Node Representation Learning for Directed Graphs

Megha Khosla ✉, Jurek Leonhardt, Wolfgang Nejdl, and Avishek Anand

L3S Resaerch Center, Hannover, Germany
{khosla,leonhardt,nejdl,anand}@l3s.de

Abstract. We propose a novel approach for learning node representations in directed graphs, which maintains separate views or embedding spaces for the two distinct node roles induced by the directionality of the edges. We argue that the previous approaches either fail to encode the edge directionality or their encodings cannot be generalized across tasks. With our simple *alternating random walk* strategy, we generate role specific vertex neighborhoods and train node embeddings in their corresponding source/target roles while fully exploiting the semantics of directed graphs. We also unearth the limitations of evaluations on directed graphs in previous works and propose a clear strategy for evaluating link prediction and graph reconstruction in directed graphs. We conduct extensive experiments to showcase our effectiveness on several real-world datasets on link prediction, node classification and graph reconstruction tasks. We show that the embeddings from our approach are indeed robust, generalizable and well performing across multiple kinds of tasks and graphs. We show that we consistently outperform all baselines for node classification task. In addition to providing a theoretical interpretation of our method we also show that we are considerably more robust than the other directed graph approaches.

Keywords: Directed Graphs · Node Representations · Link Prediction · Graph Reconstruction · Node Classification

1 Introduction

Unsupervised representation learning of nodes in a graph refers to dimensionality reduction techniques where nodes are embedded in a continuous space and have dense representations. Such node embeddings have proven valuable as representations and features for a wide variety of prediction and social network analysis tasks such as link prediction [14], recommendations [25], vertex label assignment, graph generation [7] etc.

However most of the recent node embedding methods have been focused on undirected graphs with limited attention to the directed setting. Often valuable knowledge is encoded in directed graph representations of real-world phenomena where an edge not only suggests relationships between entities, but the directionality is often representative of important asymmetric semantic information.

Prime examples are follower networks, interaction networks, web graphs, and citation networks among others.

Most of the approaches in this regime [22, 8, 19] focus on the goal of preserving neighborhood structure of nodes when embedding one space into another, but suffer from some key limitations when representing directed graphs. First, most of these node embedding techniques operate on a single embedding space and distances in this space are considered to be symmetric. Consequently, even though some of the approaches claim to be applicable for directed graphs, they do not respect the **asymmetric roles** of the vertices in the directed graph. For example, in predicting links in an incomplete web graph or an evolving social network graph, it is more likely that a directed link exists from a less popular node, say `Max Smith`, to a more popular node, say an authoritative node `Elon Musk`, than the other way around. Algorithms employing single representations for nodes might be able to predict a link between `Elon Musk` and `Max Smith` but cannot predict the direction.

Secondly, approaches like APP [26] overcome the first limitation by using two embedding spaces but are unable to differentiate between directed neighborhoods where these neighborhoods can be distinguished based on reachability. For example, for a given node v there exists a neighborhood which is reachable from v and there exists another type of neighborhood to which v is reachable. More acutely, many nodes with zero outdegree and low indegree might not be sampled because of the training instance generation strategy from its random walk following only outgoing edges. This renders such approaches **not to be robust**, a desirable and important property for unsupervised representations, for several real-world graphs.

Finally, works like HOPE [17] rely on stricter definitions of neighborhoods dictated by proximity measures like Katz [9], Rooted PageRank etc. and cannot be generalized to a variety of tasks. In addition, they do not scale to very large graphs due to their reliance on matrix decomposition techniques. Moreover, the accuracy guarantees of HOPE rely on low rank assumption of the input data. Though not completely untrue for real world data, *singular value decomposition*(SVD) operations used in matrix factorization methods are known to be sensitive even for the case of a single outlier [2]. We later empirically demonstrate in our experiments that HOPE can not be easily adapted for the node classification task as it is linked to a particular proximity matrix.

We argue that the utility and strength of unsupervised node representations is in their (1) robustness across graphs and (2) flexibility and generalization to multiple tasks and propose a simple yet robust model for learning node representations in directed graphs.

Our Contribution. We propose a robust and generalizable approach for learning **N**ode **E**mbeddings **R**especting **D**irectionality (NERD) for directed and (un)–weighted graphs. NERD aims at learning representations that maximize the likelihood of preserving node neighborhoods. But unlike the previous methods, it identifies the existence of **two** different types of node neighborhoods; one in its source role and the other in its target role. We propose an *alternating random*

walk strategy to sample such node neighborhoods while preserving their respective role information. Our alternating walk strategy is inspired from SALSA [13] which is a stochastic variation of the HITS [11] algorithm and also identifies two types of important nodes in a directed network: *hubs* and *authorities*. Roughly speaking, the paths generated with our alternating random walks alternate between hubs (source nodes) and authorities (target nodes), thereby sampling both neighboring hubs and authorities with respect to an input node. From a theoretical perspective we derive an equivalence for NERD's optimization in a matrix factorization framework. In addition, we also unearth the limitations of earlier works in the evaluation of models on directed graphs and propose new evaluation strategies for Link Prediction and Graph Reconstruction tasks in directed graphs. Finally we perform exhaustive experimental evaluation that validates the robustness and generalizability of our method.

2   Related Work

Traditionally, undirected graphs have been the main use case for graph embedding methods. Manifold learning techniques [3], for instance, embed nodes of the graph while preserving the local affinity reflected by the edges. Chen et al. [6] explore the directed links of the graph using random walks, and propose an embedding while preserving the local affinity defined by directed edges. Perrault-Joncas et al. [20] and Mousazadeh et al. [16] learn the embedding vectors based on Laplacian type operators and preserve the asymmetry property of edges in a vector field.

Advances in language modeling and unsupervised feature learning in text inspired their adaptations [8, 19, 4, 22, 23] to learn node embeddings where the main idea is to relate nodes which can reach other similar nodes via random walks. DeepWalk [19], for instance, samples truncated random walks from the graph, thus treating *walks* as equivalent of sentences, and then samples node-context pairs from a sliding window to train a *Skip-Gram* model [15]. Node2vec [8], on the other hand, uses a *biased* random walk procedure to explore diverse neighborhoods. LINE [22] which preserves first and second order proximities among the node representations can also be interpreted as embedding nodes closer appearing in random walk of length 1. VERSE [23] learns a single embedding matrix while encoding similarities between vertices sampled as first and last vertex in a PageRank style random walk.

Other works [5, 24, 10, 18] investigate deep learning approaches for learning node representations. Like most of the other methods, they also use a single representation for a node, hence ignoring the asymmetric node roles. Other downsides of these deep learning approaches are the computationally expensive optimization and elaborate parameter tuning resulting in very complex models.

Asymmetry preserving approaches. To the best of our knowledge, there are only two works [17, 26] which learn and use two embedding spaces for nodes, one representing its embedding in the source role and the other in the target role. Note that [1] does not preserve asymmetry for the nodes, which is the

main theme of this work (more comparisons and discussions on this method can be found in the Appendix). HOPE [17] preserves the asymmetric role information of the nodes by approximating high-order proximity measures like Katz measure, Rooted PageRank etc. Basically they propose to decompose the similarity matrices given by these measures and use the two decompositions as representations of the nodes. HOPE cannot be easily generalized as it is tied to a particular measure. APP [26] proposes a random walk based method to encode rooted PageRank proximity. Specifically, it uses directed random walks with restarts to generate training pairs. Unlike other DeepWalk style random walk based methods, APP does not discard the learnt context matrix, on the other hand it uses it as a second (target) representation of the node. However, the random walk employed sometimes is unable to capture the global structure of the graph. Consider a directed graph with a prominent hub and authority structure where many authority nodes have no outgoing links. In such a case any directed random walk from a source node will halt after a few number of steps, irrespective of the stopping criteria. In contrast our alternating random walks also effectively sample low out-degree vertices in their target roles, thereby exploiting the complete topological information of a directed graph.

3   The NERD Model

Given a directed graph $G = (V, E)$ we aim to learn d-dimensional ($d << |V|$) representations, Φ_s and Φ_t , such that the similarities between vertices in their respective source and target roles are preserved. We argue that that two vertices can be similar to each other in three ways: (i) both nodes in source roles (both pointing to similar authorities) (ii) both the nodes in target roles (for example both are neighbors of similar hubs) (iii) nodes in source-target roles (hub pointing to a authority). We extract such similar nodes via our *alternating random walk* strategy which alternates between vertices in opposite roles. For every vertex two embedding vectors are learnt via a single layer neural model encoding its similarities to other vertices in source and target roles. Alternatively, NERD can be interpreted as optimizing first order proximities in three types of *computational graphs* it extracts from the original graph via alternating random walks. We elaborate on this alternate view while explaining our learning framework in Section 3.2.

Notations. We first introduce the notations that would also be followed in the rest of the paper unless stated otherwise. Let $G = (V, \boldsymbol{E})$ be a directed weighted graph with N nodes and M edges. Let $w(e)$ denote the weight of edge e and $vol(G) = \sum_e w(e)$. For any vertex $v \in V$ let $d^{out}(v)$ denote the total outdegree of v, i.e. the sum of weights of the outgoing edges from v. Similarly $d^{in}(v)$ denotes the total indegree of v. For unweighted graphs, we assume the weight of each edge to be 1. Let $\Phi_s(v)$ and $\Phi_t(v)$ represent the respective embedding vectors for any node $v \in V$ in its role as source and target respectively. Let P^{in} and P^{out} denote the input and output degree distributions of G respectively. Specifically $P^{in}(v) = d^{in}(v)/vol(G)$ and $P^{out}(v) = d^{out}(v)/vol(G)$. We remark that the terms *vertex* and *node* are used interchangeably in this work.

3.1   Alternating Walks

We propose two alternating walks which alternate between source and target vertices and are referred to as *source* and *target* walks respectively. To understand the intuition behind these walks, consider a directed graph $G = (V, \boldsymbol{E})$ with N nodes. Now construct a copy of each of these N nodes and call this set V_c. Construct an undirected bipartite graph $G' = (V \cup V_c, E')$ such that for vertices $u, v \in V$ and $v_c \in V_c$, where v_c is a copy of vertex v, there is an edge $(u, v_c) \in E'$ if and only if $(u, v) \in \boldsymbol{E}$. In the directed graph G the adjacency matrix A is generally asymmetric, however, with our construction we obtain a symmetric adjacency matrix $\mathcal{A}$ for bipartite graph G'.

$$\mathcal{A} = \begin{pmatrix} 0 & A \\ A^T & 0 \end{pmatrix}. \tag{1}$$

A walk on this undirected bipartite G' starting from a vertex in V will now encounter source nodes in the odd time step and target nodes in the even time step. We call such a walk an *alternating* walk. Formally source and target alternating walks are defined as follows.

Definition 1. The Source Walk. Given a directed graph, we define *source-walk* of length k as a list of nodes $v_1, v_2, ..., v_{k+1}$ such that there exists edge (v_i, v_{i+1}) if i is odd and edge (v_{i+1}, v_i) if i is even: $v_1 \rightarrow v_2 \leftarrow v_3 \rightarrow \cdots$

Definition 2. The Target Walk. A *target walk* of length k, starting with an in-edge, from node v_1 to node v_{k+1} in a directed network is a list of nodes $v_1, v_2, ..., v_{k+1}$ such that there exists edge (v_{i+1}, v_i) if i is odd and edge (v_i, v_{i+1}) if i is even: $v_1 \leftarrow v_2 \rightarrow v_3 \leftarrow \cdots$

We now define the alternating random walk which we use to sample the respective neighborhoods of vertices in 3 types of NERD's computational graphs.

Alternating Random Walks. To generate an alternating random walk we first sample the input node for the source/target walks from the indegree/outdegree distribution of G. We then simulate source/target random walks of length ℓ. Let c_i denote the i-th node in the alternating random walk starting with node u. Then

$$\Pr(c_i = v' | c_{i-1} = v) = \begin{cases} \frac{1}{d^{out}(v)} \cdot w(v, v'), & \text{if } (v, v') \in \boldsymbol{E} \\ \frac{1}{d^{in}(v)} \cdot w(v', v), & \text{if } (v', v) \in \boldsymbol{E} \\ 0, & \text{otherwise} \end{cases}.$$

All nodes in a source/target walk in their respective roles constitute a neighborhood set for the input (the walk starts at the input node) node.

3.2   Learning Framework using Computation Graphs

As already mentioned NERD can be intuitively understood as optimizing first order proximity (embedding vertices sharing an edge closer) in 3 types of computation graphs that it extracts from G via alternating walks. In particular NERD

(a) NERD Alternating Random walks
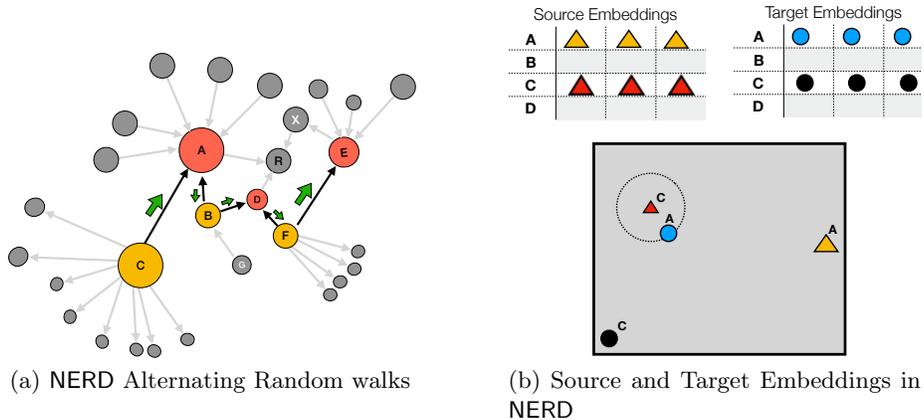
(b) Source and Target Embeddings in NERD

Fig. 1: NERD performing a source walk with input vertex C on an example graph. Node C is the input node. Nodes C, B and F are in source roles and nodes A, D and E are in in target roles. Nodes A, D, E and B,F constitute the target and source neighborhood of source node C. In Figure (b), we show two embedding representations for nodes in their source and target roles. Nodes C and A will be embedded closer in their source-target roles whereas will be far away in source-source or target-target roles.

operates on (i) directed source-target graphs (G_{st}) in which a directed edge exists between nodes of opposite roles (ii) source-source graphs (G_s) in which an undirected edge between nodes represents the similarity between two nodes in their source roles (iii) target-target graphs (G_t) in which an undirected edge between nodes represents the similarity between two nodes in their target roles. For example corresponding to a source walk say $v_1 \rightarrow v_2 \leftarrow v_3 \rightarrow \cdots$, (v_1, v_2) form an edge in G_{st}, (v_1, v_3) form an edge in G_s. For a node/neighbor pair u, v in roles r_1 and r_2 respectively in any of the computation graphs, we are interested in finding representations $\Phi_{r_1}(u)$ and $\Phi_{r_2}(v)$ in their respective roles such that the following objective is maximized.

$$\mathcal{O}(u,v) = \log \sigma(\Phi_{r_1}(u) \cdot \Phi_{r_2}(v)) + \kappa \mathbb{E}_{v' \sim P^n_{r_2}(v')} (\log \sigma(-\Phi_{r_1}(u) \cdot \Phi_{r_2}(v')), \quad (2)$$

where $\sigma(x) = \frac{1}{1+\exp(-x)}$ and $P^n_{r_2}(v')$ is the indegree or outdegree noise distribution and κ is the number of negative examples . We set $P^n_{r_2}(v') = \frac{d^{3/4}(v)}{\sum_{v \in V} d^{3/4}(v)}$, where d is the indegree (if r_2 is the target role) or outdegree (if r_2 is the source role) of vertex v. We optimize Equation (2) using Asynchronous Stochastic Gradient Descent [21].

Figure 1 shows a toy example depicting the working of NERD. The pseudocode for NERD is stated in Algorithm 1. NERD performs a total of γ walks each

walk being source walk or target walk with probability 0.5. The procedure for training a source or target walk is stated in Algorithm 2. The first vertex of the walk is the input vertex whose proximity is optimized (using negative samples) with respect to its neighbors in the opposite role (in line 14) and in the same role (in line 16). The joint training with respect to neighbors of same role can be controlled by a binary parameter JOINT.

Algorithm 1 NERD

Require: graph $G(V, \boldsymbol{E}_w)$, number of nodes to be sampled of each type n, embedding size d, number of walks γ, number of negative samples κ, JOINT $\in \{0, 1\}$
Ensure: matrix of source representations $\Phi_s \in \mathbb{R}^{|V| \times d}$ and target representations $\Phi_t \in \mathbb{R}^{|V| \times d}$

1: **function** NERD$(G, n, d, \gamma, \kappa)$
2:     Initialize Φ_s and Φ_t
3:     **for** $i = 0 \ldots \gamma$ **do**
4:         **if** (rand() > 0.5) **then**
5:             $s_1 \sim P^{out}$
6:             $W_s = $ SOURCEWALK(s_1)
7:             TRAIN$(W_s, s, \kappa,$ JOINT$)$ ▷ source role s
8:         **else**
9:             $t_1 \sim P^{in}$
10:           $W_t = $ TARGETWALK(t_1)
11:           TRAIN$(W_t, t, \kappa,$ JOINT$)$ ▷ target role t

---

We further derive closed form expression for NERD's optimization in the matrix framework as stated in the following theorem (for proof see Supplementary Material). Note that NERD (non-joint) refers to when the optimization is done only on the source/target graph.

Theorem 1. *For any two vertices i and j, NERD (non-joint) finds source ($\Phi_s(i)$) and target ($\Phi_t(j)$) embedding vectors such that $\Phi_s(i) \cdot \Phi_t(j)$ is the (i,j)th entry of the following matrix*

$$\log(vol(G) \sum_{r \in \{1, 3, \ldots, 2n-1\}} (D^{-1}\mathcal{A})^r D^{-1}) - \log \kappa,$$

where n and κ are as in Algorithm 1 and $\mathcal{A}$ is the adjacency matrix of the bipartite network G' obtained by mapping the given directed network G to G' as defined in (1).

Complexity Analysis. Sampling a vertex based on indegree or outdegree distribution requires constant amortized time by building an alias sampling table upfront. At any time only $2n$ neighbors are stored which is typically a small number as we observed in our experiments. In our experiments we set the total number of walks equal to 800 times the number of vertices. In each optimization

---

Algorithm 2 Train a source or target walk

1: **function** TRAIN(W, r, κ, JOINT)
2:     $u \leftarrow W[0]$
3:     $error = 0$
4:     **for** $i = 1, 3, \ldots 2n - 1$ **do**
5:         **for** $j = 0 \ldots \kappa$ **do**
6:             **if** $(j = 0)$ **then**
7:                 $v1 = W[i]$ ▷ neigbor in opposite role r'
8:                 $v2 = W[i + 1]$ ▷ neigbor of same role r
9:                 $label = 1$
10:            **else**   ▷ negative samples
11:                $label = 1$
12:                $v1 \sim P^n_{r'}$
13:                $v2 \sim P^n_r$
14:            $error+ =$UPDATE($\Phi_r(u), \Phi_{r'}(v1), label$)
15:            **if** (JOINT) **then**
16:                $error+ =$UPDATE($\Phi_r(u), \Phi_r(v2), label$)
17:     $\Phi_r(u) + = \ error$
18:
19: **function** UPDATE($\Phi(u), \Phi(v), label$)   ▷ //gradient update
20:     $g = (label - \sigma(\Phi(u) \cdot \Phi(v))) \cdot \lambda$
21:     $\Phi(v)+ = g * \Phi(u)$
22:     **return** $g * \Phi(v)$

---

step we use $\kappa = \{3, 5\}$ negative samples, therefore, complexity of each optimization step is $O(d\kappa)$. As it requires $O(|E|)$ time to read the graph initially, the run time complexity for NERD can be given as $O(nd\kappa N + |E|)$. The space complexity of NERD is $O(|E|)$. As our method is linear (with respect to space and time complexity) in the input size, it is scalable for large graphs.

4   Experiments

In this section we present how we evaluate NERD[1] against several state-of-the-art node embedding algorithms. We use the original implementations of the authors for all the baselines (if available). We also employ parameter tuning whenever possible (parameter settings are detailed in Supplementary Material), otherwise we use the best parameters reported by the authors in their respective papers. We perform comparisons corresponding to three tasks – Link Prediction, Graph Reconstruction and Node classification. In the next section we explain the datasets used in our evaluations.

A brief summary of the characteristics of the datasets (details are links provided in the implementation page) is presented in Table 1. We recall that *reciprocity* in a directed graph equals the proportion of edges for which an edge in

---

[1] We make available our implementation with corresponding data at `https://git.l3s.uni-hannover.de/khosla/nerd`

dataset	Size			Statistics		Details		
	$|V|$	$|E|$	$|\mathcal{L}|$	Diameter	Reciprocity	Labels	vertex	edges
Cora	23,166	91,500	79	20	0.051	✓	papers	citation
Twitter	465,017	834,797	-	8	0.003	-	people	follower
Epinion	75,879	508,837	-	15	0.405	-	people	trust
PubMed	19,718	44,327	3	18	0.0007	✓	papers	citation
CoCit	44,034	195,361	15	25	0	✓	papers	citation

Table 1: Dataset characteristics: number of nodes $|V|$, number of edges $|E|$; number of node labels $|\mathcal{L}|$.

the opposite direction exists, i.e., that are reciprocated. All the above datasets except PubMed and Cocitation datasets have been collected from [12].

Baselines. We compare the NERD model with several existing node embedding models for link prediction, graph reconstruction and node classification tasks. As baselines we consider methods like APP [26] and HOPE [17] which uses two embedding spaces to embed vertices in their two roles. We also compare against other popular single embedding based methods: DeepWalk [19], LINE [22], Node2vec [8] and VERSE [23]. All these methods are detailed in related work. As NERD is an unsupervised shallow neural method, for fair comparisons, semi-supervised or unsupervised deep models are not considered as baselines.

4.1 Link Prediction

The aim of the link prediction task is to predict missing edges given a network with a fraction of removed edges. A fraction of edges is removed randomly to serve as the *test split* while the residual network can be utilized for training. The test split is balanced with negative edges sampled from random vertex pairs that have no edges between them. We refer to this setting as the *undirected* setting. While removing edges randomly, we make sure that no node is isolated, otherwise the representations corresponding to these nodes can not be learned.

Directed link prediction. Since we are interested in not only the existence of the edges between nodes but also the directions of these edges, we consider a slight modification in the test split setting. Note that this is a slight departure from the experimental settings used in previous works where only the presence of an edge was evaluated. We posit that in a directed network the algorithm should also be able to decide the direction of the predicted edge. To achieve this, we allow for negative edges that are complements of the true (positive) edges which exist already in the test split.

We experiment by varying the number of such complement edges created by inverting a fraction of the true edges in the test split. A value of 0 corresponds to the classical undirected graph setting while a value in $(0, 1]$ determines what fraction of positive edges from the test split are inverted at most to create negative examples. It can also happen that an inverted edge is actually an edge in the network, in which case we discard it and pick up some random pair which

corresponds to a negative edge. Such a construction of test data is essential to check if the algorithm is also predicting the correct direction of the edge along with the existence of the edge. Please note that we always make sure that in the test set the number of negative examples is equal to the number of positive examples. Embedding dimensions are set to 128 for all models for both settings.

Table 2 presents the ROC-AUC (Area Under the Receiver Operating Characteristic Curve) scores for link prediction for three datasets (missing datasets show similar trends, results not presented because of space constraints). More specifically, given an embedding, the inner product of two node representations normalized by the sigmoid function is employed as the similarity/link-probability measurement for all the algorithms. Fraction 0% correspond to the undirected setting in which the negative edges in the test set are randomly picked. The 50% and 100% corresponds to directed setting in which at most 50% and 100% positive edges of test set are inverted to form negative edges. Please note that if an inverted edge is actually an edge in the network, we discard it and pick up some random pair.

Performance on Cora. VERSE outperforms others for the undirected setting in the Cora dataset. But its performance decreases rapidly in the directed setting where the algorithm is forced to assign a direction to the edge. The performance of the three directed methods (APP, HOPE and NERD) is stable supporting the fact that these methods can correctly predict the edge direction in addition to predicting a link. NERD is the next best (AUC of 0.788) and outperforms HOPE for directed setting with 50% and 100% (AUC of 0.813) test set edge reversal. This means that that whenever NERD predicts the presence of an edge it in fact also predicts the edge directionality accurately.

Performance on Twitter. For the Twitter dataset, HOPE outperforms all other methods and is closely followed by NERD for 60-40 split of training-test data. Figure 2 shows the performance of three directed graph methods: APP, HOPE and NERD on 70-30 and 90-10 training-test splits for Twitter respectively. Here we plot the AUC scores by varying the fraction of inverted edges in the test split to construct negative test edges. We omit other methods as all of them have a very low performance. We make several interesting observations here. First, HOPE which performs best for 60-40 split shows a decrease in performance with the increase in training data. We believe that the parameters for HOPE namely the attenuation factor which was tuned for best performance on a smaller amount of training data no longer might not be applicable for larger training data. This renders such a method to be very sensitive to structural changes in the graph. Second, APP's performance improves with increasing training data but is not as stable as NERD and HOPE in the directed setting when the fraction of inverted edges is increased, i.e., it does not always correctly predict the direction of an edge. Third, NERD's performance stays stable and improves on increasing the training data, which confirms our justification that it is more robust to structural changes caused by random addition/removal of edges. Moreover, at 90% training data it is the best performing method and second best but consistent (in predicting edge direction) for other splits. Finally we

observe that Twitter has a prominent hub-authority structure with more than than 99% vertices with zero out-degree. Using non-alternating directed walks on Twitter hinders APP and other similar random walk methods to fully explore the network structure as much as they could do for Cora.

Performance on Epinions. VERSE shows a high performance on Epinions in undirected setting which is not surprising as Epinions has a high reciprocity with more than 40% of the edges existing in both directions. NERD on the other hand beats the two other directed methods APP and HOPE for both the settings. As the fraction of edge reversals increases, NERD also starts performing better than VERSE. We note that even though NERD does not outperforms all methods on link prediction, it shows more robustness across datasets being the second best performing (when not the best) and is consistent in predicting the right edge direction i.e., its performance does not vary a lot (except in Epinions with high reciprocity) with increasing fraction of positive test edge inversions in the directed setting.

method	Cora			Twitter			Epinion		
	0%	50%	100%	0%	50%	100%	0%	50%	100%
DeepWalk	0.836	0.669	0.532	0.536	0.522	0.501	0.538	0.560	0.563
Node2vec	0.840	0.649	0.526	0.500	0.500	0.500	<u>0.930</u>	0.750	0.726
VERSE	**0.875**	0.688	0.500	0.52	0.510	0.501	**0.955**	<u>0.753</u>	<u>0.739</u>
APP	<u>0.865</u>	**0.841**	**0.833**	0.723	0.638	0.555	0.639	0.477	0.455
HOPE	0.784	0.734	0.718	**0.981**	**0.980**	**0.979**	0.807	0.718	0.716
LINE-1+2	0.735	0.619	0.518	0.009	0.255	0.500	0.658	0.622	0.617
LINE-1	0.781	0.644	0.526	0.007	0.007	0.254	0.744	0.677	0.668
LINE-2	0.693	0.598	0.514	0.511	0.507	0.503	0.555	0.544	0.543
NERD	0.795	<u>0.788</u>	<u>0.813</u>	<u>0.969</u>	<u>0.968</u>	<u>0.967</u>	0.906	**0.774**	**0.771**

Table 2: Link Prediction Results for directed graphs with (1) random negative edges in test set (2) 50% of the test negative edges created by reversing positive edges (3) when all positive edges are reversed to create negative edges in the test set. The top scores are shown in bold whereas the second best scores are underlined.

4.2 Graph Reconstruction

In the graph reconstruction task we evaluate how well the embeddings preserve neighborhood information. There are two separate evaluation regimes for graph reconstruction in previous works. One line of work [17], that we refer to as *edge-centric* evaluation, relies on sampling random pairs of nodes from the original graphs into their test set. These candidate edges are then ordered according to
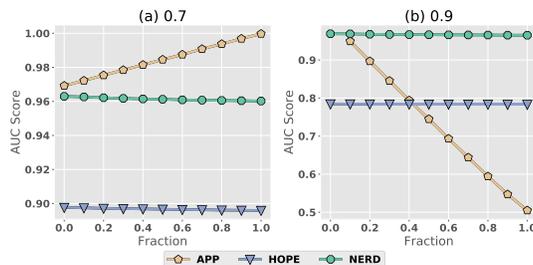
Fig. 2: Link prediction in Twitter. The y-axis shows the AUC scores and the x-axis is the maximum fraction of edges that are inverted in the test split. The models are trained on 70% and 90% of the Twitter edges respectively. The fraction on the x-axis indicates the maximum fraction of inverted positive test edges to create negative test edges. Note that the train-test split is the same over all fractions.

their similarity in the embedding space. Precision is then computed at different rank depths where the relevant edges are the ones present in the original graph. On the other hand, [23] perform a *node-centric* evaluation where precision is computed on a per-node basis. For a given node v with an outdegree k, embeddings are used to perform a k-nearest neighbor search for v and precision is computed based on how many actual neighbors the k-NN procedure is able to extract.

Directed Graph Reconstruction. We believe that the edge-centric evaluation suffers from sparsity issues typical in real-world networks and even if a large number of node pairs are sampled, the fraction of relevant edges retrieved tends to remain low. More acutely, such an approach does not model the neighborhood reconstruction aspect of graph construction and is rather close to predicting links. We adopt the node-centric evaluation approach where we intend to also compute precision on directed networks with a slight modification. In particular, we propose to compute precision for both outgoing and incoming edges for a given node. This is different from previous evaluation approaches which only considers the reconstruction of adjacency list of a node, i.e., only its outgoing neighbors. Moreover in our proposed evaluation strategy we do not assume the prior knowledge of the indegree or outdegree.

As in Link Prediction, the similarity or the probability of an edge (i, j) is computed as the sigmoid over the dot product of their respective embedding vectors. For HOPE, NERD and APP we use the corresponding source and target vectors respectively. We do not assume the prior knowledge of the indegree or outdegree, rather we compute the precision for $k \in \{1, 2, 5, 10, 100, 200\}$. For a given k we obtain the k-nearest neighbors ranked by sigmoid similarity for each embedding approach. If a node has an outdegree or indegree of zero, we set the precision to be 1 if the sigmoid corresponding to the nearest neighbor is less than 0.51 (recall that $\sigma(\boldsymbol{x} \cdot \boldsymbol{y}) = 0.5$ for $\boldsymbol{x} \cdot \boldsymbol{y} = 0$), otherwise we set it to 0. In

other cases, for a given node v and a specific k we compute $P_{out}^k(v)$ and $P_{in}^k(v)$ corresponding to the outgoing and incoming edges as

$$P_{out}^k(v) = \frac{\mathcal{N}_{out}^k \cap N^{out}(v)}{k}, \quad P_{in}^k(v) = \frac{\mathcal{N}_{in}^k \cap N^{in}(v)}{k},$$

where $\mathcal{N}_{out}^k(v)$ and $\mathcal{N}_{in}^k(v)$ are the k nearest outgoing (to whom v has outgoing edges) and incoming (neighbors point to v) neighbors retrieved from the embeddings and $N^{out}(v)$ and $N^{in}(v)$ are the actual outgoing and incoming neighbors of v. We then compute the Micro-F1 score as the harmonic mean of $P_{in}^k(v)$ and $P_{out}^k(v)$. To avoid any zeros in the denominator, we add a very small $\varepsilon = 10^{-5}$ to each precision value before computing the harmonic mean. We finally report the final precision as the average of these harmonic means over the nodes in the test set.

Results. We perform the graph reconstruction task on the Cora, Cocitation and Twitter datasets. In order to create the test set we randomly sample 10% of the nodes for Cora and Cocitation datasets and and 1% of Twitter. We plot the final averaged precision corresponding to different values of k in Figure 3.

For the Cora dataset, NERD clearly outperforms all the other models including HOPE. In particular for $k = 1$, NERD shows an improvement of 63% over HOPE which in some sense is fine tuned for this task.

The trend between NERD and HOPE is reversed for Twitter dataset, where HOPE behaves like an almost exact algorithm. This can be attributed to the low rank of the associated Katz similarity matrix. Note that only 2502 out of more than $400K$ nodes have non-zero outdegree which causes a tremendous drop in the rank of the associated Katz matrix. We recall that HOPE's approximation guarantee relies on the low rank assumption of the associated similarity matrix which seems to be fulfilled quite well in this dataset. The performance of other models in our directed setting clearly shows their inadequacy to reconstruct neighborhoods in directed graphs. For Twitter, we only show plots corresponding to HOPE and NERD as precision corresponding to other methods is close to 10^{-5}.

Again for Cocitation NERD performs the best with an improvement of around 12.5% for $k = 1$ over the second best performing method, HOPE. Once again, NERD exhibited robustness in this task as for Twitter, it is closest to the best performing method. Note that some of the methods like VERSE and APP which were sometimes better performing than NERD in link prediction show a poor performance across all datasets in graph reconstruction task. Note that this task is harder than link prediction as the model not only needs to predict the incoming and outgoing neighbors but also has no prior knowledge of the number of neighbors. Moreover the test set for this task is not balanced in the sense that for each test node the model needs to distinguish between small number of positive edges with a huge number of negative edges, for example for small k.

4.3   Node Classification

We run experiments for predicting labels in the Cora, CoCitation and PubMed datasets (labels were not available for other two datasets). We report the Micro-
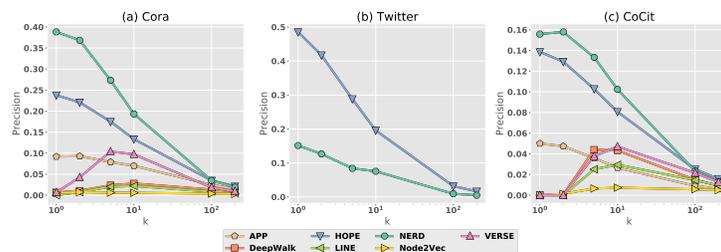
Fig. 3: Graph reconstruction for Cora, Twitter and CoCitation Networks with precision computed on both outgoing and incoming edges. NERD shows an improvement of 63.1% (for $k = 1$) as compared to HOPE in the Cora dataset. The trend is reversed in the Twitter dataset because of the exact nature of HOPE on low-rank Twitter data. For Twitter, all methods except NERD and HOPE have precision close to 10^{-5}, therefore we do not show them in the plots. NERD shows an improvement of 12.5% (for $k = 1$) as compared to HOPE in the CoCit dataset.

method	PubMed		Cora		CoCit	
	mic.	mac.	mic.	mac.	mic.	mac.
DeepWalk	73.96	71.34	64.98	**51.53**	41.92	30.07
Node2vec	72.36	68.54	65.74	49.12	41.64	28.18
VERSE	71.24	68.68	60.87	45.52	40.17	27.56
APP	69.00	65.20	64.58	47.03	40.34	28.06
HOPE	63.00	54.6	26.23	1.22	16.66	1.91
LINE-1+2	62.29	59.79	54.04	41.83	37.71	26.75
LINE-1	55.65	53.83	62.36	47.19	36.10	25.70
LINE-2	56.81	51.71	51.05	35.37	31.4	20.59
NERD	**76.70**	**74.67**	**67.75**	<u>51.30</u>	**44.84**	**33.49**

Table 3: Node Classification results in terms of Micro-F1 and Macro-F1. All results are mean of 5-fold cross validations. Each node in Cora has multiple labels, for PubMed and CoCit, there is a single label per node.

F1 and Macro-F1 scores after a 5-fold multi-label classification using one-vs-rest logistic regression. The main aim of this experiment is to show that NERD is generalizable across tasks and also performs well for a task like node classification which is not fine tuned for directed graphs. Unlike APP and HOPE, NERD also performs the best in this task over all the 3 datasets. Other single embedding based methods like DeepWalk and Node2vec also exhibit a good performance for node classification indicating that edge directionality might not be a very important factor for node labels at least for the studied datasets. HOPE which

performs very well for link prediction and graph reconstruction tasks shows a poorer performance.

As we already pointed out that HOPE is tied to particular proximity matrix and adjusting it for a task becomes much harder and non-intuitive than random walk based methods where hyper parameters can be easily fine-tuned. We also note that for HOPE the similarity between nodes i and j is determined by the effective distance between them which is computed using the Katz measure, penalizing longer distances by an attenuation factor β. The advantage of such a degrading distance measure is that it conserves the adjacency similarity of the graph, which reflects in our experiments on Graph Reconstruction. NERD on the other hand also takes into account how likely i can influence j by taking into account the likelihood of the traversal of various alternating paths between i and j. In other words, NERD constructs the local neighborhood based on how likely this neighborhood can influence the node, which helps the classifier to learn better labels on NERD trained embeddings.

5   Conclusion

We presented a novel approach, NERD, for embedding directed graphs while preserving the role semantics of the nodes. We propose an alternating random walk strategy to sample node neighborhoods from a directed graph. The runtime and space complexities of NERD are both linear in the input size, which makes it suitable for large scale directed graphs. In addition to providing advantages of using two embedding representations of nodes in a directed graph, we revisit the previously used evaluation strategies for directed graphs. We chart out a clear evaluation strategy for link prediction and graph reconstruction tasks. We observe in our experiments where we find that a method performing best in one of the tasks might perform the worst in the other task. This beats the whole idea of unsupervised learning which is supposed not to be fine tuned towards a particular task but should be performing well across different tasks. We show that the embeddings from NERD are indeed robust, generalizable and well performing across multiple types of tasks and networks. We have showcased the effectiveness of NERD by employing a shallow neural model to optimize the topology of the extracted computational graphs. In future we will employ deeper models to capture non-linearities while preserving the respective topologies.

References

1. Abu-El-Haija, S., Perozzi, B., Al-Rfou, R.: Learning edge representations via low-rank asymmetric projections. pp. 1787–1796. CIKM '17 (2017)
2. Ammann, L.P.: Robust singular value decompositions: A new approach to projection pursuit. Journal of the American Statistical Association **88**(422) (1993)
3. Belkin, M., Niyogi, P.: Laplacian eigenmaps and spectral techniques for embedding and clustering. In: Advances in Neural Information Processing Systems 14, pp. 585–591 (2002)

4. Cao, S., Lu, W., Xu, Q.: Grarep: Learning graph representations with global structural information. In: CIKM '15. pp. 891–900 (2015)

5. Cao, S., Lu, W., Xu, Q.: Deep neural networks for learning graph representations. In: Proceedings of AAAI. pp. 1145–1152. AAAI'16 (2016)

6. Chen, M., Yang, Q., Tang, X.: Directed graph embedding. In: IJCAI(2007)

7. Drobyshevskiy, M., Korshunov, A., Turdakov, D.: Learning and scaling directed networks via graph embedding. In: ECML-PKDD. pp. 634–650 (2017)

8. Grover, A., Leskovec, J.: Node2vec: Scalable feature learning for networks. In: KDD. pp. 855–864. (2016)

9. Katz, L.: A new status index derived from sociometric analysis. Psychometrika **18**(1), 39–43 (1953)

10. Kipf, T.N., Welling, M.: Variational graph auto-encoders. In: NeurIPS Workshop on Bayesian Deep Learning (NeurIPS-16 BDL) (2016)

11. Kleinberg, J.M.: Authoritative sources in a hyperlinked environment. J. ACM **46**(5), 604–632 (1999)

12. Kunegis, J.: Konect datasets: Koblenz network collection. http://konect.uni-koblenz.de (2015)

13. Lempel, R., Moran, S.: Salsa: the stochastic approach for link-structure analysis. ACM Transactions on Information Systems (TOIS) **19**(2), 131–160 (2001)

14. Liben-Nowell, D., Kleinberg, J.: The link-prediction problem for social networks. Journal of the American society for information science and technology **58**(7), 1019–1031 (2007)

15. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Proceedings of the 27th Annual Conference on Neural Information Processing Systems 2013. pp. 3111–3119 (2013)

16. Mousazadeh, S., Cohen, I.: Embedding and function extension on directed graph. Signal Process. **111**(C), 137–149 (2015)

17. Ou, M., Cui, P., Pei, J., Zhang, Z., Zhu, W.: Asymmetric transitivity preserving graph embedding. In: KDD. pp. 1105–1114. (2016)

18. Pan, S., Hu, R., Long, G., Jiang, J., Yao, L., Zhang, C.: Adversarially regularized graph autoencoder for graph embedding. In: IJCAI-18. pp. 2609–2615 (2018)

19. Perozzi, B., Al-Rfou, R., Skiena, S.: Deepwalk: Online learning of social representations. In: Proceedings of SIGKDD. pp. 701–710. KDD '14 (2014)

20. Perrault-Joncas, D.C., Meila, M.: Directed graph embedding: an algorithm based on continuous limits of laplacian-type operators. In: Advances in Neural Information Processing Systems. pp. 990–998 (2011)

21. Recht, B., Re, C., Wright, S., N., F.: Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In: Advances in Neural Information Processing Systems 24, pp. 693–701 (2011)

22. Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., Mei, Q.: Line: Large-scale information network embedding. In: Proceedings of the 24th International Conference on World Wide Web. pp. 1067–1077 (2015)

23. Tsitsulin, A., Mottin, D., Karras, P., Müller, E.: Verse: Versatile graph embeddings from similarity measures. In: Proceedings of the 2018 World Wide Web Conference. pp. 539–548 (2018)

24. Wang, D., Cui, P., Zhu, W.: Structural deep network embedding. In: Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 1225–1234. KDD '16 (2016)

25. Ying, R., He, R., Chen, K., Eksombatchai, P., Hamilton, W.L., Leskovec, J.: Graph convolutional neural networks for web-scale recommender systems. In: KDD. pp. 974–983 (2018)
26. Zhou, C., Liu, Y., Liu, X., Liu, Z., Gao, J.: Scalable graph embedding for asymmetric proximity. In: AAAI Conference on Artificial Intelligence (AAAI'17) (2017)