranking. Figure 4 shows the *Accuracy*@4 of five users with the *Mean* method and AHNEAP. As the figure shows, there is higher accuracy using AHNEAP than the *Mean* method, which achieves the highest 80% for user B02 and improves by highest about 20% for user A07. Therefore, according to all our experiments of five users, we can preliminary infer that this conclusion is universal that AHNEAP outperforms the *Mean*.

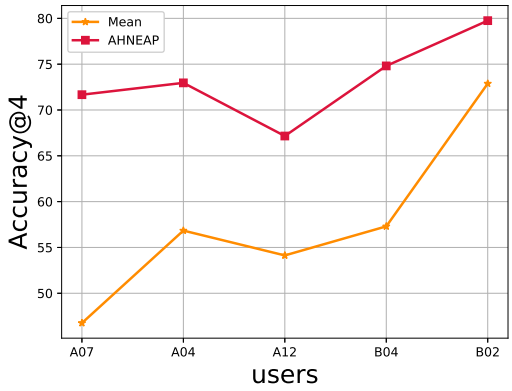


Figure 4. The comparison of prediction *Accuracy*@4 for the *Mean* and our framework AHNEAP with five users at *interval of day* scale.

Seen from all the whole process of the *Mean* method and our framework AHNEAP in Figure 5, there is only slight undulation about accuracy during all epochs in AHNEAP while the *mean* is not. In other words, we can achieve a stable state through fewer epochs using AHNEAP than the mean method, further consuming fewer resources. Thus, both in terms of accuracy and resources, we still choose the neural network to fit our context. Otherwise, the *mean* is difficult to choose the proportion of three types of parameters, but we might as well directly train the weight matrix.

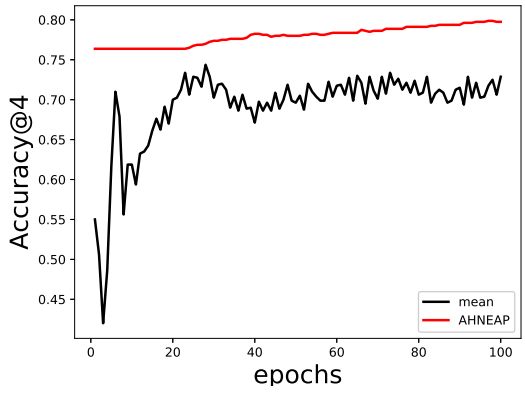


Figure 5. Prediction accuracy of B02 in all epochs.

Performance of comparison of interval and hour scale. Figure 6 shows that taking *interval of day* as time nodes are almost better than *hour of day* with five users except A12 when predicting top-4 candidate applications overall. However, it just achieves the highest about 2% for user A12 at *hour of day* temporal scale than at *interval of day* temporal scale. For users A07, A04, and B04, the accuracy is improved only by less than 1%. However, the accuracy increases by about 7% for user B02 at the scale of *interval of day*. According to the above experiments of five users, we indicate that the two temporal scales have roughly the same impact although there are little deviations for each user. In addition, fewer node pairs are needing to be trained when transforming interval information to time node

because there are fewer time nodes generated in the training process. However, from the aspect of effectiveness for a given period, the *interval of day* scale is not as effective as the *hour of day*. Which scale to choose for prediction depends on your focus.

Performance of comparison of our framework AHNEAP and baselines. Figure 7 shows the comparison of prediction accuracy for AHNEAP and baselines: MFU, MRU, Bayes, Graph Embedding (GE), and our network embedding based framework (AHNEAP) for the selected five users at two temporal scales separately. For baselines and our framework AHNEAP, the prediction accuracy varies with different users. However, the experimental results show that our framework AHNEAP outperforms other strategies significantly almost in any situation at the two temporal scales. It achieves highest about 80% for user B02 at *interval of day* temporal scale and 74% for user B04 at *interval of day* temporal scale.

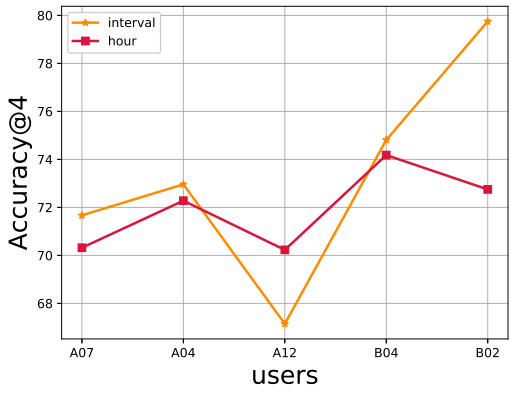


Figure 6. The comparison of two types of time patterns with prediction *Accuracy*@4 of five users: *interval of day*, *hour of day*.

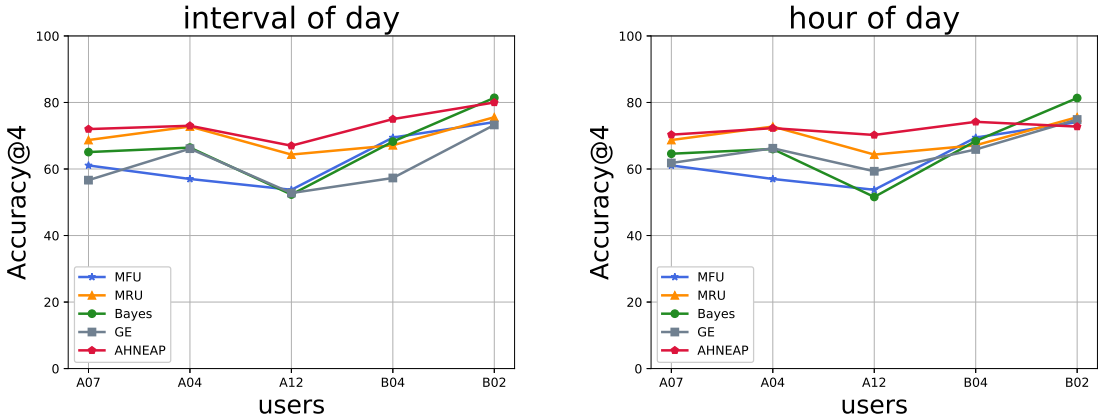


Figure 7. The comparison of prediction *Accuracy*@4 for the five models: MFU, MRU, traditional Bayes (Bayes), Graph Embedding (GE), and our network embedding based framework(AHNEAP) for the five users, respectively, at *interval of day* and *hour of day* temporal scale.

However, at *hour of day* scale, the prediction accuracy is not higher than all the four baselines with user B02. Above all, we analyze Figure 8, which describes application usage record statistics about B02. There are 14,565 application usage records of B02 and 56 applications installed. Seen from the first sub-graph, there are four Apps most frequently used. In addition, records about the top 10 most frequently used applications occupy about 91% of all records. Regardless of MFU or Bayes, the prediction accuracy is calculated based on the probability of the application usage in a certain context. About 91% of records are related to the top 10 applications, so that it is possible that MFU or Bayes can produce a higher rate than our framework AHNEAP. However, benchmarks are not friendly for newly installed apps while AHNEAP can increase possibility through the similarity

between applications, which is calculated by their features. Therefore, our framework AHNEAP is very adaptable in various situations. Even though, in the case of B02, AHNEAP is only slightly worse than the baselines.

Details of Accuracy@3–5: There is more detailed information about our experiments. Table 1 shows the *Accuracy* and *F*1 scores for each user under the two temporal scales when predicting three, four, and five candidates, leading to three observations.

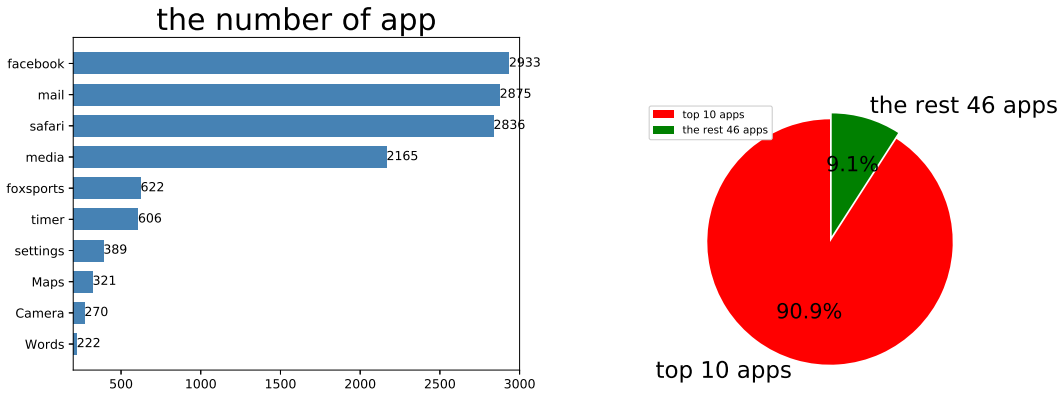


Figure 8. The figure shows the statistical distribution of application usage with user B02.

Table 1. Performance details of the proposed method with top-3 prediction candidates.

		Interval		*Hour*	
User	**Top-**	**ACCURACY**	**F1 Score**	**ACCURACY**	**F1 Score**
A04	3	0.67	0.78	0.68	0.79
	4	0.73	0.83	0.72	0.82
	5	0.78	0.86	0.77	0.85
A07	3	0.64	0.75	0.62	0.74
	4	0.72	0.82	0.70	0.81
	5	0.76	0.85	0.75	0.84
A12	3	0.60	0.72	0.63	0.75
	4	0.67	0.78	0.70	0.81
	5	0.71	0.81	0.75	0.84
B02	3	0.73	0.84	0.66	0.77
	4	0.80	0.87	0.73	0.82
	5	0.85	0.91	0.79	0.87
B04	3	0.69	0.79	0.66	0.77
	4	0.75	0.84	0.74	0.83
	5	0.80	0.88	0.77	0.86

To sum up, firstly the prediction accuracy of the method is on the increase with the k value increasing. This finding is easy to understand that, with the k value increasing, the denominator of the ratio stays the same. However, the numerator (hit counts) can only keep the same or increasing. Second, the prediction accuracy and F1 score have no inevitable correlation with the input training size according to each user's statistic in Table 2. Thus, even if it is increased when the amount of training data exceeds a certain limit, the accuracy will not be greatly improved. Third, performance under *interval of day* temporal scale is usually better than that of *hour of day* temporal scale. The results for user A12 are the only exception. However, from the aspect of effectiveness for a given period, the *interval of day* scale is not as effective as the *hour of day*.

Table 2. Statistics of the network dataset for experiments.

	Hour			Interval		
	Train		Test	Train		Test
User	**#nodes**	**#edges**	**#node Pairs**	**#nodes**	**#edges**	**#node Pairs**
A07	3822	28861	482	1278	24178	414
A04	4062	22570	370	1247	17813	307
A12	4048	28347	480	1423	24441	418
B04	3925	25540	432	1369	20716	359
B02	3478	24103	405	1251	19981	344

5. Discussion

This paper presents a network embedding based framework, AHNEAP, which leverages large scale digital App usage records to reveal the underlying patterns, and formulates the App usage prediction as a link prediction problem. Extensive empirical experiments with data from the *LiveLab* project demonstrated the effectiveness of our framework AHNEAP by comparing it mainly with *Most Recently Used* model and Bayes model.

We plan to conduct our future work in the following several aspects. First, we can use all the data records to implement embedding representation of heterogeneous networks with attributes and use users' historical records to obtain the neural network model for personalized prediction, but not focusing on a specific user. Second, there is a lot of information in the dataset that is not used in this paper. For example, this is too monotonous taking the one-hot value of cellular base station as attributes of location nodes. Attributes of time nodes do. We can consider how to use WiFi information and cellular signals to express a location in more detail on the basis of obtaining a more efficient data set. Third, in all experiments, we did not pay attention to the adjustment of the model parameters. We should design some experiments to observe the influence of parameters. Finally, we need to try more combinations of different scales of time granular when training models and making predictions.

Author Contributions: Methodology, S.L.; Writing–original draft, Y.Z.; Software, Y.Z., S.L.; Writing–review & editing, S.L.; Project administration, Y.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work is supported by the National Key Research and Development Program of China under Grant No. 2018YFB1003602, and the Key Research and Development Program of Guangdong province under Grant No. 2019B010137005.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Cao, H.; Lin, M. Mining smartphone data for App usage prediction and recommendations: A survey. *Pervasive Mobile Comput.* **2017**, *37*, 1–22. [CrossRef]
2. Kostakos, V.; Ferreira, D.; Goncalves, J.; Hosio, S. Modelling smartphone usage: A markov state transition model. In Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing, Heidelberg, Germany, 12–16 September 2016; ACM: New York, NY, USA, 2016; pp. 486–497.
3. Liao, Z.X.; Lei, P.R.; Shen, T.J.; Li, S.C.; Peng, W.C. Mining Temporal Profiles of Mobile Applications for Usage Prediction. In Proceedings of the 2012 IEEE 12th International Conference on Data Mining Workshops (ICDMW), Brussels, Belgium, 10 December 2012; IEEE: Piscataway, NJ, USA, 2012.
4. Yan, T.; Chu, D.; Ganesan, D.; Kansal, A.; Liu, J. Fast App Launching for Mobile Devices Using Predictive User Context. In Proceedings of the 10th International Conference on Mobile Systems, Applications and Services (MobiSys 2012), Low Wood Bay, UK, 25–29 June 2012.
5. Tan, Y.; Yu, K.; Wu, X.; Pan, D.; Liu, Y. Predicting app usage based on link prediction in user-app bipartite network. In *International Conference on Smart Computing and Communication*; Springer: Cham, Switzerland, 2017; pp. 191–205.

6. Huang, K.; Zhang, C.; Ma, X.; Chen, G. Predicting mobile application usage using contextual information. In Proceedings of the 2012 ACM Conference on Ubiquitous Computing, Pittsburgh, PA, USA, 5–8 September 2012; pp. 1059–1065.
7. Chen, X.; Wang, Y.; He, J.; Pan, S.; Li, Y.; Zhang, P. CAP: Context-aware App Usage Prediction with Heterogeneous Graph Embedding. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* **2019**, *3*, 1–25. [CrossRef]
8. Tan, C.; Liu, Q.; Chen, E.; Xiong, H. Prediction for mobile application usage patterns. In Proceedings of the Nokia MDC workshop, Newcastle, UK, 18 June 2012.
9. Natarajan, N.; Shin, D.; Dhillon, I.S. Which app will you use next?: Collaborative filtering with interactional context. In Proceedings of the RecSys, Hong Kong, China, 12–16 October 2013; pp. 201–208.
10. Zou, X.; Zhang, W.; Li, S.; Pan, G. Prophet: What app you wish to use next. In Proceedings of the 2013 UbiComp'13 Adjunc, Zurich, Switzerland, 8–12 September 2013; pp. 167–170.
11. Baeza-Yates, R.; Jiang, D.; Silvestri, F.; Harrison, B. Predicting the next app that you are going to use. In Proceedings of the Eighth ACM International Conference on Web Search and Data Mining, Shanghai, China, 2–6 February 2015; ACM: New York, NY, USA, 2015; pp. 285–294.
12. Shin, C.; Hong, J.-H.; Dey, A.K. Understanding and Prediction of Mobile Application Usage for Smart Phones. In Proceedings of the ACM International Conference on Ubiquitous Computing (Ubicomp), Pittsburgh, PA, USA, 5–8 September 2012; pp. 173–182.
13. Parate, A.; Böhmer, M.; Chu, D.; Ganesan, D.; Marlin, B.M. Practical prediction and prefetch for faster access to applications on mobile phones. In Proceedings of the ACM International Joint Conference on Pervasive and Ubiquitous Computing, Zurich, Switzerland, 8–12 September 2013; ACM: New York, NY, USA, 2013; pp. 275–284.
14. Rahnamoun, R.; Rawassizadeh, R.; Maskooki, A. Learning Mobile App Usage Routine through Learning Automata. *arXiv* **2016**, arXiv:1608.03507.
15. Xu, S.; Li, W.; Zhang, X.; Gao, S.; Zhan, Y.; Zhao, Y.; Zhu, W.; Sun, T. Predicting Smartphone App Usage with Recurrent Neural Networks. In *International Conference on Wireless Algorithms, Systems, and Applications*; Springer: Cham, Switzerland, 2018; pp. 532–544.
16. Xu, Y.; Zhu, Y.; Shen, Y.; Yu, J. Leveraging app usage contexts for app recommendation: A neural approach. *World Wide Web* **2019**, *22*, 2721–2745. [CrossRef]
17. Lu EH, C.; Lin, Y.W.; Ciou, J.B. Mining mobile application sequential patterns for usage prediction. In Proceedings of the 2014 IEEE International Conference on Granular Computing (GrC), Noboribetsu, Japan, 22–24 October 2014; IEEE: Piscataway, NJ, USA, 2014; pp. 185–190.
18. Hsu, K.W. Effectively mining time-constrained sequential patterns of smartphone application usage. In Proceedings of the 11th International Conference on Ubiquitous Information Management and Communication, Beppu, Japan, 5–7 January 2017; ACM: New York, NY, USA, 2017; pp. 1–8.
19. Cui, P.; Wang, X.; Pei, J.; Zhu, W. A survey on network embedding. *IEEE Trans. Knowl. Data Eng.* **2018**, *31*, 833–852. [CrossRef]
20. Perozzi, B.; Al-Rfou, R.; Skiena, S. Deepwalk: Online learning of social representations. In Proceedings of the KDD'14, New York, NY, USA, 24–27 August 2014; ACM: New York, NY, USA, 2014; pp. 701–710.
21. Grover, A.; Leskovec, J. node2vec: Scalable feature learning for networks. In Proceedings of the KDD'16, San Francisco, CA, USA, 13–17 August 2016; ACM: New York, NY, USA, 2016; pp. 855–864.
22. Qiu, J.; Dong, Y.; Ma, H.; Li, J.; Wang, K.; Tang, J. Network embedding as matrix factorization: Unifying deepwalk, line, pte, and node2vec. In Proceedings of the WSDM'18, Los Angeles, CA, USA, 5–9 February 2018; ACM: New York, NY, USA, 2016; pp. 459–467.
23. Kipf, T.N.; Welling, M. Semi-Supervised Classification with Graph Convolutional Networks. In Proceedings of the ICLR'17, Toulon, France, 24–26 April 2017.
24. Hamilton, W.; Ying, Z.; Leskovec, J. Inductive representation learning on large graphs. In Proceedings of the NIPS'17, Long Beach, CA, USA, 4–9 December 2017; pp. 1024–1034.
25. Liben-Nowell, D.; Kleinberg, J. The link-prediction problem for social networks. *J. Assoc. Inf. Sci. Technol.* **2007**, *58*, 1019–1031. [CrossRef]
26. Lin, Y.-Y.; Liu, T.-L.; Chen, H.-T. Semantic manifold learning for image retrieval. In Proceedings of the 13th annual ACM international conference on Multimedia, Hilton, Singapore, 6–11 November 2005; ACM: New York, NY, USA, 2005; pp. 249–258.

27. Liao, L.; He, X.; Zhang, H.; Chua, T.S. Attributed social network embedding. *IEEE Trans. Knowl. Data Eng.* **2018**, *30*, 2257–2270. [CrossRef]
28. Yang, J.; Leskovec, J. Overlapping community detection at scale: A nonnegative matrix factorization approach. In Proceedings of the 6th ACM International Conference on Web Search and Data Mining, Rome, Italy, 4–8 February 2013.
29. Zhao, W.X.; Huang, J.; Wen, J.R. Learning Distributed Representations for Recommender Systems with a Network Embedding Approach. In *Asia Information Retrieval Symposium*; Springer: Cham, Switzerland, 2016.
30. Li, Y.; Chen, W.; Yan, H. Learning Graph-based Embedding For Time-Aware Product Recommendation. In Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, Singapore, 6–10 November 2017.
31. Zhang, C.; Yu, L.; Wang, Y.; Shah, C.; Zhang, X. Collaborative User Network Embedding for Social Recommender Systems. In Proceedings of the 2017 SIAM International Conference on Data Mining, Houston, TX, USA, 27–29 April 2017; pp. 381–389.
32. Cen, Y.; Zou, X.; Zhang, J.; Yang, H.; Zhou, J.; Tang, J. Representation Learning for Attributed Multiplex Heterogeneous Network. *arXiv* **2019**, arXiv:1905.01669.
33. Lin, Z.; Feng, M.; Santos, C.N.d.; Yu, M.; Xiang, B.; Zhou, B.; Bengio, Y. A structured self-attentive sentence embedding. In Proceedings of the ICLR'17, Toulon, France, 24–26 April 2017.
34. Shepard, C.; Rahmati, A.; Tossell, C.; Zhong, L.; Kortum, P. LiveLab: Measuring wireless networks and smartphone users in the field. *ACM SIGMETRICS Perform. Eval. Rev.* **2011**, *38*, 15–20. [CrossRef]
35. Liu, X.; Ai, W.; Li, H.; Tang, J.; Huang, G.; Feng, F.; Mei, Q. Deriving User Preferences of Mobile Apps from Their Management Activities. *ACM Trans. Inf. Syst.* **2017**, *35*, 1–32. [CrossRef]
36. Lu, H.C.; Yang, Y.W. Mining mobile application usage pattern for demand prediction by considering spatial and temporal relations. *GeoInformatica* **2018**, *22*, 693–721. [CrossRef]
37. Liu, D.; Xiang, C.; Li, S.; Ren, J.; Liu, R.; Liang, L.; Guan, Y.; Chen, X. HiNextApp: A context-aware and adaptive framework for app prediction in mobile systems. *Sustain. Comput. Inform. Syst.* **2019**, *22*, 219–229. [CrossRef]