IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received August 27, 2019, accepted September 5, 2019, date of publication September 12, 2019,
date of current version September 24, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2940752

DeepAttr: Inferring Demographic Attributes via Social Network Embedding

XUCHENG LUO[1], MINRUI XIE[1], KUNPENG ZHANG[2], FAN ZHOU[1], AND TING ZHONG[1]
[1]School of Information and Software Engineering, University of Electronic Science and Technology of China, Chengdu 610054, China
[2]Department of Decision, Operations and Information Technologies, University of Maryland at College Park, College Park, MD 20742, USA

Corresponding author: Xucheng Luo (xucheng@uestc.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61272527, Grant 61602097, and Grant 61472064, in part by the Sichuan Science-Technology Support Plan Program under Grant 2016GZ0065, and in part by the Fundamental Research Funds for the Central Universities under Grant ZYGX2015J072.

ABSTRACT Online Social Networks (OSNs) (e.g., Facebook, Twitter, and Tecent QQ) are popular platforms for people to share information online. The providers of OSNs typically leverage demographic attributes of users in OSNs to perform business behaviors like personalized commodity recommendation, advertisement delivery, etc. Thus, understanding demographic attributes of users in OSNs is of great importance. However, some users refuse to reveal their demographic information due to privacy concerns. Therefore, inferring demographic attributes of users in OSNs with public information is an attractive topic for researchers. Most existing methods mainly focus on individual attribute inference without taking the relationship among attributes into account, which results in low precisions. In this paper, we propose a novel approach, called DeepAttr, to infer users' multiple attributes simultaneously. DeepAttr leverages existing network embedding algorithm to learn the social embedding for each user. Meanwhile, it encodes multiple attributes into structured attribute vectors which are treated as class labels. The core component of DeepAttr is a multi-layer fully connected deep neural network which captures complex nonlinear mapping between the users' social embeddings and the structured attribute vectors. Extensive experiments on a real-world dataset demonstrate that DeepAttr outperforms existing models in both single-attribute inference and multiple-attribute inference as to macro-Precision, macro-Recall, and macro-F1.

INDEX TERMS Attribute inference, deep neural network, network embedding, online social network, structured attribute vector.

I. INTRODUCTION

Online Social Networks (OSNs) allow users to express their opinions, share content, and communicate with each others. With emerging large population adhering to the OSNs, many OSN-based applications have come forth, including personalized commodity recommendation, advertisement delivery, and etc. To be a successful application, having high quality data such as precise user profiling with accurate attributes is crucial. A typical OSN usually includes networked data (e.g., user friendships, user interactions), textual data (e.g., user generated content), and user profile information (e.g., gender, age, and interests, etc.). However, due to privacy concerns, only a small fraction of users make their demographics available to the public, which makes obtaining user demographics challenging. As a result, many research efforts have been made to infer these unobserved attributes through various machine learning methods.

Existing approaches on attribute inference [1]–[8] mainly focus on exploiting information related to user behavior to predict individual attribute separately. The behavioral data used among these approaches is used to learn latent user characteristics representation, which is inspired from the intuition that users with close characteristics (e.g., similar/same on some attributes) are more likely to behave similarly. In addition, researchers in [9]–[13] use writing style in textual content to as representative features for performing attribute inference. It is based on the assumption that linguistic features behind writing may reveal OSN user's some essential characteristics to a certain extent. These approaches have several shortcomings. For example, they heavily rely on various handcrafted features, which costly involves much human intervention. They also ignore the important explicit networked data (e.g., friendship relations). In [14]–[19], they

The associate editor coordinating the review of this manuscript and approving it for publication was Gang Li.

aim to infer attributes of target users by leveraging public available friendships and structural relations among users in the OSN. They mainly build their algorithms based on a well-known social network theory: homophily indicating that friends usually demonstrate similar characteristics [20]. For example, if the majority of a user's friends are college students, s/he is likely to be in college as well.

Furthermore, combining networked and behavioral data does gain some advantages in attribute inference, as shown in [21] and [22]. They propose a social-behavior attribute network model to take the influence of friendships and behavior attributes into consideration. But there are still some drawbacks. For example, they primarily focus on single attribute inference or inferring multiple attributes separately, which neglects the dependency among different attributes. Recently, Wang *et al.* [23] propose a model (called SNE) to automatically learn user representation from user purchase history data to infer multiple attributes simultaneously. SNE maps each item to a vector in a continuous space, and the vector of the purchased items are subjected to a pooling operation to obtain the user vector representation. This approach maximizes the probability of known user attributes during training while learning the vector representation of all users. Specifically, the representation learned from purchase data makes it easier to extract useful information for inferring multiple attributes. And it fully leverages the interrelations among multiple attributes to improve the inference accuracy. Since then, the representation learning [24] has been fairly successful in the area of multiple demographic attributes inference, such as in the retail scenario [25]. However, the retail scenario is usually different from OSNs in that there are rich data in user transactions but very few social relationships among users, which results in the fact that methods developed under the retail context are not directly applicable to OSNs.

For multiple demographic attribute inference, it can be viewed as a multi-task multi-class learning problem. Formulating as multi-task learning can improve the general performance by incorporating the potential information included in related tasks [26]. One possible way is to learn multiple related tasks in parallel by leveraging a shared representation [27], [28]. Analogous to multi-task learning, capturing the interrelation between different attributes helps to improve the accuracy of inference. For example, if a user majored in software engineering in college, then his profession as a software engineer would be far more likely than as a doctor. In other words, the probability of co-occurrence of software engineering major and software engineer career is much bigger than that of software engineering and doctor, which exemplifies interrelationship between demographic attributes. In [27], the authors found that considering the effect of the interrelation between gender and age can significantly improve the inference accuracy. The experiment results indicate that circle features in mobile social network contribute much more than friend features to gender inference, while friend features have greater effect on age inference.

Therefore, in this paper we propose a social graph embedding based approach, called *DeepAttr*, to infer multiple attributes simultaneously. DeepAttr takes only the social graph and known attributes from a subset of users as input. It overcomes the weakness of current methods in the following respects. (1) Unlike many existing methods that significantly rely on user attributes that are given, DeepAttr only requires attributes of a few users, which is easy to obtain. (2) It considers the interrelationships among different attributes. In general, DeepAttr contains three key components. The first one is the social graph embedding which learns distributed representations of users purely using the social graph. The second one is the Structured Attribute Vector (SAV), which encodes multiple user attributes as a binary vector. The last one is a multi-task (MT) predictor, which maps a user representation into a corresponding SAV. In practice, OSNs typically have some users with their demographic attributes revealed. We leverage these users' vector representation and corresponding SAVs to train the MT predictor. Compared with previous approaches, DeepAttr has the following merits: 1) There is no need for collecting vast amount of user behavioral data which is time-consuming and costly; 2) It takes into account the inherent connection among different attributes by SAVs, and thus the performance of multiple attribute inference is greatly boosted; 3) To map user representations to SAVs, DeepAttr leverages a MT predictor which can grasp the complex relation between the user representations and their corresponding SAVs. To evaluate the performance of DeepAttr on both single attribute inference and multiple attribute inference, we conduct experiments on a large-scale real world OSN dataset. The results show that our approach substantially outperforms several state-of-the-art baseline approaches. We summarize the contributions of this work as follows:

1) We design a novel scheme for inferring user attributes, which only requires the information of the social graph and attributes from a small set of users. Compared with existing designs, our scheme is more practical.
2) We encode user attributes into SAV which can take into account the relationships among different attributes and simplify the multiple attribute inference.
3) We propose a multi-task predictor to infer multiple attributes simultaneously, which illustrates the existence of complicated non-linear relationship between user vectors and their corresponding SAVs.

The rest of the paper is organized as follows. In section II, we summarize the related work about attribute inferences. Section III presents the DeepAttr in detail. Section IV shows the experimental results. Section V concludes this paper and discuss the future work.

II. RELATED WORK

In this section we review the related work in attribute inference. The existing work on attribute inference can be mainly divided into four categories: behavior-based

inference, friend-based inference, hybrid inference, and other approaches.

A. BEHAVIOR-BASED INFERENCE

Many research results have shown that user behavior is highly correlated with some of their attributes. Users with same behavior tend to have same or similar attributes. For example, Hu *et al.* [1] use the web page click data to learn a discriminative model for inferring gender and age. Malmi and Weber collect the list of APPs that users have installed, and create a high dimensional spatial feature vector for each user. It compares three different dimensional reduction methods and shows that logistic regression classifier perform the best when inferring five attributes including gender, age, race, marital status, and income. Wang *et al.* [23] propose a novel SNE model to automatically learn user representation from user's purchase data for predicting multiple attributes simultaneously. Their scheme can deal with both partial-label prediction and new-user prediction. Chaabane *et al.* [3] leverage the music information that users like to infer attributes. They generate augmented music description through the corresponding Wikipedia pages and then extract the topics of all similar music. A user is inclined to demonstrate similar attributes with those that like similar music topics. Weinsberg *et al.* [4] predict user genders in a recommendation system by exploiting user ratings on movies. They compare the performance of three classifiers of Naive Bayes, Singular Value Machine (SVM) as well as Logistic Regression (LR), and find that LR shows the best performance. Xiang *et al.* [5] infer user attributes using the spatio-temporal behavior of the same user across multiple OSNs. They represent each user using a combined feature vector converted from texts and images in Google+ and Twitter. A coupled projection matrix is learned to map feature vector to attribute vector, which ultimately leads to multiple attribute inference.

B. FRIENDSHIP-BASED INFERENCE

In contrast to behavior-based attribute inference where data is difficult to obtain, friendship-based attribute inference is more feasible. The fundamental idea is originated from homophily in OSN, which means that users are more likely to have similar attributes with their friends. Gong *et al.* [18] have shown that inferring users' missing attributes is useful for link (friend relationship) prediction task. They iteratively infer attributes and predict links on the basis of the social-attribute network (SAN) model. It attempts to improve the attribute inference by using the results of latest link prediction. The superiority of the proposed method is demonstrated via a variety of experiments. Perozzi and Skiena [29] apply DeepWalk to learn all user vector representations based on friendships. They then utilize partial user vectors and their gender information to build a linear regression model to infer genders for the rest of users. Likewise, Culotta *et al.* [15] present a regression model to infer user attributes using information about followers of each website on Twitter. In [16], He *et al.* harness Bayesian network to simulate friend relationships in social network. For instance, to infer attribute value of user X, they first established a Bayesian network from X's social network. By analyzing the Bayesian network, they obtained the probability that X has attribute value A and served it as the prior probability of attribute inference. Lindamood *et al.* [17] implement partial attribute inference through modifying naive Bayesian classifier. Mislove *et al.* [14] find that OSN users with the same attributes are more likely to become friends and form a close community. According to the friend relationships of two different OSNs, they discover the local community in the networks. The attributes of other users are inferred by employing the attribute information of the users in the community.

C. HYBRID INFERENCE

Gong and Liu [21] propose a hybrid scheme which combines the social relationship and the user behavior. They design a vote distribution attack (VIAL) under the constructed social-behavior-attribute (SBA) network model to perform attribute inference. VIAL leverages the degree of similarity to iteratively distribute a fixed vote capacity from a target user to all the other users in the SBA network. The inferred attribute of the target user is the attribute value that obtains the highest vote capacity. Jia *et al.* [22] design AttriInfer, which models the social network as a pairwise Markov Random Field (pMRF) based on social structure. AttriInfer uses behaviors to learn a prior probability that each user has a considered attribute, and then computes the posterior probability that each target user has the attribute based on the pMRF model to infer attribute. AttriInfer addresses two major limitations of VIAL. First, VIAL can only use the training users with a certain attribute to infer the corresponding attribute of the target user, while AttriInfer can infer target user's certain attribute by exploiting training users who do not have that attribute. Second, VIAL needs to perform a customized random walk for each target user and infer attributes one by one, whereas AttriInfer can simultaneously infer attributes for all target users. Sun *et al.* [30] propose content-enhanced network embedding (CENE) to jointly leverage social structure and content information for attribute inference. Using the network embedding technology, user content information can be regarded as a special node in CENE model. They compare CENE with other approaches based on either friend relationships or content information and demonstrate the superiority of CENE.

D. OTHER APPROACHES

Narayanan *et al.* [9] focus on the analysis of writing style to identify author's anonymity by comparing the text of the anonymous author with a series of text corpus of known author. Zhong *et al.* [6] study attribute inference by employing the location information of users in OSN. They extract rich semantics from location information to form the feature representations of multiple attributes, and finally utilize feature dimensionality reduction to infer attribute. Similar to Zhong, Li *et al.* [7] implement attribute inference on the basis

of user-shared location information in mobile social networks (MSNs). By identifying a group of users with similar position trajectories, they exploit the users in the group whose attributes are known to infer the corresponding attribute of the target user. Fink *et al.* [10] realize gender inference merely based on Twitter users' tweets information. They use the features derived from the user's tweets to train a classifier. Likewise, Volkova *et al.* [11] extract the lexical features from the user's tweets and deduce the Twitter user's attributes by training the log-linear model. In addition, Lansley and Longley [31] explore the distribution of age and gender by means of analyzing customer's forenames. Qian *et al.* [12] harness the knowledge graph as background information to construct a comprehensive and realistic model to achieve de-anonymization and attribute inference.

In summary, existing attribute inference schemes still have limitations. Apart from friend-based inference, other approaches all need to collect a vast amount of user behavior data or content information, which is time-consuming and expensive. Besides, the existing schemes scarcely consider the correlation among different attributes. Instead, they focus on inferring multiple different attributes separately, leading to poor performance.

III. THE PROPOSED APPROACH: DEEPATTR

In this section, we first present the overall framework of DeepAttr. Then we dive into the details of each component in DeepAttr, including the network embedding, the Structured Attribute Vector (SAV), and the multi-task (MT) predictor. Finally, we show how to employ the proposed MT predictor to infer user attributes and optimize the model.

A. THE OVERALL FRAMEWORK OF DEEPATTR

OSNs typically have some users with their attributes exposed while the others are not willing to reveal due to privacy concerns. As we discussed before, user attribute information is crucial in many applications. In this work, we aim to infer multiple attributes of users by leveraging social network structure and a small fraction of known user attributes. Specifically, the attributes of users in an OSN are divided into single-value attributes and multi-value attributes. Single-value attribute refers to the attribute that has several possible values, yet each user can only have one of the values, such as gender and age. While multi-value attribute means that the attribute has multiple possible values and each user may have more than one value for that attribute, such as job and school. A user may have engaged in a variety of jobs and have studied in different schools. In this work, we focus on inferring multiple single-value attributes of users in an OSN simultaneously.

The attribute inference problem can be described as follows. For users in an OSN, let $A = \{a_1, \ldots, a_i, \ldots, a_I\}$ denote the attribute set of a users, and each attribute a_i has p_i, $i = 1, 2, \ldots, I$ different values. The total quantity of all possible attribute value is $P = \sum_{i=1}^{I} p_i$. Let $X_k = \{x_1, x_2, \ldots, x_n\}$ represents the friend list of the k^{th} user and x_n denotes a user, then each user can be expressed as (X_k, A_k). Given a set $\{(X_k, A_k), k = 1, 2, \ldots, K\}$, where K is the number of users in the OSN, a function f should be learned to infer the attribute set of users whose attributes are unknown. In summary, the formal definition of attribute inference problem addressed in this work is as below.

Definition 1 (Attribute Inference Problem): Suppose we are given 1) an undirected social graph $G = (V, E)$, 2) a same single-value attribute set $A = \{a_1, \ldots, a_i, \ldots, a_I\}$ for each node $v \in V$, and 3) some nodes of V have known attribute values while others have unknown attribute values. Attribute inference aims to output each attribute value for users whose attribute values are unknown.

As to attribute inference, both the quantity of input data and the inference performance are the concerns. Models requiring little input data are preferred since harvesting a large amount of user data is very time-consuming, expensive, and even impossible due to the restrictions from OSNs. For an OSN, the network structure and some users' attributes are typically revealed. Therefore, leveraging such kind of information as input to infer attributes is a good choice. There are some existing works such as VIAL [21], AttriInfer [22], and CNN-based approach [32] which leverage both social relationship and social behavior to infer user attributes. Our work in this paper is complementary to them. Intuitively, we need to model the relation between users and their attributes, so that we can infer the attributes from a user representation by applying the model. However, to implement this idea is not trivial. Three key issues need to be well addressed. The first one is how to represent a user. The second one is how to represent attributes of a user. The last one is how to model the relation between a user representation and the attribute representation of this user.

To solve the aforementioned issues, we propose DeepAttr which employs a multi-task (MT) predictor to model the complex relationships between user representations and their corresponding attribute representations. Specifically, DeepAttr treats the attribute inference as a classification problem. The user representations are the input vectors. The attribute representations are the class labels. The MT predictor is the classifier. Thus, DeepAttr consists of three components, which is illustrated in Figure 1. The first one is the network embedding which maps each node of the OSN to a low-dimension real-value space. That is to say, we can use a fixed length vector to represent a user. This perfectly matches the input layer of the MT predictor, which is also a fixed number of input neurons. The next component is the MT predictor, which maps the user vector to a probability distribution of different SAVs. The relation between a user vector and the probability distribution of different SAVs is intricate and MT predictor is a good choice to capture such complex relationships. The final component is the SAV. To leverage the MT predictor, we need to transform the attribute inference problem into a regression or classification problem which can be tackled

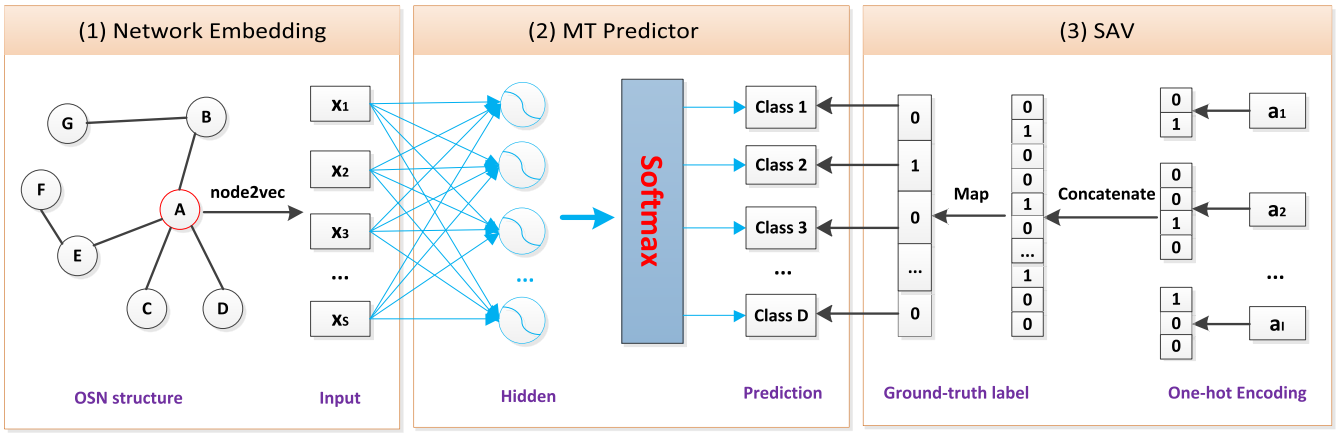


FIGURE 1. The framework of DeepAttr.

by MT predictor. The SAV is designed to turn the attribute inference problem into a classification one.

B. NETWORK EMBEDDING

DeepAttr infers users' attributes via network embedding where each node is represented as a low-dimension real-value vector. The details are described as follows. First, we regard the social structure of an OSN as an undirected graph namely user node relationship graph G, which consists of a set of user nodes V and a set of edges E representing friendship among users. Second, we use Node2Vec [33], an algorithmic framework for learning continuous feature representation for nodes in networks, to map nodes to a low-dimensional space of features that maximize the likelihood of preserving network neighbors of nodes. For embedding all nodes in G to a continuous feature space, let $F : V \rightarrow R^J$ be the mapping function from nodes to real value vector representations. Here J is the dimensionality of the embedding space. Then the objective function of Node2Vec is defined as follows:

$$\max_F \sum_{v \in V} [-\log Z_v + \sum_{n_i \in N(v)} F(n_i) \cdot F(v)], \tag{1}$$

where $Z_v = \sum_{u \in V} \exp(F(v) \cdot F(u))$, $N(v) \subset V$ is a network neighborhood of node v generated through random walk strategy. Unlike the traditional random walk, Node2Vec performs a biased random walk to balance depth-first sampling and breath-first sampling, which control walks by two parameters p and q to determine the retention of structural similarity or content similarity. After several rounds of random walks, we obtain a set of node sequences (called WalkList) of the OSN and then send them to Word2Vec [34], a tool for learning a vector to represent a word based on the context in a large corpus. Finally, we learn a real value vector $\mathbf{R_v}$ corresponding to each user $v (v \in V)$ in a J dimensional space (J generally ranges from tens to hundreds). The space complexity of the algorithm is $O(a^2|V|)$ where a is the average degree of the graph G and is usually small in real social networks.

C. STRUCTURED ATTRIBUTE VECTOR

The Structured Attribute Vector (SAV) transforms multiple single-value attributes into class labels. The building process of SAV is described as below. Our objective is to infer multiple single-value attributes $(a_1, a_2, \cdots, a_I)$ of users in the OSN. For the i^{th} attribute, we count the quantity of all attribute value occurring in the OSN and denote by $p_i, i = 1, 2, \ldots, I$. And then we construct a 0-1 vector for each attribute a_i. Each bit of the 0-1 vector corresponds to one attribute value, and the bit is set to 1 when one user has the corresponding attribute value and to 0 otherwise. Then we concatenate all the 0-1 vectors of I attributes to form a Combined Attribute Vector (CAV).

Then, the vital point of our method is to build a mapping table, in which the index is a CAV while the mapping value is a D-dimensional one-hot SAV. D is the amount of all possible CAVs, $D = \prod_{i=1}^{I} p_i$. Each possible CAV can be treated as a category. Thus D is also the number of classes. Let's begin with two attributes. Suppose there are two attribute vectors $\mathbf{c_1}$ and $\mathbf{c_2}$ for each user. Both $\mathbf{c_1}$ and $\mathbf{c_2}$ are one-hot row vectors. We first get the product of the transpose of $\mathbf{c_1}$ and $\mathbf{c_2}$, which is a matrix. Then, we concatenate the rows of the matrix sequentially to obtain the SAV. Figure 2 illustrates this process. In Figure 2, gender has two attribute values while age has five attribute values, and both of them are single-value attribute. We encode the two attributes into two one-hot vectors $\mathbf{c_1} = [1, 0]$ and $\mathbf{c_2} = [0, 1, 0, 0, 0]$, respectively. The product of $\mathbf{c_1}^T$ and $\mathbf{c_2}$ is as below:

$$\mathbf{c_1}^T \times \mathbf{c_2} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{2}$$

Thus the SAV is [0, 1, 0, 0, 0, 0, 0, 0, 0, 0]. If there are more than two attribute vectors, we can iterate the above process to get the final SAV. For example, each user has attribute vectors $\mathbf{c_1}$, $\mathbf{c_2}$, $\mathbf{c_3}$, and $\mathbf{c_4}$. Firstly, we map $\mathbf{c_1}$ and $\mathbf{c_2}$ to a one-hot vector $\mathbf{o_1}$. Then we map $\mathbf{o_1}^T$ and $\mathbf{c_3}$ to a one-hot vector $\mathbf{o_2}$. Finally, we map $\mathbf{o_2}^T$ and $\mathbf{c_4}$ to the SAV. In this way, each

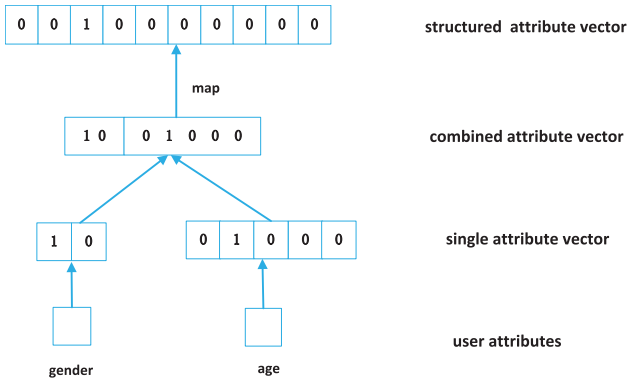


FIGURE 2. The construction of structured attribute vector.

CAV is mapped to a unique one-hot SAV. The SAV set can be denoted as $S = \{\mathbf{S_1}, \mathbf{S_2}, \ldots, \mathbf{S_D}\}$, where D is the amount of all possible CAVs. Mapping all attributes may result in high-dimension SAVs. However, it's not necessary to map all attributes to SAVs since some attributes are not correlated or slightly correlated. For a specific attribute inference task, we can select several most closely correlated attributes to construct the SAVs. Thus, the dimension of SAVs can be limited.

The mapping of multiple attributes of a user into a SAV label can make full use of the intrinsic connection among different attributes, which in turn transforms multiple attribute inference task into SAV category prediction. In this way, we transform the multi-task learning into a classification problem, such that the performance of attribute inference can be boosted due to the intrinsic ability of multi-task learning.

D. MULTI-TASK PREDICTOR

After obtaining an embedding vector $R_v(v \in V)$ in a J-dimensional space and a SAV vector $S_m(S_m \in S)$ in a D-dimensional space, we adopt supervised learning to train a multi-task (MT) predictor for inferring unknown users' attributes. The predictor mainly consists of multiple fully connected layers to learn a complicated function f which maps user embedding vectors to their corresponding SAVs:

$$f : \mathbf{R_v} \longrightarrow \mathbf{S_m}. \tag{3}$$

We take a five-layer fully connected neural network as an example to show how to obtain the parameters of MT attribute inference predictor. First, set the number of neurons in each layer as $N_L(L = 1, 2, \ldots, 5)$, and randomly initialize each layer's weight matrix $W_L(L = 1, 2, \ldots, 5)$ and bias vector $B_L(L = 1, 2, \ldots, 5)$. And then set the activation function φ for each neuron so that $a_j{}^L = \varphi(\sum_k w_{jk}{}^L a_k{}^{L-1} + b_j{}^L)$ represents the activation value of the j^{th} neuron of the L^{th} layer, where $z_j = \sum_k w_{j_k}{}^L a_k{}^{L-1} + b_j{}^L$ denotes the weighted input of the j^{th} neuron of the L^{th} layer, $w_{jk}{}^L$ denotes the link weight from the k^{th} neuron of the $(L-1)^{th}$ layer to the j^{th} neuron of the L^{th} layer, and $b_j{}^L$ denotes the bias of the j^{th} neuron of the L^{th} layer.

Next, we using some known attribute users' vector set X and their SAV set Y in the OSN to train the constructed MT predictor for getting parameters, where $X = \{\mathbf{R_1}, \mathbf{R_2}, \ldots, \mathbf{R_N}\}^T$ represents the training input, $Y = \{\mathbf{S_1}, \mathbf{S_2}, \ldots, \mathbf{S_N}\}^T$ represents the desired output. And the remaining known attribute users' vectors and corresponding SAVs serve as validation set. In order to make the output $\mathbf{S_m}(\mathbf{S_m} \in Y)$ of the MT predictor to fit all the input $\mathbf{R_v}(\mathbf{R_v} \in X)$ as much as possible, we define the following cost function:

$$C = -\frac{1}{N}\sum_X \sum_j [y_j \ln a_j{}^L + (1 - y_j)\ln(1 - a_j{}^L)], \tag{4}$$

where N is total number of training samples, and all elements in X are put into the neural network. The sum is done on all training inputs X. y_j denotes the target value of the output neurons, that is, the all elements of target vector $\mathbf{S_v}$, while $a_j{}^L$ denotes the actual value of the output neurons. By the form of the cost function, we can see that the closer the actual value of output neurons is to target value, the smaller the value the cost function. Thus, our ultimate goal is to minimize the cost function C, namely:

$$\min_{W_L, B_L} -\frac{1}{N}\sum_X \sum_j [y_j \ln a_j{}^L + (1 - y_j)\ln(1 - a_j{}^L)]. \tag{5}$$

For optimizing the above objective function, we adopt mini-batch gradient descent (MBGD) algorithm [35] to train the MT predictor. Since the error back propagation (BP) algorithm [36] is used in the training procedure, it is inevitable to take a derivative with respect to activation function. In order to prevent "gradient disappear" problem, the first four layers of hidden neurons use Rectified Linear Unit (ReLU) activation function; because the output of the MT predictor corresponds to a multi-class classification problem, the last layer of output neurons employ Softmax activation function. The ReLU function is calculated as follows:

$$\varphi(z) = max(0, z). \tag{6}$$

And the Softmax function is calculated as follows:

$$\varphi_j(z) = \frac{e^{z_j}}{\sum_{d \in D} e^{z_d}}, \tag{7}$$

where z is the weighted input of a hidden neuron, D is the number of output neurons, and $z_d(d \in D)$ denotes the weighted input of an output neuron. The Softmax function ensures that the sum of the values of all output neurons is 1, and the value of each output neuron corresponds to the probability of a combined attribute class vector. At this point, we can exploit the training set and MBGD algorithm to train the established MT predictor. To gain a stable model, the training process requires H iterations. For the $h^{th}(0 < h \leq H)$ iteration, we randomly select $r(r \ll N)$ sample points $\{E_v, S_m\}$ to calculate the gradients of the neurons' weight and bias of the output layer respectively. The formulas are as follows:

$$\nabla w_{jk}{}^L = \frac{1}{r}\sum_x a_k{}^{L-1}(a_j{}^L - y_j), \tag{8}$$

$$\nabla b_j^L = \frac{1}{r}\sum_x (a_j^L - y_j), \tag{9}$$

where $\nabla w_{jk}{}^L$ represents the gradient of weight, and $\nabla b_j{}^L$ represents the gradient of bias, $a_k{}^{L-1}$ represents the output of the k^{th} neuron of the $(L-1)^{th}$ layer. Then use the following equations to update the weight and bias of the output layer:

$$(w_{jk}{}^L)_h = (w_{jk}{}^L)_{h-1} - \eta \nabla w_{jk}{}^L, \tag{10}$$

$$(b_j^L)_h = (b_j^L)_{h-1} - \eta \nabla b_j^L, \tag{11}$$

where η denotes the learning rate of the MT predictor, and the error is passed forward by leveraging BP algorithm until all the weight and bias parameters of five layers are updated. With the increasing of iterations, the summation of cost function gradually become convergent, and the attribute inference accuracy of the validation set tend to be stable. When the number of iteration is H, the updated neuron weight matrix $W_L(L = 1, 2, \ldots, 5)$ and bias vector $B_L(L = 1, 2, \ldots, 5)$ are the desired MT predictor parameters.

E. ATTRIBUTE INFERENCE

Using the MT predictor trained in section III-D, we can infer multiple attributes of users. The output of the MT predictor is a D-dimensional probability vector, where D represents the number of output neurons in MT predictor as well as the dimension of the SAV. And each element of the probability vector means the probability value that the inferred user has the corresponding SAV. For the target user n, the process of attribute inference is as follows: Put the user vector $\mathbf{R_n}$ of n into the trained MT predictor. After the calculation of each layer of the neural network, the model outputs a probability vector. The SAV $\mathbf{S_v}$ of the target user can be found based on the index of the maximum element in the probability vector, which is consistent with the index of 1 in the SAV. Then according to the mapping table between the CAVs and the SAVs, the CAV corresponding to $\mathbf{S_v}$ is the attribute set of the target user. In our method, it is not necessary to infer the users' attributes one by one. We can obtain all the target users' attribute set by putting their user vectors into the MT predictor at the same time, which significantly improves the efficiency of attribute inference in OSN.

F. MODEL OPTIMIZATION

During the training of our MT predictor, we use the cross entropy cost function and the softmax to alleviate the slow learning issue and ReLU activation to address the ''gradient vanishing'' issue. However, due to existence of many hyper-parameters involved in MT predictor, there are still many important issues for consideration to obtain a stable and generalized MT predictor. For example, (1) model overfitting. When training samples are limited, the model is prone to be overfitting. We adopt a commonly used strategy dropout to effectively prevent the overfitting in two aspects. (a) It randomly ignores a small percentage of nodes in hidden layers in each training step. (b) It randomly makes some neurons in the hidden layers active with a different probability in each training phase. Such a randomization makes weight updating no longer depend on any fixed relationship among neurons. (2) The impact of the fixed learning rate η on the model convergence. If η remains constant throughout the training process, the loss value will show back and forth shock phenomenon, which may lead to a non-convergence situation. Theoretically, the learning rate should decrease as the number of training iteration increases, so that the loss value gradually reduces and consequently the model converges. Therefore, we leverage the learning rate decay strategy in the process of training MT predictor to speed up the convergence. Overall, the above two strategies are proved to be useful for optimizing our MT predictor. Specifically, they enhance both the stability and generalization of the proposed attribute predictor.

TABLE 1. User attributes in the dataset used for our experiments.

Attribute	Possible value
Gender	male,female
Age	15-25,25-35,35-45 45-55,55-60

IV. EVALUATION

We first describe the settings (training/testing split, parameter initialization) over the real-world dataset in our experiments and the baseline methods, then introduce the metrics used to evaluate models and finally present in detail the experimental observations regarding the advantages of our proposed method.

A. THE DATASET

We conduct our experiments over a widely used large-scale OSN dataset, namely Pokec,[1] which contains 30,622,564 undirected edges representing friend relationships among 1,632,803 user nodes. Each user has various attributes such as gender, age, height and weight, etc. Since the majority of users only disclose their genders and ages, and both of them are single-value attributes, they are chosen as target attributes for inference and evaluation. Therefore, only 1021003 users whose gender and age are public are selected for experimentation. From the practical perspective, users whose ages range between 15 and 60 are selected to conduct experiments. Moreover, we divide the age attribute into five groups, i.e., five different attribute values. The detailed attribute information of our task is show in Table 1.

B. TRAINING AND TESTING

To train a best model linking the input (user embedding vectors) to the output (user SAVs) for attribute inference, we perform some influential analysis on partial parameters, including: user embedding vector dimension, model network

[1] http://snap.stanford.edu/data/soc-pokec.html

layer, and the proportion of model training data and test data. When analyzing the influence of different parameters, except for the changes of the corresponding parameter settings, the other parameters in the MT predictor are set as follows: the number of iteration $H = 100,000$, the training batch size $r = 1,000$, the learning rate $\eta = [0.98, 0.08]$. As the number of iteration increases, the learning rate decreases from 0.98 to 0.08.

Firstly, we train a five-layer MT predictor to analyze stability and generalization of the model by using different training ratios/test ratios. The number of units (neurons) in each layer $N_L = \{N_1, N_2, N_3, N_4, N_5\} = \{1000, 800, 600, 250, 10\}$, the dimension of user embedding vector $S = 100$. We employ dropout to prevent our model from overfitting. The setting for each layer is $dropout\_vals = (1.0, 0.95, 0.95, 0.95, 1.0)$. In addition to selecting 10% of the data as the validation set, we divide the remaining 90% of the data into different training/ test ratios. With 10% as the increase unit, the training ratio is increased by 10% from 80%, and the corresponding test ratio is reduced from 80% to 10%. Secondly, we randomly choose 10% (about 100,000) of users as the testing set and 10% of user vectors and their corresponding SAVs as the validation set, the remaining 80% users are treated as training set. And the dimension of user embedding vector $J = 100$. Then the influence of the number of network layer on the inference results is evaluated by training several different hidden layer MT models. Finally, in order to analyze the impact of the dimension of the user embedding vector, we exploit Node2Vec to learn user vectors of six different dimensions. The hyper-parameters p and q are both set to 1, and the user vector of different dimensions is used as input to train a five-layer MT predictor with a training /testing ratio of 80%/10%.

In addition to analyzing the influence of some parameters on the prediction model, after deriving the best parameter settings, we need to train a best model for testing and compare the experimental results with other attribute inference methods to prove the superiority of our method.

C. BASELINES

Since DeepAttr is proposed for inferring multiple attributes simultaneously, we evaluate our model by comparing with several state-of-the-art methods. They all depend on the user vector representation which can be learned from the friend relationships in the OSN. We compare the performance of DeepAttr with single attribute inference methods and multiple-attribute inference methods. Note that VIAL [21] and the CNN-based approach [32] leverage both social relationship and social behavior to infer user attributes. However, DeepAttr only take social relationship as input. The problem addressed by DeepAttr is different from that of VIAL and the CNN-based approach. Therefore, it's not necessary to compare DeepAttr with them here.

For the single attribute inference, we compare DeepAttr with the following three classic methods.

1) **LR**: Logistic Regression (LR) [2], [4]. The input of LR here is the node vectors from Node2Vec. Since LR can only perform binary classification, we have to train two separate models for age and gender attributes, respectively.
2) **SVM**: Support Vector Machine (SVM) [1], [4]. The input of SVM is also the node vectors. Similar to LR, we also train two separate models for age and gender attributes, respectively.
3) **AttriInfer-Soc**: AttriInfer with only Social graph (AttriInfer-Soc) [22]. AttriInfer-Soc takes only social graph as input. Since it can only infer single attribute, we train two separate models for age and gender attributes, respectively.

Both LR and SVM can be directly called by the sklearn package.[2] The results of AttriInfer-Soc is reproduced by using the source code of AttriInfer.

For the multiple-attribute inference, we compare DeepAttr with the following two methods.

1) **JNE**: Joint Neural Embedding model [23]. It infers each attribute of a target user separately, and multiple attribute inference tasks can be performed in parallel. The attribute inference of a target user is correct only when multiple attributes are inferred correctly at the same time.
2) **SNE**: Structured Neural Embedding [23]. It is a typical multi-attribute inference method. This model maps several user attributes into a combined attribute vector, and then learns a linear transformation matrix between the user vector and the combined attribute vector. For a target user in the testing set, this method computes a score for each possible combined attribute vector, and the combined attribute vector with the highest score is considered as the inferred attribute set.

D. EVALUATION METRICS

Our attribute inference is a multi-class classification task. To evaluate the performance, we use the standard metrics: macro-Precision, macro-Recall and macro-F1 metrics. The macro-Precision is the inference accuracy from the user perspective while the macro-Recall is the inference accuracy from the perspective of combined attributes. In fact, the attribute inference task can be regarded as a D binary classification problem. We calculate the Precision (P) and the Recall (R) for each binary classification problem which is denoted by $(P_1, R_1), (P_2, R_2), \ldots, (P_D, R_D)$, where D is the amount of all possible CAVs. The macro-Precision and macro-Recall are computed as follows:

$$\text{macro-Precision} = \frac{1}{D}\sum_{i=1}^{I} P_i, \tag{12}$$

$$\text{macro-Recall} = \frac{1}{D}\sum_{i=1}^{I} R_i. \tag{13}$$

[2]https://scikit-learn.org/stable/

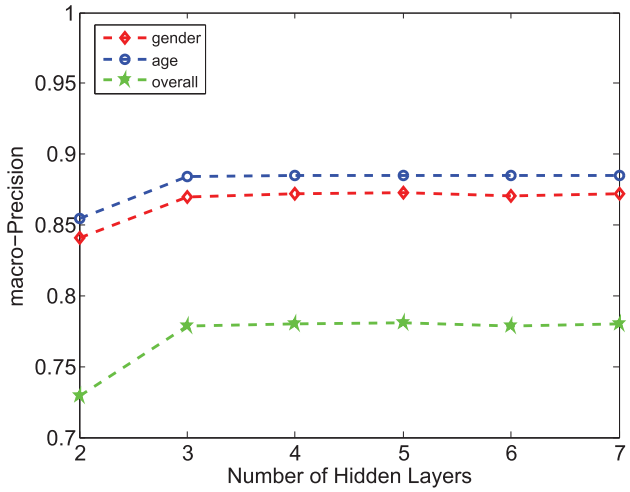


FIGURE 3. Different numbers of hidden layers v.s. macro-Precision.

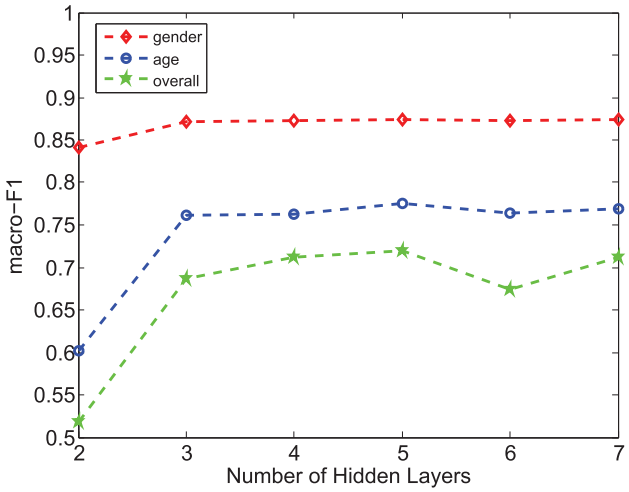


FIGURE 5. Different numbers of hidden layers v.s. macro-F1.

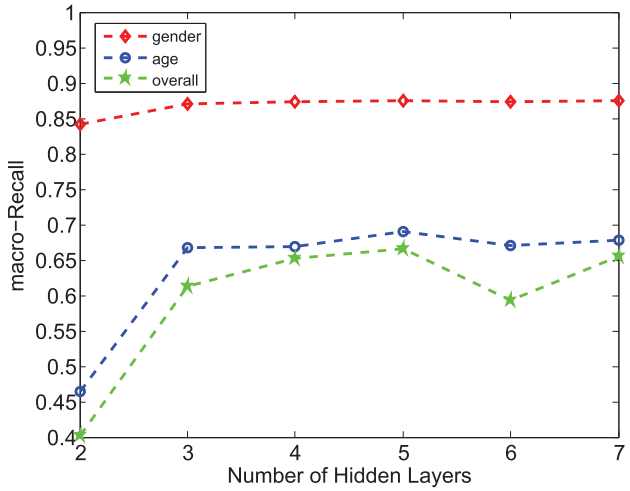


FIGURE 4. Different numbers of hidden layers v.s. macro-Recall.

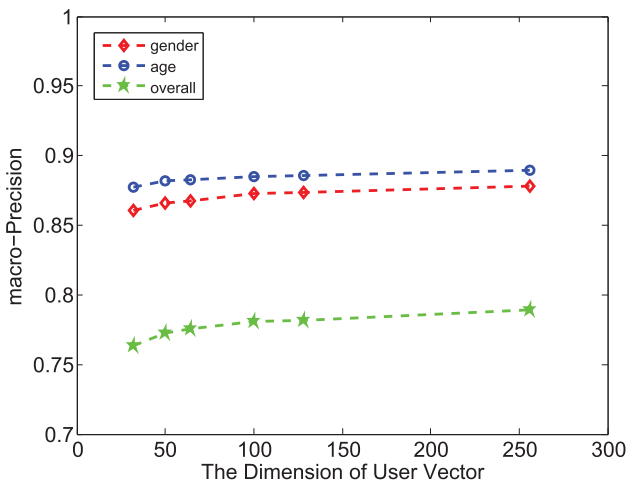


FIGURE 6. Different dimensions of user embedding vector v.s. macro-Precision.

And the macro-F1 is the harmonic mean of macro-Precision and macro-Recall:

$$\text{macro-F1} = 2 \times \frac{\text{macro-Precision} \times \text{macro-Recall}}{\text{macro-Precision} + \text{macro-Recall}}. \quad (14)$$

E. RESULTS

Table 2 shows the experimental results for single-attribute and multiple-attributes inference accuracy with different training/testing ratios. In it, macro-P and macro-R are short for macro-Precision and macro-Recall, respectively. These experiments employ a five-layer MT model with the same parameter settings as mentioned in Section IV-B. It tells us that the model achieves the highest macro-Precision, macro-Recall, and macro-F1 with the training/testing ratio of 80%/10% for both single and all attribute inference. The experimental results also show that our proposed model is stable and generalizable as long as the size of training set is enough (e.g., greater than the testing set). To find the optimal DeepAttr model, we empirically train models with different model configurations, especially the number of hidden layers and the dimension of user embedding vector.

Figure 3, 4, and 5 respectively demonstrate the macro-Precision, macro-Recall, and macro-F1 of attribute inference under DeepAttr model with different hidden layers. These figures all show that after 3 layers the model becomes stable and 5 layers achieves the best performance.

Figure 6, 7, and 8 respectively demonstrate the macro-Precision, macro-Recall, and macro-F1 of attribute inference under DeepAttr model with different user vector dimensions as input. These figures indicate that the attribute inference performance is best when the dimension of user vector is 100. Meanwhile, when the dimension of user vector fluctuates within the range of tens to hundreds, the result is fluctuating but has little influence, indicating that the user vector learned by Node2Vec contains enough friend relationship information.

In addition to the aforementioned, in contrast to other attribute inference methods, DeepAttr also shows great superiority. Figure 9, 10, and 11 separately demonstrate the

TABLE 2. Performance comparison under different training-testing ratios.

training/	*gender*			*age*			*overall*		
testing ratio	*macro-P*	*macro-R*	*macro-F1*	*macro-P*	*macro-R*	*macro-F1*	*macro-P*	*macro-R*	*macro-F1*
80%/10%	87.30%	87.57%	87.43%	88.49%	69.07%	77.58%	78.10%	66.64%	71.92%
70%/20%	86.97%	87.17%	87.07%	87.40%	64.89%	74.48%	76.93%	63.58%	69.62%
60%/30%	86.82%	86.89%	86.85%	86.96%	64.17%	73.85%	76.38%	56.17%	64.73%
50%/40%	85.86%	86.08%	85.97%	85.55%	64.49%	73.54%	74.36%	56.96%	64.51%
40%/50%	83.66%	84.48%	84.07%	84.27%	63.24%	72.26%	71.35%	56.76%	63.22%
30%/60%	81.49%	82.06%	81.77%	82.78%	62.14%	70.99%	68.28%	54.65%	60.71%
20%/70%	78.89%	79.65%	79.27%	79.94%	59.89%	68.48%	64.00%	51.71%	57.20%
10%/80%	74.63%	75.36%	74.99%	73.50%	50.71%	60.01%	56.15%	41.97%	48.04%

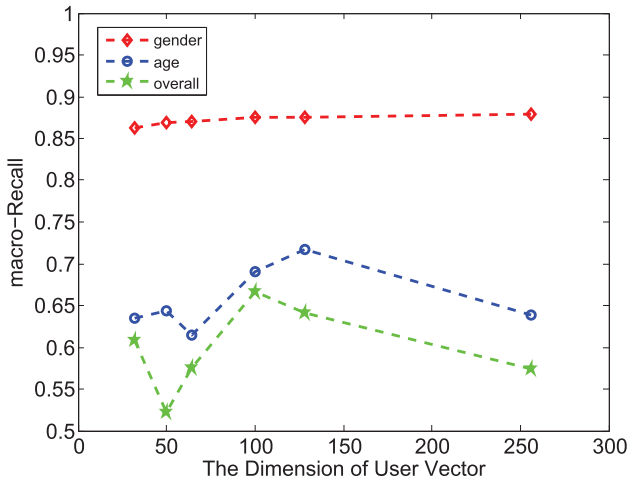


FIGURE 7. Different dimensions of user embedding vector v.s. macro-Recall.

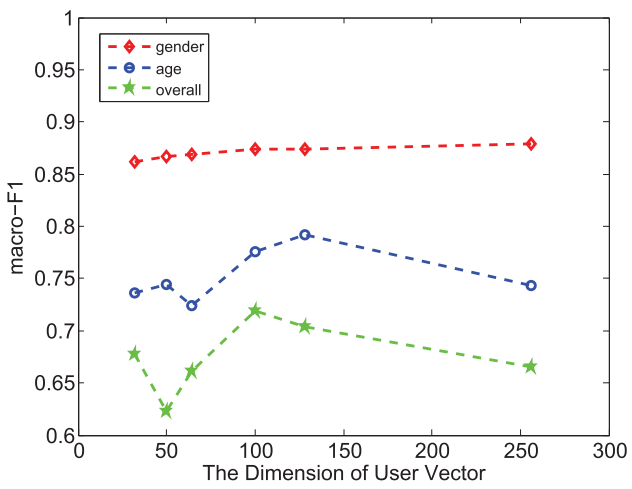


FIGURE 8. Different dimensions of user embedding vector v.s. macro-F1.

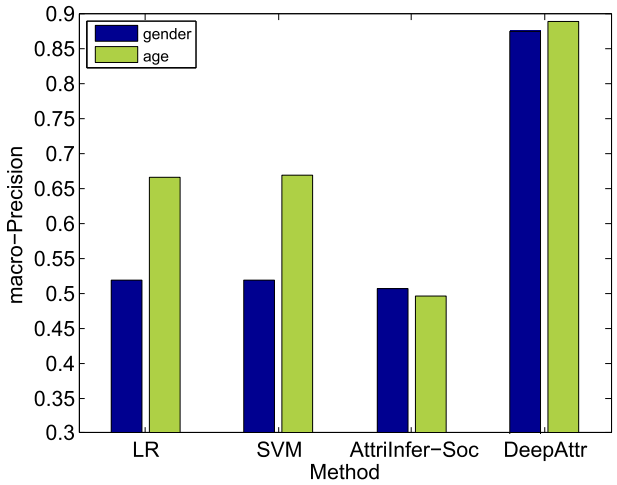


FIGURE 9. The macro-Precision for different single-attribute inference methods.

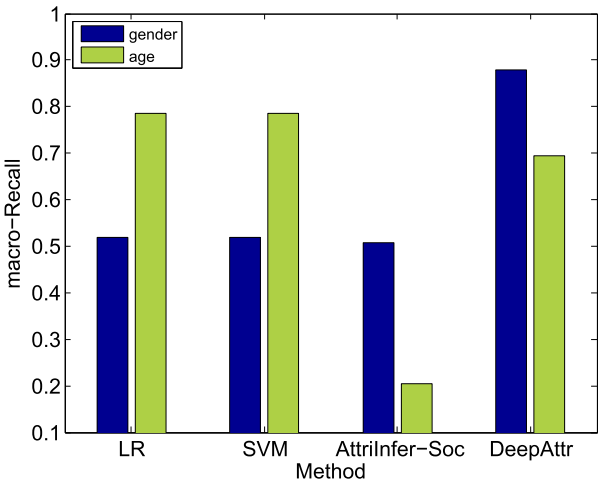


FIGURE 10. The macro-Recall for different single-attribute inference methods.

macro-Precision, macro-Recall, and macro-F1 for different single-attribute inference methods. Since LR, SVM, and AttriInfer-Soc infer two user attributes separately, we only demonstrate the performance of single attribute. From the figures we can draw the following conclusions: the macro-Precisions of DeepAttr are much higher in both attributes than LR, SVM, and AttriInfer-Soc, but the macro-Recall of age is slightly worse than LR and SVM. In terms of the combination of the two metrics, DeepAttr has achieved a big improvement in macro-F1, which explains that DeepAttr takes full advantage of the intrinsic correlation between the two attributes.

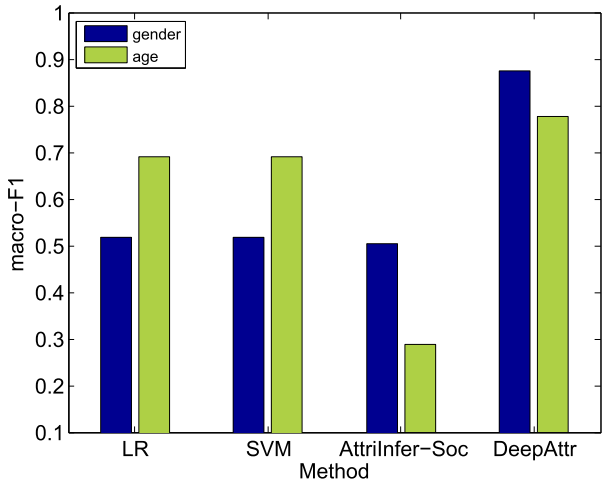


FIGURE 11. The macro-F1 for different single-attribute inference methods.

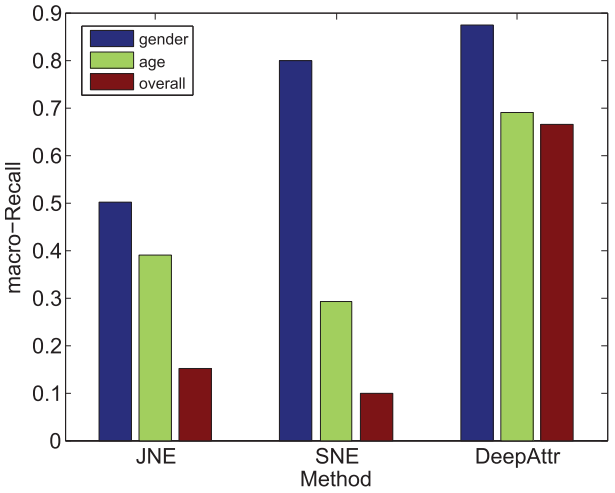


FIGURE 13. The macro-Recall for different multiple-attribute inference methods.

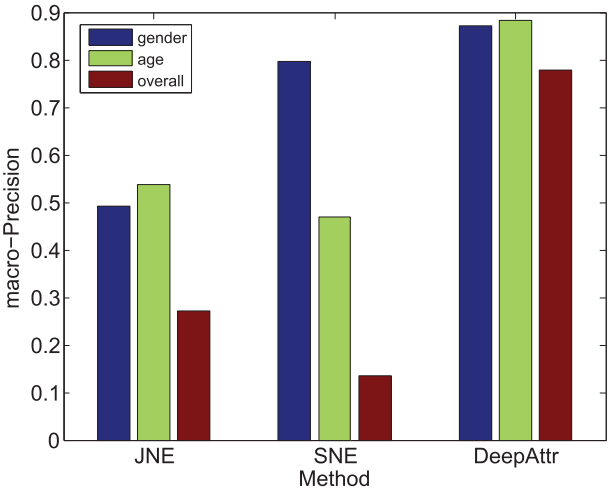


FIGURE 12. The macro-Precision for different multiple-attribute inference methods.

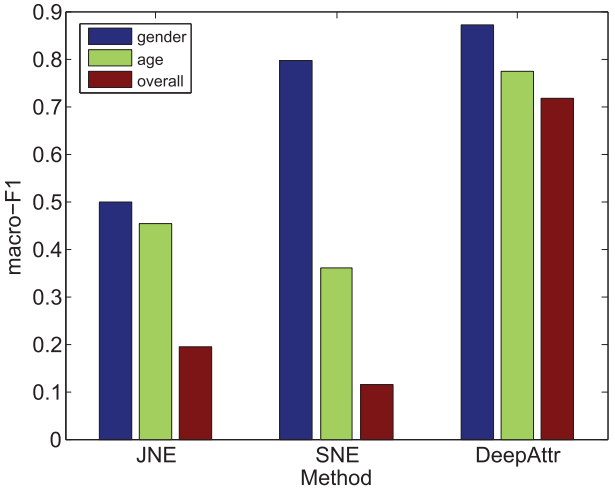


FIGURE 14. The macro-F1 for different multiple-attribute inference methods.

Figure 12, 13, and 14 separately demonstrate the macro-Precision, macro-Recall and macro-F1 for different multiple attribute inference methods. Since SNE and DeepAttr infer two user attributes simultaneously, we also show the performance in terms of these three metrics for each single attribute. The figures show that (1) DeepAttr tremendously outperforms JNE and SNE in terms of overall macro-Precision, macro-Recall and macro-F1. Compared to JNE, DeepAttr makes full use of some latent relations between different attributes (e.g., there is a strong relationship between gender and age in the dataset). It outperforms SNE because DeepAttr further maps the CAV into the SAV, which simplifies the multi-task inference problem into the multi-class inference problem. In addition, SNE utilizes the linear relationship between the user vector and the CAV, while DeepAttr learns the non-linear relation between the user vector and the SAV, which dramatically improves the performance. For the single-attribute inference, DeepAttr achieves significantly improvement in age over the baselines. In addition, we also find that the accuracy of JNE and SNE are greatly decreased as the number of attribute values increases.

V. CONCLUSION AND FUTURE WORK

In this work, we study the problem of multi-attribute inference in online social network (OSN) via social network embedding. We propose DeepAttr to infer multiple user attributes simultaneously by designing a multi-task (MT) predictor. Specifically, DeepAttr exploits Node2vec, a social network embedding method, to automatically learn user vector representation. Furthermore, DeepAttr encodes multiple user attributes into a structured attribute vector (SAV), thereby transforming the multi-attribute inference problem into a multi-class prediction problem, which fully considers the intrinsic relationship among attributes. Ultimately, DeepAttr trains a MT predictor to learn the complex non-linear mapping from user embedding vectors to their corresponding SAVs for inferring attributes simultaneously. We compare DeepAttr with several state-of-the-art approaches by

conducting several experiments on a large-scale social network with more than one million users. The experiment results demonstrate that our method substantially outperforms other methods in terms of accuracy for both single-attribute and multiple-attribute inferences. We also conduct numerous parametric analysis experiments to obtain the best parameter settings. For our future work, it is a valuable direction to add some user attribute and content information when learning user representation for further attribute inference.

REFERENCES

[1] J. Hu, H. J. Zeng, H. Li, C. Niu, and Z. Chen, "Demographic prediction based on user's browsing behavior," in *Proc. 16th Int. Conf. World Wide Web*, 2007, pp. 151–160.

[2] E. Malmi and I. Weber, "You are what Apps you use: Demographic prediction based on user's Apps," in *Proc. 10th Int. AAAI Conf. Web Social Media*, 2016, pp. 635–638.

[3] A. Chaabane, G. Acs, and M. A. Kaafar, "You are what you like! Information leakage through users' interests," in *Proc. 19th Annu. Netw. Distrib. Syst. Secur. Symp.*, 2012, pp. 1–14.

[4] U. Weinsberg, S. Bhagat, S. Ioannidis, and N. Taft, "BlurMe: Inferring and obfuscating user gender based on ratings," in *Proc. 6th ACM Conf. Recommender Syst.*, Sep. 2012, pp. 195–202.

[5] L. Xiang, J. Sang, and C. Xu, "Demographic attribute inference from social multimedia behaviors: A cross-OSN approach," in *Proc. Int. Conf. Multimedia Model.*, 2017, pp. 515–526.

[6] Y. Zhong, N. J. Yuan, W. Zhong, F. Zhang, and X. Xie, "You are where you go: Inferring demographic attributes from location check-ins," in *Proc. 8th ACM Int. Conf. Web Search Data Mining*, Feb. 2015, pp. 295–304.

[7] H. Li, H. Zhu, S. Du, X. Liang, and X. Shen, "Privacy leakage of location sharing in mobile social networks: Attacks and defense," *IEEE Trans. Dependable Secure Comput.*, vol. 15, no. 4, pp. 646–660, Jul./Aug. 2018.

[8] M. Kosinski, D. Stillwell, and T. Graepel, "Private traits and attributes are predictable from digital records of human behavior," *Proc. Nat. Acad. Sci. USA*, vol. 110, no. 15, pp. 5802–5805, 2013.

[9] A. Narayanan, H. Paskov, N. Z. Gong, J. Bethencourt, E. Stefanov, E. C. R. Shin, and D. Song, "On the feasibility of internet-scale author identification," in *Proc. IEEE Symp. Secur. Privacy*, May 2012, pp. 300–314.

[10] C. Fink, J. Kopecky, and M. Morawski, "Inferring gender from the content of tweets: A region specific example," in *Proc. 6th Int. AAAI Conf. Weblogs Social Media*, 2012, pp. 459–460.

[11] S. Volkova, Y. Bachrach, M. Armstrong, and V. Sharma, "Inferring latent user properties from texts published in social media," in *Proc. 29th AAAI Conf. Artif. Intell.*, Jan. 2015, pp. 4296–4297.

[12] J. Qian, X.-Y. Li, C. Zhang, and L. Chen, "De-anonymizing social networks and inferring private attributes using knowledge graphs," in *Proc. 35th Annu. IEEE Int. Conf. Comput. Commun.*, Apr. 2016, pp. 1–9.

[13] J. Otterbacher, "Inferring gender of movie reviewers: Exploiting writing style, content and metadata," in *Proc. 19th ACM Int. Conf. Inf. Knowl. Manage.*, Oct. 2010, pp. 369–378.

[14] A. Mislove, B. Viswanath, K. P. Gummadi, and P. Druschel, "You are who you know: Inferring user profiles in online social networks," in *Proc. 3rd ACM Int. Conf. Web Search Data Mining*, Feb. 2010, pp. 251–260.

[15] A. Culotta, N. K. Ravi, and J. Cutler, "Predicting the demographics of Twitter users from website traffic data," in *Proc. 29th AAAI Conf. Artif. Intell.*, Jan. 2015, pp. 72–78.

[16] J. He, W. W. Chu, and Z. V. Liu, "Inferring privacy information from social networks," in *Proc. 4th IEEE Int. Conf. Intell. Secur. Inform.*, 2006, pp. 154–165.

[17] J. Lindamood, R. Heatherly, M. Kantarcioglu, and B. Thuraisingham, "Inferring private information using social network data," in *Proc. 18th Int. Conf. World Wide Web*, Apr. 2009, pp. 1145–1146

[18] N. Z. Gong, A. Talwalkar, L. Mackey, L. Huang, E. Chul R. Shin, E. Stefanov, E. R. Shi, and D. Song, "Joint link prediction and attribute inference using a social-attribute network," *Trans. Intell. Syst. Technol.*, vol. 5, no. 2, p. 27, 2014.

[19] E. Zheleva and L. Getoor, "To join or not to join: The illusion of privacy in social networks with mixed public and private user profiles," in *Proc. 18th Int. Conf. World Wide Web*, 2009, pp. 531–540

[20] M. McPherson, L. Smith-Lovin, and J. M. Cook, "Birds of a feather: Homophily in social networks," *Annu. Rev. Sociol.*, vol. 27, no. 1, pp. 415–444, 2001.

[21] N. Z. Gong and B. Liu, "You are who you know and how you behave: Attribute inference attacks via users' social friends and behaviors," in *Proc. 25th USENIX Secur. Symp. Secur.*, 2016, pp. 979–995.

[22] J. Jia, B. Wang, L. Zhang, and N. Z. Gong, "AttriInfer: Inferring user attributes in online social networks using Markov random fields," in *Proc. 26th Int. Conf. World Wide Web*, Apr. 2017, pp. 1561–1569

[23] P. Wang, J. Guo, Y. Lan, J. Xu, and X. Cheng, "Your cart tells you: Inferring demographic attributes from purchase data," in *Proc. 9th ACM Int. Conf. Web Search Data Mining*, Feb. 2016, pp. 173–182.

[24] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1798–1828, Aug. 2013.

[25] P. Wang, J. Guo, Y. Lan, J. Xu, and X. Cheng, "Multi-task representation learning for demographic prediction," in *Proc. Eur. Conf. Inf. Retr.*, 2016, pp. 88–99.

[26] R. Caruana, "Multitask learning," *Mach. Learn.*, vol. 28, no. 1, pp. 41–75, Jul. 1997.

[27] Y. Dong, Y. Yang, J. Tang, Y. Yang, and N. V. Chawla, "Inferring user demographics and social strategies in mobile social networks," in *Proc. 20th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2014, pp. 15–24.

[28] Y. Ji and S. Sun, "Multitask multiclass support vector machines: Model and experiments," *Pattern Recognit.*, vol. 46, no. 3, pp. 914–924, Mar. 2013.

[29] B. Perozzi and S. Skiena, "Exact age prediction in social networks," in *Proc. 24th Int. Conf. World Wide Web*, May 2015, pp. 91–92

[30] X. Sun, J. Guo, X. Ding, and T. Liu, "A general framework for content-enhanced network representation learning," Oct. 2016, *arXiv:1610.02906*. [Online]. Available: https://arxiv.org/abs/1610.02906

[31] G. Lansley and P. Longley, "Deriving age and gender from forenames for consumer analytics," *J. Retailing Consum. Services*, vol. 30, pp. 271–278, May 2016.

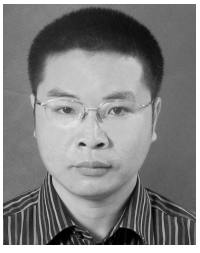
[32] X. Li, Y. Cao, Y. Shang, Y. Liu, J. Tan, and L. Guo, "Inferring user profiles in online social networks based on convolutional neural network," in *Proc. 10th Int. Conf. Knowl. Sci., Eng. Manage.* Melbourne, VIC, Australia: Springer, Aug. 2017, pp. 274–286.

[33] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2016, pp. 855–864.

[34] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proc. Adv. Neural Inf. Process. Syst.*, 2013, pp. 3111–3119.

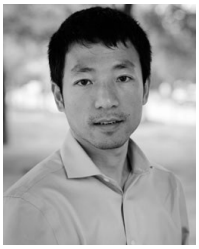
[35] M. Li, T. Zhang, Y. Chen, and A. J. Smola, "Efficient mini-batch training for stochastic optimization," in *Proc. 20th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2014, pp. 661–670.

[36] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, pp. 533–536, Oct. 1986.

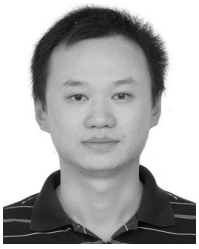


XUCHENG LUO received the M.S. and Ph.D. degrees from the University of Electronic Science and Technology of China, in 2003 and 2008, respectively, where he is currently an Associate Professor at the School of Information and Software Engineering. His research interests include machine learning, graph-structured data analysis, and network security.

MINRUI XIE received the B.S. degree in software engineering from the Chongqing University of Posts and Telecommunications, China, in 2016, and the M.S. degree from the University of Electronic Science and Technology of China, in 2019. Her current research interests include the application of network representation learning algorithms in attribute inference, network alignment, and link prediction.



KUNPENG ZHANG received the Ph.D. degree in computer science from Northwestern University. He is currently a Researcher in large-scale data analysis with particular focuses on mining social media data through machine learning, network analysis, and natural language processing techniques. He is also an Assistant Professor with the Department of Information Systems, Smith School of Business, University of Maryland at College Park, College Park. He published articles in the area of social media, text mining, network analysis, and information systems on top conference and journals. He serves as program committees for many international conferences. He is currently an Associate Editor of the *Electronic Commerce Research* journal.



FAN ZHOU received the B.S. degree in computer science from Sichuan University, China, in 2003, and the M.S. and Ph.D. degrees from the University of Electronic Science and Technology of China, in 2006 and 2011, respectively, where he is currently an Associate Professor at the School of Information and Software Engineering. His research interests include machine learning, spatio-temporal data management, and social network knowledge discovery.

TING ZHONG received the B.S. degree in computer application and the M.S. degree in computer software and theory from Beijing Normal University, Beijing, China, in 1999 and 2002, respectively, and the Ph.D. degree in information and communication engineering from the University of Electronic Science and Technology of China, Chengdu, China, where she is currently an Associate Professor with the School of Information and Software Engineering. Her research interest includes deep learning, social networks, and cloud computing.