Medley: Predicting Social Trust in Time-Varying Online Social Networks

Wanyu Lin, Baochun Li
University of Toronto, wanyu.lin@mail.utoronto.ca, bli@ece.toronto.edu

***Abstract*—Social media, such as Reddit, has become a norm in our daily lives, where users routinely express their attitude using upvotes (likes) or downvotes. These social interactions may encourage users to interact frequently and form strong ties of trust between one another. It is therefore important to predict social trust from these interactions, as they facilitate routine features in social media, such as online recommendation and advertising.**

Conventional methods for predicting social trust often accept static graphs as input, oblivious of the fact that social interactions are time-dependent. In this work, we propose *Medley*, to explicitly model users' time-varying latent factors and to predict social trust that varies over time. We propose to use functional time encoding to capture continuous-time features and employ attention mechanisms to assign higher importance weights to social interactions that are more recent. By incorporating topological structures that evolve over time, our framework can infer pairwise social trust based on past interactions. Our experiments on benchmarking datasets show that *Medley* is able to utilize time-varying interactions effectively for predicting social trust, and achieves an accuracy that is up to 26% higher over its alternatives.

I. Introduction

Social media platforms have become indispensable in our daily lives. Users routinely express their attitude towards a post or service while interacting with another user in these online platforms. Interactions with positive views, such as upvotes and likes, may encourage social interactions between users, which build stronger ties of trust. These established ties of trust form a prevalent yet complex force that drives our social decisions and can be leveraged to market services or make recommendations.

Inferring trust between a pair of users is an essential aspect of characterizing and understanding the underlying online social networks. By inferring trust in these online social networks, providers of social media platforms—such as Reddit or Twitter—can encourage positive (or discourage disruptive) user behavior. They can take preemptive actions by introducing new products or services if such ties of trust are found below a certain threshold.

Deep learning models [1], [2] have played an increasingly critical role for evaluating social trust, as they are used to learn low-dimensional embeddings for the users in online social networks. Representations learned using deep learning models can complement or even replace traditional evaluation algorithms to establish social trust, such as matrix factorization techniques [3], [4]. Among others, graph neural networks (GNNs), used for supervised learning on graph-structured data, have yielded impressive results in these domains, mainly due to their efficiency and capability of inductive learning [2], [5]. In particular, Guardian [2] proposed a framework to characterize propagation rules of social trust automatically, using the notion of localized graph convolutions [6].

This research was supported in part by the NSERC Discovery Research Program.

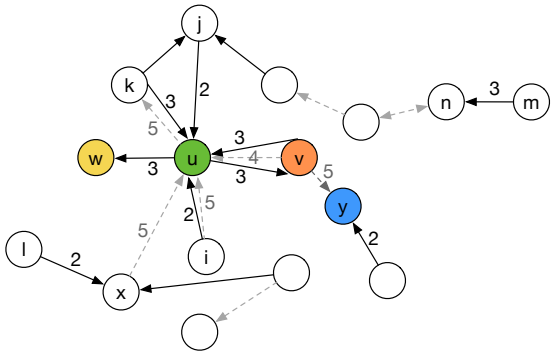


Fig. 1. A motivating example for time-aware social trust prediction (best viewed in color): The graph represents observed interactions during time interval $[0, T]$. User u and v are the target users to provide trust levels for at time $T + \Delta(T)$. The dashed and solid arrows denote interactions formed at time t_1 and t_2, respectively, where $0 < t_1 < t_2 \leq T$. The numbers are observed trust levels correspondingly.

Although these mechanisms provided reasonable performance, one major deficiency lies in their common assumption that the underlying social networks are static, omitting the fact that social interactions in online social networks are time-varying. Due to the evolving nature of online social networks over time, the strength of established trust can increase or decrease with new interactions; they may also decay with time. As shown in Fig. 1, there are multiple interactions formed between user u and v at different times, denoted with solid (new interactions) and dashed arrows (old ones). Existing models [1], [2] focused on a particular snapshot of the social interaction graph and failed to consider its time-varying features.

Ignoring time-varying dynamics in social networks can severely reduce the efficacy and optimality of existing solutions. For instance, if time information is not considered, models may mistakenly utilize future interactions to infer past ties of trust during training and testing. Therefore, it is critical to capture time features of social interactions explicitly. In

this work, we aim to infer trust between any two users based on past interactions in an online social network, explicitly incorporating associated timing information.

For a user (which is represented by a node in the social graph), it is natural to encode timing information along with other essential features. By applying the notion of localized trust convolution [2], the representations learned may well capture the evolving property of online social networks over time. However, encoding timing information is non-trivial since time is a continuous variable. Thus, our first challenge is to map the continuous-time domain into a vector space. In addition, new interactions are typically more important than old ones, as early interactions may become obsolete or irrelevant. We propose that different weights should be assigned while establishing the trust ties, e.g., more recent interactions should have higher weights for contributing to a tie of trust. Therefore, the second challenge is to find ways of discriminating the importance of social interactions formed over time, such that social trust can be well quantified in online social networks.

Considering the limitations of existing work, it is essential to develop a new model that can exploit time-varying social interactions for trust prediction in an efficient, explicit, and end-to-end manner. In this paper, inspired by the recent success of neural networks on learning graphs, we propose *Medley*, an end-to-end framework to discover hidden and predictive time-aware trust signals in online social networks, by incorporating graph attention mechanisms and functional time encoding into a unified framework. In *Medley*, we propose an attentive-trust propagation network, which is equipped with attention mechanisms to discriminate the importance of time-varying interactions. Toward this end, time-aware trust signals can be characterized by incorporating topological structures that evolve over time. Specifically, we represent both social interactions and the associated time domain in a latent space. To achieve this goal, we first apply the functional time encoding to map the continuous-time domain into a vector space, based on the classical Bochner's theorem from harmonic analysis [7].

We demonstrate the effectiveness of our proposed framework using two representative web-of-trust social networking systems: Bitcoin-Alpha and Bitcoin-OTC. We instantiate our attentive-trust propagation network with different attention operators, the inner product and the concatenation operator. We then compare *Medley* with Guardian [2], a state-of-the-art social trust evaluation model. Our extensive array of experiments on benchmarking datasets suggests that *Medley* can well capture the latent factors of time-varying interactions with an accuracy that is 27% higher than Guardian [2], showing the effectiveness of our proposed framework. We also empirically show that the choice of attention operators is critical. In general, when instantiated with the inner product operator, *Medley* achieves the best overall performance compared to existing work.

TABLE I
MATHEMATICAL NOTATIONS

Notation	Descriptions
$\|C\|$	the number of trust levels
$\mathcal{N}_u(T)$	the neighborhood of node u during time interval $[0, T]$
e^c	embedding of trust level c
x_u	initial embedding of user u
d_h, d_t, d_c	the dimension of user embedding, time embedding and interaction embedding
h_u	the latent representation of user u
h_{uv}	the hidden trust signals between user u and v
$\tilde{h}_{uv}$	predicted trust strength
$\mathbf{c}_{uv}$	the ground-truth of trust strength
$\|$	the concatenation operator
$\oplus$	weight mean aggregator
σ	non-linear functions, e.g., $\mathrm{LeakyReLU}(\cdot)$, $\mathrm{softmax}(\cdot)$
W, b	the model parameters (weight matrices and bias)[a]

[a]To simplify our notations, we unify all of the trainable matrices and learnable biases as W and b, respectively.

II. PREDICTING SOCIAL TRUST

In this paper, we consider the problem of predicting social trust in a time-varying online social network, modeled as a sequence of timestamped interaction events in a social graph, denoted as $\mathcal{G} = \{\mathbf{e}(t_1), \mathbf{e}(t_2), \cdots\}$, where $\mathbf{e}(t_i)$ represents a social interaction between a pair of nodes in the social graph at time t_i. More precisely, an interaction between nodes u and v formed at time t is represented as a directed edge $\mathbf{e}_{uv}(t)$. We also define $C = \{c_1, c_2, c_3, \cdots\}$ as the set of trust levels. Examples of such online social networks with support for trust levels include Bitcoin-OTC[1] and Pretty-Good-Privacy[2] (PGP), where $C =$\{trust, distrust\} in Bitcoin-OTC and $C =$\{Observer, Apprentice, Journeyer, Master\} in PGP.

Let $\mathcal{E}(T) = \{\langle u, v\rangle : \exists\ \mathbf{e}_{uv}(t) \in \mathcal{G}, t \leq T\}$ be the time-varying set of social interactions and $\mathcal{V}(T) = \{i : \exists\ v_i(t) \in \mathcal{G}, t \leq T\}$ be the time-varying set of users in a given online social graph $\mathcal{G}$. Without loss of generality, we assume that the time domain is represented as the time interval $[0, T]$, where T is determined by the observed social graph. $\mathcal{N}_u(T) = \{v : \exists\ \mathbf{e}_{uv}(t) \in \mathcal{E}(T), t \leq T\}$ denotes the neighborhood of node u during the time interval $[0, T]$. To model timing information, a mapping function that can map the continuous-time domain into a vector space is defined as $\Psi : t \to \mathbb{R}^{d_t}$, e.g, the encoding of time point t_1 is denoted as $\Psi(t_1)$. These social networks are evolving, where nodes and edges change over time.

[1]https://www.bitcoin-otc.com
[2]http://networkrepository.com/arenas_pgp.php

We will now formally define the problem of social trust prediction in a time-varying online social network. Given $\mathcal{G}(T) = (\mathcal{E}(T), \mathcal{V}(T))$, each observed interaction is presented as $\{(\langle u, v\rangle,\, t,\, c) \,|\, \mathbf{e}_{uv} \in \mathcal{E}(T),\, t \leq T,\, c \in C\}$, the problem of predicting social trust seeks to infer the trust level between any two users at time $T + \Delta(T)$. Specifically, the prediction model aims to quantify the probability distribution of the trust levels after a given time interval.

$$\mathcal{P} = \left(c_{uv}^{(T+\Delta(T))} | \mathcal{G}(T)\right) \tag{1}$$

For simplicity, we use $\mathbf{e}_{uv}^{c}(t)$ to denote an observed interaction that u trusts v with level c at time t, $t \leq T$ and $\mathbf{c}_{uv}$ denotes the ground truth of the corresponding trust level. The mathematical notations used in this paper are summarized in Table I.

Throughout this paper, we use a toy example to illustrate the problem of time-aware social trust prediction. Fig. 1 shows a social interaction graph observed in the time interval $[0,\, T]$, where nodes represent users, directional edges denote the interactions of user pairs, and the numbers are the associated trust levels. The dashed and solid arrows represent the interactions formed at time t_1 and t_2, respectively, where $0 < t_1 < t_2 \leq T$. Our goal is to infer whether or not u trusts v — and to what extent if such trust exists — at time $T + \Delta(T)$.

III. Preliminaries

Graph Attention (GAT) [8] is a recent proposed technique that introduces the attention mechanism into graph convolutional networks. It explicitly assigns different importance values to nodes in the same neighborhood graph. Specifically, an attention coefficient a_{ij} is first computed by an attention function $\text{attn} : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$, which measures the importance of node j to node i:

$$a_{ij} \leftarrow \text{attn}\,(Wh_i,\, Wh_j) \tag{2}$$

A softmax function is adopted to normalize attention coefficients:

$$\tilde{a}_{ij} \leftarrow \text{softmax}\,(a_{ij}) = \frac{\exp\,(a_{ij})}{\sum_{k \in \mathcal{N}_i} \exp\,(a_{ik})} \tag{3}$$

In essence, such an attention coefficient is used to capture the importance of an edge. With these coefficients, the final output features for each node then can be obtained by a linear combination of the features corresponding to them:

$$h_i \leftarrow \sigma\left(\sum_{j \in \mathcal{N}_i} \tilde{a}_{ij}\, W\, h_j\right) \tag{4}$$

where σ is a non-linear activation function, such as $\mathbf{softmax}$ shown in Eq. (3).

Theorem 1 (Bochner's Theorem [9])**.** *A continuous, translation-invariant kernel $\mathcal{K}(t_1,\, t_2) = \Phi(t_1 - t_2)$ on $\mathbb{R}^{d_t}$ is positive definite if and only if there exists a non-negative measure on $\mathbb{R}$ such that Φ is the Fourier transform of the measure.*

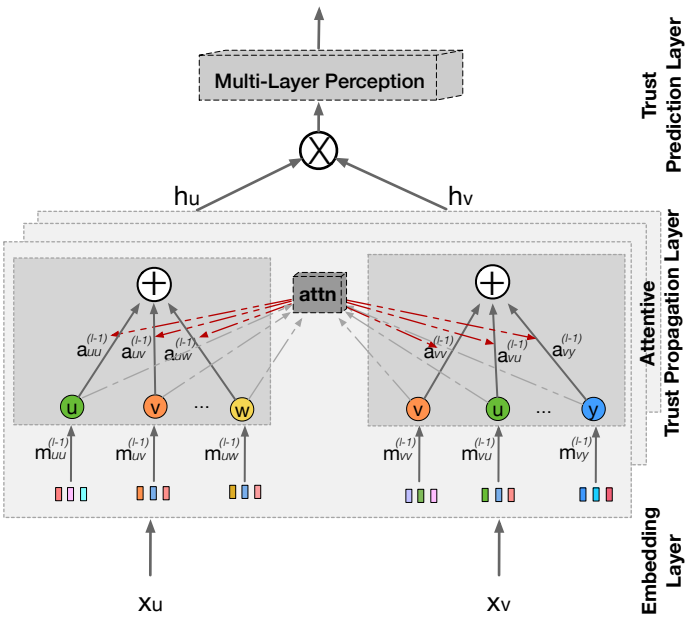


Fig. 2. Illustration of *Medley*: an embedding layer that offers an initialization of user representations, multiple attentive trust propagation layers, followed by a trust prediction layer instantiated with a multi-layer perception. The gray block "attn" represents the attention mechanism. User u and v are the target users for trust prediction.

Bochner's theorem suggests that when scaled properly, the kernel function $\Phi(t_1 - t_2)$ can be expanded as:

$$\begin{aligned}\mathcal{K}(t_1,\, t_2) &= \Phi(t_1 - t_2) \\ &= \int_{\mathbb{R}} e^{i\,\omega(t_1 - t_2)}\, p(\omega)\, d\omega = \mathbf{E}_\omega\left[\eta_\omega(t_1)\, \eta_\omega(t_2)\right]\end{aligned} \tag{5}$$

where $\eta_\omega(t) = e^{i\,\omega t}$. Both $\mathcal{K}$ and the probability $p(\omega)$ are real, thus an alternative expression of Eq. (5) can be:

$$\begin{aligned}\mathcal{K}(t_1,\, t_2) &= \mathbf{E}_\omega\left[\mathbf{cos}(\omega\,(t_1 - t_2))\right] \\ &= \mathbf{E}_\omega\left[\mathbf{cos}(\omega\, t_1)\, \mathbf{cos}(\omega\, t_2) + \mathbf{sin}(\omega\, t_1)\, \mathbf{sin}(\omega\, t_2)\right]\end{aligned} \tag{6}$$

As suggested by Rahimi & Recht [10], Eq. (6) can be further approximated with Monte Carlo integral:

$$\mathcal{K}(t_1,\, t_2) \approx \frac{1}{d_t} \sum_{i=1}^{d_t} \mathbf{cos}(\omega_i t_1)\, \mathbf{cos}(\omega_i t_2) + \mathbf{sin}(\omega_i t_1)\, \mathbf{sin}(\omega_i t_2) \tag{7}$$

where $\{\omega_i |\, i \in [1,\, d_t]\}$ are samples drawn from $\mathcal{P}(\omega)$.

IV. Medley: Design and Algorithms

In this section, we formally propose *Medley*, a deep learning-based model, to learn the time-varying dynamics of social interactions for predicting social trust in an end-to-end fashion. In what follows, we first introduce the intuition and then further describe the technical details of *Medley*.

In essence, the key component of *Medley* is the notion of graph attention mechanism (GAT) [8]. GAT is initially designed to recognize the relevant pieces of neighborhood information. In specific, the graph attention mechanism explicitly assigns different importance weights to neighbor nodes when aggregating node features. In online social networks, user-user interactions are created over time; multiple interactions between two users may be built in different time spans, as shown in Fig 1. New interactions are typically more important than old ones, as early interactions may become obsolete or irrelevant. Thus, we anticipate that different weights should be assigned while establishing the trust relationship, e.g., more recent interactions should have higher weights for contributing to a trust tie. For this purpose, we adapt graph-attention mechanism to capture relevant pieces of the time-varying neighborhood information, including established social interactions, associated timestamps, and user feature information.

To generate the time-aware embedding for a node [3], we apply a functional time encoding technique to map the continuous-time into the vector space, which then can be incorporated into the user feature information and corresponding social interactions. With this particular encoding for the time domain, each graph attention module will allow us to learn how to aggregate information, including user feature information, timing information, and corresponding social interactions, from a small graph neighborhood. By stacking multiple graph attention modules, our framework is able to propagate the local features across the entire social graph. Importantly, parameters of these localized graph attention modules are shared across all nodes, making the parameter complexity of our approach independent of the input graph size.

In a nutshell, *Medley* consists of three components: an embedding layer that offers an initialization of the user feature information, multiple attentive trust propagation layers followed by a trust prediction layer. The prediction layer is implemented with a multi-layer perception. The architecture illustration is shown in Fig. 2.

A. Embedding Layer

Following mainstream deep learning-based approaches, such as [2], we describe a user u with an embedding vector $x_u \in \mathbb{R}^{d_h}$, where d_h denotes the embedding dimension. To be more specific, user embeddings are generated based on users' social ties (or friendship), which were created whenever two users interact. As indicated by social correlation theories [11], [12], users' social behaviors, such as interactive behaviors with others, are similar to or influenced by their directly connected friends. Widely used pre-trained models for user embeddings are node2vec [13] and DeepWalk [14]. These user embeddings are used as the initial states of user representations.

B. Attentive Trust Propagation

Notably, our goal is to discover hidden and predictive trust signals in a time-varying social interaction graph. In general, an interaction graph contains not only interactions between users but also the timestamps of associated interactions. A user can express his/her attitudes (or opinions), denoted as c, to anther user during user-user interaction at time t. In other words, the learned trust signals should capture user information, the interactions between users and the timestamps of established social interactions.

Time modeling. To extract the features from the timestamps, we utilize a mapping function $\Psi : t \rightarrow \mathbb{R}^{d_t}$ to map the continuous time domain into a vector space with dimension d_t. As suggested by Xu *et al.* [7], $\Psi(t)$ can be estimated with a finite sequence formulated as:

$$\Psi(t) \leftarrow \sqrt{\frac{1}{d_t}} \left[\mathbf{cos}(\omega_1 t), \mathbf{sin}(\omega_1 t), \cdots, \mathbf{cos}(\omega_{d_t} t), \mathbf{sin}(\omega_{d_t} t)\right] \tag{8}$$

where $\{\omega_i | \, i \in [1, \, d_t]\}$ are samples drawn from distribution $\mathcal{P}(\omega)$. With this approximation, given two time points t_1, t_2, the inner product of their time encodings, denoted as $\Psi(t_1) \cdot \Psi(t_2)^\intercal$, is exactly equal to $\mathcal{K}(t_1, \, t_2)$ in Eq. (7). According to Bochner's theorem, the mapping function learning problem then can be converted to the problem of distribution learning, i.e. $p(\omega)$. The detailed definition of $\mathcal{P}(\omega)$ can be referred to as the illustration in Theorem 1. In our framework, *Medley*, the parameters of the distribution learning are jointly optimized as part of the whole model in an end-to-end fashion.

Interaction modeling. The actual type of interactions, such as trust/distrust, reveals the attitude from one user to another. More positive interactions, for example, may encourage users to interact more frequently and form stronger trust among users. These interactions implicitly indicate the trust signals among users, and therefore, they should be considered for predicting trust ties. In particular, we introduce an interaction embedding vector, denoted as $e^c \in \mathbb{R}^{d_c}$, to model the social interactions, where $c \in \{c_1, c_2, c_3, \cdots\}$. More specifically, we use one-hot encoding to represent each level of trust. For example, $C =$\{trust, distrust\}, we model them as the following one-hot representations: $[0, 1]^T$ and $[1, 0]^T$. Then a multi-layer perception is used to convert the one-hot encodings into dense vector embeddings through Eq. 9.

$$e^c_{uj} \leftarrow \left(W^c \cdot e^c_{uj}\right) \tag{9}$$

Recall that our ultimate goal is to leverage user features, social interactions, and associated time features to identify time-aware trust signals over time-varying social graphs. With the information modeling above, we build upon the message-passing architecture of graph neural networks [6], [8], [15], particularly graph attention networks, to distill comprehensive patterns of social interactions over time. In the following, we first describe the design of one-layer attentive trust propagation, and then generalize it to multiple successive layers for high-order trust propagation.

First-order attentive trust propagation. For an interaction established at time t between user u and user j with type c, we model the message from j to u, denoted as m_{uj}, with a combination of user j's embedding h_j ($h_j = x_j$ for the first

[3]This paper uses the words "node" and "user" interchangeably.

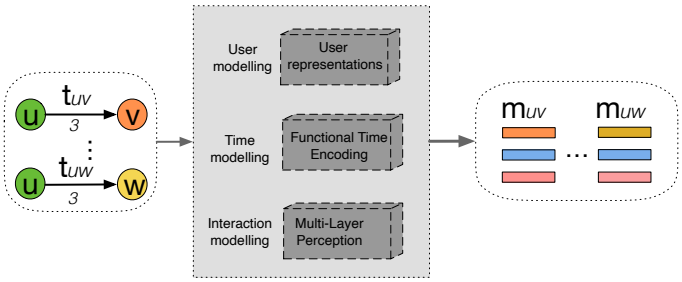


Fig. 3. An illustration of message modeling in the local neighborhood, including the functional time encoding and interaction modeling. For a target user, e.g., u, each of its interactions is transformed into a message containing three components: user representations, time embedding, and interaction representation.

layer), time embedding $\Psi(t)$ and the interaction embedding e^c_{uj} with concatenation operation.

$$m_{uj} \leftarrow \left(h_j \parallel \Psi(t_{uj}) \parallel e^c_{uj}\right) \tag{10}$$

Analogously, m_{uj} can be interpreted as the message that u consumes from j for quantifying u's attitude to others. Let's see an example in our example social network graph, originally shown in Fig. 1. For the target user u, its local interactions include $\mathbf{e}^{(3)}_{uw}(t_{uw})$, $\mathbf{e}^{(5)}_{uk}(t_{uk})$, $\mathbf{e}^{(3)}_{uv}(t_{uv})$. Messages from these interactions can be obtained via message modeling, as illustrated in Fig 3. Built upon this basis, we can aggregate the messages from the neighborhoods to fully capture the interaction pattern of user u. Mathematically, this process can be presented as:

$$h_u \leftarrow \sigma\left(\mathbf{Aggregate}\left(\{m_{uj}, \forall j \in \mathcal{N}_u(T)\}\right) \cdot W + b\right) \tag{11}$$

where W, b are trainable parameters.

As suggested by [2], the mean-based aggregator, a linear approximation of a localized spectral convolution [15], is an alternative operator for aggregation. It assumes that all interactions contribute equally to distill comprehensive information about user u. However, this may not be optimal because social influence from other users may vary dramatically, and their effects may diminish over time. Consequently, an effective learning model should allow to assign different weights to distinguish the relative importance of different interactions.

To alleviate the constraint of mean-based aggregator, we perform an attention mechanism to extract the interactions that are important to influence user u. To be more specific, Eq. (11) can be reformulated as:

$$h_u \leftarrow \sigma\left(\left(\sum_{j \in \mathcal{N}_u(T)} \tilde{a}_{uj}\, m_{uj}\right) \cdot W + b\right) \tag{12}$$

where $\tilde{a}_{uj}$ denotes the attention weight of the interaction with user j in contributing to user u's trust latent factor.

Alternative attention operators can be the inner product or concatenation. Concretely, we can use the inner product (or concatenation) of the message from j to u and the target user's own hidden message, to measure the importance of the interaction between u and j. Note that, user u's message, denoted as m_{uu}, is obtained by feeding its hidden representation h_u to a multi-layer perception (MLP). An analogy of this operator can be the similarity of users' interests. If two users have more common interests, there might be more interaction effects. Mathematically, a_{uj}, instantiated with the inner product operator, is formulated as:

$$a_{uj} \leftarrow \mathbf{attn}\left(m_{uj} W_{uj},\, h_u W_u\right) = m_{uj} W_{uj} \cdot \left(h_u W_u\right)^{\intercal} \tag{13}$$

where $W_{uj} \in \mathbb{R}^{(d_h+d_t+d_c)\times d_h}, W_u \in \mathbb{R}^{d_h \times d_h}$ are the trainable matrices that are used to distill useful interaction patterns for propagation, $h_u = x_u$ for the first layer. To obtain the relative importance of a particular interaction for user u, we use $\mathbf{softmax}$ to normalize the attentive value from Eq. (13). Then the final attention is formulated as:

$$\tilde{a}_{uj} \leftarrow \frac{\exp\left(a_{uj}\right)}{\sum_{j \in \mathcal{N}_u(T)} \exp\left(a_{uj}\right)} \tag{14}$$

Above completes the definition of single-head graph attention. We further apply multi-head attention, as suggested by Velickovic *et al.* [8]. The multi-head attention mechanism performs K independent single-head attentions in parallel, i.e $h^{(i)}_u, i = 1, \cdots, K$. We use concatenation aggregator, $\parallel$ to aggregate these K outputs from K different heads, formulated as:

$$h_u \leftarrow \Big\|_K\, \sigma\left(\left(\sum_{j \in \mathcal{N}_u(T)} \tilde{a}_{uj}\, m_{uj}\right) \cdot W + b\right) \tag{15}$$

High-order attentive trust propagation. With the first-order trust propagation modeling, we can explore high-order social interactions by staking multiple attentive trust propagation layers. Such high-order propagation is crucial to encode comprehensive trust signals to estimate the trust strength between any pair of users. Concretely, by stacking l propagation layers, a user can receive the messages propagated from its l-hop neighbors.

To generalize the first-order trust propagation to high-order counterparts, we reformulate Eq. (10) and Eq. (13)-(15) as follows:

$$m^l_{uj} \leftarrow \left(h^{(l-1)}_j \parallel \Psi(t_{uj})^{l-1} \parallel e^c_{uj}\right) \tag{16}$$

$$a^l_{uj} \leftarrow m^l_{uj} W^l_{uj} \cdot \left(h^{(l-1)}_u W^l_u\right)^{\intercal} \tag{17}$$

$$\tilde{a}^l_{uj} \leftarrow \frac{\exp\left(a^l_{uj}\right)}{\sum_{j \in \mathcal{N}_u(T)} \exp\left(a^l_{uj}\right)} \tag{18}$$

$$h^l_u \leftarrow \Big\|_K\, \sigma\left(\left(\sum_{j \in \mathcal{N}_u(T)} \tilde{a}^l_{uj}\, m^l_{uj}\right) \cdot W^l + b^l\right) \tag{19}$$

where $\{W_{uj}^l, W_u^l, W^l, b^l\}_{l=1}^L$ are trainable parameters. To lighten the notations, we unify all of the trainable matrices (e.g., $\{W_{uj}^l, W_u^l, W^l\}_{l=1}^L$) as $\{W^l\}_{l=1}^L$. These parameters are different but shared across over the nodes, and will be jointly optimized with *Medley* automatically.

C. Trust Prediction

With the above attentive trust propagation, the rich information of time-varying social interactions is encoded into the user representations h^L. To learn the time-aware trust signals between any user pair, we first concatenate the representations of associated users, which refined with L-layer attentive trust propagation. Then we fit them to a multi-layer perception followed by a softmax layer. Formally, the trust signal is formulated as Eq. (20), where W_f is a trainable weight matrix defined in the multi-layer perception, and σ is the softmax function.

$$h_{uv} = \sigma\left(W_f \cdot \left(h_u^L \parallel h_v^L\right) + b_f\right) \tag{20}$$

Due to its simplicity and expressiveness, we use the concatenation operator to obtain the hidden factor of trust signal between two users. The use of concatenation is guided by recent works of graph convolutional neural networks [2], [6]. Without loss of generality, the outcome of this step is the probabilistic prediction of the trust strength. Then we can compute the trust strength by using $\tilde{h}_{uv} = \underset{j}{\operatorname{argmax}}(h_{uv})$. The detailed forward propagation algorithm of *Medley* is shown as **procedure Medley**.

D. Model Training

Guided by prior work [2], the objective function of *Medley* is defined as the cross-entropy loss between the predicted values and the ground-truth trust strength from the observed set $\mathcal{G}(T)$. Formally, it is formulated as:

$$\mathcal{L} = -\frac{1}{|\mathcal{E}(T)|} \sum_{\mathbf{e}_{uv} \in \mathcal{E}(T)} \log h_{uv,\mathbf{c}_{uv}} + \lambda \cdot ||\Theta||_2^2 \tag{21}$$

where $\Theta = \{\{W_l, b^l\}_{l=1}^L, W_f, b_f\}$ denotes all trainable model parameters, and λ controls the L_2 regularization strength to prevent over-fitting. In particular, we adopt Adam [16] as the optimizer in our implementation, as it has been shown to be effective in updating the model parameters [6].

procedure Medley: PREDICTING SOCIAL TRUST (I.E, FORWARD PROPAGATION)

Generate initial states of user representations for $\mathcal{G}(T)$
$h_u^0 \leftarrow x_u$, for all $u \in \mathcal{V}(T)$
▷ User representations
for all $u \in \mathcal{V}(T)$ **do**
  for $l = 1 \cdots L$ **do**
    for $i = 1 \cdots K$ **do**
      ▷ Attention weights
      for $j \in \mathcal{N}_u(T)$ **do**
        $m_{uj}^l \leftarrow \left(h_j^{(l-1)} \parallel \Psi(t_{uj})^{l-1} \parallel e_{uj}^c\right)$
        $a_{uj}^l \leftarrow m_{uj}^l W_{uj}^l \cdot \left(h_u^{(l-1)} W_u^l\right)^\intercal$
        $\tilde{a}_{uj}^l \leftarrow \frac{\exp\left(a_{uj}^l\right)}{\sum_{j \in \mathcal{N}_u(T)} \exp\left(a_{uj}^l\right)}$
      ▷ Attentive aggregation from local neighborhood
      $h_u^{l(i)} \leftarrow \sigma\left(\left(\sum_{j \in \mathcal{N}_u(T)} \tilde{a}_{uj}^l \, m_{uj}^{(l)}\right) \cdot W^l + b^l\right)$
    ▷ Multi-head aggregation
    $h_u^l \leftarrow \|_K \, \sigma\left(\left(\sum_{j \in \mathcal{N}_u(T)} \tilde{a}_{uj}^l \, m_{uj}^{(l)}\right) \cdot W^l + b^l\right)$
  ▷ Trust strength vector
for all $e_{uv} \in \mathcal{E}(T)$ **do**
  $h_u \leftarrow h_u^L$
  $h_v \leftarrow h_v^L$
  $h_{uv} = \sigma\left(W_f \cdot (h_u \parallel h_v) + b_f\right)$

TABLE II
STATISTICAL DESCRIPTION OF BITCOIN-ALPHA AND BITCOIN-OTC DATASETS.

DATASET	# OF NODES	# OF TRUST EDGES	# OF DISTRUST EDGES	AVE. DEGREE
BITCOIN-ALPHA	3,775	22,650	1,536	12.79
BITCOIN-OTC	5,881	32,029	3,563	12.1

V. EXPERIMENTAL RESULTS

A. Description of Datasets Used

Our experiments are conducted on two real-world and benchmarking datasets — Bitcoin-Alpha[4] and Bitcoin-OTC[5]. These two datasets are both coming from the sites that focus on having an open market in which users can make transactions using Bitcoins. As Bitcoin accounts are anonymous, these sites adopt the concept of "web-of-trust" to provide safety and security. Specifically, these sites allow users to positively (or negatively) rate others they trust (or distrust), for maintaining a record of users' reputation to prevent transactions with fraudulent and risky users. Each rating interaction is time-stamped, which we use to construct the time-varying interaction graph. We obtain these two datasets from [17], [18]. The statistics of the datasets are presented in Table II.

Data preparation. We split the time-stamped interactions chronologically into 70%-15%-15% for training, validation, and testing according to their timestamps. More precisely, the 30% of interactions were removed from the interactive network to compose the validation and test set[6]. The remaining 70% of interactions compose the observed social interaction graph. With such split, we ensure that the interactions in the

[4]https://btc-alpha.com/en/

[5]https://www.bitcoin-otc.com

[6]PGP and Advogato datasets, used in Guardian [2], are no longer applicable for evaluations in our context as the collected datasets are a snapshot of the network and do not contain timestamps.

validation and test set are post-hoc interactions, or "future" interactions, relative to the observed social graph or training set.

These social interaction graphs are time-varying, where users/edges are typically changing over time. To reflect the time-varying dynamics of these social graphs, we define the users in the training set as *observed users*. The new users appearing during validation or testing period are called *unobserved users*. Typically the new users show up during validation and testing may not have many interactions among themselves. To contain a certain amount of "future" interactions created by newly added users for validation and testing, we randomly pick 10% of users for each dataset, mask them in the training period, and treat them as unobserved users by only considering their interactions during validation and testing. This process ensures that the appropriate amount of future interactions among the unobserved users will show up during validation and testing.

TABLE III
TESTING ACCURACY ON BITCOIN-ALPHA ON OBSERVED USERS (%)

METHODS	AUC	F1-MICRO	F1-WEIGHTED	AP
Medley-IP	**92.7**	**92.9**	**92.4**	**98.3**
Medley-CAT	90.1	91.6	90.5	97.6
GUARDIAN	66.9	84.6	77.6	91.7

TABLE IV
TESTING ACCURACY ON BITCOIN-ALPHA ON UNOBSERVED USERS (%)

METHODS	AUC	F1-MICRO	F1-WEIGHTED	AP
Medley-IP	**90.4**	**91.9**	**90.9**	**97.9**
Medley-CAT	88.0	91.5	90.1	97.4
GUARDIAN	65.5	86.1	79.7	92.0

B. Experimental Settings

Baseline for comparisons. Since few approaches are capable of managing deep learning on social trust prediction in time-varying settings, we have no baseline to have a comprehensive comparison. An alternative option is Guardian [2], the state-of-the-art method on deep learning-based social trust prediction. Guardian [2] was proposed to characterize social trust using the notion of localized graph convolutions [6]. This method focuses on a particular snapshot of the social interaction graph and does not consider its time-varying features. Comparing them is not expected. However, for effectiveness demonstration, we use Guardian [2] as our baseline and compare the prediction performance of *Medley* with it. For a fair comparison, we use the same data preparation for Guardian, splitting the data chronologically for training, validation and testing, and evaluating the observed users and unobserved users, respectively. The main difference is that the timestamps of the interactions are removed from the data, as they do not utilize the data features. We do not include NeuralWalk [1], as previous work (Guardian [2]) has shown its superiority over this method.

As we mentioned in Sec. IV-B, to model the importance of an interaction between two users, the inner product and concatenation are alternative attention operators. To see how different attention operators affect the prediction performance, we instantiate the attentive trust propagation of *Medley* with the inner product and concatenation operator, respectively. For simplicity, we call them *Medley-IP* and *Medley-CAT* accordingly.

Evaluation metrics. As for the evaluation metrics, we employ three standard metrics to measure the prediction accuracy, including the area under the ROC curve (AUC), F-measure, and the average precision (AP). Specifically, due to the label imbalance in the datasets, we report F1-micro and F1-weighted values for the F-measure. Among others, F1-weighted value is an alternative F-measure metric that accounts for label imbalance. Note that larger values of these metrics indicate better prediction accuracy. Unless otherwise stated, all experiments are run 5 times to ensure statistical significance. Specifically, we report the average values over 5 runs, as commonly did in the literature [2].

All our experiments are performed on a machine with Intel E5-2650 v4 Broadwell 24-core 2.2GHz CPU, NVIDIA P100 Pascal GPU with 12GB HBM2 memory and 800GB SSD.

Parameter settings. We implemented our proposed framework in PyTorch[7]. Guided by prior work, node2vec [13] was used to generate the initial embeddings for each user[8]. The embedding dimension was fixed to 100 for all datasets. In terms of hyperparameters, we applied a grid search for hyperparameters: the learning rate was tuned amongst $\{0.0001, 0.001, 0.01\}$, the attention head was in $\{1, 2, 3, 4, 5\}$, the time dimension was in $\{20, 40, 60, 80, 100, 120\}$, and the coefficient of L_2 normalization was searched in $\{10^{-5}, 10^{-4}\}$. We used the Xavier initializer [19] and the Adam SGD optimizer [16] for all models. In addition, the maximum epoch was set as 50, and early stopping strategy was performed. Concretely, we terminate the training process if the validation AUC does not increase for 3 successive epochs. Without specification, we report the results under the hyperparameters with the best performance overall — 2 attentive trust propagation layers with 2 attention heads, the time dimension of 100, a dropout ratio of 0.1, the learning rate of 0.0001, and normalization coefficient of 10^{-5}. For the detailed parameter settings of Guardian, refer to [2].

C. Performance Comparisons

The dataset Bitcoin-Alpha is used to evaluate the performance of different approaches. We report the testing accuracy on observed users (seen during training period) and unobserved users (unseen during training period), respectively. The results are reported in Table III and Table IV. *Medley* offers the best AUC with 38.0% improvement on unobserved users as

[7]https://pytorch.org

TABLE V
TESTING ACCURACY ON BITCOIN-OTC ON OBSERVED USERS (%)

METHODS	AUC	F1-MICRO	F1-WEIGHTED	AP
Medley-IP	**72.2**	**86.9**	83.7	**93.3**
Medley-CAT	69.0	**86.9**	**83.8**	92.0
GUARDIAN	66.0	85.9	80.4	91.6

TABLE VI
TESTING ACCURACY ON BITCOIN-OTC ON UNOBSERVED USERS (%)

METHODS	AUC	F1-MICRO	F1-WEIGHTED	AP
Medley-IP	**73.3**	86.9	**84.3**	**93.6**
Medley-CAT	69.7	**87.2**	**84.3**	92.3
GUARDIAN	66.7	86.1	80.7	92.0

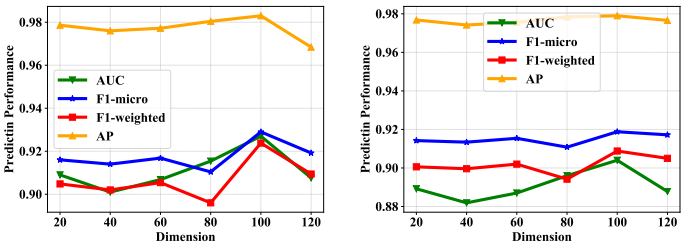
compared to Guardian — the state-of-the-art solution — and even higher improvement on observed users, about 38.6%. The increases in performance are significant. In terms of F-measure and AP, both *Medley-IP* and *Medley-CAT* achieve substantial improvement, which implies the powerful learning capability of our attentive trust propagation mechanisms.

To test that *Medley* does not rely on datasets, we also evaluated our framework on Bitcoin-OTC. As shown in Table V and Table VI, *Medley* consistently offers the best AUC by increasing the accuracy 8.6% for observed users and 9.0% for unobserved users. The results reported successfully verify that our proposed attentive trust propagation layers are able to characterize the latent factors of time-varying interactions to establish effective social trust.

Guardian is not able to offer comparable performance on both datasets, which indicates that ignoring the time-varying dynamics of social interactions may not be sufficient to capture the complex ties of trust among users. We also observe that the choice of attention operators is critical. In general, *Medley-IP* achieves the best performance on both datasets, which shows that the attention mechanism instantiated with the inner product operator effectively captures the time-varying interactions for social trust prediction. By comparing the prediction performance on observed users and unobserved users, e.g., Table III and Table IV, we can conclude that both *Medley* and Guardian can be generalized to unseen nodes/users, indicating that they inherit the inductive property of graph neural networks.

D. Parameter Analysis

We investigate how the prediction performance varies with the hyperparameters in our proposed framework. In particular, we focus on the dimension of the functional time encoding and the number of attention heads in this analysis. Unless otherwise stated, the reported results are averaged over 10 runs.



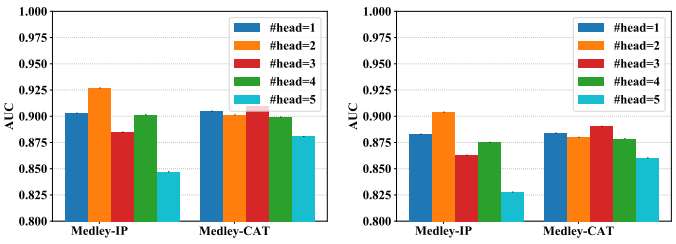
(a) Observed Users on Bitcoin-Alpha

(b) Unobserved Users on Bitcoin-Alpha

(c) Observed Users on Bitcoin-OTC

(d) Unobserved Users on Bitcoin-OTC

Fig. 4. Prediction Performance vs. Dimension of Time Encoding.



(a) Observed Users on Bitcoin-Alpha

(b) Unobserved User on Bitcoin-Alpha

(c) Observed Users on Bitcoin-OTC

(d) Unobserved Users on Bitcoin-OTC

Fig. 5. AUC Comparisons.

Dimension of time encoding. As stated in Eq. (8), the continuous-time domain is transformed into a vector space with dimension d_t. To investigate how the time dimension influences the prediction performance, we vary the time dimension from 20 to 120 with a step size of 20. The remaining

model setups are the default settings as reported in V-B when varying d_t. We identify that the performance trend is similar when the model is instantiated with different attention operators. Due to the space limitation, we report the results for the attention mechanism with the inner product operator. We observe that $d_t = 100$ achieves the best AUC overall. In terms of F-measure and the average precision, we observe slightly better prediction accuracy when increasing the dimension of the time features. It is possible to achieve better performance by carefully tuning other hyperparameters in different time dimension settings.

Number of attention heads. Another hyperparameter we analyze is the number of heads used for multi-head attention. We analyze the impacts on both attention mechanisms, one instantiated with the inner product and the other with the concatenation operator. We set the number of total hidden units as 100 and other parameters in default settings when varying the number of heads, for a fair comparison. Specifically, we vary the number of heads to be $1, 2, 3, 4, 5$, respectively. We report the AUC comparisons of *Medley-IP* and *Medley-CAT*, as shown in Fig. 5. We observe that *Medley* benefits from the multi-head mechanisms in general. More specifically, *Medley-CAT* achieves higher accuracy with 3 heads on Bitcoin-Alpha and 2 heads on Bitcoin-OTC, respectively. *Medley-IP*, equipped with a 2-head attention with the inner product operator, gives the best performance overall.

VI. Related Work

This section discusses two research lines relevant to our work: social trust evaluation and recent progress of graph neural network-based learning developed for graph-structured data.

A. Social Trust Evaluation

Social interactions among users are time-dependent, which leads to the dynamic evolution of trust in online social networks. Most of the existing works focus on the static social graph and do not explicitly capture the evolving nature of the online social networks. These models can be categorized into three categories: walk-based models, matrix factorization-based models, and deep learning models.

Walk-based models: ModelTrust [20] and TidalTrust [21] compute trust by searching the paths throughout the network. Multiple paths between users are aggregated to be the estimates of trust. AssessTrust [22] and OpinionWalk [23] model the value of trust using statistical distributions in three-valued subjective logic.

Matrix factorization-based models: Zheng *et al.*'s work [3] and Matri [4] are designed based on matrix factorization. User-user trust ties are analogous to user-item pairs in a recommender system. Matri [4] is designed to combine trust tendency and trust propagation under a collective matrix factorization framework, while Zheng *et al.*'s work further considers the similarity of users' trust rating habits.

Deep learning models: NeuralWalk [1] is designed to capture the trust propagation and aggregation rules using machine learning techniques. The core of this model is WalkNet, a neural network architecture designed to model single-hop trust propagation and aggregation. Guardian [2] is proposed to characterize social trust using the notion of localized graph convolutions [6].

To capture the evolving nature of the online social networks, PowerTrust [24] proposes to perform trust computation periodically to ensure that the computed trust values are up-to-date. PeerTrust [25] allows users to choose the temporal window for dynamic aging of old interactions/experiences as per their needs.

B. Graph Neural Networks

Graph Neural Networks (GNNs) have shown their effectiveness and obtained state-of-the-art performance on different graph tasks [8], [15], [26], [27], such as node classification, link prediction, and graph classification. There are several variants of GNNs, such as graph convolutional networks [15], graph attention networks [8], and graph isomorphism networks [28]; all share a similar feature learning strategy. For each node, GNNs refine its node features by aggregating the features from its neighbors and combining them with its own features. These GCN-based approaches consistently outperform techniques based upon matrix factorization or random walks (e.g, node2vec [13], Line [29], and DeepWalk [14]). Their success has led to a surge of interest in applying the architecture of GNNs to applications ranging from recommendation systems [30], drug design [31], to social trust prediction [2], [5].

The analysis of time-dependent social interaction is essential for understanding trust in time-varying online social networks. However, none of the existing work investigates time-aware trust prediction automatically. In this work, we argue that incorporating timing information can provide more compelling insights into how social trust is established. Therefore, we expect to deliver effect trust prediction based on past interactions in a time-varying online social network.

VII. Conclusion

We introduced a new framework, called *Medley*, to exploit time-dependent social interactions for trust prediction on time-varying online social networks. In this framework, we explicitly model users' time-varying latent factors to discover hidden and predictive time-aware trust signals in online social networks. By applying the functional time encoding, we represent the social interactions and the timing information into a latent space. The core of *Medley* is the attentive-trust propagation network, which is equipped with attention mechanisms to capture various importance weights for interactions that evolve over time. Extensive experiments on two real-world datasets have demonstrated the rationality and effectiveness of our proposed *Medley*. By incorporating topological structures that evolve over time, our framework can infer pairwise social trust based on past social interactions.

REFERENCES

[1] G. Liu, C. Li, and Q. Yang, "NeuralWalk: Trust Assessment in Online Social Networks with Neural Networks," in *Proc. INFOCOM*. IEEE, 2019.

[2] W. Lin, Z. Gao, and B. Li, "*Guardian*: Evaluating Trust in Online Social Networks with Graph Convolutional Networks," in *Proc. INFOCOM*. IEEE, 2020.

[3] X. Zheng, Y. Wang, M. A. Orgun, Y. Zhong, and G. Liu, "Trust Prediction with Propagation and Similarity Regularization," in *Proc. AAAI*, 2014.

[4] Y. Yao, H. Tong, X. Yan, F. Xu, and J. Lu, "Matri: A Multi-Aspect and Transitive Trust Inference Model," in *Proc. WWW*. ACM, 2013.

[5] W. Lin, "Social Media Analytics with Graph Convolutional Networks," Ph.D. Dissertation, University of Toronto, 2020.

[6] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive Representation Learning on Large Graphs," in *Proc. the Neural Information Processing Systems (NeurIPS)*, 2017.

[7] D. Xu, C. Ruan, E. Korpeoglu, S. Kumar, and K. Achan, "Self-Attention with Functional Time Representation Learning," in *Proc. the Neural Information Processing Systems (NeurIPS)*, 2019.

[8] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph Attention Networks," in *Proc. International Conference on Learning Representations (ICLR)*, 2018.

[9] L. H. Loomis, *Introduction to Abstract Harmonic Analysis*. Courier Corporation, 2013.

[10] A. Rahimi and B. Recht, "Random Features for Large-Scale Kernel Machines," in *Proc. the Neural Information Processing Systems (NeurIPS)*, 2008.

[11] M. McPherson, L. Smith-Lovin, and J. M. Cook, "Birds of A Feather: Homophily in Social Networks," *Annual Review of Sociology*, vol. 27, no. 1, pp. 415–444, 2001.

[12] P. V. Marsden and N. E. Friedkin, "Network Studies of Social Influence," *Sociological Methods & Research*, vol. 22, no. 1, pp. 127–151, 1993.

[13] A. Grover and J. Leskovec, "node2vec: Scalable Feature Learning for Networks," in *Proc. SIGKDD*. ACM, 2016.

[14] B. Perozzi, R. Al-Rfou, and S. Skiena, "DeepWalk: Online Learning of Social Representations," in *Proc. SIGKDD*. ACM, 2014.

[15] T. N. Kipf and M. Welling, "Semi-Supervised Classification with Graph Convolutional Networks," in *Proc. International Conference on Machine Learning (ICML)*, 2017.

[16] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," in *Proc. International Conference for Learning Representations (ICLR)*, 2015.

[25] L. Xiong and L. Liu, "PeerTrust: Supporting Reputation-Based Trust for Peer-to-Peer Electronic Communities," *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, vol. 16, no. 7, pp. 843–857, 2004.

[17] S. Kumar, F. Spezzano, V. Subrahmanian, and C. Faloutsos, "Edge Weight Prediction in Weighted Signed Networks," in *Proc. International Conference on Data Mining (ICDM)*. IEEE, 2016.

[18] S. Kumar, B. Hooi, D. Makhija, M. Kumar, C. Faloutsos, and V. Subrahmanian, "Rev2: Fraudulent User Prediction in Rating Platforms," in *Proc. International Conference on Web Search and Data Mining (WSDM)*. ACM, 2018.

[19] X. Glorot and Y. Bengio, "Understanding the Difficulty of Training Deep Feedforward Neural Networks," in *Proc. International Conference on Artificial Intelligence and Statistics*, 2010.

[20] P. Massa and P. Avesani, "Controversial Users Demand Local Trust Metrics: An Experimental Study on epinions.com Community," in *Proc. AAAI*, 2005.

[21] J. Golbeck, J. Hendler *et al.*, "FilmTrust: Movie Recommendations using Trust in Web-Based Social Networks," in *Proc. International Conference on Consumer Communications and Networking*. IEEE, 2006.

[22] 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 *Proc. INFOCOM*. IEEE, 2014.

[23] 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 *Proc. INFOCOM*. IEEE, 2017.

[24] R. Zhou and K. Hwang, "PowerTrust: A Robust and Scalable Reputation System for Trusted Peer-to-Peer Computing," *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, vol. 18, no. 4, pp. 460–473, 2007.

[26] M. Zhang and Y. Chen, "Link Prediction Based on Graph Neural Networks," in *Proc. the Neural Information Processing Systems (NeurIPS)*, 2018.

[27] W. Lin, Z. Gao, and B. Li, "Shoestring: Graph-Based Semi-Supervised Classification With Severely Limited Labeled Data," in *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 4174–4182.

[28] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How Powerful are Graph Neural Networks?" in *Proc. International Conference on Learning Representations (ICLR)*, 2018.

[29] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "Line: Large-Scale Information Network Embedding," in *Proc. WWW*. ACM, 2015.

[30] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec, "Graph Convolutional Neural Networks for Web-Scale Recommender Systems," in *Proc. SIGKDD*. ACM, 2018.

[31] M. Zitnik, M. Agrawal, and J. Leskovec, "Modeling Polypharmacy Side Effects with Graph Convolutional Networks," *Bioinformatics*, vol. 34, no. 13, pp. i457–i466, 2018.