CS-Trust2Vec: Context-Sensitive Trust Network Embedding Model

Onat Özdemir
onat.ozdemir@metu.edu.tr
Middle East Technical University
Ankara, Turkey

Pınar Karagöz*
pinark@metu.edu.tr
Middle East Technical University
Ankara, Turkey

İsmail Hakkı Toroslu*
toroslu@metu.edu.tr
Middle East Technical University
Ankara, Turkey

ABSTRACT

Trust network embedding is a network modeling task aiming to map nodes in social trust networks to low dimensional vector space. Although there are trust network embedding models proposed in the literature, these models focus on the topological structure of the trust network and disregard additional supportive information such as the context of trust. Context-specificity is an often-mentioned property of the trust, and it has been studied well in sociological and psychological terms. However, it has not been elaborated on within computational trust modeling. In this paper, we propose a trust network embedding model (CS-Trust2Vec) that integrates the context information into the embedding process to obtain more accurate representations of the nodes in trust networks. Due to the lack of a dataset that contains context information, we also crawled a dataset and propose a method to generate context-labeled trust networks from datasets that contain contexts of trust relationships. We validate the efficiency and accuracy of our model using the crawled dataset on the link prediction task. Experiments indicate that the proposed model shows remarkable performance on the link prediction task, and integrating the context information into the embedding process increases the quality of the generated node embeddings.

KEYWORDS

social trust network, network embedding, link prediction

1 INTRODUCTION

The growth of e-commerce websites and social media platforms has increased interest in trust and reputation systems. Trust in such systems can be either implicitly inferred or explicitly given by users in the form of friendship/like or block/dislike requests. Furthermore, trust relationships between the users can be represented as a directed graph called trust network where nodes represent users, and edges represent the trust relationships between these users. Trust networks can be signed or unsigned depends on the existence of distrust relationships.

Analyzing and modeling trust in online social networks is crucial for many areas such as recommendation systems [15], malicious user detection [24], and fake news filtering [4]. Modeling trust in very large social networks is a hard problem due to the highly noisy nature of these networks. Online social networks span trust relationships from many different contexts, based on judgments of reliability, dependability, and competence [17].

This paper primarily focuses on the trust network embedding problem. Trust network embeddings are the low-dimensional vector representations of the users in online social networks. Once embeddings are obtained, hidden patterns and features of the network can be observed, and one can use them for many different subsequent tasks such as link prediction [6], edge sign prediction [23], and network visualization [7].

Previous studies proposed various trust network embedding models: SiNE [21] is a deep learning-based method that constructs its objective function based on Structural Balance Theorem [1], STNE [23] is a skip-gram based model which uses a special negative sampling strategy and embeds 16 different types of trust transfer patterns, SDGNN [9] is a graph neural network-based model which uses both Structural Balance Theorem and the Status Theorem [13] to build its objective function. Most of these models focus on the topological structure and the graph features of trust networks to generate embeddings and disregard the context property of the trust.

Context of trust defines on which topic a person trusts another person. In real life, trust relationships are generally formed in the context of a specific topic. For example, it is hard to directly deduce that a person trusting someone in sports will also trust in politics. Therefore, it is important to analyze each relationship in its own context. Although the context of trust has been studied well in sociological and psychological terms [18, 19], to the best of our knowledge, it has not been studied in computational trust modeling, especially in generating trust embedding.

In this paper, we study the trust network embedding problem. We are particularly interested in analyzing the impacts of context on trust modeling to fill the mentioned gap in the literature. Arguably, studying the context of trust is very challenging from many different aspects. For instance, during the dataset review, we observed that there is no trust network dataset that contains explicit context information of the trust relationships. As an additional research effort, we crawled a social network dataset that contains categorized trust relationships and propose an algorithm to generate a context-labeled trust network from datasets with explicit context information. Last but not least, we also propose a context-sensitive trust network embedding model that has a remarkable performance on the link prediction task.

*Equal contribution.

2 BACKGROUND

Although the focus of this paper is primarily on analysis of trust networks, since social trust networks are a special type of graph, our problem is directly related to the graph embedding problem.

Graph embedding is a well-studied network analysis problem, and many embedding techniques are proposed. The models in the literature can be classified under two approaches: shallow embedding and deep embedding.

2.0.1 *Deep Embedding.*

In deep embedding approach, for a given graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$, research objective is to find a function $f : \mathcal{V} \rightarrow \mathbb{R}^d$ that maps each node $v \in \mathcal{V}$ to d-dimensional vector space $\mathbb{R}^d$. After the parameters of the encoder function f are optimized, the embeddings of the nodes can be obtained by passing the feature vectors of the nodes to f as inputs.

With the emergence of the Deep Learning, many deep embedding models are proposed to solve the graph embedding problem. SiNE [21] is one of the deep learning-based undirected signed graph embedding models. Its objective function is based on the Structural Balance Theorem, and it uses node triplets to train its deep neural network. BESIDE [3] is another deep learning-based model. It uses the Structural Balance Theorem to model triads and the Status Theorem to model bridge edges in its objective function.

2.0.2 *Shallow Embedding.*

Shallow embedding models store the embeddings of the nodes in a $|\mathcal{V}| \times d$ embedding lookup table. Based on the objective function, the vector representations in the embedding lookup table are updated in each optimization step. Therefore, unlike deep embedding models, outputs of the shallow embedding models are directly the low dimensional vectorial representations of the nodes in the network. The main limitation of the shallow embedding models is that they are *inherently transductive* [8], which means these models can only generate embedding for the nodes that are currently present in the network.

With the success of the skip-gram method in the field of word embedding, the same approach is adapted to graph embedding problem. DeepWalk [16] and Node2vec [6] are two of the most popular shallow graph embedding models. While DeepWalk generates unbiased random walks, Node2vec has a biased random walk strategy to generate positive samples for each node in the network. These two models are designed for unsigned networks and hence have poor performances on the signed network embedding task.

3 RELATED WORK

Since the paper focuses on both trust network embedding problem and the context-specific nature of trust, this section is divided into two subsections. In the first subsection, we briefly introduce the sociological and psychological studies on the context nature of trust. In the second section, we provide information about the state-of-the-art trust network embedding models.

3.1 Context Specific Nature of Trust

Context-dependence is an often-mentioned property of the trust. Context of trust defines on which topic a person trusts another person.

In [18], this property is described as, "Is trust/reputation context dependent? If we trust a doctor when she is recommending a medicine it does not mean we have to trust her when she is suggesting a bottle of wine. The reputation as a good sportsman does not help if we are looking for a competent scientist. It seems clear that the answer is yes: trust and reputation are context dependent properties".

Although there are sociological and psychological studies [18, 19] on the context of trust, the only trust model that considers the context nature of trust we have found is [10]. In [10], the authors propose an algorithm to extract domain-aware trust networks from the Heterogeneous Trust Networks (HTN) to increase the performances of widely used trust propagation algorithms such as MoleTrust [14], and TidalTrust [5].

There are, however, studies on different context types such as social context [2, 4, 25] and location context [2]. However, context categorization is out of the scope of this study.

In addition to the lack of context-aware computational trust models, we could not find any trust datasets that contain contextual trust relationships. To overcome this issue, we collected our dataset, whose details can be found in subsection 6.1.

3.2 Trust Network Embedding

During the literature review, we have found two models outperformed others on the edge sign prediction task: STNE [23] and SDGNN [9]. On the edge sign prediction task, the models predict whether a given relationship is a trust or distrust relationship.

STNE is a skip-gram-based signed trust network embedding model. It uses Structural Balance Theorem during the random walking phase to generate negative samples in addition to positive ones and also considers how trust is transferred between nodes by embedding incoming and outgoing trust transfer patterns for each node.

SDGNN is a graph neural network-based signed network embedding model. Its loss function models sign of the edges, directions of the edges using the Status Theorem, and the triadic relationships using the Structural Balance Theorem.

We tested these models on widely used trust datasets such as Epinions, Slashdot, and WikiRfa. The details and the results of the analysis can be found in Appendix A.

4 PROBLEM DEFINITION

In the paper, many notations and symbols are used. To avoid confusions and misunderstandings, we listed commonly-used notations in Table 1.

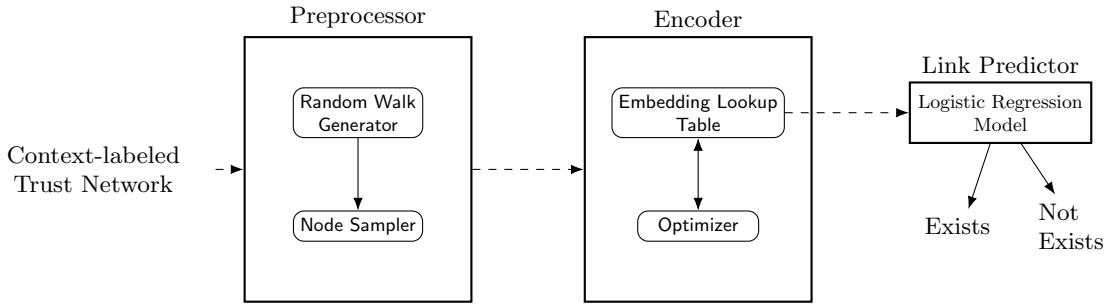

Figure 1: Architecture of the CS-Trust2Vec

Table 1: List of commonly-used notations

Notation	Description
d	Dimension of embedding
L_u^{out}/L_u^{in}	Outward/inward latent features of u
CT_u^{out}/CT_u^{in}	Outward/inward context tendency features of u
ϕ_{uv}	Context label of the trust relationship (u,v)
γ	Similarity measure for context labels
δ_u^+/δ_u^-	Positive/negative sample set of u
z_u	Embedding of node u
l_w	Maximum walk length
l_c	Length of the context window
n_w	Number of walks per node
n_{neg}	Number of negative samples per positive sample
n_{epoch}	Number of epochs

Definition 1. (*Directed Graph*). A directed graph is a graph where all the edges are directed from one node to another. Formally, it is a 4-tuple $\mathcal{G}(\mathcal{V}, \mathcal{E}, s, t)$ where

- $\mathcal{V}$ is set of nodes
- $\mathcal{E}$ is set of edges
- $s : \mathcal{E} \to \mathcal{V}$ is source function such that for a given $e \in \mathcal{E}$, $s(e)$ gives the source node of e.
- $t : \mathcal{E} \to \mathcal{V}$ is target function such that for a given $e \in \mathcal{E}$, $t(e)$ gives the target node of e.

Definition 2. (*Context-labeled Trust Network*). A context-labeled trust network is a directed graph that stores the context information of the trust relationships as edge attributes. Formally, it is a 6-tuple $\mathcal{G}(\mathcal{V}, \mathcal{E}, \mathcal{L}, s, t, \lambda)$ where

- $\mathcal{V}$ is set of nodes that represent the users in the network
- $\mathcal{E}$ is set of edges that represent the trust relationships between the users
- $\mathcal{L}$ is set of context labels
- $s : \mathcal{E} \to \mathcal{V}$ is source function such that for a given $e \in \mathcal{E}$, $s(e)$ gives the trustor of the trust relationship e.
- $t : \mathcal{E} \to \mathcal{V}$ is target function such that for a given $e \in \mathcal{E}$, $t(e)$ gives the trustee of the trust relationship e.
- $\lambda : \mathcal{E} \to \mathcal{L}$ is labeling function such that for a given $e \in \mathcal{E}$, $\lambda(e)$ gives the context label of e.

For a given $e \in \mathcal{E}$ where $s(e) = u$ and $t(e) = v$, $\lambda(e)$ is denoted as ϕ_{uv} in the following sections of this paper.

Definition 3. (*Trust Network Embedding Model*). For a given trust network $\mathcal{G}(\mathcal{V}, \mathcal{E}, s, t)$, a trust network embedding model is a function $f : \mathcal{V} \to \mathbb{R}^d$ that maps $u \in \mathcal{V}$ to d-dimensional vector z_u by preserving network properties.

Definition 4. (*Link Prediction*). Link prediction is the problem of deciding whether for a given (u, v) node pair, $(u, v) \in \mathcal{E}$ or $(u, v) \notin \mathcal{E}$.

Given a context-labeled trust network $\mathcal{G}(\mathcal{V}, \mathcal{E}, \mathcal{L}, s, t, \lambda)$, the problem is to generate d - dimensional vector representation $z_u \in \mathbb{R}^d$ for all $u \in \mathcal{V}$ such that a logistic regression model trained with the generated node embeddings achieves high accuracy on the link prediction task.

Our motivation is to design a trust network embedding model $f : \mathcal{V} \to \mathbb{R}^d$ that integrates $\mathcal{L}$ and λ into the embedding process to solve the trust network embedding problem.

5 PROPOSED MODEL

Our model consists of three modules: Preprocessor, Encoder, and Link Predictor. The architecture diagram of the model can be seen in Figure 1.

5.1 Preprocessor

5.1.1 *Random Walk Generator.*

To generate training samples, we propose a special biased random walk strategy. The purpose of the Random Walk Generator is to generate random walks using the proposed strategy.

For each node, the module generates n_w many random walks with a maximum length of l_w. The first edge in each walk is randomly chosen. Assuming that trust should be propagated between either the same or similar contexts, the following edges are randomly selected from a non-uniform probability distribution P^+ whose weights are determined by the similarities between the context label of the previously

chosen edge and the candidate edges. For a given previously chosen edge e_{prev}, probability of choosing e_x as the next edge is given in Equation 1 as

$$P^+(e_x \mid e_{prev}) = \frac{\gamma(\lambda(e_{prev}), \lambda(e_x))}{\sum_{e \in outgoing(u)} \gamma(\lambda(e_{prev}), \lambda(e))} \quad (1)$$

where $t(e_{prev}) = u$, $outgoing(u) = \{(u, v) \mid (u, v) \in \mathcal{E}\}$, and $e_x \in outgoing(u)$.

The pseudocode of the random walk generation algorithm can be found in algorithm 1.

5.1.2 *Node Sampler.*

The Node Sampler is used to generate positive and negative samples using the random walks generated by the Random Walk Generator to train the model.

Even if there is no edge between two nodes, if they are in the same random walk chain, using the Structural Balance Theorem [1] we can claim that they are positively related if the distance between them in the chain is shorter than l_c.

For each random walk, a context window of length l_c traverses the random walk, one node at a time. In each step, set of nodes covered by the context window are added to the positive sample set of the first node of the context window.

In Figure 2, node s is the first node of the context window, nodes pointed by the dashed arrows are added to the δ_s^+. In the next step, the same procedure will be applied to generate positive samples for node 2.

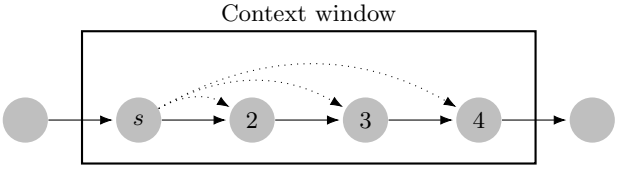

Figure 2: Positive sampling

In this paper, we focus on unsigned trust networks. Since unsigned trust networks do not contain distrust information, to generate negative samples, we use node pairs without any relationships. During the positive sampling, for each positive sample, $v \in \delta_u^+$, n_{neg} number of nodes that are not neighbours of the u are randomly chosen and added to the $\delta_u^-(v)$.

The pseudocode of the node sampling algorithm can be found in algorithm 2.

5.2 Encoder

5.2.1 *Embedding Lookup Table.*

Embeddings of the nodes in the network are stored in a $|\mathcal{V}| \times d$ matrix named "Embedding Lookup Table". Each node embedding consists of four different features L_u^{out}, L_u^{in}, CT_u^{out} and CT_u^{in}. Representation of the Embedding Lookup Table can be seen in Figure 3.

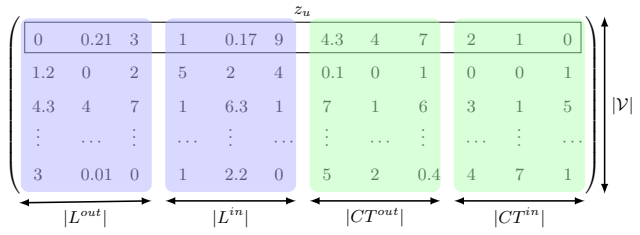

Figure 3: Representation of the Embedding Lookup Table

5.2.2 *Optimizer.*

In the optimizer module, the aim is to learn L_u^{out}, L_u^{in}, CT_u^{out} and CT_u^{in} features for all $u \in \mathcal{V}$. In our model, we adapt the objective function proposed in [23] to context-labeled trust networks.

Model uses the function given in Equation 2

$$f(u, v) = L_u^{out} \cdot L_v^{in} + (CT_u^{out} + CT_v^{in}) \cdot \phi_{uv} \quad (2)$$

to measure the u's tendency to trust v.
Function given in Equation 3 is the objective function for the given (u, v) pair

$$max\ J_{(u,v)} = h(u, v) + \sum_{i=1}^{n_{neg}} h'(u, v'^{(i)}) \quad (3)$$

where $v \in \delta_u^+$ and $v'^{(i)} \in \delta_u^-(v)$ and

$$h(u, v) = log\sigma(f(u, v))$$
$$h'(u, v') = log\sigma(-f(u, v'))$$

Objective function for the model is given in Equation 4.

$$max\ J = \sum_{u \in V} \sum_{v \in \delta_u^+} J_{(u,v)} \quad (4)$$

Since the negative samples are selected from non neighbour pairs, during the optimization of the negative pairs (u, v') context tendency parameters CT_u^{out} and $CT_{v'}^{in}$ are not updated.

Derivatives for the objective function Equation 3 are given in Equation 5, Equation 6, Equation 7, Equation 8, and Equation 9.

$$\frac{\partial J_{(u,v)}}{\partial L_u^{out}} = (1 - \sigma(f(u, v)))L_v^{in} - \sum_{i=1}^{n_{neg}} (1 - \sigma(-f(u, v'^{(i)})))L_u^{out} \quad (5)$$

$$\frac{\partial J_{(u,v)}}{\partial L_v^{in}} = (1 - \sigma(f(u, v)))L_u^{out} \quad (6)$$

$$\frac{\partial J_{(u,v)}}{\partial L_{v'^{(i)}}^{in}} = -(1 - \sigma(-f(u, v'^{(i)})))L_u^{out} \quad (7)$$

$$\frac{\partial J_{(u,v)}}{\partial CT_u^{out}} = (1 - \sigma(f(u, v)))\phi_{uv} \quad (8)$$

$$\frac{\partial J_{(u,v)}}{\partial CT_v^{in}} = (1 - \sigma(f(u, v)))\phi_{uv} \quad (9)$$

To optimize the model, we use Stochastic Gradient Ascent and update the parameters using the algorithm described in algorithm 3.

5.3 Link Predictor

The Link Predictor module contains a logistic regression model to predict whether there exists a trust relationship between the given node pair (u, v) or not.

Since the link prediction is a binary classification task, it is logical to utilize a logistic regression model as used in [9, 23]. Formally, the Link Predictor can be seen as a function $f : \mathbb{R}^{2d} \rightarrow \{0, 1\}$. For a given (u, v) node pair, $f(z_u \oplus z_v)$ is defined in Equation 10 as

$$f(z_u \oplus z_v) = \begin{cases} 0, & \text{predicts } (u, v) \notin \mathcal{E} \\ 1, & \text{predicts } (u, v) \in \mathcal{E} \end{cases} \tag{10}$$

where $\oplus$ is the concatenation operation.

Although there are different algebraic operations used in the literature [9, 23] such as Hadamard product, sum, and average, these operations are commutative. Since concatenation operation is non-commutative, $z_u \oplus z_v$ is expected to be different from the $z_v \oplus z_u$. Hence, the model becomes sensitive to the directions of the edges while making predictions.

6 EXPERIMENTS

6.1 Dataset

The dataset used in experiments consists of nearly 80k entries favorited by 10k users. Entries and user information are crawled from a Turkish online social media platform called Ekşi Sözlük [1].

We create an edge from the source user to the target user for each favorite entry to generate the trust network. The context of the created relationship is determined by the categories assigned to the topic of the entry. The statistics of the network can be seen in Table 2.

Table 2: Statistics of the network

# of Users	10069
# of Relationships	73468
# of Categories	28

For the relationships with a single category, context can be directly stored as a label of the category. However, if the relationships can have one or more categories as in the dataset we collected, we need a different approach to store context information.

In [10], the authors propose an algorithm to generate context-labeled trust networks from datasets that contain relationships with multiple categories. The paper basically suggests creating an edge between the source and the target user for each category that the relationship has. However, this method is not scalable since it causes the creation of an extensive number of edges when the dataset contains lots of possible category combinations.

In our model, for each relationship, we keep the context information in a k-length array where k is the number of different categories. For each category, there is a designated index in the array, and if the topic has that category, the value in the index assigned to that category is set to 1 and otherwise to 0. If there are multiple relationships between the user pair we apply vector addition to the context arrays of the relationships to obtain the final context label.

6.2 Experimental Setup

6.2.1 *Baselines.*

- **Node2vec :** Node2vec [6] is a widely-used skip-gram based unsigned graph embedding model. It uses a special biased random walk strategy to generate low dimensional vector representations obeying both homophily and structural equivalence.

- **Trust2Vec :** Trust2Vec is a non context-sensitive variation of the CS-Trust2Vec. Trust2Vec generates random walks using uniform distribution and does not have CT features. We use the Trust2Vec to analyse the impact of the context information on the link prediction task.

6.2.2 *Parameter Setting.*

For the baseline models and CS-Trust2Vec, parameters given in Table 3 are common. Hence, we use the same parameter values given in Table 3 during the evaluations of the baselines and CS-Trust2Vec.

Table 3: Parameter Settings for the experiments

Dimension of L	96
Dimension of CT	56
l_c	5
l_w	10
n_w	20
n_{epoch}	5
n_{neg}	3
γ	cosine similarity

We test the Node2vec with different p and q combinations. Although the best result is obtained when $p = 2$ and $q = 1$, the performance differences between the combinations are not remarkable.

6.3 Experiment Results

We evaluate the models on the link prediction task. The edges in the network are split as 80% training edges and 20% as test edges. We repeat the tests 5 times with different train-test splits and get the average of the scores to obtain the final results. Results of the link prediction task can be found in Table 4.

[1] https://eksisozluk.com/

Table 4: Performances of the models on the link prediction task

	Node2vec	CS-Trust2Vec	Trust2Vec
F1-Macro	0.615	**0.759**	0.744
AUC	0.654	**0.831**	0.820

As can be seen in Table 4, there is a remarkable accuracy difference between the results of the Node2vec and CS-Trust2Vec models. Furthermore, by looking at the results of the CS-Trust2Vec and Trust2Vec, we can claim that integrating context information into the embedding process increases the quality of the generated node embeddings.

6.4 Ablation Study

In this subsection, we do an ablation study on the embedding features of our model.

We test our model on three different cases: only using the Latent feature (L), only using the Context Tendency feature (CT), and using the combination of both embedding features (L + CT). During the experiment, the values given in Table 3 are used in settings of the parameters. Both in L and L + CT cases, we use random walks generated by the Random Walk Generator to obtain positive samples for the nodes and select the negative samples from the non-neighbor node pairs. However, since ϕ_{uv} information is required to optimize the CT feature, in the case where we just use the CT feature, only the edges in the network are used in the optimization of CT, and random walks are not generated. The results of the ablation study on embedding features can be seen in Table 5.

Table 5: Ablation study on embedding features

	L	CT	L + CT
F1-Macro	0.743	0.740	0.755
AUC	0.813	0.811	0.824

As can be seen in Table 5, the best F1-Macro and AUC scores are obtained when both L and CT features are used in the embedding process.

7 CONCLUSION

In this paper, we propose a trust network embedding model that integrates the context information of trust relationships into the embedding process and investigate the impacts of context information on the trust network embedding task. We validate the accuracy of our model on the link prediction task. Experiments indicate that using the context information in trust embedding models remarkably increases the accuracies of the models. In future work, we plan to extend our model for signed trust networks.

REFERENCES

[1] Dorwin Cartwright and Frank Harary. Structural balance: a generalization of heider's theory. *Psychological review*, pages 277–93, 1956.

[2] Yukuo Cen, Jing Zhang, Gaofei Wang, Yujie Qian, Chuizheng Meng, Zonghong Dai, Hongxia Yang, and Jie Tang. Trust relationship prediction in alibaba e-commerce platform. *IEEE Transactions on Knowledge and Data Engineering*, 32(5):1024–1035, 2020. doi: 10.1109/TKDE.2019.2893939.

[3] Yiqi Chen, Tieyun Qian, Huan Liu, and Ke Sun. "bridge": Enhanced signed directed network embedding. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, CIKM '18, page 773–782, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450360142. doi: 10.1145/3269206.3271738. URL https://doi.org/10.1145/3269206.3271738.

[4] Seyed Mohssen Ghafari, Shahpar Yakhchi, Amin Beheshti, and Mehmet Orgun. Social context-aware trust prediction: Methods for identifying fake news. In Hakim Hacid, Wojciech Cellary, Hua Wang, Hye-Young Paik, and Rui Zhou, editors, *Web Information Systems Engineering – WISE 2018*, pages 161–177, Cham, 2018. Springer International Publishing. ISBN 978-3-030-02922-7.

[5] Jennifer Ann Golbeck. *Computing and applying trust in web-based social networks*. PhD thesis, 2005.

[6] Aditya Grover and Jure Leskovec. Node2vec: Scalable feature learning for networks. KDD '16, page 855–864, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450342322. doi: 10.1145/2939672.2939754. URL https://doi.org/10.1145/2939672.2939754.

[7] Leonardo Gutiérrez-Gómez and Jean-Charles Delvenne. Unsupervised network embedding for graph visualization, clustering and classification, 2019.

[8] William Hamilton, Rex Ying, and Jure Leskovec. Representation learning on graphs: Methods and applications. 09 2017.

[9] Junjie Huang, Huawei Shen, Liang Hou, and Xueqi Cheng. Sdgnn: Learning node representation for signed directed networks, 2021.

[10] Cuiqing Jiang, Shixi Liu, Zhangxi Lin, Guozhu Zhao, Rui Duan, and Kun Liang. Domain-aware trust network extraction for trust propagation in large-scale heterogeneous trust networks. *Know.-Based Syst.*, 111(C):237–247, November 2016. ISSN 0950-7051. doi: 10.1016/j.knosys.2016.08.019. URL https://doi.org/10.1016/j.knosys.2016.08.019.

[11] Srijan Kumar, Francesca Spezzano, VS Subrahmanian, and Christos Faloutsos. Edge weight prediction in weighted signed networks. In *Data Mining (ICDM), 2016 IEEE 16th International Conference on*, pages 221–230. IEEE, 2016.

[12] Srijan Kumar, Bryan Hooi, Disha Makhija, Mohit Kumar, Christos Faloutsos, and VS Subrahmanian. Rev2: Fraudulent user prediction in rating platforms. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pages 333–341. ACM, 2018.

[13] Jure Leskovec, Daniel Huttenlocher, and Jon Kleinberg. Signed networks in social media. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, page 1361–1370, New York, NY, USA, 2010. Association for Computing Machinery. ISBN 9781605589299. doi: 10.1145/1753326.1753532. URL https://doi.org/10.1145/1753326.1753532.

[14] P. Massa and P. Avesani. Controversial users demand local trust metrics: An experimental study on epinions.com community. In *Proceedings of the National Conference on Artificial Intelligence*, volume 1, pages 121–126, 2005. Cited By :220.

[15] Makbule Gulcin Ozsoy and Faruk Polat. Trust based recommendation systems. In *2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2013)*, pages 1267–1274, 2013. doi: 10.1145/2492517.2500276.

[16] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk. *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, Aug 2014. doi: 10.1145/2623330.2623732. URL http://dx.doi.org/10.1145/2623330.2623732.

[17] Yi Qian and Sibel Adali. Foundations of trust and distrust in networks: Extended structural balance theory. *ACM Trans. Web*, 8(3), July 2014. ISSN 1559-1131. doi: 10.1145/2628438. URL https://doi.org/10.1145/2628438.

[18] Jordi Sabater and Carles Sierra. Review on computational trust and reputation models. *Artificial Intelligence Review*, 24(1): 33–60, September 2005. ISSN 0269-2821. doi: 10.1007/s10462-

004-0041-5. URL https://doi.org/10.1007/s10462-004-0041-5.

[19] Wanita Sherchan, Surya Nepal, and Cecile Paris. A survey of trust in social networks. *ACM Comput. Surv.*, 45(4), August 2013. ISSN 0360-0300. doi: 10.1145/2501654.2501661. URL https://doi.org/10.1145/2501654.2501661.

[20] Jiliang Tang, Huiji Gao, Huan Liu, and Atish Das Sarma. eTrust: Understanding trust evolution in an online world. In *KDD*, 2012.

[21] Suhang Wang, Jiliang Tang, Charu Aggarwal, Yi Chang, and Huan Liu. Signed network embedding in social media. In Nitesh Chawla and Wei Wang, editors, *Proceedings of the 17th SIAM International Conference on Data Mining, SDM 2017*, Proceedings of the 17th SIAM International Conference on Data Mining, SDM 2017, pages 327–335. Society for Industrial and Applied Mathematics Publications, 2017. doi: 10.1137/1.9781611974973.37.

[22] Robert West, Hristo S. Paskov, Jure Leskovec, and Christopher Potts. Exploiting social network structure for person-to-person sentiment analysis. *Transactions of the Association for Computational Linguistics*, 2:297–310, 2014. doi: 10.1162/tacl_a_00184. URL https://aclanthology.org/Q14-1024.

[23] Pinghua Xu, Wenbin Hu, Jia Wu, Weiwei Liu, Bo Du, and Jian Yang. Social trust network embedding. In *2019 IEEE International Conference on Data Mining (ICDM)*, pages 678–687, 2019. doi: 10.1109/ICDM.2019.00078.

[24] Jiahui Yu, Kun Wang, Peng Li, Rui Xia, Song Guo, and Minyi Guo. Efficient trustworthiness management for malicious user detection in big data collection. *IEEE Transactions on Big Data*, pages 1–1, 2017. doi: 10.1109/TBDATA.2017.2761386.

[25] Xiaoming Zheng, Yan Wang, Mehmet A. Orgun, Guanfeng Liu, and Haibin Zhang. Social context-aware trust prediction in social networks. In Xavier Franch, Aditya K. Ghose, Grace A. Lewis, and Sami Bhiri, editors, *Service-Oriented Computing*, pages 527–534, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.

A STNE AND SDGNN COMPARISON

We tested STNE and SDGNN models in five widely used trust datasets: Epinions [20] Slashdot [13], WikiRfa [22], Bitcoin-OTC [11, 12] and Bitcoin-Alpha [11, 12].

Models were tested in the Google Colab platform. The hardware setup used during the tests was Intel(R) Xeon(R) 2vCPU @ 2.3GHz, 13GB RAM, Nvidia Tesla T4 GPU. STNE was trained for one epoch using just the CPU, and SDGNN was trained for 100 epochs using T4 GPU. Hyperparameters were set using the recommended settings given in the papers.

Accuracy results of SDGNN for Epinions and Slashdot datasets could not be calculated since the execution time of the model exceeded the 12 hours time limit.

The F1-micro, F1-binary, F1-Macro, and AUC scores can be found in Table 6.

Table 6: Accuracy comparisons of STNE and SDGNN on the edge sign prediction task

		F1-micro	F1-binary	F1-macro	AUC
Epinions	STNE	0.932	0.961	0.851	0.926
	SDGNN	-	-	-	-
Slashdot	STNE	0.847	0.905	0.756	0.875
	SDGNN	-	-	-	-
WikiRfa	STNE	0.841	0.903	0.738	0.857
	SDGNN	0.863	0.915	0.785	0.893
Bitcoin-OTC	STNE	0.932	0.963	0.801	0.870
	SDGNN	0.929	0.960	0.793	0.912
Bitcoin-Alpha	STNE	0.950	0.973	0.740	0.858
	SDGNN	0.950	0.973	0.730	0.882

B ALGORITHM DETAILS

B.1 Random Walk Generation Algorithm

Algorithm 1: Random Walk Generation

Data: Context-Labeled Graph $\mathcal{G}(\mathcal{V}, \mathcal{E}, \mathcal{L}, s, t, \lambda)$, number of walks per node n_w, maximum walk length l_w
Result: list of random walks walkList
walkList = []
for *i in range*(n_w) **do**
  for $u \in \mathcal{V}$ **do**
    walk = $[u]$
    while $len(walk) < l_w$ **do**
      $outgoing(u) = \{(u, v) \mid (u, v) \in \mathcal{E}\}$
      if $outgoing(u) == \emptyset$ **then**
        break
      else if $len(walk) == 1$ **then**
        randomly choose an edge from $outgoing(u)$ as e_{next}
      else
        foreach $e \in outgoing(u)$ **do**
          $P^+(e) = \gamma(\lambda(e_{prev}), \lambda(e))$
        end
        normalize P^+ by dividing with total sum of weights
        randomly choose an edge from P^+ as e_{next}
      end
      add $t(e_{next})$ to walk
      $e_{prev} = e_{next}$
    end
    add walk to walkList
  end
end

B.2 Node Sampling Algorithm

Algorithm 2: Node Sampling

Data: Context-Labeled Graph $\mathcal{G}(\mathcal{V}, \mathcal{E}, \mathcal{L}, s, t, \lambda)$, walkList, length of the context window l_c, number of negative samples per positive sample n_{neg}
Result: positive sample set δ^+ and negative sample set δ^-

```
for walk in walkList do
    for i in range(walkLength - 1) do
        window = walk[i + 1 : min(i + l_c + 1, walkLength)]
        u = window[0]
        for v in window[1:] do
            add v to δ_u^+
            randomly choose n_neg number of nodes ∈ V such that for each chosen node v',
            (u, v') ∉ E and v' ∉ δ_u^+
            add the chosen nodes to δ_u^-(v)
        end
    end
end
```

B.3 Optimization Algorithm

Algorithm 3: Optimization

Data: Context-Labeled Graph $\mathcal{G}(\mathcal{V}, \mathcal{E}, \mathcal{L}, s, t, \lambda)$, positive sample set δ^+, negative sample set δ^-, L^{out}, L^{in}, CT^{out}, CT^{in}, and n_{epoch}
Result: Updated L^{out}, L^{in}, CT^{out}, and CT^{in}

```
for i in range(n_epoch) do
    Shuffle V
    for u ∈ V do
        for v ∈ δ_u^+ do
            Update L_u^out via Equation 5
            Update L_v^in via Equation 6
            if (u, v) ∈ E then
                Update CT_u^out via Equation 8
                Update CT_v^in via Equation 9
            for v' ∈ δ_u^-(v) do
                Update L_v'^in via Equation 7
            end
        end
    end
end
```