Robust Network Enhancement from Flawed Networks

Anonymous Author(s)

ABSTRACT

Real-world networks are often noisy or error-prone, which may harm the performance of network analysis or modeling. This paper solves the problem of reliable network reconstruction from a flawed network, namely network enhancement. Specifically, it aims to detect the noisy links and predict the missing links in a network simultaneously. We propose E-Net, an end-to-end graph neural network, to leverage the mutual influence of the two tasks to achieve both the goals more effectively. Because on one hand, detecting noisy links can benefit the accuracy of predicting missing links; and on the other hand, the performance of predicting missing links can provide indirect supervision for detecting noisy links when the labeled noisy links are unavailable. Besides, the model can be scaled up to large networks and can preserve the local and global structural characteristics following a proposed lazy subgraph extraction mechanism. The experimental results on several genres of large networks demonstrate the superiority of our model. Our implementation is available at: https://github.com/anonymous-Enet/E-Net.

KEYWORDS

network enhancement, robust model, social network

ACM Reference Format:

Anonymous Author(s). 2020. Robust Network Enhancement from Flawed Networks. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Networks are ubiquitous in domains including social network analysis, bioinformatics, chemistry, etc. They offer rich generic connectivity patterns that can help us understand the relational data. However, most real networks are error-prone and structurally flawed due to incomplete sampling [6], imperfect measurements [2], individual non-response and dropout [19], etc. Generally, the flawed structures can be caused by two kinds of flawed links: *noisy links* and *missing links*. Specifically, *noisy links* are those that are observed in the constructed networks but do not really exist in the real world (false positives). For example, in a mobile network, an occasional call between a delivery driver and a customer is more likely to be a noisy link than a actual "social" relationship. *Missing links* are indeed in the real world yet they are unobserved in the constructed network (false negatives). For example, if building links in a mobile network according to calling interactions, some infrequently interactive but actual "social" relationships will be discarded. The flawed networks may harm the performance of network analysis and modeling, which demands an effective way to reconstruct reliable networks from them—i.e., removing the noisy links and completing the missing links.

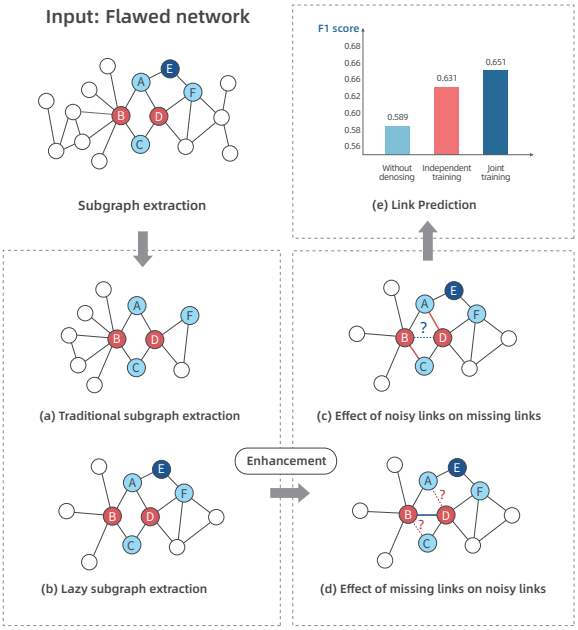


Figure 1: Illustration of predicting missing links and detecting noisy links in a unified framework. The dashed lines denote the missing links to be predicted or the noisy links to be detected given the existing links denoted by the solid lines.

One straightforward way to deal with the problem is to adopt the heuristic metrics, such as Common Neighbors, Jaccard Coefficient, Preferential Attachment, Adamic-Adar, etc. [15, 17, 28], to predict missing links and detect noisy links simultaneously, i.e., complete a missing link if the score measured by one of the above metrics is quite high and remove a noisy link if the score is significantly low. Some existing works proposed additional metrics such as node correlation [18] and link reliability [7] to identify the missing and noisy links in one framework. However, such unified measurement ignored the mutual influence of the two kinds of links. Take an example in Figure 1 to illustrate the necessity of this mutual influence. Figure 1(c) explains the influence of noisy links on the missing links. When predicting the missing link between node B and D, if the links A-D and B-C are detected as noisy links and removed beforehand, the connections between B and D will be weakened. Thus the likelihood of creating a link between B and D will be reduced. Figure 1(d) explains the influence of missing links on the noisy links. When detecting whether the link A-D or B-C is noisy or not, if the link B-D is predicted as a missing link and added beforehand, the connection between A and D or the connection between B and C will be strengthened according to the structural homophily, which

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

will decrease the likelihood of removing the link A-D or B-C. The example indicates that the mutual influence between the two tasks can boost the performance of each other. Thus, *how to capture the mutual influence of the missing links and the noisy links is the main challenge to be addressed in the paper.*

In addition, super large networks prevent us from leveraging the whole network to infer the missing or noisy links. Existing works directly extracted subgraphs consisting of one-hop or two-hop neighbors of queried nodes [3, 23, 25]. Such subgraphs may still be too large when hub nodes are traversed. For example in Figure 1(a), a large number of neighbors will be expanded when node B is traversed. In addition, the exactly expanded one-hop or two-hop neighbors lose global structural characteristics. As we can see, if A-D and B-C are detected as noisy links and removed before predicting the relationship between B and D, B and D will be disconnected in the one-hop subgraph. However, if node E which carries more global structure evidence is included in the subgraph (Cf. Figure 1(b)), there will be a path B-A-E-F-D that connects B and D even if A-D and B-C are removed. Thus, *how to extract a small-sized subgraph but carries both the local and graph structural characteristics together is another challenge to be dealt with.*

Our contributions can be summarized as follows:

- We propose to jointly identify the noisy links and the missing links in one framework. Specifically, we propose an end-to-end graph neural network model, named as the Enhanced Network model (abbreviated as E-Net) which can capture the mutual influence between the two tasks.
- We propose a lazy subgraph extraction approach, which enables our model to scale up to large networks while capture both the global and local structure characteristics.
- Extensive experiments on several networks demonstrate that our model can obtain 10.5% improvement in terms of F1 comparing with the model without denoising the networks, and obtain 3.1% improvement comparing with the method identifying the missing links and noisy links independently (Cf. Figure 1(e) for details).

2 PROBLEM FORMULATION

Here we introduce some definitions and formulate the problem.

Definition 1. ***Flawed Network.*** *In practice, we often obtain a network by sampling nodes and edges from a complete network or creating a network following some heuristic rules. Thus, it is inevitable that the obtained network will contain incorrect information, especially at the edge level (as obtaining a reliable node set is much easier)*[1]*. We define this kind of incomplete network as a flawed network, mainly containing two kinds of flawed links:* ***noisy links*** *which are observed in the network but do not exist in the real world, and* ***missing links*** *which indeed exist in the real world but are unobserved in the network.*

Formally, we represent a flawed network as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of nodes with $|\mathcal{V}|$ nodes, and $\mathcal{E}$ is the set of edges with $|\mathcal{E}|$ edges. We denote $\mathcal{A}$ as the adjacency matrix of $\mathcal{G}$ and $\mathcal{D}$ as the degree matrix of $\mathcal{A}$. We also augment $\mathcal{G}$ with a node attribute matrix $\mathcal{X}$. Based on the above definitions, we define the problem of network enhancement as follows:

[1]We target at flawed links and leave flawed nodes in the future.

Problem 1. ***Network Enhancement.*** *Given a flawed network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and a state matrix $\mathcal{Y} = [\mathcal{Y}_{ij}]_{i,j=1\ldots|\mathcal{V}|}$, where $\mathcal{Y}_{ij} \in \{1, 0, ?\}$ denote s confirmed existing, confirmed absent and uncertain links, our goal is to infer the uncertain links ($\mathcal{Y}_{ij} = ?$) in $\mathcal{Y}$, i.e., predict the missing links ($\epsilon_{ij} \notin \mathcal{E}$ and $\mathcal{Y}_{ij} = ?$) and detect the noisy links ($\epsilon_{ij} \in \mathcal{E}$ and $\mathcal{Y}_{ij} = ?$). The two tasks of missing link prediction and noisy link detection comprise the network enhancement problem.*

When solving network enhancement problem, even though it is hard to obtain both the labels of missing links and noisy links, it is not persuasive to learn the missing links and the noisy links in a complete unsupervised way. To reduce the difficulty of learning, we suppose the labels of missing links ($\epsilon_{ij} \notin \mathcal{E}$ and $\mathcal{Y}_{ij} = 1$) can be obtained in some way (e.g., we hold out some existing links as the missing links), while the labels of noisy links ($\epsilon_{ij} \in \mathcal{E}$ and $\mathcal{Y}_{ij} = 0$) are not available. Then we leverage the labels of missing links to supervise both the missing link prediction and the noisy link detection, i.e., the links satisfy $\epsilon_{ij} \notin \mathcal{E}$ and $\mathcal{Y}_{ij} = 1$ serve as the training data of our model, where the noisy link detection is encapsulated as an component in the model without direct supervisions. Formally, we use q to denote two queried nodes with observed label which indicates whether the two nodes should be linked or not, and use $Q = \{q^{(m)}\}_{m=1}^{M}$ to denote all the labeled queries in $\mathcal{G}$. In the following parts, a symbol with superscript $^{(m)}$ denotes that it corresponds to the m-th query.

3 OUR APPROACH

In this section, we first introduce the proposed subgraph extraction mechanism, which paves the way for the subsequent network enhancement model. Then given a query q and its corresponding extracted subgraph G, we encapsulate the missing link prediction and noisy link detection in E-Net, an end-to-end graph neural network, to capture the mutual influence of the two objectives.

3.1 Lazy Subgraph Extraction

Direct computation on the entire network is costly in general. We thus aim to extract a local subgraph for each pair of queried nodes. Some works have verified that computing on local subgraphs can well approximate a wide range of heuristic network metrics within a bounded error [1, 23, 25]. However, most of them simply extracted a fixed-size subgraphs or kept the whole one-hop or two-hop neighbors, totally ignored the global structure. Moreover, fixed-hop subgraphs may still be too large when hub nodes are traversed. Other sampling methods such as GraphSAGE [8] uniformly sampled neighbors, thus ignored the reliability of links.

We propose a lazy subgraph extraction approach to capture the reliable local and global structural characteristics through random walks with restart. Given a pair of the queried nodes, instead of keeping all the fixed-hop neighbors, we conduct several random walks with restart from each of the two queried nodes. The probability transition matrix can be expressed as $\mathbf{P} = (1-\lambda)\mathbf{I} + \lambda\mathcal{A}\mathcal{D}^{-1}$, where $\mathcal{D}$ is the degree matrix with diagonal value $\mathcal{D}_{ii} = \sum_j \mathcal{A}_{ij}$, $\mathbf{I}$ is the identity matrix, the teleport (or restart) probability $\lambda \in (0, 1]$ controls the probability of staying at the current node or jumping to a random neighbor, which enables preserving both the local and global topological structures. This is also similar to personalized Pagerank, in which $\mathbf{I}$ characterizes the nodes' personalized preferences and is filled with some real values [9].

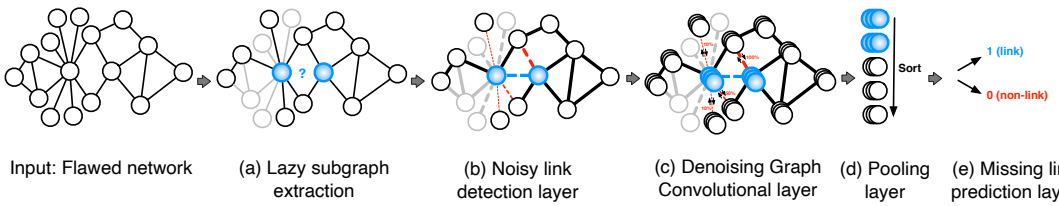


Figure 2: Model structure of E-Net. The dashed lines denote the uncertain links ($\mathcal{Y}_{ij}$ =?) and the thickness of links denotes their reliability.

We also suggest extracting multiple subgraphs for each query to avoid overfitting. That is, for m-th query $q^{(m)} \in Q$, we extract c subgraphs and form them as a set of <subgraph, query> pair, i.e., $\{<G_1^{(m)}, q^{(m)}>, <G_2^{(m)}, q^{(m)}>, ..., <G_c^{(m)}, q^{(m)}>\}$. The resulting $M \times c$ pairs for all the M queries constitute our training data, where the subgraphs can be viewed as a kind of structural evidence. Also, this kind of extraction can situate our model in an inductive setting, which can be used to predict links on unseen subgraphs.

3.2 E-Net

In E-Net, we aim to jointly predict the missing link and detect the noisy links in an extracted subgraph (Cf. Figure 2). The input subgraph and each layer of E-Net are introduced here.

The Input Subgraph. Let $G = (V, E)$ be a subgraph of a query q extracted from $\mathcal{G}$ with node set $V = \{v_1, \ldots, v_n\}$ and edge set $E = \{e_{ij} | v_i, v_j \in V\}$. The adjacency matrix is denoted as $\mathbf{A} \in \mathbb{R}^{n \times n}$. The attribute matrix is denoted as $\mathbf{X}$, where $\mathbf{x}_i$ is the attribute vector of v_i. We also record the relative position of each node in G to the query q as additional features by applying Double-Radius Node Labeling (DRNL) [25], as the nodes closer to the queried nodes will provide more important evidence. Specifically, for the two queried nodes v_i and v_j, we assign label $t = 1$ to them respectively. Then, for any node $v_k \in G$ with $(d(v_k, v_i), d(v_k, v_j)) = (1, 1)$, we assign label $t = 2$, where $d(\cdot, \cdot)$ is the shortest path between two nodes. Nodes with double-radius $(1, 2)$ or $(2, 1)$ get label $t = 3$ and nodes with $(2, 2)$ get $t = 4$. So on and so forth. The position labels $\mathbf{t}_i$ of all the nodes in G comprise the relative position matrix $\mathbf{T}$.

Noisy link detection layer. How to efficiently choose heuristic scores with different properties for unsupervised noisy link detection is a long-standing problem since no supervision can be given. Thus we tactfully utilize the missing links to provide indirect supervision. We use a *noise scoring function*, $s(\cdot, \cdot)$, to measure link reliability. The lower the score is, the more likely the link will be noisy. In this paper we instantiate the function as a weighted average of K heuristic score functions:

$$s(v_i, v_j) = \sum_{k=1}^{K} w_k \cdot s_k(v_i, v_j), \tag{1}$$

where $s_k(\cdot, \cdot)$ is the k-th heuristic score to estimate the node similarity and w_k is its corresponding learnable weight, which can be indirectly guided by the objective of predicting missing links. We use Common Neighbors, Jaccard Coefficient, Preferential Attachment, Adamic-Adar, Resource Allocation and cosine similarity of two nodes' attributes as these heuristic scores to guarantee the detection capability. Thus, given the noise scoring function $s(\cdot, \cdot)$ of each link, the subgraph G can be transformed into a denoised weight matrix $\mathbf{A}^* \in \mathbb{R}^{n \times n}$, in which the (i, j)-th entry $\mathbf{A}^*_{ij} = s(v_i, v_j) * \mathbf{A}_{ij}$.

Denoising graph convolutional layers. The denoising graph convolutional layers is a generalization of the classical graph convolutional network. The general idea is to reduce or prevent the messages aggregated from noisy links when performing node aggregation. Although Graph Attention Network (GAT) [21] also puts different attentions on neighbors, the attention contribution of node i on node j is different from that of node j on node i. And it was experimentally verified that GAT performed the same or worse than GCN in noisy graphs [29]. This is because GAT introduces too many parameters, which may result in overfitting to noise. In our problem, the influence of node i on node j is assumed to be equivalent to that of node j on node i, as noisy links exhibit global influence on the whole graph. Thus, the denoised weight matrix $\mathbf{A}^*$ can be considered as a global weight function when conducting node aggregation. We then define the following denoising graph convolution layer for calculating the forward-pass update of node:

$$Z^{l+1} = f\left(\frac{\tilde{\mathbf{A}}^*}{\left\|\tilde{\mathbf{A}}^*\right\|_2} Z^l W^l\right), \tag{2}$$

where $\tilde{\mathbf{A}}^* = \mathbf{A}^* + \mathbf{I}$, $Z^l \in \mathbb{R}^{n \times d^l}$ is the outputs of l-th graph convolution layer and $Z^0 = [\mathbf{X}; \mathbf{T}]$, $W^l \in \mathbb{R}^{d^l \times d^{l+1}}$ is a layer-specific graph convolution parameters, and f is a nonlinear activation function. With $\mathbf{A}^*$, the information flow along the noisy links will be decreased or prevented, which helps us build more robust model. Considering the over-smoothing problem [13, 14], we further concatenate the outputs of all graph convolution layers to get a high-quality one, i.e., $Z^{1:L} := \left[Z^1, ..., Z^L\right]$, where L is the number of graph convolution layers and $Z^{1:L} \in \mathbb{R}^{n \times \sum_1^L d^l}$.

Pooling layer. Since our goal is to predict the existence of a link between the two queried nodes, the graph-level representation should 1) capture the local structures of the queried nodes and 2) emphasize the two queried nodes. First, we adopt SortPooling [26] to get the graph-level representation, where the dimension of the last denoising graph convolutional layer d^L is fixed as 1. We sort all the nodes in the subgraph according to the continuous output values $Z^L \in \mathbb{R}^{n \times 1}$ and concatenate the node embeddings of top $\bar{n}$ nodes as $Z \in \mathbb{R}^{\bar{n} \times \sum_1^L d^l}$. Since the attributes of queried nodes can provide additional information, we further add them on and obtain the final graph embedding as $\boldsymbol{H} = [\mathrm{CONV}(Z); \mathbf{x}_q^{(1)}, \mathbf{x}_q^{(2)}]$, where

$\mathbf{x}_q^{(1)}$ and $\mathbf{x}_q^{(2)}$ are the node attributes of two queried nodes, CONV denotes several 1-D convolutional layers and max-pooling layers applied on the concatenated hidden vectors $\boldsymbol{Z}$.

Missing link prediction layer. Finally, we predict the link between the two queried nodes based on the obtained subgraph representation, i.e., $\hat{\mathcal{Y}}_q = \text{softmax}(\text{MLP}(\boldsymbol{H}))$, where MLP denotes a multilayer perceptron followed by a softmax.

3.3 Model Learning

Main objective. We optimzie the cross-entropy loss function $\mathcal{L}_{missing} = -\sum_{m=1}^{M}\sum_{b=1}^{d^L} \mathcal{Y}_{q^{(m)}b} \cdot \ln \hat{\mathcal{Y}}_{q^{(m)}b}$, where $\mathcal{Y}_{q^{(m)}}$ and $\hat{\mathcal{Y}}_{q^{(m)}}$ are the ground truth and the predicted probability corresponding to the query $q^{(m)}$ respectively.

Auxiliary denoising objectives. Recall that the denoised weight matrix $\mathbf{A}^*$ is only guided by $\mathcal{L}_{missing}$, which is difficult to learn. So we add an additional regularization to minimize the graph Laplacian quadratic form on node attribute $\mathbf{X}$:

$$\mathcal{L}_{denoise} = tr(\mathbf{X}^T\mathbf{L}^*\mathbf{X}),\ \mathbf{X}^T\mathbf{L}^*\mathbf{X} = \sum_{e_{ij}\in E}(\mathbf{x}_i - \mathbf{x}_j)^2\mathbf{A}^*_{ij}, \quad (3)$$

where $\mathbf{X}^T\mathbf{L}^*\mathbf{X}$ is the Laplacian quadratic form on signal $\mathbf{X}$, i.e., a smooth graph signal model. It has been widely used in various learning problems (e.g., regularization, semi-supervised learning on graphs) [20, 27]. We explain why we adopt it as follows:

1) Global Smoothing. This objective enables two nodes with similar attributes close to each other in the denoised network $\mathbf{A}^*$. According to Eq.(3), if v_i and v_j have similar attributes, the link e_{ij} is less likely to be a noisy one, which forces $\mathbf{A}^*_{ij}$ to be a large value.

2) Sparsity. In most scenarios, it is desirable that the obtained graph is sparse [4, 16]. Here, we also want the denoised network $\mathbf{A}^*$ to be sparse, i.e., some entries of $\mathbf{A}^*_{ij}$ can be reduced to zero if edge e_{ij} is highly probable to be a noisy one. According to [4], we know that:

$$\sum_{i\neq j}\mathbf{A}^*_{ij} = n - \text{tr}(\mathbf{A}^*) = \text{tr}\left(\mathbf{I} - \mathbf{A}^*\right) = \text{tr}(\mathbf{L}^*). \quad (4)$$

Since the node attribute matrix $\mathbf{X}$ is observed, minimizing the trace of graph Laplacian quadratic form $tr(\mathbf{X}^T\mathbf{L}^*\mathbf{X})$ in Eq.(3) is equivalent to minimizing $tr(\mathbf{L}^*)$. Thus, Eq.(3) can also be considered as a sparsity penalty of the denoised weight matrix, which exactly matches our sparsity objective.

Joint loss function. $\mathcal{L} = \mathcal{L}_{missing} + \alpha\mathcal{L}_{denoise}$, where $\alpha > 0$ is a trade off between $\mathcal{L}_{missing}$ and $\mathcal{L}_{denoise}$.

4 EXPERIMENTS

Datasets. We explain the datasets as below.

1) Mobile network. We use a large-scale mobile network provided by PPDai Company[2], spanning one year from January 1st, 2018, to January 1st, 2019. We create a link if two users (nodes) made a call, and obtain user's demographic information (including age, sex, birthplace, educational level and so on) as user attributes. Fortunately, PPDai Company provides us some information of anomalous nodes (i.e., identified as fraudsters, intermediaries or delivery drivers), so we can intuitively choose those calls from anomalous nodes with short call duration or call times as our noisy link ground truth. And we randomly remove 10% of links to serve as our missing link samples. The PPDai network containing the simulated missing links and the identified noisy links is the input flawed network $\mathcal{G}$, and after removing them, we get a real clean network $\mathcal{G}_{\text{clean}}$.

2) Citation network. Cora, Citeseer and Pubmed are used. These networks are well-constructed and widely used; therefore, we assume that each of them is error-free, which serves as our clean network $\mathcal{G}_{\text{clean}}$. To quantify the ability to identify flawed links, we generate observed flawed networks by randomly removing links (creating the missing link samples), and randomly adding nonexistent links (constituting the noisy link samples). After adding the simulated noisy links and missing links, we get the input flawed network $\mathcal{G}$.

Baselines. We compare against several baselines: 1) *HEU.* A logistic regression classifier based on five popular heuristics (HEU), including Common Neighbors (CN), Jaccard Coefficient, Preferential Attachment (PA), Adamic-Adar (AA) and Resource Allocation (RA), which only takes graph structure into consideration; 2) *ATT.* A MLP classifier based on the attributes of two query nodes without consideration of graph structures; 3) *ENS.* An ensemble (ENS) of node attributes and all heuristic scores, following a MLP classifier; 4) *Node2vec [5].* A network embedding method, which learns latent topological features from network structures; 5) *SEAL [25].* A link prediction method based on GNNs, also applied on the extracted subgraphs; 6) *SEAL (lazy).* SEAL using lazy subgraph extraction for ablation study. 7) *GAT (lazy).* GAT based on lazy subgraphs. 8) *E-Net (fix).* A variant E-Net which removes the mutual influence between noisy links and missing links. Specifically, we fix all w_k as the same value (averaged of all heuristic scores) rather than training the noise scoring function $s(\cdot,\cdot)$; 9) *E-Net (s-).* Another variant E-Net, where the auxiliary denoising regularization is removed.

Our flawed link candidates in the test set contains all the links with $\mathcal{Y}_{ij}$ =?, where we control the ratio of real and fake missing links as well as the ratio of real and fake noisy links as around 1:5. We set train/validation/test size as 0.8:0.1:0.1. The reported results are averaged over 10 runs.

4.1 Experimental Results

Missing Link Prediction. Table 1 shows the performance of missing link prediction on all the datasets. From the results, we can see that our model consistently achieves the best or the second best performance on all the datasets (+6.2% in terms of AUC and +9.7% in terms of F1). Compared with *HEU*, *ATT* and *ENS* that only using node attributes and heuristics measures, our model can discover new structural and node-specific attributes. Compared with latent feature based methods *Node2vec* and *SEAL*, our model can prevent the messages propagated along flawed links. Moreover, *SEAL* even fails on PPDai dataset due to its high memory and time cost when meeting with lots of hub nodes. The performance gain of *E-Net* with respect to *E-Net (fix)* is correlated with the mutual influence between noisy links and missing links.

Noisy Link Detection. The noisy link detection is conducted based on a list of observed links ranked according to their noisy scores $s(\cdot,\cdot)$. We compare against some unsupervised measures: 1)*ATT*, a cosine similarity between two query nodes; 2)*CN, Jaccard, PA, AA, RA*, five heuristic measures; 3)*ENS*, the mean of above six

[2]PPDai, the leading online lending platforms in China.

Table 1: Experimental results with standard deviation on missing link prediction.

	PPDai		Cora		Citeseer		Pubmed	
	AUC	F1	AUC	F1	AUC	F1	AUC	F1
HEU	0.593 ± 0.04	0.345 ± 0.08	0.745 ± 0.03	0.561 ± 0.15	0.692 ± 0.06	0.509 ± 0.09	0.639 ± 0.05	0.424 ± 0.14
ATT	0.860 ± 0.04	0.598 ± 0.07	0.570 ± 0.02	0.286 ± 0.02	0.584 ± 0.07	0.286 ± 0.05	0.820 ± 0.03	0.500 ± 0.03
ENS	0.834 ± 0.06	0.603 ± 0.07	0.754 ± 0.05	0.420 ± 0.05	0.720 ± 0.06	0.398 ± 0.07	0.813 ± 0.02	0.489 ± 0.02
Node2vec	0.736 ± 0.05	0.443 ± 0.06	0.741 ± 0.09	0.406 ± 0.09	0.707 ± 0.09	0.389 ± 0.10	0.908 ± 0.07	0.609 ± 0.08
SEAL	-	-	0.781 ± 0.10	0.496 ± 0.17	0.772 ± 0.03	0.521 ± 0.05	0.962 ± 0.01	0.750 ± 0.03
SEAL (lazy)	0.892 ± 0.03	0.615 ± 0.03	0.808 ± 0.12	0.589 ± 0.18	0.793 ± 0.05	0.552 ± 0.06	**0.969 ± 0.02**	0.751 ± 0.04
GAT (lazy)	0.884 ± 0.04	0.608 ± 0.03	0.795 ± 0.12	0.535 ± 0.17	0.791 ± 0.06	0.548 ± 0.06	0.932 ± 0.04	0.642 ± 0.05
E-Net (fix)	0.918 ± 0.01	0.697 ± 0.02	0.890 ± 0.02	0.631 ± 0.04	0.871 ± 0.01	0.598 ± 0.03	0.952 ± 0.01	**0.762 ± 0.02**
E-Net (s-)	0.921 ± 0.02	0.703 ± 0.03	**0.898 ± 0.01**	**0.651 ± 0.04**	**0.883 ± 0.01**	0.607 ± 0.04	0.943 ± 0.01	0.720 ± 0.01
E-Net	**0.930 ± 0.03**	**0.712 ± 0.03**	0.891 ± 0.02	0.633 ± 0.03	0.875 ± 0.03	**0.614 ± 0.05**	0.947 ± 0.01	0.741 ± 0.01

Table 2: Precision score on noisy link detection. Note that we set the number of selected links (i.e., the denominator) as the number of real noisy links, which means precision equals to recall, F1 and accuracy metric under this setting.

	PPDai	Cora	Citeseer	Pubmed
ATT	0.173	0.139	0.143	0.166
CN	0.297	0.297	0.267	0.297
Jaccard	0.288	0.288	0.242	0.288
PA	0.218	0.218	0.117	0.218
AA	0.241	0.241	**0.271**	0.241
RA	0.237	0.237	0.260	0.273
ENS	0.144	0.237	0.117	0.156
NE	-	0.117	0.051	0.024
E-Net	**0.348**	**0.319**	0.248	**0.298**

Table 3: Running time (s/epoch) comparison of different subgraph extraction approaches.

	PPDai	Cora	Citeseer	Pubmed
Enclosing	>2days	108.9	354.2	>1days
Lazy	**650.0**	**70.8**	**170.1**	**2013.2**

Table 4: Experimental results on missing link prediction with different subgraph extraction approaches.

	Cora		Citeseer	
	AUC	F1	AUC	F1
Enclosing	0.883	0.621	0.858	0.584
Lazy	**0.891**	**0.633**	**0.875**	**0.614**

scores; 4)*NE* [22], whose denoised edge weights are taken as noisy scores. For evaluation, we adopt a standard metric [18], defined as the ratio of true positive noisy links to the number of selected links. We pick up the last C links according to $s(\cdot,\cdot)$ (serve as selected links) and C_n is the number of truly observed noisy links in them. Thus precision equals C_n/C. Table 2 shows that our method can maintain a stable performance across all datasets, while the other baselines sometimes perform poorly on certain datasets and *NE* even can not deal with PPDai due to its computational complexity. The performance gap between *E-Net* and *ENS* suggests that the missing link objective does provide useful guidance for learning more reasonable weights rather than the average.

Table 5: Performance of node classification on Cora.

Networks	Macro-F1	Micro-F1	Weighted-F1
Flawed network $\mathcal{G}$	0.613	0.629	0.626
Enhanced network $\mathcal{G}^*$	**0.662**	**0.677**	**0.676**
Clean network $\mathcal{G}_{\text{clean}}$ (**Ground Truth**)	0.778	0.786	0.785

The Effect of Lazy Subgraph Extraction. We compare with the *Enclosing* method [24], which directly extracted 2-hop neighbors of query nodes. Table 3 presents the running time of two different subgraph extraction approaches both applied on *E-Net*. Table 4 shows the effectiveness of two subgraph extraction approaches (we do not evaluate enclosing version on Pubmed and PPDai due to their expensive space and time cost), which demonstrates the lazy version's superior performance. All of these suggest that in practical applications, the lazy version is performed as a fast and accurate approach to extract subgraphs, which is especially advanced for large graphs and graphs containing lots of hub nodes, as in PPDai.

Node Classification on the Enhanced Network. Given an input flawed network $\mathcal{G}$, we will reconstruct the enhanced network $\mathcal{G}^*$ via *E-Net*, and further compare with the real clean network $\mathcal{G}_{\text{clean}}$ by taking node classification as our downstream task. Specifically, when constructing $\mathcal{G}^*$, we first remove the links in noisy link candidate whose ranking is in last C according to $s(\cdot,\cdot)$, where we set C as the number of real noisy links, and then add the missing links predicted by our model in the missing link candidate set. We then apply node2vec [5] (adopting their default parameters) to get node embedding and follow a MLP for label prediction. We set train:test split as 0.8:0.2. Table 5 reports the comparison between the performance on different networks constructed based on Cora, which demonstrates that enhancing the flawed network based on our model is really helpful for downstream task here.

5 CONCLUSION

In this work, we study the problem of network enhancement, and propose E-Net, an end-to-end GNN model that considers the mutual influence between noisy links and missing links. We further propose a lazy subgraph extraction method to reduce computational cost. We demonstrated the significancy of our methods on various datasets. Our future directions include analysing the model robustness and defense strategy under network adversarial attacks on graphs.

REFERENCES

[1] Ziv Bar-Yossef and Li-Tal Mashiach. 2008. Local Approximation of Pagerank and Reverse Pagerank. In *CIKM*. 279–288.

[2] Carter T. Butts. 2003. Network inference, error, and informant (in)accuracy: a Bayesian approach. *Social Networks* 25, 2 (2003), 103 – 140.

[3] Yen-Yu Chen, Qingqing Gan, and Torsten Suel. 2004. Local Methods for Estimating Pagerank Values. In *CIKM*. 381–389.

[4] Giorgio Gnecco, Rita Morisi, and Alberto Bemporad. 2015. Sparse Solutions to the Average Consensus Problem via Various Regularizations of the Fastest Mixing Markov-Chain Problem. *IEEE Transactions on Network Science and Engineering* 2, 3 (July 2015), 97–111.

[5] Aditya Grover and Jure Leskovec. 2016. Node2Vec: Scalable Feature Learning for Networks. In *Proceedings of the 22th International Conference on Knowledge Discovery and Data Mining (KDD'16)*. 855–864.

[6] Kossinets Gueorgi. 2006. Effects of missing data in social networks. *Social Networks* 28, 3 (2006), 247 – 268.

[7] Roger Guimerà and Marta Sales-Pardo. 2009. Missing and spurious interactions and the reconstruction of complex networks. *PNAS* 106, 52 (Dec 2009), 22073–22078.

[8] William L. Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. In *Proceedings of the 31th International Conference on Neural Information Processing Systems (NIPS'17)*.

[9] Glen Jeh and Jennifer Widom. 2003. Scaling Personalized Web Search. In *WWW*. ACM, New York, NY, USA, 271–279.

[10] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR'15)*.

[11] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*.

[12] Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. 2018. Predict then Propagate: Graph Neural Networks meet Personalized PageRank. In *Proceedings of the 6th International Conference on Learning Representations (ICLR'18)*.

[13] Qimai Li, Zhichao Han, and Xiao Ming Wu. 2018. Deeper Insights into Graph Convolutional Networks for Semi-Supervised Learning. In *AAAI*. AAAI.

[14] Yujia Li, Richard Zemel, Marc Brockschmidt, and Daniel Tarlow. 2016. Gated Graph Sequence Neural Networks. In *ICLR* (proceedings of iclr'16 ed.).

[15] David Liben-Nowell and Jon Kleinberg. 2003. The Link Prediction Problem for Social Networks. In *Proceedings of the 12th ACM International Conference on Information and Knowledge Management (CIKM'03)*. 556–559.

[16] F. Lin, M. Fardad, and M. R. Jovanović. 2012. Identification of sparse communication graphs in consensus networks. In *Allerton*. 85–89.

[17] Linyuan Lü, Ci-Hang Jin, and Tao Zhou. 2009. Similarity index based on local paths for link prediction of complex networks. *Physical review. E, Statistical, nonlinear, and soft matter physics* 80 (10 2009), 046122.

[18] Liming Pan, Tao Zhou, Linyuan Lü, and Chin-Kun Hu. 2016. Predicting missing links and identifying spurious links via likelihood analysis. *Scientific Reports* 6 (03 2016), 22955.

[19] Joseph Schafer and John Graham. 2002. Missing Data: Our View of the State of the Art. *Psychological Methods* 7 (06 2002), 147–177.

[20] Alexander J. Smola and Risi Kondor. 2003. Kernels and Regularization on Graphs. In *Learning Theory and Kernel Machines*. Springer Berlin Heidelberg, Berlin, Heidelberg, 144–158.

[21] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *ICLR*.

[22] Bo Wang, Armin Pourshafeie, Marinka Zitnik, Junjie Zhu, Carlos D. Bustamante, Serafim Batzoglou, and Jure Leskovec. 2018. Network enhancement: a general method to denoise weighted biological networks. *Nature Communications* (2018).

[23] Jia Xu, Hongyan Liu, Zou Li, Jun He, Xiaoyong Du, and Yuanzhe Cai. 2010. Local Methods for Estimating SimRank Score. In *APWeb*. 157–163.

[24] Muhan Zhang and Yixin Chen. 2017. Weisfeiler-Lehman Neural Machine for Link Prediction. In *Proceedings of the 23rd International Conference on Knowledge Discovery and Data Mining (KDD'17)*. ACM, New York, NY, USA, 575–583.

[25] Muhan Zhang and Yixin Chen. 2018. Link Prediction Based on Graph Neural Networks. In *Proceedings of the 32th International Conference on Neural Information Processing Systems*. Curran Associates Inc., USA, 5171–5181.

[26] Muhan Zhang, Zhicheng Cui, Marion Neumann, and Yixin Chen. 2018. An End-to-End Deep Learning Architecture for Graph Classification. In *AAAI*.

[27] Dengyong Zhou and Bernhard Schölkopf. 2004. A Regularization Framework for Learning from Graph Data. In *Proceedings of the 19th International Conference on Machine Learning (ICML'04)*.

[28] Tao Zhou, Linyuan Lü, and Yi-Cheng Zhang. 2009. Predicting missing links via local information. *The European Physical Journal B* 71, 4 (Oct 2009), 623–630.

[29] Dingyuan Zhu, Ziwei Zhang, Peng Cui, and Wenwu Zhu. 2019. Robust Graph Convolutional Networks Against Adversarial Attacks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '19)*. Association for Computing Machinery, New York, NY, USA, 1399–1407. https://doi.org/10.1145/3292500.3330851

A APPENDIX

A.1 Notations

In most cases, we use lower-case letters for scalars (e.g., l), upper-case letters for sets (e.g., V and E), bold lower-case letters for column vectors (e.g., $\mathbf{x}$), and bold upper-case letters for matrices (e.g., $\mathbf{A}$). When indexing the matrices, $\mathbf{A}_{ij}$ denotes the element at the i-th row and the j-th column, and $\mathbf{A}_i$ denotes the vector at the i-th row. The major notations used in our proposed model can be found in Table 6.

Table 6: Description of some major notations

Notation	Description
$\mathcal{G}$, $\mathcal{G}^*$	The flawed network and the enhanced network
$\mathcal{V}$	The node set of $\mathcal{G}$
$\mathcal{E}$, ϵ_{ij}	The edge set of $\mathcal{G}$ and its corresponding edge
$\mathcal{Y}$, $\mathcal{Y}_{ij}$	The true state matrix and state label of ϵ_{ij}
$\hat{\mathcal{Y}}$	The predicted state matrix
$\mathcal{X}$	The node attribute matrix of $\mathcal{G}$
$\mathcal{A}$	The adjacency matrix of $\mathcal{G}$
$\mathcal{D}$	The degree matrix of $\mathcal{A}$
G	The input local subgraph
V, v_i	The node set of G and its corresponding node
E, e_{ij}	The edge set of G and its corresponding edge
$\mathbf{T}$, $\mathbf{t}_i$	The relative position matrix and the position label of v_i
$\mathbf{X}$, $\mathbf{x}_i$	The node attribute matrix of G and the attribute vector of v_i
$\mathbf{A}$, $\mathbf{A}^*$	The adjacency matrix and the denoised weight matrix of G
$\mathbf{L}^*$	The Laplacian matrix of A^*
$\mathbf{P}$	The probability transition matrix of lazy random walk
$s(\cdot,\cdot)$	The noise scoring function
q, $q^{(m)}$	The query of the two nodes and the m-th query
Z^l	The output of the l-th denoising graph convolution layer
$\boldsymbol{H}$	The subgraph representation
d^l	The output dimension of the l-th graph convolution layer
L	The number of graph convolution layer

A.2 Dataset Details

Detailed dataset statistics can be found in Table 7.

Table 7: Dataset statistics

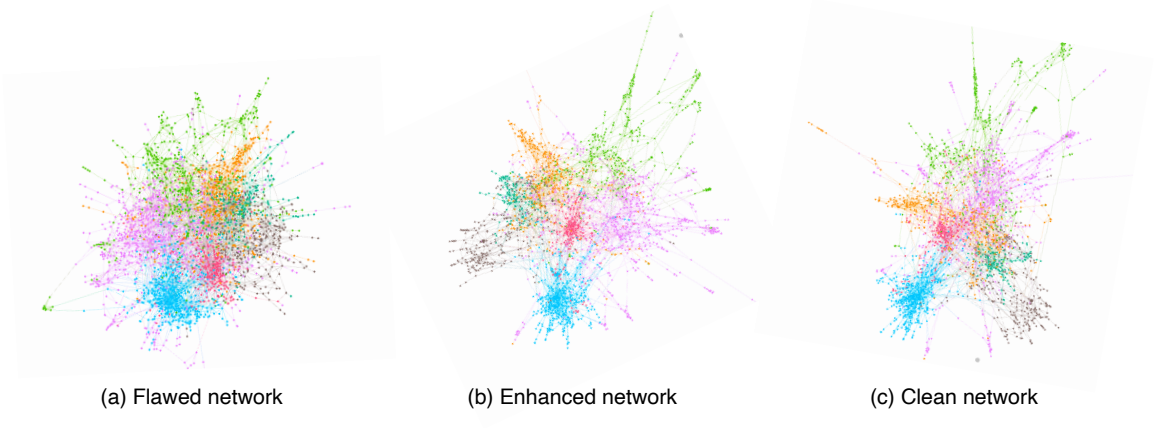
Dataset	Type	#Nodes	#Edges	#Attributes	Noisy Rate	Missing Rate
PPDai	Mobile network	83,286	114,422	776	10.6%	10%
Cora	Citation network	2,708	5,429	1,433	10%	10%
Citeseer	Citation network	3,327	4,732	3,703	10%	10%
Pubmed	Citation network	19,717	44,338	500	10%	10%

A.3 Experimental Details

We implemented all the models in Pytorch with the Adam optimizer for optimization [10]. Early stopping strategy is implemented if the performance ceases to improve or only improves in a small range ($1e$-3) for 7 successive epochs on the validation set. All experiments are conducted on a single machine with an Intel Xeon E5 and one NVIDIA TITAN GPU.

A.4 Visualization of the enhanced network

Figure 3 further visualizes the networks, where the enhanced network exhibits more discernible clustering compared with the input flawed network, and is more similar to the clean network.



(a) Flawed network (b) Enhanced network (c) Clean network

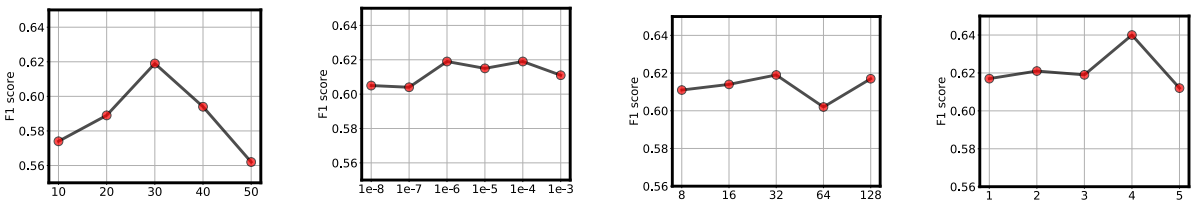
Figure 3: Visualization of Cora networks, where different colors present different node categories.

A.5 Parameter Analysis

In this section, we analyze four crucial hyper-parameters, which are the number of lazy walks when extracting subgraph n_{lazy}, the sparsity coefficient α, the hidden dimension of GNN layers d^l (here, we let all GNN layers have the same dimension) and the number of GNN layers L respectively.

In order to further determine the optimal parameters settings, we vary the values of n_{lazy}, α, d^l and L to better observe how the performance of missing link prediction will change. In detail, we study $n_{\text{lazy}} \in \{10, 20, 30, 40, 50\}$, $\alpha \in \{1e\text{-}8, 1e\text{-}7, 1e\text{-}6, 1e\text{-}5, 1e\text{-}4, 1e\text{-}3\}$, $d_l \in \{8, 16, 32, 64, 128\}$, $L \in \{1, 2, 3, 4, 5\}$ respectively. It should be noted that the the number of lazy walks n_{lazy} remains at 30 while studying α, d^l and L, the sparsity coefficient α remains at $1e$-4 while studying n_{lazy}, d^l and L, the hidden dimension of GNN layers d^l remains at 32 while studying n_{lazy}, α and L, and the number of GNN layers L remains at 4 while studying n_{lazy}, α and d^l. Here, we present some suitable values of these parameters for reference. We report the F1 score on Cora dataset in the task of missing link prediction in Figure 4. As shown, remarkably, we can see that E-Net can achieve relatively good performance regardless of parameters changing (still better than the baseline methods), which answers the question **Q5**.

More specifically, from Figure 4(a), we observe that our model yields good results when n_{lazy} equals to 30. It can be explained that few local structural information is captured when n_{lazy} is set to be small, thus can not provide us enough information, while too many observations would in turn prevent us from capturing the most important part of graph structure. From Figure 4(b), we can see that we can maintain our results greater than 0.61 when α is set to be larger than or equal to $1e$-6, which suggests the necessity of our auxiliary denoising objective. From Figure 4(c), the results seem less sensitive to the hidden dimension of GNN layers. From Figure 4(d), the performance becomes better on the whole if more GNN layers are used, and it only slightly drops after stacking 5 layers. This makes intuitive sense, since GNN with deeper layers give the model more capacity to represent graphs. However, when the layer number is too much (equals to 5), the performance starts to drop slightly. This observation is due to the fact that GNN tends to be over smoothing after stacking too many layers, which is consistent with many conclusions in previous works [11–13]. Since we concatenate all the outputs of each GNN layers to generate final high-quality node representation, which avoids this problem to some extent.



(a) n_{lazy}: # of lazy walks (b) α: sparsity coefficient (c) d^l: hidden dimension of GNN (d) L: # of GNN layers

Figure 4: Hyper-parameter analysis, where the y-axes are fixed in the same range.