NeuralWalk: Trust Assessment in Online Social Networks with Neural Networks

Guangchi Liu
Stratifyd, Inc., Charlotte, NC
Email: luke.liu@stratifyd.com

Chenyu Li
Stratifyd, Inc., Charlotte, NC
Email: chenyu.li@stratifyd.com

Qing Yang
University of North Texas, Denton, TX
Email: qing.yang@unt.edu

***Abstract*—Assessing the trust between users in a trust social network (TSN) is a critical issue in many applications, e.g., film recommendation, spam detection, and online lending. Despite of various trust assessment methods, a challenge remaining to existing solutions is how to accurately determine the factors that affect trust propagation and trust fusion within a TSN. To address this challenge, we propose the NeuralWalk algorithm to cope with trust factor estimation and trust relation prediction problems simultaneously. NeuralWalk employs a neural network, named WalkNet, to model single-hop trust propagation and fusion in a TSN. By treating original trust relations in a TSN as labeled samples, WalkNet is able to learn the parameters that will be used for trust computation/assessment. Unlike traditional solutions, WalkNet is able to accurately predict unknown trust relations in an inductive manner. Based on WalkNet, NeuralWalk iteratively assesses the unknown multi-hop trust relations among users via the obtained single-hop trust computation rules. Experiments on two real-world TSN datasets indicate that NeuralWalk significantly outperforms the state-of-the-art solutions.**

Keywords*—*Trust Assessment, Social Network, Neural Network, Machine Learning

I. Introduction

A trust social network (TSN) can be regarded as a graph where the nodes are users and the edges are the trust relations among users. Assessing the trust between two users, commonly referred to as trustor and trustee, is a fundamental research problem in a TSN. Trust assessment in a TSN is essential in many applications, including online lending[1], malicious website identification [1], social network analysis [2], [3], vehicular network [4] and active friending [5]. Research work on trust assessment in a TSN can be roughly categorized into two groups: trust modelling and trust inference. Different theoretical frameworks, including Beta distribution [6], Dempster-Shafer (DS) theory of evidence [7] and subjective logic (SL) [8] have been adopted to design various trust models. The major limitation of these studies is that they do not explicitly consider the multi-hop trust inference problem in a TSN while designing trust models. On the other hand, most of the existing trust inference algorithms facilitating multi-hop trust inference in TSNs, e.g., TidalTrust [9], TrustRank [10] and MoleTrust [11], commonly work with simplified trust models, e.g., using a binary number to indicate whether a user is trustworthy.

Although some recent works, e.g., AssessTrust [12] and OpinionWalk [13], make an attempt to fill the gap by taking a joint consideration of both trust modelling and trust inference in TSNs, they have difficulty in connecting sophisticated trust models to real-world datasets. For example, trust ratings are commonly used by a user to quantify the trust of another; however, AssessTrust and OpinionWalk assume that trust is a three-valued opinion rather than a scalar number. As such, trust ratings were transformed into trust opinions by heuristic methods, which caused errors in trust assessment.

A. Proposed Solution

Unlike traditional solutions to the multi-hop trust assessment problem, we propose to "learn" the entire trust computation process, instead of mathematically defining it. The proposed solution is called NeuralWalk that employs a neural network architecture, WalkNet, to capture trust propagation and fusion in a TSN. Previous work has shown that a trust opinion indeed propagates from one user to another, and several trust opinions (of a trustee) can be fused to derive a new trust opinion. As such, the discounting operator and combining operator were proposed to model these two operations [8]. Existing literature [14], [12], [13] tried to understand these operations, however, the underlying mechanisms for trust propagation and fusion are still not clear. In WalkNet, the operations are implemented by the connections of neurons in a neural network. The parameters of these two operations are learned from training datasets.

In addition to the lacked understanding of trust opinion operations, a practical issue faces multi-hop trust assessment is that the trust among users, in most existing datasets, is commonly represented as a rating, rather than an opinion. A trust rating is either an ordinal or categorical value, indicating different trust levels; therefore, it must be first converted into an opinion for accurate trust assessment. To address this issue, an opinion is learned from a trust rating from a layer of neural network in the WalkNet. After trust computation, the resulting opinions will then be transformed into scalar numbers to match the original trust ratings in the datasets. This is achieved by a softmax function that transforms an opinion to a multinomial distribution, from which the most likely trust rating can be obtained. In the end, we use the cross entropy function as the loss function to facilitate the back propagation algorithm that is essential for the training process.

With WalkNet enabling single-hop trust assessment, we further propose an algorithm called NeuralWalk to support multi-hop trust assessment. NeuralWalk is an iterative algorithm and there are two steps in each iteration: training and inference. During the training step, the WalkNet is trained by minimizing the cross entropy between predicted trust ratings and labels (ground truths). After training, the parameters for

[1]http://blog.lendingclub.com/facebook-and-lending-club-looks-like-its-working/

transforming a trust rating to an opinion, as well as those for discounting and combining operators, can be learned. In the inference step, NeuralWalk iteratively employs the trained WalkNet to perform single-hop trust inference upon unknown trust relations. Based on single-hop trust inference results in each iteration, the algorithm will "walk" through the entire TSN, in a breadth first search (BFS) manner, to conduct multi-hop trust assessment.

B. Technical Challenges

NeuralWalk addresses three technical challenges that were not adequately researched in existing trust assessment solutions. First of all, to the best of our knowledge, most existing multi-hop trust assessment methods, accounting for both trust propagation and fusion, are deductive. That means the trust operations in these models and algorithms must obey rules or logic derived from assumptions based on cognitive recognition. These rules and/or assumptions, however, have been criticized for either being unrealistic or unable to offer flexible trust assessments in different systems. For example, the statement that "the enemy of my enemy is my friend" could be true or false under different circumstances [15]. As a result, one has to customize the rules and assumptions used in trust assessment and fine-tune the corresponding trust models empirically; otherwise, the deductive solutions will barely capture the nature of trust propagation and fusion. To address this challenge, NeuralWalk models trust propagation and fusion among users in an inductive manner. In the WalkNet, the rules for trust propagation and fusion are learned from training samples, obtained from real-world datasets. In this way, NeuralWalk is automatically adapted to the most numerically-fitted model, eliminating any manual efforts.

Secondly, some previous multi-hop trust assessment models [12], [13], [16] use evidence, e.g., positive, negative, and uncertain ones, as the input to accurately quantify a trust relation. However, the information is often unavailable in almost all existing datasets. Moreover, in many scenarios, the information will not be allowed to collect, e.g., due to privacy concerns. As a matter of fact, only ordered or categorical trust ratings are provided in most trust social networks. To the extent of our knowledge, there is very limited research on how to effectively convert from a ordered or categorical trust rating to a trust opinion, or vice versa. To break this limitation, NeuralWalk incorporates the transformation into WalkNet, i.e., ordered or categorical trust ratings are converted to trust opinions through a layer of neural network. The parameters of the layer are jointly learned as those for opinion operators are learned.

In the end, trust propagation and fusion in a TSN can hardly be addressed by an end-to-end neural network. This is because the network topologies between users in a TSN are inconsistent and complex, especially when they are multiple hops away from each other. As a result, it is difficult, if not impossible, for an end-to-end neural network, whose architecture is consistent, to precisely capture trust propagation and trust fusion. To address this challenge, WalkNet is designed to handle single-hop trust propagation and fusion only, upon which the architecture of WalkNet is consistent. Based on the learned WalkNet, NeuralWalk algorithm iteratively searches a TSN, in a BFS manner, to realize multi-hop trust assessment.

C. Contributions

The contributions of this paper lie in the following four aspects. First, we propose for the first time a unique neural network based solution for multi-hop trust assessment in a trust social network. Second, existing trust models are too sophisticated to be validated by existing TSN datasets, however, the long-known problem is addressed by the proposed WalkNet that provides a mechanism to freely map between trust ratings and trust opinions. Third, experiment results demonstrate that not only the WalkNet is a valid model for trust computation but also the NeuralWalk algorithm is able to offer accurate trust assessment, i.e., achieving state-of-the-art performance, compared to previous methods. In the end, the WalkNet model in NeuralWalk can be substituted with any other machine learning models, providing opportunities for more accurate trust assessment. This feature also makes NueralWalk a fundamental framework that can be easily adapted to other networking applications.

II. Preliminaries

A. Problem Formulation

A trust social network (TSN) is modeled as a directed graph $G(V, E, W)$ where vertex $i \in V$ represents a user, and edge $e_{ij} \in E$ denotes that user i has a trust relation to j. The weight $w_{ij} \in W$ on edge e_{ij} indicates how user i trusts j. It is modeled as either a real number [9] or a feature vector [8], [12], [13], resulting in different accuracies in trust assessments. The trust assessment problem can be formulated as follows. *Given a trust social network $G(V, E, W)$, $\forall$ i and j, s.t. $i, j \in V$, $\exists$ at least one path from i to j, how to compute i's trust in j where $\{j \in V, j \neq i\}$.*

B. Trust Opinion

To enable accurate trust assessment, vector-based trust opinions [8], [12], [13], [16] are recently adopted to model the trust propagation and fusion among users in a TSN. In these models, an *original trust* relation is considered a probabilistic distribution over three different states, i.e., belief, distrust, and uncertainty. The probability of a user being trustworthy, untrustworthy, or uncertain is determined by the observed evidence for each corresponding state. As such, the trust relation between users i and j can be represented as an opinion $\omega_{ij} = \langle \alpha_{ij}, \beta_{ij}, \gamma_{ij} \rangle$ [16], where α_{ij}, β_{ij}, and γ_{ij} denote the amounts of positive, negative, and uncertain evidence, respectively.

C. Opinion Operators

Besides original trust relations, the prediction of *potential trust* relations in TSNs has attracted many attentions in recent years. Potential trust is used to express the unknown yet possible trust relation between two users who do not have explicit trust relation. Given three users i, s and j, and trust opinions ω_{is} and ω_{sj}, it is possible to predict i's potential trust in j, based on s's recommendation of j. Potential trust can be computed based on original trust opinions via opinion operations. The trust propagation and trust fusion are two basic operations used to conduct trust computation in TSNs [8], [12], [13], [16]. Two opinion operators, namely discounting and combining operators, are proposed in literature [8], [12], [13], [16] to facilitate modeling trust propagation and trust fusion in TSNs. Given three users i, s and j, and two existing opinions

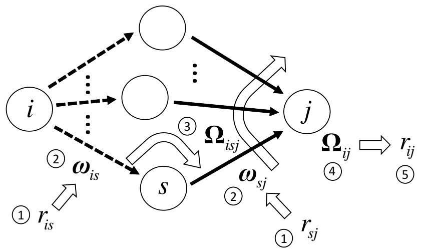


Fig. 1: Illustration of trust assessment between two users.

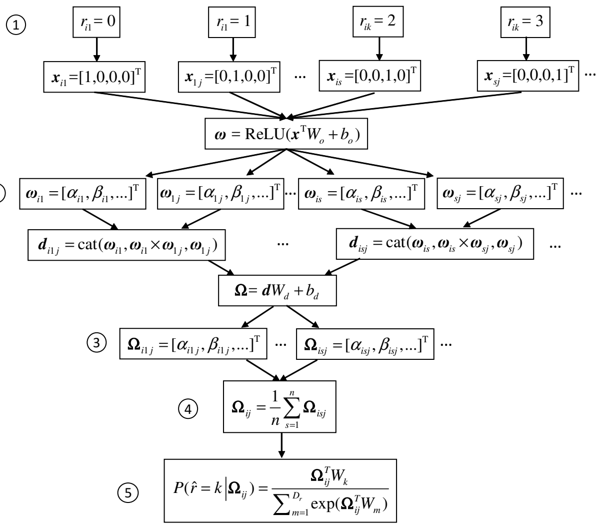

① $r_{i1}=0$ $r_{i1}=1$ $r_{ik}=2$ $r_{ik}=3$
$x_{i1}=[1,0,0,0]^T$ $x_{1j}=[0,1,0,0]^T$ ... $x_{is}=[0,0,1,0]^T$ $x_{sj}=[0,0,0,1]^T$ ...
$\omega=\mathrm{ReLU}(x^TW_o+b_o)$
② $\omega_{i1}=[\alpha_{i1},\beta_{i1},...]^T$ $\omega_{1j}=[\alpha_{1j},\beta_{1j},...]^T$ ... $\omega_{is}=[\alpha_{is},\beta_{is},...]^T$ $\omega_{sj}=[\alpha_{sj},\beta_{sj},...]^T$ ...
$d_{i1j}=\mathrm{cat}(\omega_{i1},\omega_{i1}\times\omega_{1j},\omega_{1j})$ ... $d_{isj}=\mathrm{cat}(\omega_{is},\omega_{is}\times\omega_{sj},\omega_{sj})$ ...
$\Omega=dW_d+b_d$
③ $\Omega_{i1j}=[\alpha_{i1j},\beta_{i1j},...]^T$ ... $\Omega_{isj}=[\alpha_{isj},\beta_{isj},...]^T$ ...
④ $\Omega_{ij}=\frac{1}{n}\sum_{s=1}^{n}\Omega_{isj}$
⑤ $P(\hat{r}=k|\Omega_{ij})=\frac{\Omega_{ij}^TW_k}{\sum_{m=1}^{D_r}\exp(\Omega_{ij}^TW_m)}$

Fig. 2: Illustration of the architecture of WalkNet.

ω_{is} and ω_{sj}, the *discounting operator* $\Delta(\omega_{is},\omega_{sj})$ yields an opinion Ω_{ij}, expressing i's potential trust in j, based on s's recommendation of j. To distinguish from original opinion, Ω_{ij} is used to denote i's potential opinion on j. Let Ω^1_{ij} and Ω^2_{ij} be two potential opinions, the *combining operator* $\Theta(\Omega^1_{ij},\Omega^2_{ij})$ produces a new opinion Ω_{ij} that fuses Ω^1_{ij} and Ω^2_{ij}.

III. WalkNet

Although there are several works on modeling trust propagation and trust fusion [14], [17], [18], [12], [13], to the best of our knowledge, most existing methods accounting for these two operations are deductive. That means these trust models obey the laws of logic derived from assumptions, based on cognitive recognition. These assumptions, however, have been criticized for being unrealistic in practice [15]. What's worse, all parameters in the deductive trust models have to be determined empirically, which seriously impacts the accuracy of trust assessments. Therefore, it is worth exploring new ways for modeling trust propagation and fusion in an inductive manner.

A. Overview of WalkNet

Instead of rigidly defining the trust operators, we propose to "learn" the operators with the *WalkNet*, a neural network. For the sake of simplicity, we use an example to illustrate how the WalkNet works. As shown in Fig. 1, the process of user i assessing the trust of j can be decomposed into five steps. The first step is to retrieve the original trust ratings (e.g., i's trust in s) from existing datasets, and convert these ratings into trust opinions, as shown in step 2. In most existing TSN datasets, trust between users is represented by an ordered or categorical scalar, which indicates the i's trust rating of s. The scalar representation of trust hinders the application of trust opinion, resulting inaccurate trust assessments. To address this issue, a transformation from scalar trust ratings to vector based trust opinions becomes essential, which will be discussed in Section III-B. Based on the trust opinions (e.g., ω_{is}), discounting operations will be carried out in step 3 to obtain the potential opinions (e.g., Ω_{ij}), which will be elaborated in Section III-C. In step 4, i's all potential opinions on j will be combined to yield a new opinion Ω_{ij}, finishing the trust assessment process. Finally, in step 5, a softmax-based transformation is applied to convert obtained trust opinion (e.g., Ω_{ij}) into a trust rating. In the following, we will describe the details of each step, based on the illustration in Fig. 2.

B. Rating to Opinion

Existing research has shown that a vector-based trust opinion is able to accurately model trust and thus ensures precise trust assessment [12], [13], [16]. On the other hand, when datasets for trust social networks were collected, scalar numbers were commonly used to represent the trust relations among users. To close this gap, it is worth studying the mechanism that effectively transforms ratings to opinions and vice versa.

For two users i and j, let's assume i's trust in j is denoted as a scalar number r_{ij}. Then, we can use an one-hot vector $\boldsymbol{x}_{ij} \in \mathbb{R}^{D_l}$ to represent the trust rating. For instance, in the Advogato dataset [19], the trust relations between users are quantified by four ratings: apprentice, observer, journeyer, and master. In this example, as the trust rating falls into four possible levels, the four corresponding one-hot vectors can be represented as

$$\begin{cases} [1,0,0,0]^{\mathrm{T}} \\ [0,1,0,0]^{\mathrm{T}} \\ [0,0,1,0]^{\mathrm{T}} \\ [0,0,0,1]^{\mathrm{T}} \end{cases}.$$

Obviously, the dimension of a one-hot vector D_l is determined by the number of possible trust levels in the dataset.

For an one-hot vector $\boldsymbol{x}_{ij}$, WalkNet employs a non-linear transformation to convert it into an opinion[2] $\boldsymbol{\omega}_{ij} \in \mathbb{R}^{D_o}$. The non-linear transformation is expressed as

$$\boldsymbol{\omega}_{ij} = \mathrm{ReLU}(\boldsymbol{x}_{ij}^{\mathrm{T}} W_o + \boldsymbol{b}_o),$$

where $W_o \in \mathbb{R}^{D_l \times D_o}$ and $\boldsymbol{b}_o \in \mathbb{R}^{D_o}$. The ReLU here refers to rectified linear unit that is a widely used activation function in neural networks. The ReLU function is defined as $\mathrm{ReLU}(x) = \max(0,x)$. According to [20], [21], ReLU is a preferred activation function in feedforward neural network, due to its efficacy in computation and convergence. Because we assume a trust opinion (see Section II-B) is built upon three types of evidence, i.e., positive, negative and uncertain, D_o is set to be 3 here. The value of D_o, however, can be any positive integer greater than 3 to account for trust models considering more than three features.

C. Discounting Operation

Based on [12], [13], the discounting operation of two opinions, e.g., ω_{is} and ω_{sj}, can be expressed as $\Omega_{ij} = \Delta(\omega_{is},\omega_{sj})$.

[2]Since vectors are usually denoted as bold symbols in a neural networks, we use a bold Greek letter to denote an opinion.

A discounting operator Δ in fact recombines the three components in ω_{is} and ω_{sj} to generate Ω_{ij}. That also implies that the interactions of components $\alpha_{is}, \beta_{is}, \gamma_{is}$, $\alpha_{sj}, \beta_{sj}, \gamma_{sj}$ determine the resulting opinion Ω_{ij}. It is unfortunately not clear that how these interactions are carried out when trust propagates within a TSN.

To this end, we make use of a neural network to address this problem. We use pairwise multiplications of components in ω_{is} and ω_{sj} to represent the interactions among them. The pairwise multiplications will result in a set of parameters,

$$\begin{aligned}\boldsymbol{\omega}_{ik}^{\mathrm{T}} \times \boldsymbol{\omega}_{kj}^{\mathrm{T}} \quad = \quad & [\alpha_{is}\alpha_{sj}, \alpha_{is}\beta_{sj}, \alpha_{is}\gamma_{sj}, \\ & \beta_{is}\alpha_{sj}, \beta_{is}\beta_{sj}, \beta_{is}\gamma_{sj}, \\ & \gamma_{is}\alpha_{sj}, \gamma_{is}\beta_{sj}, \gamma_{is}\gamma_{sj}],\end{aligned}$$

which can be used as the building blocks for reconstructing the discounting operation. Together with components $\alpha_{is}, \beta_{is}, \gamma_{is}$, α_{sj}, β_{sj}, and γ_{sj}, we can obtain $\boldsymbol{d}_{ij} = \mathrm{cat}(\boldsymbol{\omega}_{is}^{\mathrm{T}}, \boldsymbol{\omega}_{is}^{\mathrm{T}} \times \boldsymbol{\omega}_{sj}^{\mathrm{T}}, \boldsymbol{\omega}_{sj}^{\mathrm{T}})$, where $\mathrm{cat}(\cdot)$ means concatenating. Note that $\boldsymbol{d}_{ij}$ serves as the determining factor for Ω_{ij} as follows. With another linear transformation, $\boldsymbol{d}_{ij}$ can be converted into the form of the resulting opinion

$$\boldsymbol{\Omega_{ij}}{=}\boldsymbol{d}_{ij}W_d + \boldsymbol{b}_d,$$

where $W_d \in \mathbb{R}^{(D_o^2+2D_o)\times D_o}$ and $\boldsymbol{b}_d \in D_o$. As $D_o = 3$ in this example, we have $\boldsymbol{\omega}_{is}^{\mathrm{T}} \times \boldsymbol{\omega}_{sj}^{\mathrm{T}} \in \mathbb{R}^9$ and $W_d \in \mathbb{R}^{15\times 3}$. We further add a bias vector $\boldsymbol{b}_d$ to account for the bias in the discounting operation.

D. Combining Operation

Combining operation is used to fuse multiple trust opinions into a consensus one. Let $\Omega_{i1j} = \langle \alpha_{i1j}, \beta_{i1j}, \gamma_{i1j} \rangle$ and $\Omega_{i2j} = \langle \alpha_{i2j}, \beta_{i2j}, \gamma_{i2j} \rangle$ be i's two opinions of j's trust. Then, the combining operation $\Theta(\Omega_{i1j}, \Omega_{i2j})$ [16] yields a new opinion $\langle \alpha_{ij}, \beta_{ij}, \gamma_{ij} \rangle$ where

$$\begin{cases} \alpha_{ij} = \alpha_{i1j} + \alpha_{i2j} \\ \beta_{ij} = \beta_{i1j} + \beta_{i2j} \\ \gamma_{ij} = \gamma_{i1j} + \gamma_{i2j} \end{cases}.$$

When there are n opinions to be combined, it can be expressed as $\Theta(\Omega_{i1j}, \Omega_{i2j}, \cdots, \Omega_{inj})$, due to the associative property of the combining operator. Therefore, the combining operation can be extended as

$$\Theta(\Omega_{i1j}, \Omega_{i2j}, \cdots \Omega_{inj}) = \langle \alpha_{ij}, \beta_{ij}, \gamma_{ij} \rangle, \tag{1}$$

where
$$\begin{cases} \alpha_{ij} = \alpha_{i1j} + \alpha_{i2j} + \cdots + \alpha_{inj} \\ \beta_{ij} = \beta_{i1j} + \beta_{i2j} + \cdots + \beta_{inj} \\ \gamma_{ij} = \gamma_{i1j} + \gamma_{i2j} + \cdots + \gamma_{inj} \end{cases}.$$

Based on the above analysis, WalkNet employs a matrix multiplication to implement the combining operator. We represent the to-be-combined opinions in a matrix as $M_\Omega = [\boldsymbol{\Omega}_{i1j}, \boldsymbol{\Omega}_{i2j}, \cdots, \boldsymbol{\Omega}_{inj}]$, where $\boldsymbol{\Omega}_{isj} \in \mathbb{R}^{D_0}$ for every possible $s = 1, 2, \cdots, n$. Note that $\boldsymbol{\Omega}_{isj} = [\alpha_{isj}, \beta_{isj}, \gamma_{isj}]^{\mathrm{T}}$ when $D_o = 3$. Therefore, $M_\Omega \in \mathbb{R}^{D_0} \times n$. Let $\boldsymbol{I} \in n \times 1$ be a vector with all ones $[1, 1, \cdots 1]^T$, then Eq. 1 can be rewritten as

$$\Theta(\Omega_{i1j}, \Omega_{i2j}, \cdots \Omega_{inj}) = M_\Omega \boldsymbol{I}. \tag{2}$$

In addition, to normalize the resulting opinion, we have $\Theta(\Omega_{i1j}, \Omega_{i2j}, \cdots \Omega_{inj}) = \dfrac{1}{n} M_\Omega \mathbf{I}$, in the WalkNet.

E. Opinion to Rating

So far, WalkNet is able to represent the discounting and combining operations. In the last step, WalkNet transforms the computed opinions into scalar trust values, for training and validation purposes. The transformation is realized via a softmax function, which is a generalization of the logistic function. It squashes a multi-dimensional vector to a probability distribution over certain scalar numbers (trust ratings). Given an opinion Ω_{ij}, the probability that it is corresponding to the rating k can be denoted as $P(\hat{r}_{ij} = k \,|\boldsymbol{\Omega}_{ij})$ where $\hat{r}_{ij}$ is the inferred trust rating. By exploiting a softmax function, it becomes

$$P(\hat{r}_{ij} = k \,|\boldsymbol{\Omega}_{ij}) = \frac{e^{\boldsymbol{\Omega}_{ij}^T \mathbf{W}_k}}{\sum_{m=1}^{D_r} e^{\boldsymbol{\Omega}_{ij}^T \mathbf{W}_m}}. \tag{3}$$

The output of this step will be a multinomial probability distribution over all possible ratings. The rating with the highest probability is considered i's inferred trust in j.

F. Loss Function

The main reason of introducing WalkNet is to leverage original trust relations in a TSN to train the parameters for opinion operations, and then use the trained WalkNet to predict potential trust relations. To facilitate the training process, we use the cross entropy as the loss function. The function is expressed as

$$\mathcal{L} = -\sum_{e_{ij}\in E} \sum_{k=1}^{D_r} \mathrm{I}[r_{ij} = k] \log P(\hat{r}_{ij} = k), \tag{4}$$

where the Iverson bracket $\mathrm{I}[r_{ij} = k]$ evaluates to 1 if $r_{ij} = k$, and 0 otherwise. It is clear that Eq. 4 is minimized when all known trust ratings r_{ij} are correctly predicted by the WalkNet. By applying the back-propagation algorithm [22] on WalkNet, the undetermined matrices W_o, $\boldsymbol{b}_o$, W_d, $\boldsymbol{b}_d$ and $W_m, m = 1, 2, \cdots, D_r$ in Eq. 3 can be estimated such that Eq. 4 is minimized.

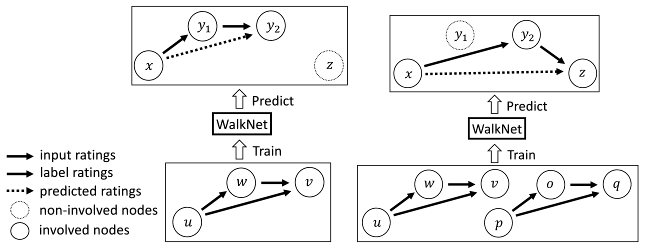
IV. NeuralWalk

With WalkNet and a given TSN, the NeuralWalk algorithm "Walks" through the TSN and computes trust with WalkNet alternately. In this section, we present the details NeuralWalk algorithm, in two folds. Firstly, we explain how the algorithm "walks" through a TSN and compute trustees' trust. Secondly, we elaborate the algorithm with details of how to compute trust via the WalkNet, rather than the original opinion operations.

A. The "Walking" Process

The "Walking" process upon a TSN, denoted as G, is accomplished in K iterations, where K is the maximum iteration number. Intuitively, it is the depth NeuralWalk "Walks" through the TSN. It can be any integer greater then 1. We use the example shown in Fig. 1 to illustrate how the algorithm "walks" in G. For convenience, we use $N_{in}(u)$ and $N_{out}(u)$ to denote the in-neighbors and out-neighbors of a node $u \in G$, respectively. In-neighbors are the nodes that connect to u, while out-neighbors are those u connects to. In addition, since G will be updated during different iterations, we use $G^{(k)}$ to denote the TSN in the kth iteration. Notably, we use $G^{(0)}$ to denote the original TSN in the beginning.

In the kth iteration, for each node (trustor) $i \in G$, the algorithm starts from i and conducts a BFS-style search to



(a) kth Iteration (b) $k+1$th Iteration

Fig. 3: An illustration of how NeuralWalk 'Walks" through a TSN $G^{(k)}$ and computes potential trust relations.(a) Based on the training set (e.g.,$r_{uw}^{(k)}$, $r_{wv}^{(k)}$, $r_{uv}^{(0)}$) found in $G^{(k)}$, a WalkNet is trained based on $\mathcal{T}^{(k)}$ and used to predict $r_{xy_2}^{(k+1)}$ based on $r_{xy_1}^{(k)}$ and $r_{y_1y_2}^{(k)}$. Note that $r_{xz}^{(k+1)}$ cannot be computed at the current iteration. (b) In $k+1$th iteration, NeuralWalk is able to collect more training samples for $\mathcal{T}^{(k+1)}$ and learn a new WalkNet. Based on the previously predicted rating $r_{xy_2}^{(k+1)}$ and original rating $r_{y_2z}^{(0)}$, it is able to predict the potential trust rating $r_{xz}^{(k+2)}$, where z is 2-hops way from x in $G^{(k+1)}$. Intuitively, NeuralWalk "Walks" from y_2 to z.

find all of its out-neighbors S, i.e., 1-hop neighbors. If $k=0$, i's trust ratings on $s \in S$ will be original trust ratings only, since $r_{is}^{(0)}, \forall s \in S$ comes from $G^{(0)}$. Otherwise, it can be either original trust ratings or potential trust ratings predicted in previous iteration. Based on S, NeuralWalk conducts one more level of BFS-style search on each $s \in S$ and reaches nodes that are *at most* 2-hops away from i in $G^{(k)}$. We denote the set of these nodes as J. NeuralWalk will collect training and inference samples for WalkNet from i, S and J. More specifically, NeuralWalk processes the nodes in J in two cases:

Case 1. If the node $j \in J \cap S$ and $r_{ij}^{(k)} = r_{ij}^{(0)} \in G^{(0)}$ (i.e., r_{ij} is an original trust relation in the original graph), the NeuralWalk collects all of potential and original trust ratings on 2-hop path(s) from i to j, i.e., $(r_{is}^{(k)}, r_{sj}^{(k)})$ as input, where $s \in N_{out}(i) \cap N_{in}(j)$. With $r_{ij}^{(0)}$ as a label, a training sample is formed:

$$\left\{(r_{is}^{(k)}, r_{sj}^{(k)}) \forall s \in N_{out}(i) \cap N_{in}(j), r_{ij}^{(0)}\right\}. \tag{5}$$

Case 2. If the node $j \in J$ and $j \notin S$, the NeuralWalk collects all of the 2-hop path(s) from i to j, i.e., $(r_{is}^{(k)}, r_{sj}^{(k)})$, where $s \in N_{out}(i) \cap N_{in}(j)$ to form an inference sample:

$$\left\{(r_{is}^{(k)}, r_{sj}^{(k)}) \forall s \in N_{out}(i) \cap N_{in}(j)\right\}. \tag{6}$$

Notably, predicted trust relations, i.e.,$r_{ij} \in G^{(k)}/G^{(0)}$, will not be considered as labels here. The purpose is to guarantee that the ground truths in the training set will always be correct. By traversing all of the nodes $i \in G$ as is introduced above, the algorithm will obtain a training set $\mathcal{T}^{(k)}$ and inference set $\mathcal{I}^{(k)}$ for the current iteration.

With $\mathcal{T}^{(k)}$, a WalkNet for the current iteration, denoted as $\mathcal{W}^{(k)}$, can be learned. By applying $\mathcal{W}^{(k)}$ on $\mathcal{I}^{(k)}$, the potential trust relation $r_{ij}^{(k)}$ for each $r_{is}^{(k)}, r_{sj}^{(k)} \in \mathcal{I}^{(k)}$ can be computed. In the end, an updated TSN for the next iteration, $G^{(k+1)}$, will be obtained by filling the computed trust relations into $G^{(k)}$. With $G^{(k+1)}$, the algorithm goes into the next iteration, until the iteration number is reached or all of the potential trust relations have been identified. As the TSN is filled with more potential trust relations, more training samples yielding Eq. 5 will be found. Therefore, we have $|\mathcal{T}^{(k+1)}| \geq |\mathcal{T}^{(k)}|$.

B. Boolean Matrix Operation

After showing the process of "walking" in a straightforward way, we present how NeuralWalk is implemented with Boolean matrix operations. By using Boolean matrix operations, NeuralWalk can be significantly accelerated with matrix-efficient tool-kits in a parallel or distributed environment. In addition, a further increase in computing speed can be expected if the Boolean matrix operation is implemented with GPUs.

For a TSN $G^{(k)}$ in iteration k, we use $\mathbb{G}^{(k)}$ to denote its adjacent matrix, where $\mathbb{G}^{(k)}[i,j]=1$ if there is an original or potential trust rating between node i and j, otherwise we have $\mathbb{G}^{(k)}[i,j]=0$. For any $i \in G^{(k)}$, we use a vector

$$\mathbf{s}_i^{(k)} = \left[e_{i1}^{(k)}, e_{i2}^{(k)}, \cdots, e_{ij}^{(k)}, \cdots, e_{iN}^{(k)}\right]^{\mathrm{T}}$$

to indicate if i has trust ratings in all trustees from $\mathbb{G}^{(k)}$, i.e., $\mathbf{s}_i^{(k)} = \mathbb{G}^{(k)}[i,:]$. Note that the index of any nonzero element in $\mathbf{s}_i^{(k)}$ is the index of a 1-hop neighbor of i in $G^{(k)}$. Based on $\mathbb{G}^{(k)}$ and $\mathbf{s}_i^{(k)}$, we conduct a matrix multiplication to get another vector

$$\mathbf{j}_i^{(k)} = \left(\mathbb{G}^{(k)}\right)^{\mathrm{T}} \times \mathbf{s}_i^{(k)} = \begin{bmatrix} e_{i1}^{(k)} e_{11}^{(k)} + \cdots + e_{iN}^{(k)} e_{N1}^{(k)}, \\ e_{i1}^{(k)} e_{12}^{(k)} + \cdots + e_{iN}^{(k)} e_{N2}^{(k)}, \\ \cdots \\ e_{i1}^{(k)} e_{1N}^{(k)} + \cdots + e_{iN}^{(k)} e_{NN}^{(k)} \end{bmatrix}. \tag{7}$$

In addition, to track each element $e_{ix}^{(k)} e_{xj}^{(k)}$ in Eq. 7, we also keep a matrix

$$\mathbb{J}_i^{(k)} = \left(\mathbb{G}^{(k)}\right)^{\mathrm{T}} \otimes \mathbf{s}_i^{(k)} = \begin{bmatrix} e_{i1}^{(k)} e_{11}^{(k)}, \cdots, e_{iN}^{(k)} e_{N1}^{(k)}, \\ e_{i1}^{(k)} e_{12}^{(k)}, \cdots, r_{iN}^{(k)} e_{N2}^{(k)}, \\ \cdots \\ r_{i1}^{(k)} e_{1N}^{(k)}, \cdots, r_{iN}^{(k)} e_{NN}^{(k)}, \end{bmatrix},$$

where $\otimes$ denotes element-wise product. Typically, there will be special paths $e_{ii}^{(k)} e_{ij}^{(k)}$ $\forall 1 < x < N$ in $\mathbb{J}_i^{(k)}$. These paths are essentially 1-hop edges from i to j. Thereby, we set $e_{ii}^{(k)} = 0$ in $\mathbf{s}_i^{(k)}$, such that any nonzero element $\mathbb{J}_i^{(k)}[s,j]$ only indicate a 2-hops path from i through s to j.

By carrying out an element-wise multiplication between $\mathbf{s}_i^{(0)}$ and $\mathbf{j}_i^{(k)}$, we get a third vector

$$\mathbf{m}_i^{(k)} = \mathbf{s}_i^{(0)} \otimes \mathbf{j}_i^{(k)} = \begin{bmatrix} \left(e_{i1}^{(k)} e_{11}^{(k)} + \cdots + e_{iN}^{(k)} e_{N1}^{(k)}\right) e_{i1}^{(0)}, \\ \left(e_{i1}^{(k)} e_{12}^{(k)} + \cdots + e_{iN}^{(k)} e_{N2}^{(k)}\right) e_{i2}^{(0)}, \\ \cdots \\ \left(e_{i1}^{(k)} e_{1N}^{(k)} + \cdots + e_{iN}^{(k)} e_{NN}^{(k)}\right) e_{iN}^{(0)} \end{bmatrix}.$$

With $\mathbf{s}_i^{(k)}$, $\mathbf{m}_i^{(k)}$ and $\mathbb{J}_i^{(k)}$, we can find out the training set $\mathcal{T}^{(k)}$ as introduced in the previous session. Specifically, a nonzero element, whose index is j in $\mathbf{m}_i^{(k)}$, indicates that there exists 2-hops path(s) from i to j, as well as an 1-hop original trust relation from i to j in $G^{(0)}$. These two-hops paths can be retrieved from $G^{(k)}$ with the nonzero elements indices from

$\mathbb{J}_i^{(k)}[j,:]$. For example, $\mathbb{J}_i^{(k)}[j,s]=1$ indicates that there is a 2-hops path from i through s to j in $G^{(k)}$. Therefore, these two-hops paths and $\mathbf{s}_i^{(k)}[j]$ will compose of a training sample defined in Eq. 5.

On the other hand, the algorithm conducts another element-wise multiplication between $\overline{\mathbf{s}_i^{(k)}}$ and $\mathbf{j}_i^{(k)}$ to obtain a forth vector

$$\mathbf{p}_i^{(k)} = \overline{\mathbf{s}_i^{(k)}} \otimes \mathbf{j}_i^{(k)} = \begin{bmatrix} \left(e_{i1}^{(k)}e_{11}^{(k)} + \cdots + e_{iN}^{(k)}e_{N1}^{(k)}\right)\overline{e_{i1}^{(k)}}, \\ \left(e_{i1}^{(k)}e_{12}^{(k)} + \cdots + e_{iN}^{(k)}e_{N2}^{(k)}\right)\overline{e_{i2}^{(k)}}, \\ \cdots \\ \left(e_{i1}^{(k)}e_{1N}^{(k)} + \cdots + e_{iN}^{(k)}e_{NN}^{(k)}\right)\overline{e_{iN}^{(k)}} \end{bmatrix},$$

where $\overline{\mathbf{s}_i^{(k)}}$ is the negation of $\mathbf{s}_i^{(k)}$, i.e., turning nonzero elements in $\mathbf{s}_i^{(k)}$ into zeros and vice versa. With $\mathbf{p}_i^{(k)}$ and $\mathbb{J}_i^{(k)}$, we can find out the inference set as introduced in the previous session. Specifically, a nonzero element, whose index is j in $\mathbf{p}_i^{(k)}$, indicates that there exists 2-hops path(s) from i to j, but no 1-hop trust relation from i to j in $G^{(k)}$. Notably, any original or potential trust ratings already in $G^{(k)}$ will be ignored by multiplying with $\overline{\mathbf{s}_i^{(k)}}$. This mechanism will prevent a trust rating be repeatedly computed in a loop. Similarly, these two-hops paths can be retrieved from $G^{(k)}$ with the nonzero elements indices from $\mathbb{J}_i^{(k)}[j,:]$. These two-hops paths will compose of an inference sample in Eq. 6.

In the end, a WalkNet $\mathcal{W}^{(k)}$ can be trained with $\mathcal{T}^{(k)}$. $G^{(k+1)}$ is then updated with the inference results from applying $\mathcal{W}^{(k)}$ on $\mathcal{I}^{(k)}$.

C. NeuralWalk Algorithm

Algorithm 1 NeuralWalk(G, V, H)

```
Require: A directed graph $G^{(0)}$ with node set $V$ and iteration number $H$.
Ensure: $i$'s trust rating on $j$, $\forall i, j \in V, i \neq j$.
 1: $k \leftarrow 0$
 2: Initialize $\mathbb{G}^{(k)}$ based on $G^{(k)}$
 3: while $k < H$ do
 4:     $\mathcal{T}^{(k)} \leftarrow \{\}$, $\mathcal{I}^{(k)} \leftarrow \{\}$
 5:     for all $i \in V$ do
 6:         Generate $\mathbf{j}_i^{(k)}$, $\mathbb{J}_i^{(k)}$, $\mathbf{m}_i^{(k)}$, $\mathbf{s}_i^{(k)}$ and $\mathbf{p}_i^{(k)}$ from $\mathbb{G}^{(k)}$
 7:         Generate $\mathcal{T}_i^{(k)}$ from $\mathbf{m}_i^{(k)}$, $\mathbb{J}_i^{(k)}$, $\mathbf{s}_i^{(0)}$ and $G^{(k)}$
 8:         $\mathcal{T}^{(k)} \leftarrow \mathcal{T}^{(k)} \cup \mathcal{T}_i^{(k)}$
 9:         Generate $\mathcal{I}_i^{(k)}$ from $\mathbf{p}_i^{(k)}$, $\mathbb{J}_i^{(k)}$, $\mathbf{s}_i^{(0)}$ and $G^{(k)}$
10:         $\mathcal{I}^{(k)} \leftarrow \mathcal{I}^{(k)} \cup \mathcal{I}_i^{(k)}$
11:     end for
12:     Train WalkNet $\mathcal{W}^{(k)}$ with $\mathcal{T}^{(k)}$
13:     Use $\mathcal{W}^{(k)}$ to infer $\mathcal{I}^{(k)}$ and update $G^{(k+1)}$ with the inference results
14:     $k \leftarrow k + 1$
15: end while
16: return $G^{(k+1)}$
```

The pseudo-code of NeuralWalk is shown in Algorithm 1. In line 2, the adjacent matrix $\mathbb{G}^{(k)}$ is generated from $G^{(k)}$. From line 5 to line 11, training set $\mathcal{T}^{(k)}$ and inference set $\mathcal{I}^{(k)}$ are generated via collecting training and inference samples $\mathcal{T}_i^{(k)}$ and inference set $\mathcal{I}_i^{(k)}$ for each node $i \in V$. In line 12, $\mathcal{T}^{(k)}$ is used to train a WalkNet $\mathcal{W}^{(k)}$. In line 13, $\mathcal{W}^{(k)}$ is used on the inference set, and the newly computed trust ratings are updated to $G^{(k+1)}$.

Now we analyze the time complexity of the NeuralWalk algorithm. The time complexity from lines 5 is $O(|V|)$. The time complexity of generating $\mathbf{j}_i^{(k)}$, $\mathbb{J}_i^{(k)}$, $\mathbf{m}_i^{(k)}$ and $\mathbf{p}_i^{(k)}$ are $O(|V|^2)$, $O(|V|^2)$, $O(|V|)$ and $O(|V|)$, respectively. Therefore, the time complexity from line 5 to line 11 is $O(|V|^3)$. However, since this part is accomplished based on Boolean matrix operations, it can be efficiently implemented. As the training time is proportional to the sample size[3], the time complexity of line 12 is $O(|V|^2)$. Similarly, the time complexity of line 13 is also $O(|V|^2)$, since the size of inference set is $|V|^2$. Finally, because $H \ll V$, we can conclude that the time complexity of NeuralWalk is $O(|V|^3)$.

V. Experiments and Result Analysis

The performance of NeuralWalk (NW) will be compared to state-of-the-art trust assessment solutions, including OpinionWalk (OW) [13], Matri [23] and TidalTrust (TT) [9]. Two real-world datasets, Advogato [19] and Pretty Good Privacy (PGP) [24], will be used in the evaluation. The focus of the evaluation will be how accurately different algorithms are, regarding to trust assessment.

A. Datasets

The first dataset, Advogato, is obtained from an online software development community where a connection between two users represents one's trust in anther's ability in software development. The trust between users is quantified by four different levels: apprentice, observer, journeyer, and master. The second dataset, PGP, is collected from a public key certification network where an edge from a node to another indicates that the node certificates the trust of the other node. The trust between nodes in PGP is also divided into four levels. The different levels in both datasets reflect different trust values, so we use numbers 1, 2, 3 and 4 to indicate them. The statistics of the datasets are summarized in TABLE I.

TABLE I: Statistics of the Advogato and PGP Datasets

Dataset	# of Vertices	# of Edges	Avg Deg	Diameter
Advogato	6,541	51,127	19.2	4.82
PGP	38,546	31,7979	16.5	7.7

B. Experimental Setup

In the evaluation, we use the same parameter settings for OW, Matri and TT, as they were in the papers [13], [23], [9], to initialize the trust opinions between users. As the trust ratings in the datasets are already categorical values, they are directly fed into the NW as the input. The only hyperparameters of NW are (1) the dimension of opinion D_o, and (2) the max number of opinions to be combined, i.e., n in Section III-D. D_o indicates the number of hidden features in a trust opinion, and we set $D_o = 16$ as the performance of NW with $D_o \geq 16$ tends to be consistent. To keep the training set robust enough, the paths $(r_{is}^{(0)}, r_{sj}^{(0)})$ from the original graph will be considered

[3]The training and inference time are also impacted by the parameter scale of WalkNet. We treat it as a constant number because it does not depends on the graph size.

as the prior choice for opinion combining. Consequentially, we set $n = 32$ because majority (93% and 95%) of the nodes in Advogato and PGP have out degree less than 32. When the out degree of a node in $G^{(0)}$ is less than 32, in the evaluation, we randomly select paths composed of computed potential trust ratings as compensation. If the combined opinions are still less then 32, the blank positions will be padded as zeros. The entire NW algorithm is developed using python 3.6. Specially, WalkNet is implemented by PyTorch 0.4.

Because OW and TT are deductive algorithms, there is no need to separate a certain portion of the datasets for the training purpose. For these algorithms, we randomly select a trustor u from the datasets and find all its 1-hop neighbors v's. For each neighbor node v, if there exists at least one path from u to v, we remove the edge (u, v) from the datasets. The original trust ratings from u to v's are considered the ground truths. With the updated datasets, OW and TT algorithms are executed to estimate u's trust in v. The estimated trust values are then compared to the ground truths to determine the algorithms' accuracy in trust assessment. We randomly select 1000 trustors from Advogato or PGP, in the evaluation, to obtain statistically significant results.

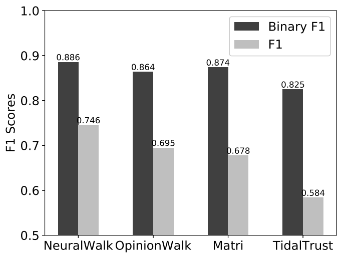
For NW and Matri algorithms, the datasets are divided into two parts: one for training and the other for testing. For Matri, we first identify all trustor-trustee pairs (u, v) in the datasets and treat the trust ratings between them as the labels. Then, 20% of edges are randomly selected and removed from the graph, to compose the testing set. The remaining graph is used as the training set. After Matri is trained, it is used to the estimate the trust values of the edges in the testing set. The difference between the estimated and ground truth values is recorded and treated as the algorithm's accuracy in trust assessment. Similarly, for NW, we split the datasets as 80% and 20% for training and testing sets, respectively.

C. Evaluation Metrics

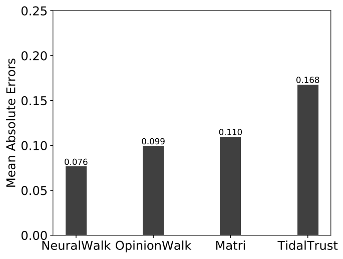
The metrics used to evaluate the accuracy of trust assessment offered by these algorithms are F1 score, mean absolute error (MAE), and binary F1 score. Because NW is essentially a categorical classifier, F1 score is the most appropriate metric. However, OW, Matri and TT aim at computing a continuous trust value, for a given trustor-trustee pair. Therefore, to make a fair comparison, MAE is used in the evaluation. To evaluate NW using MAE, the four trust levels are converted to 0.1, 0.4, 0.7 and 0.9, respectively. To obtain an F1 score, for Matri, OW and TT, the estimated trust values are rounded to the closest categorical trust value. Last but not the least, it is commonly useful to know whether a node is trustworthy or not, so the four different trust levels are collapsed into two levels, i.e., trustworthy and untrustworthy, to obtain binary F1 scores for different algorithms. The original first and second levels of trust are merged into one level and the other two levels into another level.

D. Accuracy

Using the Advogato dataset, we evaluate the trust assessment accuracy of different algorithms. As shown in Fig. 4a and Fig. 4b, NW offers the highest accuracy, in terms of F1 score, binary F1 score, and MAE. It is intriguing to note that NW achieves an F1 score as high as 0.746, which is 0.051 higher than OW – the best solution in the literature. As the range of an F1 score is $[0, 1]$, a 0.051-higher F1 score is considered a substantial improvement. The binary F1 scores of NW and other algorithms are higher than their F1 scores, which makes sense as the binary F1 score only provides a coarse accuracy measurement. Not surprisingly, all algorithms achieve a > 0.8 F1 score, as shown in Fig. 4a. The observation indicates that trust assessment becomes much easier if it is solely for determining whether a user is trustworthy or not. Nevertheless, NW achieves the best performance, with a binary F1 score of 0.886. This is still better than the second best algorithm – Matri. As shown in Fig. 4b, NW's outstanding performance is also confirmed when MAE is used to measure the accuracy in trust assessment. Specifically, the MAE of NW is as low as 0.076, which is nearly 25% lower than the second best solution – OW.



(a) 4 Class and Binary F1 Scores



(b) Mean Absolute Error

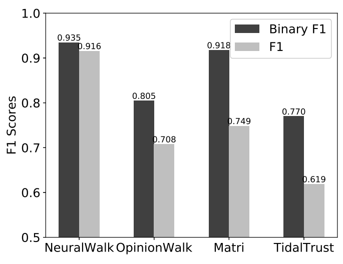
Fig. 4: Accuracy of different algorithms on Advogato.

To confirm NW's performance does not depend on datasets, we then evaluate NW, as well as all other algorithms, using the PGP dataset. As shown in Fig. 5a and Fig. 5b, all algorithms perform better, compared to their performance in Advogato. For example, the NW achieves the highest F1 score of 91.6%. It is 16.7% higher than Matri, the second best one. It is interesting to note that the accuracy of OW is lower than Matri, which is not the case when the Advogato dataset was used. That implies neither Matri nor OW will perform well consistently, in both Advogato and PGP. As we expected, the binary F1 scores of NW and all other algorithms are higher than their F1 scores. Among all the solutions, NW achieves the highest binary F1 score 0.935. As shown in Fig. 5b, the overall trend of MAE is similar to that of Advogato. NW still performs the best while Matri is the second best. In point of fact, the MAE of NW is only 0.054, which is nearly 50% lower than that of Matri (the second best).

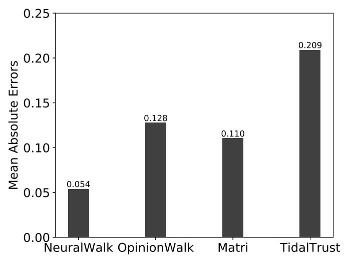
From the above experiments, we conclude that NW significantly outperforms existing (either inductive or deductive) solutions to trust assessment, in both the Advogato and PGP datasets. The main reason is that NW inherited the capability of 3VSL, in terms of accurately modeling trust. Moreover, the machine learning based (discounting and combining) opinion operations enable more precise trust computation, which is lacked in the original 3VSL trust model.

E. Robustness

In addition to accuracy, we are interested in NW's robustness to training sample size. As an inductive method, NW's performance may degrade when not enough training samples are available. The method of evaluating NW's robustness to sample size is to adjust the ratio between the training and testing sets. Specially, we compare the performance of NW



(a) 4 Class and Binary F1 Scores

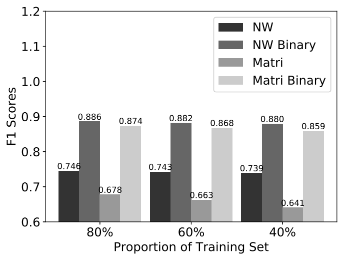


(b) Mean Absolute Error

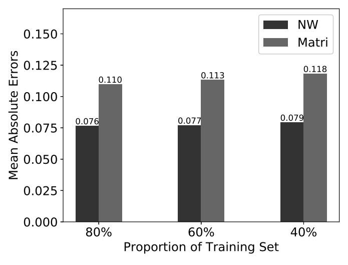
Fig. 5: Accuracy of different algorithms on PGP.

when the train set is 80%, 60% and 40% of the entire dataset, respectively. We compare NW to Matri and ignore OW and TT algorithms because Matri is an inductive solution while the others are not.

As shown in Fig. 6, the degraded performance of NW is not notable when the ratio is reduced to 40% in the Advogato dataset. In Fig. 6a, the F1 scores of NW are 0.743 and 0.739, when the training set ratios are 60% and 40%, respectively. The F1 scores of NW are only 0.003 and 0.007 smaller than the best one where the ratio is 80% (where the F1 score is 0.746). On the other hand, the F1 scores of Matri are 0.663 and 0.641 when ratios are 60% and 40%, respectively. Matri degrades 0.015 and 0.037 from the best 0.678, i.e., Matri shows worse robustness. For binary F1 score, NW's performance drops 0.004 and 0.006 from the best 0.886, which can be ignored. The binary F1 scores of Matri are 0.006 and 0.015 lower than that of the best case, when the the ratio is reduced to 60% and 40%, respectively. As shown in Fig. 6b, the MAE of NW is increased 0.001 and 0.003 from the best 0.076. Overall, we do not observe drastic performance difference for NW, when the training/testing ratio is adjusted.



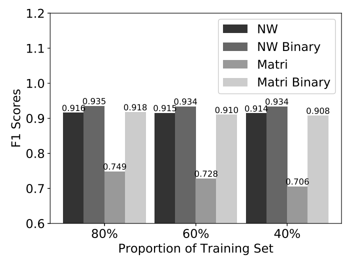
(a) 4 Class and Binary F1 Scores



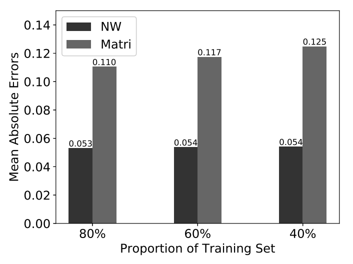
(b) Mean Absolute Error

Fig. 6: Impact of training size on Advogato.

Using the PGP dataset, similar yet even better observations can be found in Fig. 7. In Fig. 7a, the F1 scores of NW are 0.916, 0.915 and 0.914, when the training sets are 80%, 60% and 40%, respectively. The performance difference is too subtle to be notable. On the other hand, the F1 scores of Matri are 0.749, 0.728 and 0.706, when the ratios are 80%, 60% and 40%, respectively. The performance of Matri degrades much faster than NW. Minor performance degradation is also observed for NW, if binary F1 score is used in evaluations. Finally, as shown in Fig. 7b, NW's MAE does not change significantly when the ratio is changed from 80% to 60% and then to 40%. In summary, we conclude that NW is robust to the training sample size, i.e., it offers a fairly stable performance even if smaller amount of training samples are available.



(a) 4 Class and Binary F1 Scores



(b) Mean Absolute Error

Fig. 7: Impact of training size on PGP.

VI. Related Work

Previous study in trust assessment mainly focuses in two folds, i.e., trust inference and trust modeling. Trust inference focuses on conducting multi-hop trust assessment in complicated network. Assuming trust is a scalar number and the trust value is derived from the paths between a trustor and trustee, several research works are proposed [11], [9]. The methods proposed in [11], [9] aim at computing the trustworthiness by performing path searches in the network and models trust propagation as flow decaying and aggregation in the network. However, these methods are lack of accurate ways for modeling trust. On the other hand, trust modeling focuses on representing the existing trust between two user with a mathematical model. Conventional methods model trust as statistic distribution upon evidences or observations [7], [8]. The advantage of these methods is that they are able to model the complicated nature of trust, in an explainable way. However, the limitation is these methods that the are unable to handle complex networks due to the limitations identified in [12]. Combining both trust modeling and inference, AssessTrust [12] and OpinionWalk [13] are proposed to conduct trust assessment in complex networks, such as online social networks. However, a non-trivial problem remaining to AssessTrust and OpinionWalk is that their parameters have to be determined empirically, which limits their accuracy and is impractical in trust assessment systems. Recently, matrix factorization-based methods [25], [23], such as Matri, are proposed to conduct both trust modeling and inference in an inductive manner. By minimizing the error between the observed trust values and the inner products of corresponding trustor/trustee vectors, the unobserved trust values can be predicted via conducting inner products upon their feature vectors. The main limitation of these methods is that the output of this models are continues scalars indicating the strength of trust, therefore cannot be accurately fitted into discrete trust ratings existed in most of the trust datasets. In addition, none of them accounts for the trust fusion operation, therefore their accuracy are impacted.

The idea of WalkNet is inspired by the recent studies in graph representation learning [26], [27], [28], [29]. In [26], an approach called DeepWalk is proposed to learn distributed representations of nodes in a social network, based on the co-occurrence of the nodes in a local area. The distributed representations are vectors with same dimensions. The tie strength between two nodes in a social network can hence be measured by the distance between the vectors representing them. Based upon DeepWalk, improvements are made in [27] to account for the dynamics and scalability of social networks. In [28], a framework called GraphSage is proposed to predict the feature of a target node via learning and aggregating the feature representations of the k hop nodes around it. Similar to GraphSage, in [29], a model called R-GCN is proposed to predict the feature (link type) of a target edge by the feature (entity type) representations of nodes around it.

Note that the WalkNet model and NeuralWalk algorithm are different from the aforementioned works. On the one hand, WalkNet makes use of the interactions of edge features in serial and parallel typologies to predict the feature of a certain edge. On the other hand, WalkNet is regarded as an end-to-end solution of single-hop trust assessment only. Working with WalkNet, the NeuralWalk algorithm conduct multi-hop trust assessment across TSNs in a BFS manner.

VII. Conclusions

We proposed a machine learning driven algorithm, NeuralWalk, to tackle the trust assessment problem in trust social networks. Unlike traditional solutions, NeuralWalk employs a neural network architecture, called WalkNet, to model single-hop trust propagation and trust combination. The parameters in WalkNet are learned, when the NeuralWalk algorithm searches within a trust social network, via the standard training framework of neural networks. Based on the learned single-hop trust computation, i.e., discounting and combining operations, NeuralWalk iteratively conducts multi-hop trust assessment in a BFS manner. Experiments are conducted against two real-world datasets, and results demonstrate that NeuralWalk outperforms the state-of-the-art solutions. As a fundamental framework, NeuralWalk can be extended to cope with more complex trust inference algorithms. The obtained transformation mechanism, for trust ratings and trust opinions, can be used to covert existing datasets to support the validation of sophisticated trust models.

Acknowledgment

The work is supported by Stratifyd, Inc. and National Science Foundation (NSF) grant NSF CNS-1761641. The authors would like to thank the insightful comments from Xiaoyu (Derek) Wang and Wenwen Dou. The authors would also like to thank the help from Michael Schlichtkrull and Yuan Yao for code and datasets.

References

[1] X. Niu, G. Liu, and Q. Yang. Trustworthy website detection based on social hyperlink network analysis. *IEEE Transactions on Network Science and Engineering*, pages 1–1, 2018.

[2] Guangchi Liu, Qing Yang, Honggang Wang, Shaoen Wu, and M. P. Wittie. Uncovering the mystery of trust in an online social network. In *2015 IEEE Conference on Communications and Network Security (CNS)*, pages 488–496, Sep. 2015.

[3] X. Li, Q. Yang, X. Lin, S. Wu, and M. Wittie. Itrust: interpersonal trust measurements from social interactions. *IEEE Network*, 30(4):54–58, July 2016.

[4] T. Cheng, G. Liu, Q. Yang, and J. Sun. Trust assessment in vehicular social network based on three-valued subjective logic. *IEEE Transactions on Multimedia*, pages 1–1, 2019.

[5] De-Nian Yang, Hui-Ju Hung, Wang-Chien Lee, and Wei Chen. Maximizing acceptance probability for active friending in online social networks. In *19th ACM SIGKDD*, pages 713–721, 2013.

[6] Lik Mui. *Computational models of trust and reputation: Agents, evolutionary games, and social networks*. PhD thesis, Massachusetts Institute of Technology, 2002.

[7] Glenn Shafer. *A mathematical theory of evidence*, volume 42. Princeton university press, 1976.

[8] AUDUN Josang. A logic for uncertain probabilities. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 09(03):279–311, 2001.

[9] Jennifer Golbeck, James Hendler, et al. Filmtrust: Movie recommendations using trust in web-based social networks. In *IEEE CCNC*, pages 282–286, 2006.

[10] Zoltán Gyöngyi, Hector Garcia-Molina, and Jan Pedersen. Combating web spam with trustrank. In *VLDB '04*, pages 576–587, 2004.

[11] Paolo Massa and Paolo Avesani. Controversial users demand local trust metrics: An experimental study on epinions. com community. In *Proceedings of the National Conference on artificial Intelligence*, volume 20, page 121, 2005.

[12] G. Liu, Q. Yang, H. Wang, X. Lin, and M. P. Wittie. Assessment of multi-hop interpersonal trust in social networks by three-valued subjective logic. In *IEEE INFOCOM 2014 - IEEE Conference on Computer Communications*, pages 1698–1706, April 2014.

[13] G. Liu, Q. Chen, Q. Yang, B. Zhu, H. Wang, and W. Wang. Opinionwalk: An efficient solution to massive trust assessment in online social networks. In *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*, pages 1–9, May 2017.

[14] Audun Jøsang, Stephen Marsh, and Simon Pope. Exploring different types of trust propagation. In *Trust management*, pages 179–192. Springer, 2006.

[15] Audun Jøsang, Tanja Ažderska, and Stephen Marsh. Trust transitivity and conditional belief reasoning. In *IFIP International Conference on Trust Management*, pages 68–83. Springer, 2012.

[16] Guangchi Liu. Trust assessment in online social networks. *Ph.D. dissertations in computer science*, Jul 2017.

[17] Christian Borgs, Jennifer Chayes, Adam Tauman Kalai, Azarakhsh Malekian, and Moshe Tennenholtz. *A Novel Approach to Propagating Distrust*, pages 87–105. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[18] Cai-Nicolas Ziegler and Georg Lausen. Propagation models for trust and distrust in social networks. *Information Systems Frontiers*, 7(4):337–358, 2005.

[19] Raph Levin. Advogato. http://www.advogato.org/, 2014.

[20] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[21] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, pages 1097–1105, USA, 2012. Curran Associates Inc.

[22] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

[23] Yuan Yao, Hanghang Tong, Xifeng Yan, Feng Xu, and Jian Lu. Matri: A multi-aspect and transitive trust inference model. In *Proceedings of the 22Nd International Conference on World Wide Web*, WWW '13, pages 1467–1476, New York, NY, USA, 2013. ACM.

[24] Chung-Wei Hang, Yonghong Wang, and Munindar P. Singh. Operators for propagating trust and their evaluation in social networks. In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems*, pages 1025–1032, 2009.

[25] Xiaoming Zheng, Yan Wang, Mehmet A. Orgun, Youliang Zhong, and Guanfeng Liu. Trust prediction with propagation and similarity regularization. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, AAAI'14, pages 237–243. AAAI Press, 2014.

[26] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, pages 701–710, New York, NY, USA, 2014. ACM.

[27] A. Zhiyuli, X. Liang, and Z. Xu. Learning distributed representations for large-scale dynamic social networks. In *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*, pages 1–9, May 2017.

[28] William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *NIPS*, 2017.

[29] Michael Sejr Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. *CoRR*, abs/1703.06103, 2017.