future internet

Article

Mobile App Start-Up Prediction Based on Federated Learning and Attributed Heterogeneous Network Embedding

Shaoyong Li [1,2], Liang Lv [3], Xiaoya Li [1] and Zhaoyun Ding [4,*]

1 College of Mathematics and Computer Science, Changsha University, Changsha 410083, China; shaoyongli@ccsu.edu.cn (S.L.); b20190302335@stu.ccsu.edu.cn (X.L.)
2 College of Computer Science, National University of Defense Technology, Changsha 410073, China
3 School of Computer Science and Engineering, Tsinghua University, Beijing 410083, China; lvl16@tsinghua.org.cn
4 College of Systems Engineering, National University of Defense Technology, Changsha 410073, China
* Correspondence: zyding@nudt.edu.cn

Abstract: At present, most mobile App start-up prediction algorithms are only trained and predicted based on single-user data. They cannot integrate the data of all users to mine the correlation between users, and cannot alleviate the cold start problem of new users or newly installed Apps. There are some existing works related to mobile App start-up prediction using multi-user data, which require the integration of multi-party data. In this case, a typical solution is distributed learning of centralized computing. However, this solution can easily lead to the leakage of user privacy data. In this paper, we propose a mobile App start-up prediction method based on federated learning and attributed heterogeneous network embedding, which alleviates the cold start problem of new users or new Apps while guaranteeing users' privacy.

Keywords: mobile app start-up prediction; federated learning; app usage; prediction

check for updates

Citation: Li, S.; Lv, L.; Li, X.; Ding, Z. Mobile App Start-Up Prediction Based on Federated Learning and Attributed Heterogeneous Network Embedding. *Future Internet* **2021**, *13*, 256. https://doi.org/10.3390/fi13100256

Academic Editor: Paolo Bellavista

Received: 3 September 2021
Accepted: 28 September 2021
Published: 7 October 2021

1. Introduction

The popularization of mobile terminals and mobile networks has resulted in changes to our daily life. Various services of mobile terminals are basically provided through Apps, such as news reading services provided by TouTiao; social networking services offered by MicroBlog; travel recommendations, hotel reservations and other services provided by Ctrip; entertainment services provided by PlayerUnknown's Battlegrounds (PUBG); and other game software. With the development of communication network technology and the popularization of mobile devices, the service types of mobile Apps are becoming more diversified, and the number is increasing. The start-up prediction of installed Apps has practical implications, which can help users reduce the time before using Apps. Furthermore, it takes time for mobile Apps to start up and load the latest content, especially for game Apps, the loading time of which can reach 20 s. This can be reduced by preloading the Apps into memory. However, the premise of preloading an App is to accurately predict the App that a user will use, so as to reduce the waste of resources and various adverse consequences caused by preloading an inaccurate mobile App. The purpose of mobile App start-up prediction is to predict the mobile Apps that users are most likely to use in the next period, which can realize the preloading of Apps, save the start-up time of Apps for users, and provide better use experience [1].

The general predictive recommendation algorithm is based on multi-user data, such as the most classic collaborative filtering algorithm; for example, in shopping, the similarity of items purchased between multiple users can be calculated, and then users with the highest similarity can recommend products to each other. However, application startup prediction is highly personalized, so at present, most mobile App start-up prediction algorithms are only trained and predicted based on single-user data. The simplest example is to count the proportion of the time used by the installed software in the software, which is used to predict which application will be the next to operate. The algorithms cannot integrate the data of all users to mine the correlation between users, and cannot alleviate the cold start problem of new users or newly installed Apps. Although there are mobile App start-up prediction algorithms based on multi-user data, the integration of multi-party data is required, which can easily lead to the leakage of user privacy data [2]. The current general solution is the distributed learning of centralized computing, which collects user data, uploads it to the central server, and conducts distributed learning in the central server to obtain the recommended model, and then recommends items that users may be interested in based on the model and the current situation. In the whole process, the user serves only as the producer of the data. After the data is gathered on the central server, the user will lose the ownership of the data and cannot control the storage, access, usage, or flow of the data. Consequently, the risk of information being leaked, misused, or even used for improper profit is increasing. As for big data systems, they have organizational boundaries and lack of matched solutions for supervision and compliance operations. However, in the era of big data where privacy protection has attracted much attention, various countries have begun to attach importance to the legislation in terms of privacy protection. Data producers still lack the control over data, and due to the frequent occurrence of personal information leakage, users have more distrust of data custodians. The Federated Learning [3] solution proposed by Google AI Lab in 2016 can alleviate the problem of data privacy leakage in centralized data recommendation. The basic idea of federated learning is to store the data locally, train the model locally, and then upload the model parameters to the cloud to perform the model aggregation.

To address the above problems, we propose the FL-AHNEAP method based on the federated learning and AHNEAP [4] method. The FL-AHNEAP method can realize the mobile App start-up prediction under the condition of privacy protection, so as to adapt to the mobile App start-up prediction scenario. While integrating multi-user data, the idea of federated learning is applied to alleviate the privacy leakage problem in the distributed learning of centralized computing. At the same time, it can alleviate the cold start problem of users to a certain extent.

The paper comprises the following parts: Section 1. Review of current research results of mobile App start-up prediction and federated learning. Section 2. A multi-user data prediction method, FL-AHNEAP, is proposed based on federated learning and the AHNEAP method. Section 3. The terminal load and communication overhead of the FL-AHNEAP method are analyzed based on the Android terminal. Section 4. We use the LiveLab dataset to conduct related experiments on the proposed method and analyze the influence of various factors on FL-AHNEAP. Section 5. Summary of the main work of this paper, and outlook for further work.

2. Related Work

2.1. Mobile App Start-Up Prediction

Mobile App start-up prediction is a kind of prediction recommendation issue. Its purpose is to predict the mobile Apps that users are most likely to use in the next period, so as to realize the preloading of Apps and save the start-up time of Apps for users. The information that the mobile App start-up prediction method relies on for modeling is mainly related to the usage patterns and the context of the user's App.

App usage patterns refer to the App's usage cycle, usage sequence, etc. The temporal-sequence correlation can be studied from two perspectives: the usage temporal sequence of all installed Apps, and each App's separate usage temporal sequence. The usage temporal sequence of all installed Apps refers to the start-up sequence of Apps in a period of time, whereas each App's separate usage temporal sequence refers to the start-up sequence of the App in that period of time. By analyzing the temporal sequence of each App separately, we can determine the time rule for the usage of the App, and then analyze the temporal sequence of all Apps, and find the periodicity of user behavior [5]. For example, Reference [5] divides the usage temporal sequence of each mobile App according to the length of the time slot, so as to mine the periodicity and recent behavior of users using mobile Apps. Reference [6] uses the variant of App's usage temporal sequence to perform hierarchical clustering on the number of startups of each mobile App in a unit of time to determine a variety of user behaviors. The separate analysis of each App's usage temporal sequence attaches great importance to studying the time rule of each App, but ignores the correlation between Apps. For example, after paying with Alipay, the purchaser usually opens an SMS to check the balance of their bank card. To a certain extent, the startup sequence of different Apps has a sequence association, which may be caused by the user's own habits, or by the automatic jump between Apps. Therefore, some studies predict the next App the user is most likely to launch based on the usage sequence of all installed Apps. Reference [7] exploits the ideas presented in word2vec to model the App's temporal sequence documents and uses a Gaussian-based method to identify the context of each App action to extract session features. According to the principle of Tree Augmented Naive Bayes (TAN) algorithm, the prediction model is established based on the basic features of spatiotemporal context (such as Time, Latitude, Longitude, Charging Cable, etc.); Reference [8] uses a Bayesian network to draw the relationship between the App and the last used App, time, location, and the user profile (such as whether the mobile phone mode is in vibration or mute), and calculates the probability of each App according to the network.

Context-based mobile App start-up prediction can often achieve higher accuracy after fusing multi-party contextual information. Some studies are based on probabilistic graphic models to realize the context-based mobile App start-up prediction, such as the conditional probability model [9], Markov model [10,11], and Bayesian network [7,8,12]. Baeza et al. proposed an effective Bayesian network-based personalized classification method, using the joint features from the basic contextual information and the App session-related contextual information, such as the context change when an App is started, to solve the App prediction problem [7]. Reference [12] utilizes sensor data to extract contextual information, and uses a Bayesian model to predict.

In recent years, researchers have collected local App usage records of mobile terminals and exploited machine learning methods to establish prediction models, such as inferring Apps from temporal profiles [6,13] and mining App usage patterns for prediction [14], etc., which can improve prediction accuracy. XU Shijian et al. defined the mobile App start-up prediction problem as a multi-label classification problem, and proposed a classification model based on LSTM using temporal-sequence correlation and contextual information as prediction features [15]. XU Yanan et al. mined the App usage context patterns based on the contextual information, and proposed a neural network approach to learn both user characteristics and App characteristics, and introduced various sampling methods to address the problem of unbalanced data in the historical use of multiple Apps by multiple users [16]. However, the App-related characteristics that can be collected from smart mobile devices are becoming increasingly diversified, such as the location information, device configuration information, and various sensor data, when the App is in use, and the performance of extracting features from this information by machine learning methods such as LSTM is not sufficient for prediction.

To date, some research has used network representation learning methods to mine the correlation between different types of data of App–time–location. TAN Yaowen et al. constructed a User–App bipartite graph based on the network footprint data. This bipartite graph not only extracts the User–App correlation, but also expresses the similarity between users who used the same application [17]. Thus, an App usage prediction method based on link prediction was proposed. However, the bipartite graph does not use contextual information related to App usage, and research [18] showed that contextual information, such as location and time information or the last used App, is conducive to App prediction. CHEN Xinlei et al. proposed a method called CAP [16], which includes the representational learning module and personalized prediction. Firstly, multi-user data are used to construct a network representing App usage records, including four node types, i.e., App, time, location, App type, and three edge types, i.e., App–location, App–time, App–App. The representational learning module is trained to obtain the node embedding representation, and then the personalized user profile is calculated based on the individual historical data. Personalized prediction is undertaken based on user profiles combined with App node embedding representation. The CAP method uses the relationships between edges to represent App attributes—time, location, and type—but the nodes in the network do not extract the feature attribute of the nodes based on external information, which is not conducive to the calculation of the embedding representation of new nodes and affects the prediction accuracy. CEN et al. proposed an attributed network representation learning method—GATNE [19]—and constructed a heterogeneous network based on historical records and combined external information to construct feature attributes for the nodes of the heterogeneous network. At the same time, they proposed the GATNE-I mode, which can effectively deal with the existence of new nodes; however, this method is not designed for mobile App start-up scenario.

2.2. Federated Learning

Federated learning is also a kind of distributed learning, but differs from traditional distributed learning. The current distributed machine learning is still a kind of data centralized computing, but after the data is concentrated, the central server divides the sample subsets, and each sample subset trains the model and then aggregates the parameters of each model. Therefore, the samples in distributed machine learning are independent and identically distributed, and the sample size is uniformly distributed, so the training process of each subset is relatively similar. However, in federated learning, instead of concentrating data, algorithms are distributed to various devices to perform calculations. Compared with distributed learning, federated learning has more training subsets. In the mobile App start-up prediction scenario, a terminal user is equivalent to a training subset, and the training subsets located in different terminals may not be independent and identically distributed. In recent years, there have been many related studies on joint optimization, mainly for communication overhead and communication security.

The current research reduces the communication overhead from two perspectives: reducing the communication rounds and reducing the dimensionality of communication data. Reference [20] uses the optimization method of the synchronous SGD algorithm [21] to improve the asynchronous stochastic gradient descent algorithm, so as to reduce the communication rounds between the client and the server, perform more calculations on the client, and quicken the model convergence. There are also studies using the same idea to improve the SVRG algorithm and apply it to federated learning [22]. In order to solve the problem of multiple communication rounds in federated learning based on the first-order stochastic gradient descent algorithm, YANG et al. proposed a quasi-Newton method-based vertical federated learning framework for logistic regression under the additively homomorphic encryption scheme [23]. LIU et al. proposed the Federated Stochastic Block Coordinate Descent (FedBCD) to effectively reduce the communication rounds for VFL [24]. The device and the server side need to interact with the complete model, and the large model needs to transmit more data. For the model update data uploaded on the device side, Low Rank, Random Mask, Subsampling, Probabilistic quantization, and other methods are used to reduce the dimensionality or compress the uplink data to reduce communication overhead [25].

According to the idea of federated learning, only the updated data of the model is used to interact between the client and the server, but the updated data still contains the characteristics of the client and may be reproduced. Some studies achieve privacy security in communication from the perspective of data encryption. For example, References [26,27] adopted differential privacy methods to solve this problem. Reference [28] improved secure multi-party protocols for federated learning, to ensure that individual updates can be read only when enough users submit updates. In addition, the design of the framework and the system has been studied to realize the protection of communication data. YANG et al. introduced a comprehensive secure the federated learning framework, covering three types of federated learning, building data networks among organizations based on federated mechanisms, and allowing knowledge to be shared without compromising user privacy [29]. CHENG et al. proposed a novel lossless privacy-preserving tree-boosting system known as SecureBoost in the setting of federated learning. The learning process of the system is executed by multiple parties with partially common user samples but different feature sets. It was theoretically proven that the system is as accurate as other non-privacy protection methods, and can also not disclose any user-related privacy data [30].

In addition to the study on communication overhead and communication security, there are many other efforts on federated learning, such as the application of federated learning, other security issues, and optimal design in federated learning. In terms of the application of federated learning, research has been conducted on the specific architecture of federated learning [31], the application of federated learning to the field of mobile recommendation [32], and the combination of deep learning and edge computing to intelligently utilize the collaboration among devices and edge nodes to exchange the learning parameters [33]. In view of the problems that the client is autonomous in federated learning and can easily deviate from the prescribed process of the model, some studies have proposed the generation of low-dimensional surrogates of model weight vectors, which are then used to detect anomalous clients at the server side [34]. Previous studies did not take the heterogeneity of data and privacy constraints into consideration. Reference [35] proposed a heterogeneous federated learning approach to analyze human behavior and recognize human emotions based on EEG technology, and train machine learning models over heterogeneous data, while preserving the data privacy of each party. Regarding studies on the optimization of the aggregate parameter server, it is believed that the centralized parameter updates are easily affected by server failure; therefore, a blockchained federated learning architecture has been proposed, where mobile devices' local learning model updates are exchanged and verified by leveraging the blockchain [36].

3. FL-AHNEAP for Mobile App Start-Up Prediction under Privacy Protection

3.1. Basic Idea

The AHNEAP [4] method performs mobile App start-up prediction based on a single-user data training model. It is a mobile App startup prediction method based on representation learning on the attributed heterogeneous network. In an attributed heterogeneous network, there are different nodes, and each node has its own attribute information. There are also multiple edge relationships between the nodes. In this article, this means that a network contains three kinds of nodes: time, location, and application. Each node has its own attribute information. Accordingly, there are three types of edge relationships between time and application, location and application, and preorder application. The method includes three steps:

- Data pre-processing: extracting time, location, App information, and their relationships from the user's historical App usage records to generate a heterogeneous network, and assigning attribute information to each node in the network;
- Representation learning on the attributed heterogeneous network: employing the random walk method in the attributed heterogeneous network to generate training sample pairs to train the representation learning model for the attributed heterogeneous network;
- Link prediction model based on the neural network: integrating three pieces of contextual information—time, location, and previous App—to predict the probability of links jointly generated by current time, location, previous App node, and other App nodes. Moreover, the processing of new nodes is included in the design of the AHNEAP method, and the new Apps in the network are represented by the new nodes. Therefore, the AHNEAP method alleviates the cold start problem of new Apps to a certain extent.

In practical situations, users with few or no historical records cannot be predicted according to the AHNEAP method, and the integration of multi-user data can mine the correlation between users and alleviate the cold start problem of new users to a certain extent. At present, the training based on multi-user data generally depends on centralized computing of data, and historical records generated by each user need to be stored in a centralized manner. The AHNEAP method is a context-based prediction method, and the leakage of context information may result in risks to the safety of the user's property or life. Therefore, users' privacy issues need to be considered in the context of improvement based on the AHNEAP method to integrate multi-user data. The idea of federated learning is proposed to alleviate the risk of privacy leakage in data centralized computing.

In the mobile App start-up prediction scenario, the training process of each user is the same. Therefore, we intend to improve the AHNEAP method by integrating the multi-user App usage records, which is the basic idea of horizontal federated learning. Based on the idea of federated learning, the data pre-processing step is performed locally, and in order to ensure the user's personalized prediction, the FL-AHNEAP method only applies federated learning to the representation learning on the attributed heterogeneous network, and the link prediction step based on the neural network is still performed locally. To summarize, FL-AHNEAP contains the following three steps:

- Data pre-processing: In the context of federated learning, model training is performed on the terminal, and obviously data pre-processing is also performed on the terminal;
- Network representation learning under federated learning: Based on the idea of federated learning, the representation learning model for the attributed heterogeneous network is used to integrate multi-user data, and the FederatedAveraging algorithm is used as the optimization algorithm;
- Personalized link prediction: The personalized link prediction model is trained based on the neural network.

The data pre-processing and personalized link prediction are basically the same as the AHNEAP method, and the most important step is to improve the network representation learning. As shown in Figure 1, the terminal obtains the representation learning model for the attributed heterogeneous network from the server, using the local dataset to train the model, and then the terminal uploads the model updates to the server. The server performs model aggregation according to the model updates uploaded by each terminal. After the terminal and the server complete the training of the representation learning model for the attributed heterogeneous network, the terminal obtains the latest shared model from the cloud, calculates the node embedding of the local dataset through the representation learning model for the attributed heterogeneous network as the input, and trains the personalized link prediction model based on the neural network.

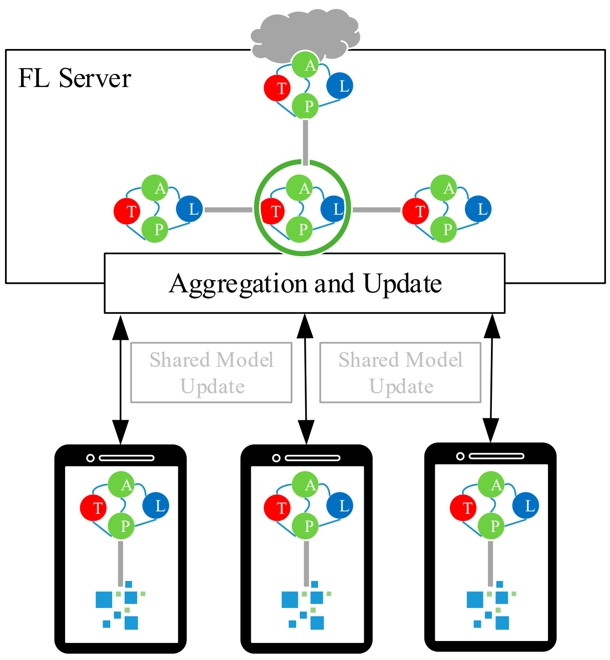

Figure 1. Framework of the FL-AHNEAP method.

3.2. Network Representation Learning under Federated Learning

The representation learning model for the attributed heterogeneous network in AHNEAP uses the following formula to calculate the node's embedding:

$$v_{i,r} = h_z(x_i) + \alpha_r M_r^T U_i a_{i,r} + \beta_r D_z^T x_i \quad (1)$$

where r represents the edge type in the attribute heterogeneous network built in AHNEAP, and there are three types of node: time, location, and application, and three types of edge: time application, location application, and preamble application. Therefore, r is a positive integer and satisfies $1 \leq r \leq 3$. x_i denotes a feature of the node i. $h_z(x_i)$ is a transformation function, whose role is to calculate the influence of x_i in the embedding representation of the node, and is the base embedding for node i. According to the self-attention mechanism [37], $a_{i,r}$ is the weight of the edge embedding vector of node i in the sub-network of edge type r, and is computed according to $a_{i,r} = softmax\left(w_r^T tanh(W_r U_i)\right)$, and is actually a simple feedforward neural network. w_r^T, W_r, represent the transformation probability matrix, which requires the training of the model, and optimization U_i represents the hidden layer state in the self-attention mechanism. $tanh()$ is the activation function of $W_r U_i$, and $softmax()$ is a normalized function, which converts the result from negative infinity to positive infinity to the probability 0–1. U_i is a vector composed of the edge embedding vectors of node i in three sub-networks, which represents the relationship between node i and its neighbor nodes. Each node aggregates the mean value of the features of a certain number of neighbor nodes in each sub-network, as the edge embedding vector of the node in the sub-network U_i is formed by concatenating the edge embedding vectors of node i in three sub-networks. M_r is the transformation matrix of the edge embedding vector; M_r^T represents the transpose of M_r, and the ith one-dimensional vector in M_r represents the influence of the edge embedding vector of the node i in the sub-network of the edge type r on the embedding representation of node i. α_r represents the influence coefficient of the edge embedding vector of each node on the embedding representation of the node in the sub-network with edge type r. D_z is a feature transformation matrix on node i's corresponding node type z, designed for the purpose of calculating the embedding representation vector of the new node, and the new nodes are isolated from all nodes in the network. D_z represents the similarity relation between node i and all nodes in the network from the perspective of features similarity; D_z^T is the transpose of D_z. β_r represents the influence coefficient of the similarity relation between features in the sub-network with edge type r on the node embedding.

In this section, we use the FederatedAveraging algorithm to realize the parameter aggregation of the network representation learning model. In Equation (1), $h_z(x_i)$ is actually obtained by multiplying a feature matrix and a transformation matrix. There is a transformation matrix H, and there are transformation matrices M and D, and two transformation matrices w and W in $a_{i,r}$, and finally there is a transformation matrix T for node edge embedding. The updates of model parameters are actually the updates of the above six transformation matrices. Algorithm 1 is pseudo code of network representation learning based on federated learning.

Algorithm 1. Network Representation Learning under Federated Learning.

```
Inputs: Nodes, Features, Neighbors.
Model parameters: transformation matrix in network representation learning model $H_0$, $M_0$, $D_0$, $w_0$, $W_0$, $T_0$.
Server:
    Choose b samples from the training set M;
    Initialize the model parameters $\boldsymbol{H_0}$, $\boldsymbol{M_0}$, $\boldsymbol{D_0}$, $\boldsymbol{w_0}$, $\boldsymbol{W_0}$, $\boldsymbol{T_0}$;
for each round $\boldsymbol{t} = \mathbf{1}, \mathbf{2}, \mathbf{3}, \ldots$:
        Randomly select $\boldsymbol{C}$ clients $\boldsymbol{S_t}$;
        Parallel execution on each worker node $\boldsymbol{k} \in \boldsymbol{S_t}$:
        $\boldsymbol{H^k_{t+1}}, \boldsymbol{M^k_{t+1}}, \boldsymbol{D^k_{t+1}}, \boldsymbol{w^k_{t+1}}, \boldsymbol{W^k_{t+1}}, \boldsymbol{T^k_{t+1}} = \textbf{ClientUpdate}\left(\mathbf{k}, \boldsymbol{H^k_t}, \boldsymbol{M^k_t}, \boldsymbol{D^k_t}, \boldsymbol{w^k_t}, \boldsymbol{W^k_t}, \boldsymbol{T^k_t}\right)$;
        Calculate the parameter updates: $\mathrm{H_{t+1}} = \sum_{\mathrm{k=1}}^{\mathrm{C}} \frac{\mathrm{n_k}}{\mathrm{n}} \mathrm{H^k_t}$, $\mathrm{M_{t+1}}$, $\mathrm{D_{t+1}}$, $\mathrm{w_{t+1}}$, $\mathrm{W_{t+1}}$, $\mathrm{T_{t+1}}$
in a similar way;
ClientUpdate(k, $\boldsymbol{H^k_t}$, $\boldsymbol{M^k_t}$, $\boldsymbol{D^k_t}$, $\boldsymbol{w^k_t}$, $\boldsymbol{W^k_t}$, $\boldsymbol{T^k_t}$):
      Divide the training set into B parts according to the batch size $\boldsymbol{B}$;
for each epoch $\boldsymbol{e}$ from 1 to $\boldsymbol{E}$:
        for each batch $\boldsymbol{b} \in$ B:
        Train the representation learning model for attributed heterogeneous network, and
update $\boldsymbol{H^k_{t+1}}, \boldsymbol{M^k_{t+1}}, \boldsymbol{D^k_{t+1}}, \boldsymbol{w^k_{t+1}}, \boldsymbol{W^k_{t+1}}, \boldsymbol{T^k_{t+1}}$;
                    return $\boldsymbol{H^k_{t+1}}, \boldsymbol{M^k_{t+1}}, \boldsymbol{D^k_{t+1}}, \boldsymbol{w^k_{t+1}}, \boldsymbol{W^k_{t+1}}, \boldsymbol{T^k_{t+1}}$;
```

Assuming that the completion of the shared model requires multiple rounds of communication between the terminal and the server, there is a fixed set of clients K, and E rounds of local training, and each round is divided into small batches using all the local datasets. At the beginning of each round of communication t, C clients $\boldsymbol{S_t}$ are randomly selected, and the server sends the information about the current shared model to the selected clients. Then, each selected client performs local computation based on the shared model and its local datasets, trains the representation learning model for the attributed heterogeneous network, updates the model parameters, and sends the updated model parameters to the server. The server aggregates these updated model parameters and carries out the weighted average of the model parameters according to the proportion of the number of training samples of each user ($\boldsymbol{n_k}$) to the number of training samples of all users participating in this round of training (n). The specific process of each round of communication basically includes the following three steps:

- Select a certain proportion of users from all client users to participate in this round of training;
- Each selected client trains the shared model obtained from the cloud using local data;
- The server waits for and obtains the updated model parameters of all selected clients, and aggregates the model parameters according to the proportion of client training samples to all training samples.

3.3. Analysis of Cold Start Prediction

The AHNEAP method includes the processing of new nodes. In the mobile App start-up prediction scenario, a new App appears as a new node in the network, and the

FL-AHNEAP method is designed based on the AHNEAP method, so the FL-AHNEAP method copes with the cold start problems of new Apps to a certain extent. Therefore, we focus on analyzing the cold start problem of new users in this section.

The cold start problem is a classic problem in prediction and recommendation, and is an extreme problem of data sparseness. It has been frequently studied. However, for most recommended algorithms, effective solutions have not been obtained. Generally, various auxiliary data are used to achieve cold start prediction of new things or new users. In terms of the sources of auxiliary information, there are three kinds of cold start prediction methods at present [38]. The first does not consider the prediction of auxiliary content at all, and does not avail of any auxiliary information, such as random recommendation, mean method, and mode method. In general, we do not take this method into account. The second kind of auxiliary information is obtained from the source network or the destination network, and is solved by using user and item score records and auxiliary information. The third method is interview-based cold start prediction, using questionnaire surveys and other means to directly ask about new users' preferences. However, in addition to the above three kinds of cold start recommendation methods, group recommendation can also alleviate the cold start problem to a certain extent. As the name suggests, group recommendation is to recommend things to a group [39]. Firstly, the prediction scores of all users in the group are aggregated to obtain the prediction score of the group, and then the degree of deviation between each user in the group and the prediction score of the group is calculated, and finally the group consensus score is calculated according to the weighted group prediction score and the deviation degree of users. The higher the group prediction score, the smaller the user deviation, and the greater the value of the consensus score. The goal of group recommendation is to solve the consensus score, and recommend the item with the highest score or the top k item groups with the highest score to the group.

Reference [6] describes two user cold start strategies for the mobile App startup prediction problem. The first strategy is to find similar users with the same App, and the second is to select the historical records of some users to synthesize the historical records of new users. Both strategies are based on the premise that other user data is available. However, in the idea of federated learning, the App information and historical records of users are isolated from each other, and the only thing that can be obtained from the user terminal is the model parameters. Therefore, these two strategies are not feasible. In the FL-AHNEAP method, the only additional information required for new users to access is the representation learning model for the attributed heterogeneous network placed in the cloud, so the first three cold start prediction methods mentioned above are not feasible. The basic idea of the FL-AHNEAP method is to select a certain number of users, generate models on the user terminals, and then aggregate these model parameters in the cloud to generate a shared model. This is similar to the group recommendation concept. Reference [39] introduces an aggregation strategy, which aggregates and calculates the score data of each member based on the characteristics, influence, and other information of group members, and the simplest one is the mean strategy. The FL-AHNEAP method uses the FederatedAveraging algorithm to implement cloud model aggregation, and calculates the mean value of a group of users' model parameters as the parameters of the final shared model, which is similar to the mean aggregation strategy in group recommendation. Therefore, new users can obtain the shared model from the cloud for direct prediction.

When a new user has no historical record at all, the prediction can only be made based on time or location information according to the shared model. We can learn from the analysis of experimental results in Reference [4] that the accuracy based on the time information is higher than that based on the location information. Thus, we calculate the current time and the node embeddings of all Apps based on the model, and then calculate the cosine similarity between the time node embedding and the node embeddings of all Apps. Then we select the top K Apps with the highest similarity as the candidate set. However, in actual situations, it is impossible for a new user to have no historical record at all. Even if there is only one record, the prediction can be made based on the previous App.

The cosine similarity between the embedding representation of the context node (v_c) and the embedding representation of the App node (v_a) is:

$$\cos(c,a) = \frac{\sum_{i=1}^{n} v_c^i v_a^i}{\sqrt{\sum_{i=1}^{n} {v_c^i}^2}\sqrt{\sum_{i=1}^{n} {v_a^i}^2}} \tag{2}$$

where n represents the dimension of node embedding; v_c^i denotes the ith element of the ith embedding representation of the time or the previous App node. Then the user's cold start prediction problem is actually to find an application a_i^u, so that the cosine similarity is highest in a certain context c:

$$\max(\cos(c, a_i^u)),\ a_i^u \in apps_u \tag{3}$$

where $apps_u$ represents the App set of user u; a_i^u denotes the ith application of user u.

4. Terminal Load and Communication

The FL-AHNEAP method is improved based on federated learning, so the process of model training is performed on the terminal. However, the computing power of the terminal is limited. Thus, we need to consider whether the terminal is capable of undertaking the model training of the FL-AHNEAP method and whether the communication overhead between the terminal and the cloud will impose a burden on users. Next, we analyze the terminal load from the spatiotemporal complexity of the FL-AHNEAP method, and analyze the communication overhead of the terminal.

4.1. Analysis of Terminal Load

The training process of the FL-AHNEAP method is divided into two parts: the representation learning on the attributed heterogeneous network and the link prediction model based on the neural network.

The representation learning on the attributed heterogeneous network uses the noise contrastive estimation (NCE) loss function to optimize the model parameters. According to the idea of NCE, for each sample, N other labels are first sampled to generate negative samples. At first, we calculate the input node embedding vector $v_{i,r}$ according to Formula (4), and then construct the optimization model parameters of log-likelihood function of the binary logistic regression:

$$\begin{aligned} l_i &= \log\left(sigmoid(W_p^T v_{i,r})\right) + \textstyle\sum_{k=1}^{K} \log(1 - sigmoid(W_n^T v_{i,r})) \\ &= \log\left(sigmoid\left(W_p^T v_{i,r}\right)\right) + \textstyle\sum_{k=1}^{K} \log(sigmoid(-W_n^T v_{i,r})) \end{aligned} \tag{4}$$

When computing the node embedding v_i, in the mobile App startup prediction scenario stated in this paper, the embedding of the same node in the three sub-networks needs to be obtained respectively. The main computing overhead comes from the process of generating node embedding representation from node features and node neighbor information. The process is mainly matrix multiplication, and the time complexity is $O(NRDL)$, where N represents the number of nodes, R denotes the edge type, D is the dimension of the node embedding, and L represents the number of neighbor nodes selected for the generation of node edge embedding. The main computing overhead of using the NCE loss function to optimize the model is the matrix multiplication in logistic regression. The time complexity is $O(NRD(K+1))$, and K represents the number of negative samples for each sample. The experiment of Reference [2] shows that the terminal training round has little effect on the FL-AHNEAP method, assuming five rounds of terminal training. The edge type $R = 3$, the node embedding dimension $D = 20$, the number of neighbors $L = 10$, and K is generally set to 5, and the time complexity depends on the number of nodes N. When there are an increasing number of historical records, the number of nodes in the network generated based on the records will also increase, and the computing load on the terminal will become greater.

In light of Reference [2], only a simple neural network with a single hidden layer is constructed in the link prediction based on neural network, so the main computing overhead is still matrix multiplication. The time complexity is $O(N_r N_a D + N_r N_a N_a) = O(N_r N_a(D + N_a))$, where N_r represents the number of historical records, and N_a represents the number of applications. There are not many applications for a user, so $(D + N_a) \ll N_r$. The time complexity depends on the number of historical records of users participating in the training. In the same manner, when there are an increasing number of historical records, the computing load on the terminal will increase.

In this paper, we adopt the App: AID Learning to run the terminal part of the FL-AHNEAP method on the Android system to calculate the model training time. AID Learning is a Linux virtual machine running on an Android terminal that supports a graphical interface. It is a framework and platform that supports the development of deep neural networks, with the most popular deep learning framework built in. We use the following two mobile devices with different configurations for verification:

- Huawei P20: equipped with OS Android 10, HiSilicon Kirin 970 processor, CPU frequency 2.36GHz, 6GB RAM, 128GB ROM;
- Huawei Nova2S: equipped with OS Android 9, HiSilicon Kirin 960 processor, CPU frequency 1.8GHz, 4GB RAM, 64GB ROM.

We use following three kinds of training data for testing

- One month's data of user A11: a total of 745 records; 170 nodes and 2550 training sample pairs were generated;
- One month's data of user D03: a total of 2718 records; 570 nodes and 6694 training sample pairs were generated;
- Data of user B02 in the past year: 14,565 records in total; 3427 nodes and 37,602 training sample pairs were generated.

The test results are shown in Table 1. The values in the table are the mean values after multiple tests. The total running time includes the whole step of data preprocessing, training of the representation learning model for the attributed heterogeneous network, training of the link prediction model based on the neural network, and model testing, whereas the data processing time only includes the time consumed by the data pre-processing step. Obviously, the mobile phone configuration will affect the time consumption. The configuration of P20 is better than that of Nova2S. Therefore, P20 generally consumes less time than Nova2S in both data processing time and total running time. The total running time of P20 is about 56∼68% of that of Nova2S, whereas the data processing time is about 60∼70% . The total running time minus the data processing time yields the time for model training and model testing; this time for P20 is about 51∼65% of that of Nova2S.

Table 1. Comparison of terminal training duration.

	User	Number of Records	Number of Nodes	Training Sample Pairs	Data Processing Time	Total Running Time
P20	A11	745	170	2550	29″	1′20″
	D03	2718	570	6694	48″	3′7″
	B02	14,565	3427	37,602	10′32″	37′09″
Nova2S	A11	745	170	2550	48″	2′18″
	D03	2718	570	6694	1′12″	4′35″
	B02	14,565	3427	37,602	14′4″	56′47″

Through the analysis of the test results, we can learn that the more training data, the longer the consumption time; the less training data, the shorter the consumption time; and the total running time increases exponentially similar to the number of training sample pairs. As the user with the most historical records in a month, user D03 spends 3′7″ on P20, and 4′35″ on Nova2S. As one of the users with the most historical records in a year, user B02 spends 32′43″ on P20 and 56′37″ on Nova2S. It can be seen that when the amount of training data is within a certain range, the consumption of the terminal load in the FL-AHNEAP method is still within an acceptable range. However, as the training data grows, the running time also increases, and it takes nearly half an hour or even an hour of training, which is unfriendly for users. Therefore, we still need to consider a variant of the FL-AHNEAP method in the future to make it an incremental training method.

At present, the RAM of most users' smart phones is still 4 GB, but the RAM of newly launched mobile phones is generally 6 GB, and mobile phones with 8 GB RAM are continuously being introduced. Apparently, from the perspective of space complexity, the space consumption of RAM in the terminal is sufficient for FL-AHNEAP. In addition, the storage capacity of smart phones is constantly expanding, from 64 G to 128 G, and gradually to 256 G; this capacity is enough to store the historical records continuously generated by users. Similarly, however, as a result of the growth of time, the storage capacity occupied by historical records is also increasing. Although the capacity is sufficient, occupying too much capacity of the mobile phone is not friendly to the user experience. Therefore, it is necessary to consider a variant of the FL-AHNEAP method in the future to make it an incremental training method.

4.2. Analysis of Terminal Communication Overhead

In the FL-AHNEAP method, end users need to interact with the cloud. According to Algorithm 1, the communication items are mainly six transformation matrices, the dimensions of which are mainly related to the characteristic dimension of the node features, the node embedding dimension, and the edge in the network. We selected the matrix with the largest dimension for analysis. It can be drawn from the Reference [2] that there are three edge types in the network. The attribute feature dimension is set to 50, and the node embedding can be set to 20 according to experiments; then, there are up to floating-point type elements in a matrix. Five matrix parameters need to be passed, so a total of floating-point type elements need to be passed. A floating-point type occupies 8 bytes, so it occupies a total of about 14 KB. We exploit the array form in Python to store six matrices, and then store the six matrices in the same dictionary. We store the structure and data of the dictionary in a file, and the final file size is about 33 KB; that is, the interaction required for each communication is a file of about 33 KB. For users, the communication overhead of each round is almost negligible.

5. Experimental Results and Analysis

5.1. Experiment Settings

Dataset: Figure 2 shows the statistics of 34 users' historical records in a month. Among these, user D05 has the least with only 45 records, whereas user D03 has the most with a total of 2718 records. It can be seen that the number of users' historical records within a month can be divided into five stages from 0 to 2500. In order to make the experimental results more credible and make the user data more random, users in each stage are selected separately, and finally the data of ten users, i.e., A01, B07, A11, D00, A03, A12, B06, B08, A04, and A07, are selected for the experiment.

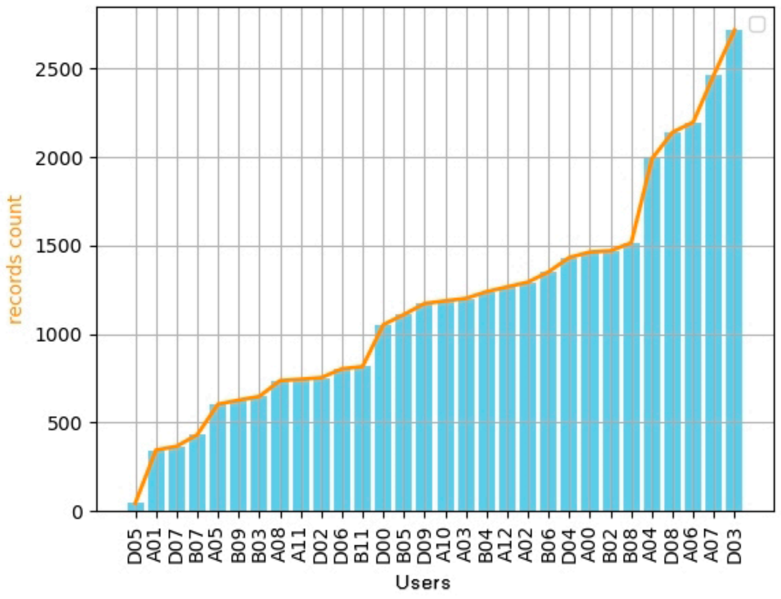

Figure 2. LiveLab dataset: statistics of 34 users' historical records in a month.

Data pre-processing: Based on the idea of horizontal federated learning, the model used by each terminal is the same, and the model parameter settings are also the same. In the representation learning model for the attributed heterogeneous network, the parameter dimension setting is related to the feature vector, and the feature information about the location node is related to the number of connected base stations in the historical records. However, in horizontal federated learning, each terminal needs to get the same feature, so the dimension of the feature vector needs to be fixed. The App node features are obtained according to the App node types. The frequency of each type of application is counted, a fixed number of high-frequency types are selected, and the remaining types are classified into one category. In the same manner, the location node features are obtained according to the coverage area of the base station, the usage frequency of each coverage area is counted, a fixed number of high-frequency base stations LAC are selected, and the remaining types are classified into one category. When a user's App types or the number of coverage areas is less than a fixed number, 0 is used to fill the one-hot encoding value to obtain the feature vector of the same dimension.

Basic settings: In this section, we conduct an experiment and evaluation on the proposed FL-AHNEAP method still based on the LiveLab App usage dataset, using the prediction accuracy as the evaluation criterion. During terminal training, we use 80% of the selected training dataset as the training set and 20% as the test set.

In this experiment, ten users—A01, B07, A11, D00, A03, A12, B06, B08, A04, and A07—were selected as the representative users to realize the mobile App start-up prediction method based on federated learning. Because it is a simulation experiment, and there are restrictions on the number of users in the dataset, we did not select many users to participate in the training, and did not randomly selected users for the training. In each round of communication, the clients corresponding to these 10 users were directly selected to perform the aggregation of the shared model parameters, which is equivalent to the parallelization of synchronous data in distributed learning, and all user data were selected to train the shared model. Then each terminal trained a personalized link prediction model based on the neural network according to the shared model. During prediction, each terminal first generates the node embeddings of the current time, location, and previous App according to the shared model, and then aggregates the node embeddings as the input, obtains the prediction probability based on the neural network model, and generates the candidate App set.

5.2. Experimental Results

AHNEAP vs. FL-AHNEAP: Figure 3 compares the accuracy between the AHNEAP method and the FL-AHNEAP method. The FL-AHNEAP method uses 10 users for 20 rounds of communication, and each user is trained for five rounds at the terminal. As for the AHNEAP method, each user is trained for 100 rounds independently. FL-Strict strictly implements the FL-AHNEAP method, selecting 15 users for training and conducting 20 rounds of communication. The users are trained at the terminal for five rounds, and five users are randomly selected to participate in one round of communication according to a one-third ratio. It can be seen from Figure 3 that the experimental results of FL-AHNEAP and FL-Strict are not much different, but the accuracy of the two is generally lower than that of the AHNEAP method of single-user data. Among these, the accuracy of user D00 is the highest, reaching 72%, whereas user B08 has the lowest accuracy, of only 40%. For user A12, the accuracy of the FL-AHNEAP method is 17.5% lower than that of the AHNEAP method, which is the largest difference among all users, whereas the difference of user A03 is the smallest, at only 2.5% lower. This is because the FL-AHNEAP method destroys the structural property of the network to a certain extent; from the perspective of federated learning, it is equivalent to sacrificing part of the accuracy to obtain privacy protection.

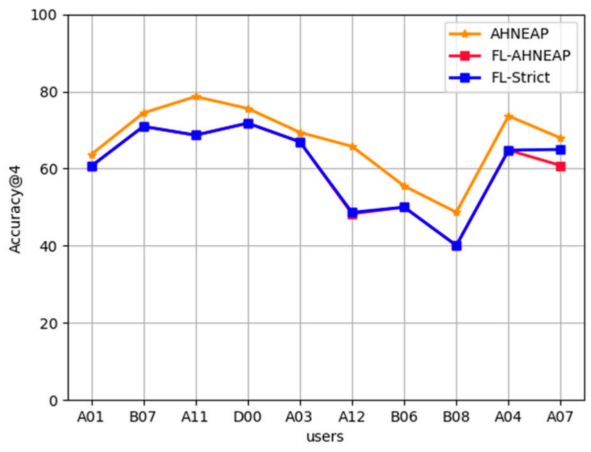

Figure 3. AHNEAP vs. FL-AHNEAP.

In the figure, the abscissa consists of 10 users arranged from left to right based on the number of historical records. From the results of AHNEAP, it is obvious that each user has different App usage patterns, the accuracy of App prediction based on AHNEAP method is different, and there is no direct relationship between the accuracy of prediction and the number of historical records. User A11 does not have the most historical records, but has the highest accuracy rate of 79%, whereas user B08 has significantly more historical records, but has the lowest accuracy rate of only 48%.

Effect of communication rounds on FL-AHNEAP: Figure 4 compares the effect of communication rounds on FL-AHNEAP. Five users are selected to train the shared model. Each time, the communication terminal only performs one round of training for the representation learning on the attributed heterogeneous network, and a total of 20 communication rounds are performed. It can be seen from the figure that the communication rounds have little effect on the prediction accuracy. For users D00, B07, and B08, there is almost no effect, whereas user A04 has a slight fluctuation, and A07 is relatively unstable. Generally speaking, however, the fluctuation range of prediction accuracy is not large; thus, in this case, it may only need one round of communication to obtain better results.

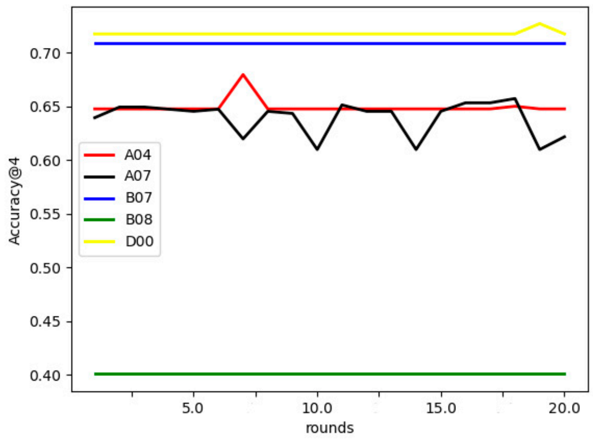

Figure 4. Effect of communication rounds on FL-AHNEAP.

Effect of terminal communication rounds on FL-AHNEAP: Figure 5 compares the effect of terminal communication rounds on FL-AHNEAP. Ten users are selected to participate in training and 20 communication rounds are performed. Each time, the communication terminal performs one round, five rounds, and 10 rounds of training, respectively, for the representation learning on the attributed heterogeneous network. It can be seen from the figure that the prediction accuracy rates of 10 users in the three cases almost overlap. For users A01 and A07, the prediction accuracy rate of one round of training is slightly higher. In can be drawn from the experiment stated in Reference [2] that the AHNEAP method can converge quickly, so it can be inferred that the terminal training round has little effect on FL-AHNEAP. To reduce the consumption of terminal resources, there is no need to perform multiple rounds of training on the terminal. According to the FL-AHNEAP method, the terminal will train a personalized neural network, so the overlap of the prediction accuracy in the three cases can be explained.

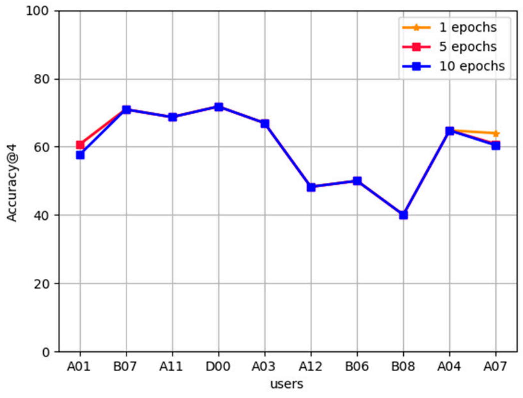

Figure 5. Effect of terminal communication rounds on FL-AHNEAP.

Effect of the number of users on FL-AHNEAP: Figure 6 compares the effect of the number of users on FL-AHNEAP. Each time, the communication terminal only performs one round of training for the representation learning on the attributed heterogeneous network, and a total of 20 communication rounds are performed. We select five users, 10 users, and 15 users, respectively, for the training, and conduct the prediction on a common group of five users. It can be seen from the figure that there is basically no difference in the accuracy of training with different numbers of users. This is because, in the FL-AHNEAP method, only the shared model is used to train the representation learning on the attributed heterogeneous network, whereas the link prediction model based on the neural network is used to realize the personalized prediction of users.

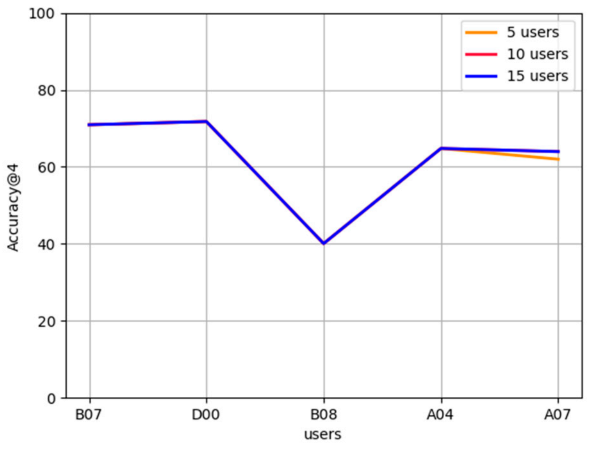

Figure 6. Effect of the number of users on FL-AHNEAP.

Effect of the number of applications used by users on the accuracy of prediction: Figure 7 describes the effect of the number of applications used in each user's training set on the accuracy of prediction. The FL-AHNEAP method uses 10 users for 20 rounds of communication, and each user is trained for five rounds at the terminal. It shows that for the three users B07, A11, and D00 with high prediction accuracy, the number of applications used in the training set is relatively small, whereas for B08 with the lowest prediction accuracy, the number of applications used in the training set is the largest. It can be basically inferred that the number of applications used by users has a certain impact on the prediction accuracy, and further research can be undertaken in the future.

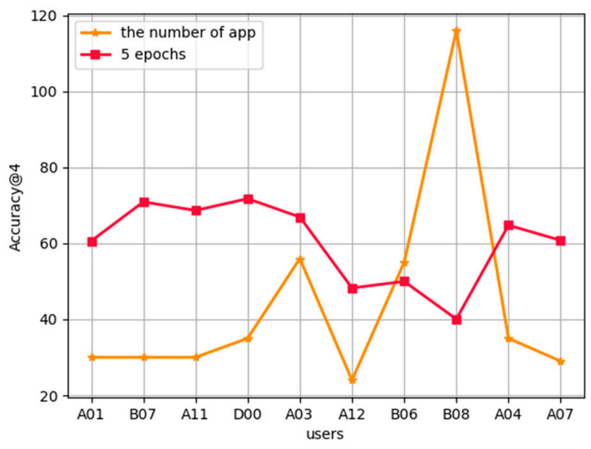

Figure 7. Effected of the number of used applications on FL-AHNEAP.

Analysis of cold start of new users: In the FL-AHNEAP method, each user can obtain a shared model, that is, a representation learning model for the attributed heterogeneous network. Therefore, for the cold start problem of new users, we can calculate the embedding of each new code based on the shared model, then calculate the cosine similarity of the node embedding, and select the node with higher similarity as the prediction result. Figure 8 shows the prediction accuracy results of the shared model (obtained after training based on the data of ten users: A01, B07, A11, D00, A03, A12, B06, B08, A04, and A07) to five users: D03, A06, A02, B03, and D04, who have not participated in the training. If a new user does not have a historical record at all, we can only make predictions based on the current time and location information. It can be seen that, except for D03, the prediction accuracy obtained by time and location information is almost the same, and all accuracies are higher than 20%. If a new user has some historical records, the prediction accuracy of time, location, and previous App can be considered at the same time. It can be seen that the prediction accuracies based on the previous App are much higher than that of time and location information, and all are higher than 50%. User D03 achieves the highest result, reaching 71%, whereas user D04 is the lowest, but also has an accuracy of 51%. In actual situations, it is impossible for a new user to have no App usage record at all, so the candidate App set can be obtained by selecting the previous App information.

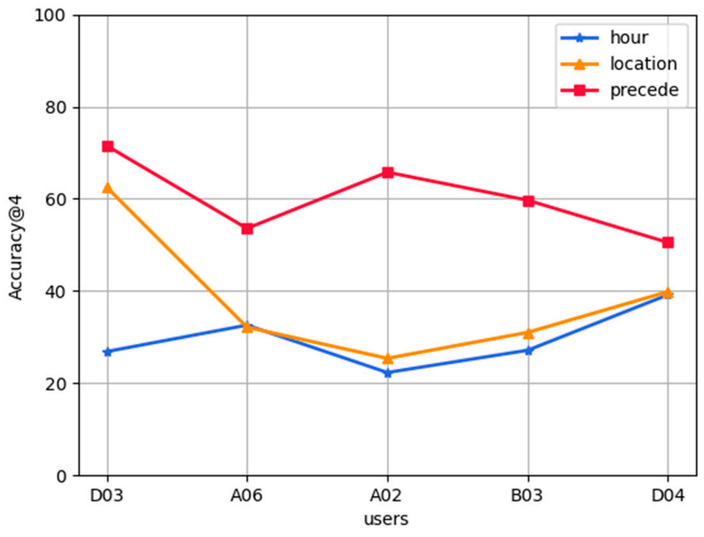

Figure 8. Cold start prediction of new users.

Effect of the number of user historical records on FL-AHNEAP: Figure 9 shows the effect of the number of user historical records on the FL-AHNEAP method. One month, three month, and six month data of five users are selected for training, including 80% for the training set and 20% for the test set. The figure shows that the number of user historical records has little effect on the FL-AHNEAP method, and the fluctuation of the prediction accuracy of each user is small. When only one month's data of each user is selected for the training, the accuracy tends to be slightly higher.

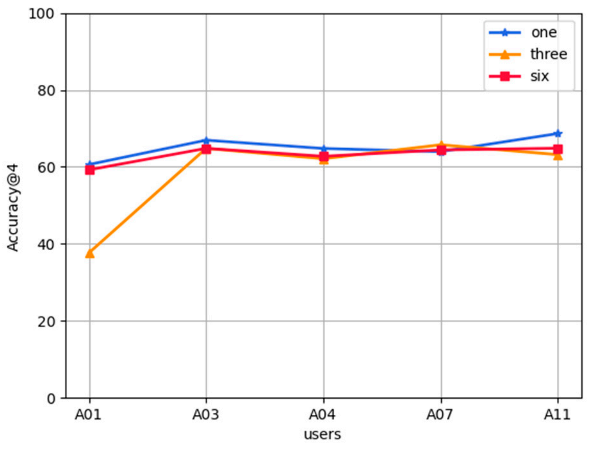

Figure 9. Effect of user data size on FL-AHNEAP.

Effect of historical data on FL-AHNEAP: We can learn from the terminal load analysis that the user data size will directly affect the time consumed by the FL-AHNEAP method in terminal training. Therefore, it is necessary to analyze the effect of historical data on FL-AHNEAP, e.g., the effect of three month data and one month data on FL-AHNEAP. We select the historical data of one month, three months, and six months to predict the same data of the next month, as shown in Figure 10. It can be seen that the accuracy of using three month and six month data to predict the next month's data is slightly higher than the accuracy of using one month data, whereas the accuracy of using three month data and six month data is basically the same. It can be inferred that historical data has a certain impact on accuracy, but when the time span of historical data is long, the gap gradually decreases. More specific experiments can be performed to obtain the critical point, by measuring the historical data of the appropriate time period for the prediction accuracy and terminal load.

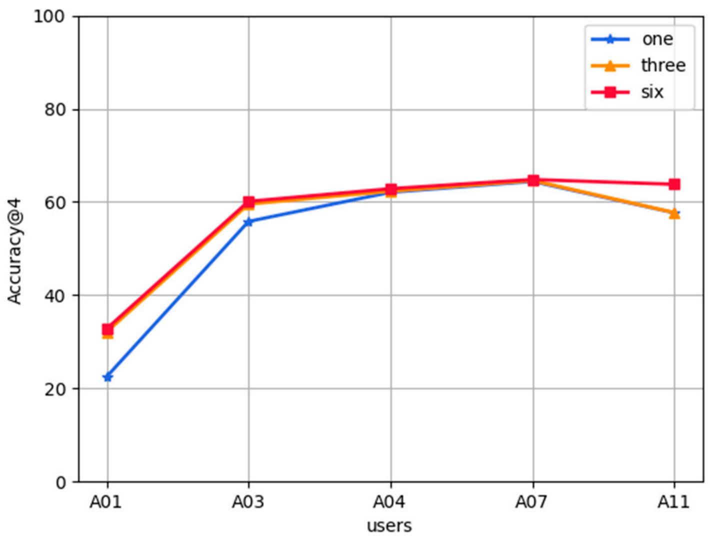

Figure 10. Effect of historical data on FL-AHNEAP.

6. Conclusions

The training of single-user data has strong personalized features, but when a user has little or no historical record, the user will not be able to make a good prediction. In general, the integration of multi-user data can often achieve higher accuracy. However, the integration of multi-user data generally requires centralized management, which is not conducive to the privacy protection of users. Combining federated learning and AHNEAP methods, in this paper we propose FL-AHNEAP, a mobile App start-up prediction method under privacy protection. It integrates multi-user data to train a representation learning model for the attributed heterogeneous network, and the terminal separately trains the link prediction model based on the neural network to achieve personalized predictions for users. Experimental analysis shows that, in the LiveLab dataset, the accuracy of the FL-AHNEAP method is slightly lower than that of the AHNEAP method based on single-user data because the model aggregation destroys the structure of the user's heterogeneous network. However, through the analysis of cold start experiments for new users, it is shown that the FL-AHNEAP method can alleviate the cold start problem of new users to a certain extent. Under the assumption that users have previous Apps, the prediction accuracy can reach 71%.

However, the FL-AHNEAP method under privacy protection only uses FederatedAveraging, a simple federated learning optimization algorithm, and even if the idea of federated learning is used, the user's privacy is not absolutely secure. There are many aspects worthy of discussion in the context of mobile App start-up prediction based on federated learning, such as data security in the communication process and other algorithms for model aggregation. The FL-AHNEAP method destroys the network structure when performing model aggregation. In the future, it can be considered how to realize not only the aggregation of parameters, but also the merger of the network under the federated learning mechanism. Although the idea of federated learning realizes the privacy protection of user data to a certain extent, it cannot guarantee absolute data security. Studies have shown that user-related information can be extracted from model parameters, and FL-AHNEAP can be optimized from the perspective of federated optimization, such as model aggregation algorithm, homomorphic encryption, and differential privacy, to further ensure privacy and security. Moreover, model training is resource intensive, and for the terminal, model training consumes a large proportion of resources. Due to the importance of attracting terminal holders to participate in federated learning training, the study of user incentive mechanisms is a possible future research direction.

Author Contributions: Conceptualization, Z.D.; Project administration, S.L.; Resources, X.L.; Supervision, L.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Not applicable, the study does not report any data.

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

References

1. Ca, O.H.; Lin, M. Mining smartphone data for app usage prediction and recommendations: A survey. *Pervasive Mob. Comput.* **2017**, *37*, 1–22. [CrossRef]
2. Han, D.; Li, J.; Lei, Y. A recommender system to address the Cold Start problem for App usage prediction. *Int. J. Mach. Learn. Cybern.* **2018**, *10*, 2257–2268. [CrossRef]
3. Brendan Mcmahan, D.R. Federated Learning: Collaborative Machine Learning without Centralized Training Data [EB/OL]. 2017. Available online: https://ai.googleblog.com/2017/04/federated-learning-collaborative.html (accessed on 27 September 2021).
4. Zhou, Y.; Li, S.; Liu, Y. Graph-based Method for App Usage Prediction with Attributed Heterogeneous Network Embedding. *Future Internet* **2020**, *12*, 58. [CrossRef]
5. Lu, E.H.; Lin, Y.; Ciou, J. Mining mobile application sequential patterns for usage prediction. In Proceedings of the IEEE International Conference on Granular Computing, Noboribetsu, Japan, 22–24 October 2014; pp. 185–190.
6. Liao, Z.; Lei, P.; Shen, T. Mining temporal profiles of mobile applications for usage prediction. In Proceedings of the 12th IEEE International Conference on Data Mining Workshops, Washington, DC, USA, 10 December 2012; pp. 890–893. [CrossRef]
7. Baeza-Yates, R.; Jiang, D.; Silvestri, F. 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, 31 January–6 February 2015; pp. 285–294.
8. Huang, K.; Zhang, C.; Ma, X. Predicting mobile application usage using contextual information. In Proceedings of the 2012 ACM Conference on Ubiquitous Computing, New York, NY, USA, 5 September 2012; pp. 1059–1065. [CrossRef]
9. Chang, T.E.C.; Qi, L. Prediction for mobile application usage patterns. In Proceedings of the Nokia MDC Workshop, Newcastle, UK, 18–19 June 2012.
10. Natarajan, N.; Shin, D.; Dhilloni, S. Which app will you use next?: Collaborative filtering with interactional context. In Proceedings of the Seventh ACM Conference on Recommender Systems, New York, NY, USA, 12 October 2013; pp. 201–208. [CrossRef]
11. Zou, X.; Zhang, W.; Li, S. Prophet: What app you wish to use next. In Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing, New York, NY, USA, 8–12 September 2013; pp. 167–170.
12. Shin, C.; Hong, J.; Deya, K. Understanding and prediction of mobile application usage for smart phones. In Proceedings of the 2012 ACM Conference on Ubiquitous Computing, New York, NY, USA, 5–8 September 2012; pp. 173–182.
13. Kostakos, V.; Ferreira, D.; Gonçalves, J. Modelling smart phone 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; pp. 486–497.
14. Yan, T.; Chu, D.; Ganesan, D. Fast app launching for mobile devices using predictive user context. In Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services, New York, NY, USA, 25–29 June 2012; pp. 113–126.
15. Xusx, Z.; Li, W. *Predicting Smart Phone App Usage with Recurrent Neural Networks*; Springer: Berlin/Heidelberg, Germany, 2018.
16. Xu, Y.; Zhu, Y.; Shen, Y. Leveraging app usage contexts for app recommendation: A neural approach. *World Wide Web* **2019**, *22*, 2721–2745. [CrossRef]
17. Tan, Y.; Yu, K.; Wu, X. Predicting app usage based on link prediction in user-app bipartite network. In Proceedings of the Smart Computing and Communication-Second International Conference, Shenzhen, China, 10–12 December 2017; Volume 10699, pp. 191–205.
18. Chen, X.; Wang, Y.; He, J. CAP: Context-aware app usage prediction with heterogeneous graph embedding. *IMWUT* **2019**, *3*, 1–25. [CrossRef]
19. Cen, Y.; Zou, X.; Zhang, J. Representation learning for attributed multiplex heterogeneous network. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Anchorage, AK, USA, 4–8 August 2019; pp. 1358–1368.
20. Mcmahan, B.; Moore, E.; Ramage, D. Communication-efficient learning of deep networks from decentralized data. In Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, Ft. Lauderdale, FL, USA, 20–22 April 2017; Volume 54, pp. 1273–1282.

21. Pan, X.; Chen, J.; Monga, R.; Bengio, S.; Rafal, J. Revisiting distributed synchronous SGD. *arXiv* **2017**, arXiv:1702.05800.
22. Konecný, J.; Mcmahan, H.B.; Ramage, D.; Richtárik, P. Federated optimization: Distributed machine learning for on-device intelligence. *arXiv* **2016**, arXiv:1610.02527.
23. Yang, K.; Fan, T.; Chen, T.; Shi, Y.; Yang, Q. A quasi-newton method based vertical federated learning framework for logistic regression. *arXiv* **2019**, arXiv:1912.00513.
24. Liu, Y.; Kang, Y.; Zhang, X.; Li, L.; Cheng, Y.; Chen, T.; Hong, M.; Yang, Q. A communication efficient vertical federated learning framework. *arXiv* **2019**, arXiv:1912.11187.
25. Konecný, J.; Mcmahan, H.B.; Yu, F.X.; Richtárik, P.; Suresh, A.T.; Bacon, D. Federated learning: Strategies for improving communication efficiency. *arXiv* **2016**, arXiv:1610.05492.
26. Mcmahanh, H.B.; Ramage, D.; Talwar, K.; Zhang, L. Learning differentially private language models without losing accuracy. *arXiv* **2017**, arXiv:1710.06963.
27. Bonawitz, K.; Ivanov, V.; Kreuter, B. Practical secure aggregation for federated learning on user-held data. *arXiv* **2016**, arXiv:1611.04482.
28. Bonawitz, K.; Ivanov, V.; Kreuter, B.; Marcedone, A.; McMahan, H.B.; Patel, S.; Ramage, D.; Segal, A.; Seth, K. Practical secure aggregation for privacy-preserving machine learning. In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, New York, NY, USA, 30 October 2017; pp. 1175–1191. [CrossRef]
29. Yang, Q.; Liu, Y.; Chen, T. Federated machine learning: Concept and applications. *ACMTIST* **2019**, *10*, 12:1–12:19. [CrossRef]
30. Cheng, K.; Fan, T.; Jin, Y. Secure boost: A loss less federated learning framework. *IEEE Intell. Syst.* **2021**. [CrossRef]
31. Malle, B.; Giuliani, N.; Kieseberg, P. *The More the Merrier—Federated Learning from Local Sphere Recommendations. Lecture Notes in Computer Science: Volume 10410 Machine Learning and Knowledge Extraction-First IFIP TC5, WG8. 4,8.9, 12.9 International Cross-Domain Conference*; Springer: Berlin/Heidelberg, Germany, 2017; pp. 367–373.
32. Chen, F.; Dong, Z.; Li, Z.; He, X. Federated meta-learning for recommendation. *arXiv* **2018**, arXiv:1802.07876.
33. Wang, X.; Han, Y.; Wang, C. In edge AI: Intelligentizing mobile edge computing, caching and communication by federated learning. *IEEE Netw.* **2019**, *33*, 156–165. [CrossRef]
34. Li, S.; Cheng, Y.; Liu, Y.; Wang, W.; Chen, T. Abnormal client behavior detection in federated learning. *arXiv* **2019**, arXiv:1910.09933.
35. Gao, D.; Ju, C.; Wei, X.; Liu, Y.; Chen, T.; Yang, Q. HHHFL: Hierarchical heterogeneous horizontal federated learning for electroencephalography. *arXiv* **2019**, arXiv:1909.05784.
36. Kim, H.; Park, J.; Bennis, M.; Kim, S.-L. On-device federated learning via blockchain and its latency analysis. *arXiv* **2018**, arXiv:1808.03949.
37. Lin, Z.; Feng, M.; Dos Santos, C.N. A structured self-attentive sentence embedding. In Proceedings of the 5th International Conference on Learning Representations, Toulon, France, 24–26 April 2017.
38. Shi, L.; Zhao, W.X.; Shen, Y. Local representative-based matrix factorization for cold start recommendation. *ACM Trans. Inf. Syst.* **2017**, *36*, 22:1–22:28. [CrossRef]
39. Zhang, Y.J.; Meng, X.W. Research on Group Recommender Systems and Their Applications. *Chin. J. Comput.* **2016**, *4*, 745–764.