FeatNet: Large-scale Fraud Device Detection by Network Representation Learning with Rich Features

Chao Xu
Baidu X-Lab
Beijing, China
xuchao09@baidu.com

Zhentan Feng
Baidu X-Lab
Beijing, China
fengzhentan@baidu.com

Yizheng Chen
Baidu X-Lab
Sunnyvale, CA, USA
chenyizheng01@baidu.com

Minghua Wang
Baidu X-Lab
Beijing, China
wangminghua01@baidu.com

Tao Wei
Baidu X-Lab
Sunnyvale, CA, USA
lenx@baidu.com

ABSTRACT

Online fraud such as search engine poisoning, groups of fake accounts and opinion fraud is conducted by fraudsters controlling a large number of mobile devices. The key to detect such fraudulent activities is to identify devices controlled by fraudsters. Traditional approaches that fingerprint devices based on device metadata only consider single device information. However, these techniques do not utilize the relationship among different devices, which is crucial to detect fraudulent activities. In this paper, we propose an effective device fraud detection framework called FeatNet, which incorporates device features and device relationships in network representation learning. Specifically, we partition the device network into bipartite graphs and generate the neighborhoods of vertices by revised truncated random walk. Then, we generate the feature signature according to device features to learn the representation of devices. Finally, the embedding vectors of all bipartite graphs are used for fraud detection. We conduct experiments on a large-scale data set and the result shows that our approach can achieve better accuracy than existing algorithms and can be deployed in the real production environment with high performance.

CCS CONCEPTS

• **Computing methodologies** → *Machine learning approaches*; *Machine learning algorithms*;

KEYWORDS

fraud detection; heterogeneous information network; network representation learning; node embedding

ACM Reference Format:
Chao Xu, Zhentan Feng, Yizheng Chen, Minghua Wang, and Tao Wei. 2018. FeatNet: Large-scale Fraud Device Detection by Network Representation Learning with Rich Features. In *11th ACM Workshop on Artificial Intelligence and Security (AISec '18), October 19, 2018, Toronto, ON, Canada.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3270101.3270109

1 INTRODUCTION

With the rapid development of the mobile internet, both account and device information are essential for anti-fraud authentication [2]. For example, fraudsters could register different fake accounts to participate in the sales promotion of e-commerce companies in order to get the discount vouchers or steer prices [8]. Traditionally, a special rule is applied to make sure each account can only get one voucher from one device, since the cost of faking accounts is much cheaper than building a mobile device cluster. Therefore, device fingerprinting technique becomes a core technology to identify unique devices. However, research [21] has demonstrated that only device fingerprint is not enough to be used for authentication, and it could be harmful for the authentication system because the fingerprints carry a lot of similarity, even across models and brands in some particular scenario. To this end, we propose a framework for fraud device detection called FeatNet. The relational and statistical information with the devices is mined in this framework. We utilize relationships between the devices, including mobile devices in the same Wi-Fi local network, or shared by one same person. Specifically, we construct a graph where the vertices are device, WI-FI AP, and user account, and the edges between them denote “device connects WI-FI AP” and “account logins in device” correspondingly. The intuition is that the same WIFI and bad accounts may be used by multiple devices controlled by the fraudsters. In addition to relational information, statistics extracted from devices can also strongly indicate whether they are used by fraudsters. For example, devices that are rooted, have installed repackaged apps, and always being charged have a higher likelihood of being used by fraudsters.

Traditionally, to find anomalies of devices by their relationships, label propagation methods [20], or rules like vertex degree counts are applied in risk management, both of which fail to utilize the global network information. Recently, research in the area of network representation learning [1] [19] [24] utilizes intrinsic information in network and learns distributed representations of vertices or edges. However, there are challenges to transfer these work to fraud detection for the following reasons: the scale of dataset is up to hundreds of million; the dictionary of device ID is too large to be embedded; the statistical features of vertices should be integrated

AISec '18, October 19, 2018, Toronto, ON, Canada
© 2018 Association for Computing Machinery.
ACM ISBN 978-1-4503-6004-3/18/10…$15.00
https://doi.org/10.1145/3270101.3270109

into network structure of similar importance. We propose FeatNet to solve these challenges by making the following contributions:

- We define the device network for detecting fraud devices, which links devices together by physical or manmade time-related relationships. In the device network, risky devices are inclined to form clusters and we predict the risk probability by incorporating device features with device relationships.
- We propose FeatNet, an unsupervised NRL framework for the device network: large-scale heterogeneous device network is partitioned to learn the distributed representation of devices; to preserve the semantic closeness, the feature signature of vertex is introduced in neighborhood sampling strategy.
- We conduct experiments on the spamdexing detection of Baidu mobile search with tens of millions of devices. Results show that FeatNet has significant improvement over baseline algorithms with high scalability in practice.

This paper proceeds as follow: related work of fraud detection and NRL methods are reviewed in section 2. In section 3, after the definition of device network, the FeatNet framework is introduced: the objective function of the problem is designed and studied firstly; then we define feature signature and discuss the graph partition and vertex sampling strategy; we briefly outline the learning procedure of the overall framework lastly. In section 4, experiments on spamdexing detection is conducted with practical data. We conclude with FeatNet framework and discuss some directions for future works in section 5.

2 RELATED WORK

Extensive research has been done to predict the label of nodes by extracting structural information in the network. The conventional method [3] [9] generates hand-crafted node features based on network properties such as degree centrality and neighborhood connectivity. Recently, Network Representation Learning (NRL) has attracted much attention, which learns the feature representations of vertices by defining and optimizing the objective function. LINE [24] preserves first order and second order proximity of vertices and concatenated the two distributed representations, however, it is insufficient to take full advantage of network. GraRep [4] generalizes LINE to incorporate information from neighborhoods beyond 2-step, but computing of high-order proximities is time-consuming and does not scale practically. SDNE [26] generalizes LINE to a semi-supervised deep model and adapts auto-encoder to embed the similarity vector of vertex. NEU [28] approximates higher order proximity matrix to enhance the performance of any NRL methods, while it supports tens of thousands of vertices with millions of edges even though.

An alternative approach is to sample the vertex's neighbors by random walks and learn the representation from the resulting context. DeepWalk [19] generates linear context of vertices by truncated random walks and adopts skip-gram model for the representation learning, which exploit different orders of proximity. Node2vec [6] generalizes DeepWalk with BFS and DFS of random walks and explores diverse neighborhoods. TADW [27] proves that random walks based method is equivalent to matrix factorization. Except for the equivalence of structural neighborhoods in device network, many device features exhibit a strong homophily (e.g., groups of products of the same type are controlled by fraudsters). Therefore, the methods such as node2vec can be outperformed by latent representations that better capture device homophily (as we soon show).

More generally, feature learning in heterogeneous network is challenging since it is hard to preserve the concept of node-context with arbitrary types of vertices and incomparable weight of edges. PTE [23] generalizes LINE to heterogeneous text network by partitioning the whole network, and utilizes both labeled and unlabeled data. To address the link prediction problem in academic network, studies [5] [10] [22] define the composite relations as meta-paths between two nodes and adopts meta-path-based random walks to generate heterogeneous vertex neighbors biased by vertex type, while it is difficult to model similarities between nodes without connected mate-paths. LSHM [11] assumes that the labels and tags of vertices are inter-dependent due to the influence of neighbors, and learns the latent node representations by extending homogeneous label propagation model. These research collectively demonstrate that the embedding of heterogeneous network differs on specific problems.

In addition, NRL with extra information is another interesting topic. There is usually rich information in vertices and edges in device network, such as device hardware information, historical risk labels, etc. The approaches above mainly devote to preserve the network structure, but they cannot be generalized to deal with features trivially. TADW [27] incorporates text features of vertices into NRL under the framework of matrix factorization. Most embedding frameworks, such as SDNE, are inherently transductive and can only generate embedding for a single fixed graph. The alternative method is neighborhood aggregation algorithm. GCN [12] operates convolutional neural network (CNN) directly on graphs and introduces a graph-based model for semi-supervised classification. GraphSAGE [7] leverages node feature information to efficiently generate node embedding for previously unseen data. CANE [25] notices the different aspects of vertex when it interacts with different neighbors. It uses a CNN to obtain text embeddings according to structure based objective as well as text based objective. However, CNN requires large amount of labeled samples which is impractical in anti-fraud scenario, and the tuning of deep model involves many parameters, which makes the gained knowledge hard to transfer to related problems.

3 FEATURE LEARNING FRAMEWORK

In this section, we firstly formulate the embedding method of the device network with various features of vertex, and then introduce the approach of FeatNet and the components of our algorithm.

3.1 Problem Definition

Definition 1. Device network is a social network of devices linked by physical or man-made relationships, which consists of type I vertex and type II vertex. Type I vertex is the device itself, identified by device fingerprint. Type II vertex includes Wi-Fi AP, user account, IP address and so on. Note that we take mobile device as Type I vertex because we want to predict the risk level of devices in spamdexing detection. In other scenario, any node can be considered as Type I vertex. Wi-Fi AP is identified by the concatenation of BSSID (Basic Service Set Identification) and SSID (Service Set Identifier). User account is identified by the account ID.

The device network, denoted as $G_L = (V, E, W, X, Y)$, is a partially labeled information network. $V = V_0 \cup V_1 \cup ... \cup V_{N-1}$ are N disjoint sets of vertices, where V_0 are type I vertices and others are type II vertices. $X \in R^{\chi}$ is the feature vertor of nodes' attributes, where χ is the dimension of feature, $Y \in R$ is the label set of each node. We will discuss how to extract node feature X in Section III.B. E is connections between vertexes, which can be directed or undirected, weighted or unweighted. In our approach, the weight of the edge between v_i and v_j is defined as the reciprocal of the days since the connection is established. We aim to learn the mapping function $f : X \rightarrow R^d, d << \chi$, which embeds information to low-dimensional feature representations.

In device network, type I vertex is connected to type II vertex, and there is no connection between different types of type II vertices. Hence, a reasonable solution is to partition the network into some homogeneous bipartite networks where there is only one type of edge, then sample the context of vertices alternatively from these subnets. More importantly, partitioning by type II vertex accelerates the convergence of optimization with high variance of edge's weights and makes the result interpretable in risk management scenario. Let the type I vertices be V_0, type II vertices be $V_1, ..., V_{N-1}$. Then we partition the network G_L into $N - 1$ bipartite graphs $G_1, ..., G_{N-1}, G_N = (V_0, V_n, E)$, where $n = 1, ..., N - 1$. The following node representation is based on these subnets.

3.2 FeatNet

3.2.1 Target of Optimization. The representation learning of one bipartite graph can be formulated to a maximum likelihood optimization problem. First, the source vertex is v_i. The conditional probability of generating context from vertex v_i to vertex v_j is defined by the dot product of their representations as:

$$p(v_j|v_i) = \frac{exp(\vec{u}_j^T \cdot \vec{u_i})}{\sum_{v_k \in V} exp(\vec{u}_j^T \cdot \vec{u_i})} \tag{1}$$

Where $\vec{u_i}$ is the representation vector of vertex v_i. Let the set of neighborhoods be N, $v_j \in N$ is a sampled neighbor of v_i. For each source vertex $v_i \in V$, equation (1) defines the conditional distribution $p(\cdot|v_i)$ over the vertices set. We aim to maximize the log-probability in terms of context structures. Since each step in sampling is independent, the objective of sampling a network neighborhood of v_i is to maximize:

$$\sum_{v_i \in V} log(\prod_{v_j \in N} p(v_j|v_i))$$

We replace $p(v_j|v_i)$ with equation (1). Let d is size of the neighborhoods N, the objective function conditioned on the feature representation can be calculated as:

$$\sum_{v_i \in V} [-d \cdot log(\sum_{v_k \in V} exp(\vec{u}_k^T) \cdot \vec{u_i}) + \sum_{v_j \in N} exp(\vec{u}_j^T \cdot \vec{u_i})] \tag{2}$$

By maximizing the objective function equation (2), we can learn the representation vector $\vec{u_i}$ of each vertex $\vec{v_i}$. The calculation of sum of each $v_k \in V$ is able to be optimized by negative sampling in large-scale network. However, the computation of equation (2) still meets with daunting challenges that will be discussed in the next section.

3.2.2 Sampling Strategy. Next, we introduce the neighborhood sampling strategy with a fixed window size d. The neighborhoods of a node are not restricted to just immediate neighbors but can have vastly different structures depending on the sampling strategy. Node2vec [6] proposed a biased neighborhood sampling strategy between BFS and DFS by introducing the return parameter p and in-out parameter q.

In the traversal, let the previous vertex be $v_i \in V_0$, the current vertex be $v_c \in V_n$, then the next sampled vertex v_j must be back in V_0, and no path length equals to 1 between $\vec{v_i}$ and $\vec{v_j}$ in the bipartite graph. In this case, the unnormalized transition probability of node c is set to $\pi_{cj} = \alpha(i, j) \cdot w_{cj}$, where

$$\alpha(i, j) = \begin{cases} \frac{1}{p}, if(i = j) \\ \frac{1}{q}, if(i \neq j) \end{cases}$$

For each vertex v_i in the bipartite graph, we generate random walk sequences by normalized π_{cj}, and produce fixed length neighborhood observations. After sampling, the skip-gram model [17] is used to update our representations in accordance with our objective function Equation (2). The method of learning distributed representation of words includes dimensionality reduction on the word co-occurrence matrix [13], skip-gram model, GloVe [18], etc. The work [16] defines the specific loss function to prove that skip-gram with negative sampling is an explicit matrix factorization of the word co-occurrence matrix. There is no advantage of using any approach over others and much of the performance gains are due to system design choices and hyper-parameter optimizations [14], so we evaluate the performance of FeatNet to specific task, which will be stated in section 4.

3.2.3 Embedding With Feature Signature. However, there are limitations of the aforementioned approach. First, embedding the vertex by a sequence of device ID ignores the domain knowledge contained in device features. Second, embedding device IDs can make the model overfit labeled devices. Third, the quantity of device ID is too large to train the skip-gram model. Next, we introduce feature signature to solve this problem.

Definition 2. Feature signature is a string that represents the characteristics of each device, which is constructed by expert domain knowledge and other labeled information such as records of fraud.

By integrating feature signature into the context of each vertex, we are able to utilize all features of devices as well as historical labels. For example, there are tags with security expert knowledge, such as antivirus_scan_security, host_repackage, debug_mode_on etc., along with the rich text features and numeric features. We preprocess these features to construct the feature signature, which is described as below:

(1) Text features are represented as list L_1.

(2) Categorical features, such as records of fraud, is individually labeled into disjoint subsets to text list, e.g., "risk: loan fraud" is to "RISK_1". Categorical features are represented as list L_2.
(3) Numeric feature is normalized and mapped to feature bucket, then marked with text tag, eg. "day_average_power: 0.65" is to "AVG_PWR_6". Numeric features are represented as list L_3.
(4) L_1, L_2, L_3 are concatenated into list L. An example of L is: ["BOARD_MSM8953", "RESOLUTION_1080*1920", "PRODUCT_OPPO R9S", "IS_ROOTED", "ACCOUNT_NUM_LARGE", "DEBUG_MODE_ON", "AVG_SIGNAL_STRENGTH_HIGH", "AVG_PWR_6", "RISK_1",...].
(5) Skip-gram model is used to convert L to a fixed length vector V_k.
(6) Finally, continuous values in V_k are mapped to buckets. Then the feature signature string is formed by concatenating numbers in each bucket..

Text feature is the most common feature in spamdexing detection, so other features are converted to text feature in feature the list L. We use skip-gram model to represent the sequence of V_s with a low dimensional vector. Once the vectors of each feature signature are learned, the representation vector of the device can be obtained by averaging all the vectors of its neighbors' feature signatures. Since the representations of devices leveraged by the fraudsters inclines to cluster apparently, the representations of device's neighbors reveal the potential risk of the device. Note that the dimension of V_k k is flexible to avoid over-fitting of the model. Besides, k affects the dictionary size of feature signature s, which is proportional to cost of computing equation (2). Hierarchical softmax is used to reduce the complexity of computing to $O(log(s))$.

3.2.4 Overall Framework. The overall learning framework is detailed in Algorithm 1. First, the device network is partitioned to $N-1$ bipartite subnets. Second, the neighbors of vertices as well as the feature signature of neighbors are sampled in all subnets. Third, the representation of each vertex is learned by the feature signature of neighbors. Let $\vec{u_{ni}}$ be the distributed representation of v_i in the subgraph $G_n, n = 1, ..., N-1$. $\vec{u_{ni}}$ will be the output to downstream prediction task as $N-1$ individual feature slot of v_i.

In the stage of neighbor sampling, the time complexity is $O(dn|V|)$, where d is sampling length of neighborhood; n is the number of walks; $|V|$ is the number of vertices in the network. In the stage of stochastic gradient descent, the training of skip-gram takes $O(|V|log(s))$ time, where s is the dictionary size of feature signature, $s << |V|$. Thus the overall time complexity of FeatNet is $O(|V|log(s))$.

Algorithm 1 FeatNet Algorithm

```
function FEATURELEARNING(($Graph \quad G = (V, E, W)$))
    for $G_i$ in partitionGraph($G$) do
        append bipartiteGraphWalk($G_i$) to $walks$
        $u_i$ = skip-gram($walks$)
        output $u_i$
    end for
end function

function PARTITIONGRAPH($G = (V, E, W)$,
                                nodeType = $N$)
    for $i$ from 1 to $N - 1$ do
        append ($V_0 \& V_i, E, W$) to $subgraph$
    end for
return $subgraph$
end function

function BIPARTITEGRAPHWALK($G_i = (V_i, E, W)$)
    for $v$ in $G_i$ do
        constructNodeSigature($v$)
    end for
    for $i$ from 1 to $walk\_num$ do
        for vertex in $G_i$ do
            for $d$ from 1 to $sampling\_length$ do
                $cur\_vertex = vertex\_neighbor[-1]$
                $v$ = sampleNextNeighbor($cur\_vertex$)
                append $v$ to $vertex\_neighbor$
                append $v.sig$ to $sig\_neighbor$
            end for
            append $sig\_neighbor$ to $sig\_walk$
        end for
    end for
return $sig\_walk$
end function
```

4 EVALUATION

4.1 Data Sets

We conduct experiments of spamdexing detection on a real-world datasets of Baidu mobile search to find devices possessed by fraudsters. By constructing relationships of device to Wi-Fi AP and user account, we can integrate device features with network struture. The entire dataset has 13,567,732 devices IDs, 31,427,140 BSSIDs and 136,006 user account IDs. The average degree of device vertex is 31.3.

4.2 Baseline Methods

We compare FeatNet with the following baseline algorithms. For comparisons, the dimension of distributed representation is set to 200 consistently.

(1) node feature skip-gram. The skip-gram model is used to embed the features of node, then training classifier only with node features, regardless of the structural information. The minimum word count is set to 5.
(2) matrix factorization. The factorization of vertex co-occurrence matrix is adopted to embed the network structure. We take the top 200 values as the low-dimensional representation vector of each vertex.
(3) LINE. LINE is a NRL method suitable for learning information networks. We adopt both the first order and second order proximity of LINE model. The number of negative samples

is set to 5. The dimensions of the first and the second order representations are both set to 100.

(4) node2vec. Node2vec defines a flexible notion of vertex's neighborhood and proposes a biased random walk sampling method based on Deepwalk. We apply node2vec (scala implementation with Spark) to device network to learn the structural informaiton between devices. The values for p and q are set to 1; number of walks is set to 20; walk length is set to 50.

(5) FeatNet-1. FeatNet-1 uses device ID and Wi-Fi AP relationships.

(6) FeatNet-2. FeatNet-2 uses device ID, Wi-Fi AP and user account relationships.

4.3 Performance

We evaluate the performance of these approaches in both classification and clustering tasks. In binary classification task, the results is compared by the standard evaluation metric AUC with random forest classifier on Spark. In clustering task with DBSCAN, the effectiveness is measured by F1 score, homogeneity score and completeness score.

4.3.1 Performance on Sample Dataset. To demonstrate the embedding result of FeatNet, we select device node with high degree as the sample dataset, which has 20000 edges of device ID/BSSID/user account ID relationships. There are 9330 device IDs, 5525 of which are spam; 3805 of which are not spam. There are 7339 BSSIDs and 138 user account IDs.

For binary classification task, we evaluate the performance of fraud detection on the sampled dataset with various percentage of training data with labels. As shown in Table 1, by incorporating device feature with network structural information, FeatNet-2 consistently achieves improvement among the baselines in different training ratios. The performance of methods that take only node or network information, such as node feature skip-gram and matrix factorization, is inferior to other baselines. While, due to a small quantity of device ID/user account relationships in this dataset, FeatNet-2 hardly makes contribution to AUC score. The average accuracy of FeatNet-1 is 0.824; and he average accuracy of FeatNet-2 is 0.851.

Table 1: AUC values on sampled dataset with sampled training ratios.

Algorithms	20%	40%	60%	80%
node feature skip-gram	0.755	0.766	0.772	0.771
matrix factorization	0.765	0.778	0.779	0.780
LINE	0.780	0.792	0.799	0.798
node2vec	0.790	0.808	0.808	0.811
FeatNet-1	0.926	0.927	0.927	0.930
FeatNet-2	0.928	0.928	0.929	0.931

We also conducted clustering tasks to assess the performance of the methods. In the clustering task, we randomly select a portion of vertices as training set, then repeat the clustering for 10 times and report the average Macro-F1 and Micro-F1 score. As shown in Table 2 and Table 3, FeatNet consistently and significantly improves the performance of network embedding on both evaluation tasks.

Table 2: Micro-F1 on sampled dataset with sampled training ratios.

Algorithms	20%	40%	60%	80%
node feature skip-gram	0.639	0.651	0.658	0.659
matrix factorization	0.539	0.569	0.615	0.618
LINE	0.666	0.667	0.667	0.667
node2vec	0.676	0.677	0.679	0.68
FeatNet-1	0.678	0.677	0.681	0.682
FeatNet-2	0.681	0.682	0.683	0.683

Table 3: Macro-F1 on sampled dataset with sampled training ratios.

Algorithms	20%	40%	60%	80%
node feature skip-gram	0.628	0.663	0.668	0.67
matrix factorization	0.518	0.57	0.614	0.617
LINE	0.737	0.74	0.742	0.745
node2vec	0.74	0.741	0.743	0.745
FeatNet-1	0.742	0.775	0.801	0.867
FeatNet-2	0.745	0.787	0.829	0.875

Table 4: Homogeneity Score and Completeness Score.

Algorithms	homogeneity	completeness
node feature skip-gram	0.006	0.008
matrix factorization	0.005	0.003
LINE	0.001	0.073
node2vec	0.012	0.055
FeatNet-2	0.193	0.042

Table 4 reports the homogeneity score and completeness score of the clustering results. The homogeneity score of FeatNet is much higher than others, which means that there are many clusters containing only data points that are members of a single class in the FeatNet representation. Figure 1 shows the visualization of the device network embedding. The results are mapped to the 2-D space using the t-SNE package with learned FeatNet embedding as input. By investigating suspicious clusters that have high homogeneity scores (such red plus signs in the lower left corner), we discovered hundreds of device groups sharing similar format of SSID and BSSID, which are likely controlled by fraudsters. These devices and user accounts associated with them are added to the blacklist to recude fraud risk.

4.3.2 Performance on Large-Scale Dataset. The large-scale dataset has 39,483,849 edges of device ID/BSSID/user account ID relationships. There are 13,567,732 device IDs, 1,260,835 of which are spam; 12,306,897 of which are not spam. There are 31,427,140 BSSIDs and 136,006 user account IDs. The dictionary size of feature signature is 362,452 when $k = 15$.

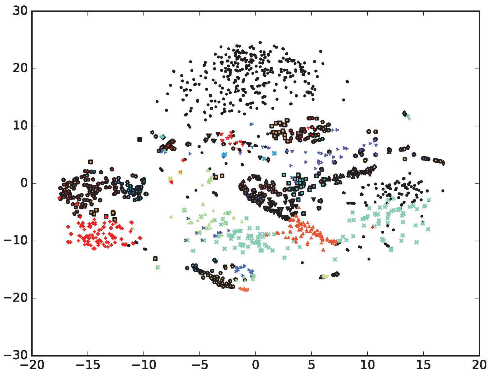

Figure 1: Visualization results on FeatNet.

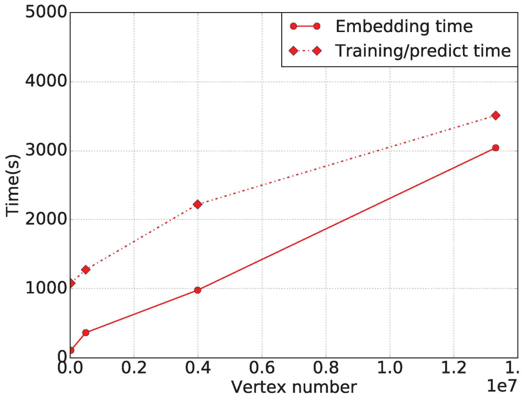

Figure 2: Scalability.

For binary classification task, Table 5 compares the performance on the large-scale dataset with various percentage of training data with labels. Node feature skip-gram, matrix factorization, node2vec and FeatNet run on a Spark cluster. For FeatNet, we use 200 executors, with 12g memory and 3 vCores for each executor; for each of the other methods, we use 200 executors, with 14g memory and 3 vCores for each executor. Matrix factorization fails in this evaluation for the magnitude of data. Node2vec takes device ID as node context, so it fails in the embedding stage as the dictionary size of device ID becomes too large. The open source project of LINE [24] is multiple threads stand-alone mode, and the experiment on CentOS server (128GB memory, Xeon E5-2620 v3 2.40GHz) fails with out of memory error on the large-scale dataset.FeatNet-2 achieves stable improvement in various training ratios than other baselines.

Table 5: AUC values on large-scale dataset with sampled training ratios.

Algorithms	20%	40%	60%	80%
node feature skip-gram	0.698	0.700	0.699	0.705
FeatNet-1	0.874	0.878	0.880	0.881
FeatNet-2	0.875	0.880	0.883	0.885

The average accuracy of FeatNet-1 is 0.809; The average accuracy of FeatNet-2 is 0.811.

4.3.3 Scalability. The scalability of FeatNet-2 is tested on a spark cluster. The number of devices from 9,224 to 13,567,732, which is randomly sampled from the whole dataset. Spark configurations are as follows: num-executors=200, executor-cores=3, executor-memory=8g.

Some optimization from previous work [15] makes the sampling procedure efficient. As shown in Figure 2 FeatNet-2 scales nicely with learning time increasing linearly, and completes learning of the whole dataset in 59 minutes.

4.3.4 Parameter Sensitivity. We employ grid search to select the best hyper-parameters of FeatNet. There are 5 hyper parameters: dimension of feature signature k, return parameter p, in-out parameter q,truncated sampling length d and number of walks n. The training ratio is fixed to 80%, and the parameters not tested are set to default values: $k = 15, p = q = 10, d = 50, n = 20$.

Figure 3 shows that k improves performance at the cost of increased computation time, while the performance drops when the dimension becomes too large. When the ratio of return parameter p to in-out parameter q is near 1, the performance of FeatNet improves, which means the fraud devices cluster more efficiently (Low p/q leads the walk to backtrack a step, and high p/q encourages outward exploration in network [6]). We also examine that the performance improves with the enhancement of d, because the length of context increases. When d becomes large enough, the performance of FeatNet converges. Similarly, n improves performance with an upper limit, at the cost of proportional time consumption. In conclusion, the performance of FeatNet is stable when theses hyper parameters vary within a reasonable range.

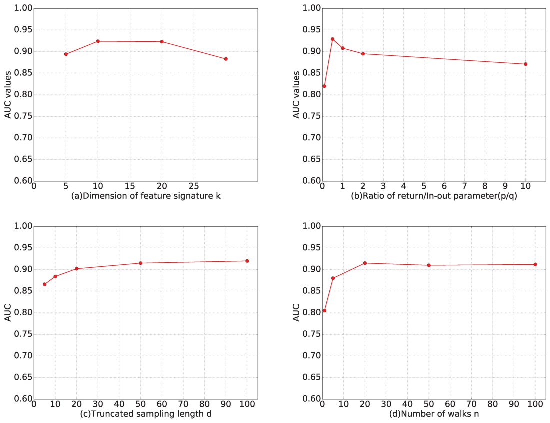

Figure 3: Parameter Sensitivity.

5 DISCUSSION AND CONCLUSION

In this paper, we have proposed FeatNet, a network representation learning framework for heterogeneous information networks with explicit domain features. The sampling strategy that FeatNet applies in homogeneous bipartite graphs utilizes overall structural information of device network. FeatNet also introduces feature signature which adapts detailed device information in the embedding procedure. In addition, FeatNet is easy to extend to other device-device relationships. Finally, the whole FeatNet framework is implemented on Spark, which brings excellent scalability.

In conventional anti-spam solutions, there is agreement on the necessity of moderate data collection to fight against underground economy. Our dataset was collected by the SDK bundle of Baidu mobile search app with privacy notice provided. To anonymize the data, device fingerprints and AP mac addresses are hashed and personally identifiable fields are deleted. Adversarial evasion to data collection is possible, e.g., change Wi-Fi AP name, but some features may reflect such evasion attempt, e.g., the device may be rooted in order to manipulate certain fields.

For future work, further studies are needed to improve FeatNet, such as to strengthen the theoretical justifications of the construction of feature signature. We also would like to investigate the applicability of our algorithm in other representation learning tasks. Another intriguing direction is to apply FeatNet as a representational layer in the deep learning architecture.

ACKNOWLEDGMENTS

We thank Guangzhu Wu for discussions and suggestions, and we also acknowledge the anonymous reviewers for their helpful comments.

REFERENCES

[1] Amr Ahmed, Nino Shervashidze, Shravan Narayanamurthy, Vanja Josifovski, and Alexander J Smola. 2013. Distributed large-scale natural graph factorization. In *Proceedings of the 22nd international conference on World Wide Web*. ACM, 37–48.

[2] Hristo Bojinov, Yan Michalevsky, Gabi Nakibly, and Dan Boneh. 2014. Mobile device identification via sensor fingerprinting. *arXiv preprint arXiv:1408.1416* (2014).

[3] Stephen P Borgatti. 2005. Centrality and network flow. *Social networks* 27, 1 (2005), 55–71.

[4] Shaosheng Cao, Wei Lu, and Qiongkai Xu. 2015. Grarep: Learning graph representations with global structural information. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. ACM, 891–900.

[5] Yuxiao Dong, Nitesh V Chawla, and Ananthram Swami. 2017. metapath2vec: Scalable representation learning for heterogeneous networks. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 135–144.

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

[7] William L. Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. In *NIPS*.

[8] Aniko Hannak, Gary Soeller, David Lazer, Alan Mislove, and Christo Wilson. 2014. Measuring price discrimination and steering on e-commerce web sites. In *Proceedings of the 2014 conference on internet measurement conference*. ACM, 305–318.

[9] Keith Henderson, Brian Gallagher, Lei Li, Leman Akoglu, Tina Eliassi-Rad, Hanghang Tong, and Christos Faloutsos. 2011. It's who you know: graph mining using recursive structural features. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 663–671.

[10] Zhipeng Huang and Nikos Mamoulis. 2017. Heterogeneous Information Network Embedding for Meta Path based Proximity. *arXiv preprint arXiv:1701.05291* (2017).

[11] Yann Jacob, Ludovic Denoyer, and Patrick Gallinari. 2014. Learning latent representations of nodes for classifying in heterogeneous social networks. In *Proceedings of the 7th ACM international conference on Web search and data mining*. ACM, 373–382.

[12] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

[13] Rémi Lebret and Ronan Collobert. 2013. Word emdeddings through hellinger PCA. *arXiv preprint arXiv:1312.5542* (2013).

[14] Omer Levy, Yoav Goldberg, and Ido Dagan. 2015. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics* 3 (2015), 211–225.

[15] Aaron Q Li, Amr Ahmed, Sujith Ravi, and Alexander J Smola. 2014. Reducing the sampling complexity of topic models. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 891–900.

[16] Yitan Li, Linli Xu, Fei Tian, Liang Jiang, Xiaowei Zhong, and Enhong Chen. 2015. Word Embedding Revisited: A New Representation Learning and Explicit Matrix Factorization Perspective.. In *IJCAI*. 3650–3656.

[17] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).

[18] Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 1532–1543.

[19] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 701–710.

[20] Usha Nandini Raghavan, Réka Albert, and Soundar Kumara. 2007. Near linear time algorithm to detect community structures in large-scale networks. *Physical review E* 76, 3 (2007), 036106.

[21] Jan Spooren, Davy Preuveneers, and Wouter Joosen. 2015. Mobile device fingerprinting considered harmful for risk-based authentication. In *Proceedings of the Eighth European Workshop on System Security*. ACM, 6.

[22] Yizhou Sun, Rick Barber, Manish Gupta, Charu C Aggarwal, and Jiawei Han. 2011. Co-author relationship prediction in heterogeneous bibliographic networks. In *Advances in Social Networks Analysis and Mining (ASONAM), 2011 International Conference on*. IEEE, 121–128.

[23] Jian Tang, Meng Qu, and Qiaozhu Mei. 2015. Pte: Predictive text embedding through large-scale heterogeneous text networks. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1165–1174.

[24] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. Line: Large-scale information network embedding. In *Proceedings of the 24th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 1067–1077.

[25] Cunchao Tu, Han Liu, Zhiyuan Liu, and Maosong Sun. 2017. Cane: Context-aware network embedding for relation modeling. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Vol. 1. 1722–1731.

[26] Daixin Wang, Peng Cui, and Wenwu Zhu. 2016. Structural Deep Network Embedding. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 1225–1234.

[27] Cheng Yang, Zhiyuan Liu, Deli Zhao, Maosong Sun, and Edward Y Chang. 2015. Network Representation Learning with Rich Text Information.. In *IJCAI*. 2111–2117.

[28] Cheng Yang, Maosong Sun, Zhiyuan Liu, and Cunchao Tu. 2017. Fast network embedding enhancement via high order proximity approximation. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI*. 19–25.