# Outlier Aware Network Embedding for Attributed Networks


Sambaran Bandyopadhyay, Lokesh N, M Narasimha Murty

sambaran.ban89@gmail.com, sambband@in.ibm.com

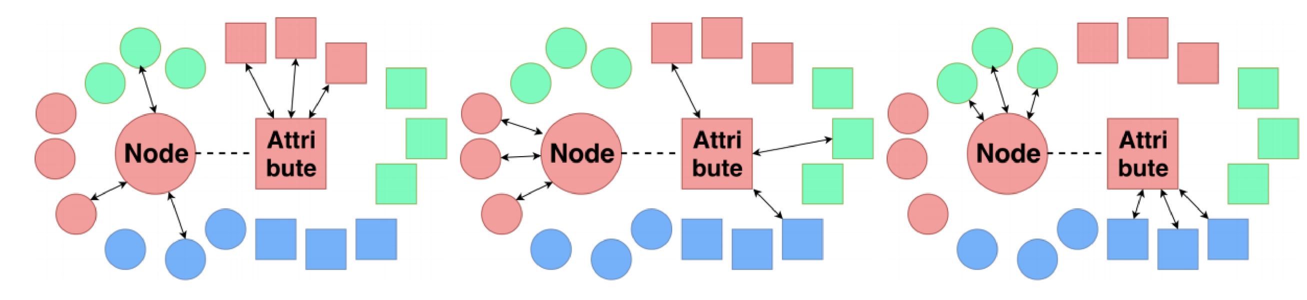
IBM Research, India    Indian Institute of Science, Bangalore



AAAI 2019 - Paper ID: 1715


## Introduction and Motivation

- *In an attributed network, each node comes with some content within (also known as **node attributes**).*
- *Existing attributed network approaches work well when the network is **consistent between structure and attributes**.*
- *Real world networks often have anomalous (**outlier**) nodes, which affect the embeddings of other nodes in the network. Thus all the downstream network mining tasks may fail miserably in the presence of such outliers.*
- *We proposed an **unsupervised O**utlier aware **N**etwork **E**mbedding algorithm (**ONE**) for attributed networks, which minimizes the effect of the outlier nodes, and hence generates robust network embeddings.*
- ***To the best of our knowledge, this is the first work to propose a completely unsupervised algorithm for attributed network embedding integrated with outlier detection.***




- *(a) **Structural Outlier**: The node has edges to nodes from different communities.*
- *(b) **Attribute Outlier**: The attributes of the node are similar to attributes of the nodes from different communities.*
- *(c) **Combined Outlier**: Node belongs to a community structurally but it has a different community in terms of attribute similarity.*

## Problem Formulation and Solution Approach

- An attributed information network is represented by a graph as $\mathcal{G} = (V, E, C)$
- $V = \{v_1, v_2, \cdots, v_N\}$ is the set of nodes, $E \subset \{(v_i, v_j) | v_i, v_j \in V\}$ is the set of edges.
- $N \times N$ dimensional adjacency matrix of the graph $\mathcal{G}$ is $A = (a_{i,j})$, where $a_{i,j} = w_{v_i, v_j}$ if $(v_i, v_j) \in E$, and $a_{i,j} = 0$ otherwise.
- $C$ is a $N \times D$ matrix where $C_{i \cdot} \in \mathbb{R}^D$ is the attribute vector associated with the node $v_i \in V$.
- $C_{id}$ is the value of the attribute $d$ for the node $v_i$. For example, if there is only textual content in each node, $c_i$ can be the tf-idf vector for the content of the node $v_i$.
- For a given network $\mathcal{G}$, network embedding is a technique to learn a function $f : v_i \mapsto \mathbf{y_i} \in \mathbb{R}^K$, i.e., it maps every vertex to a $K$ dimensional vector, where $K < min(N, D)$.
- The representations should preserve the underlying semantics of the network.
- **Joint Loss Function**:

$$\mathcal{L} = \mathcal{L}_{str} + \alpha \mathcal{L}_{attr} + \beta \mathcal{L}_{dis}$$

$$0 < O_{1i}, O_{2i}, O_{3i} \leq 1 \ , \ \forall v_i \in V$$

$$\sum_{i=1}^{N} O_{1i} = \sum_{i=1}^{N} O_{2i} = \sum_{i=1}^{N} O_{3i} = \mu$$

$$W \in \mathcal{O}_K \iff W^T W = \mathcal{I}$$

- **Learning from the Link Structure**:

$$\mathcal{L}_{str} = \sum_{i=1}^{N} \sum_{j=1}^{N} \log\left(\frac{1}{O_{1i}}\right)(A_{ij} - G_{i\cdot} \cdot H_{\cdot j})^2 \quad (1)$$

- **Learning from the Attributes**:

$$\mathcal{L}_{attr} = \sum_{i=1}^{N} \sum_{d=1}^{C} \log\left(\frac{1}{O_{2i}}\right)(C_{id} - U_{i\cdot} \cdot V_{\cdot d})^2 \quad (2)$$

- **Connecting Structure and Attributes**: We want to find a matrix $W$ which minimizes $||G - WU||_F$.

$$\mathcal{L}_{dis} = \sum_{i=1}^{N} \sum_{k=1}^{K} \log\left(\frac{1}{O_{3i}}\right)\left(G_{ik} - U_{i\cdot} \cdot (W^T)_{\cdot k}\right)^2 \quad (3)$$

- If we restrict $W$ to be an orthogonal matrix, then a **closed form solution** can be obtained from the solution concept of Procrustes problem:

$$W^* = \underset{W \in \mathcal{O}_K}{\mathrm{argmin}} ||G - UW^T||_F \quad (4)$$

where $W^* = XY^T$ with $X\Sigma Y^T = \mathrm{SVD}(G^T U)$, $\mathcal{O}_K$ is the set of all orthogonal matrices of dimension $K \times K$.

## Update Rules and Experimental Setup

$$G_{ik} = \frac{G_{ik}^{num1} + \beta \log\left(\frac{1}{O_{3i}}\right)(W_{k\cdot} \cdot U_{i\cdot})}{\log\left(\frac{1}{O_{1i}}\right)(H_{k\cdot} \cdot H_{k\cdot}) + \beta \log\left(\frac{1}{O_{3i}}\right)}$$

$$G_{ik}^{num1} = \log\left(\frac{1}{O_{1i}}\right) \sum_{j=1}^{N} (A_{ij} - \sum_{k' \neq k} G_{ik'} H_{k'j}) H_{kj}$$

$$H_{kj} = \frac{\sum_{i=1}^{N} \log\left(\frac{1}{O_{1i}}\right)(A_{ij} - \sum_{k' \neq k} G_{ik'} H_{k'j}) G_{ik}}{\sum_{i=1}^{N} \log\left(\frac{1}{O_{1i}}\right) G_{ik}^2}$$

$$O_{1i} = \frac{\left(\sum_{j=1}^{N} (A_{ij} - G_{i\cdot} \cdot H_{\cdot j})^2\right) \cdot \mu}{\sum_{i=1}^{N} \sum_{j=1}^{N} (A_{ij} - G_{i\cdot} \cdot H_{\cdot j})^2}$$

$$O_{2i} = \frac{\left(\sum_{d=1}^{D} (C_{id} - U_{i\cdot} \cdot V_{\cdot d})^2\right) \cdot \mu}{\sum_{i=1}^{N} \sum_{j=1}^{D} (C_{id} - U_{i\cdot} \cdot V_{\cdot d})^2} \qquad O_{3i} = \frac{\left(\sum_{k=1}^{K} (G_{ik} - W_{i\cdot} \cdot U_{\cdot k})^2\right) \cdot \mu}{\sum_{i=1}^{N} \sum_{k=1}^{K} (G_{ik} - W_{i\cdot} \cdot U_{\cdot k})^2}$$

**Lemma 1** The joint cost function decreases after each iteration of the set of update rules.

- To check the performance of the algorithms in the presence of outliers, we manually planted a total of 5% outliers (with equal numbers for each type) in each dataset.

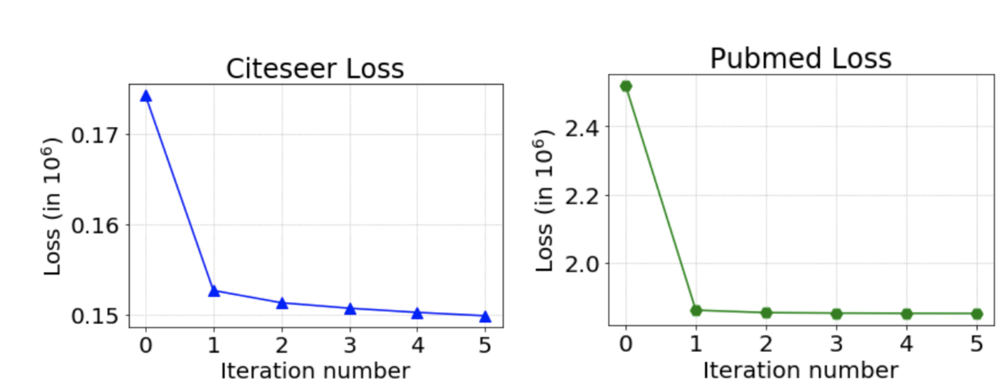



Figure 2: Values of Loss function over different iterations of ONE for Citeseer and Pubmed (seeded) datasets

## Experimental Results

- Observations:
  - Though **SEANO** uses 20% labeled data as the extra supervision to generate the embeddings, its accuracy is always close (or less) to **ONE** which is completely unsupervised in nature.
  - ONE converges very fast on real datasets. But updating most of the variables in this framework takes $O(N)$ time, which leads to $O(N^2)$ runtime for ONE without any parallel processing.

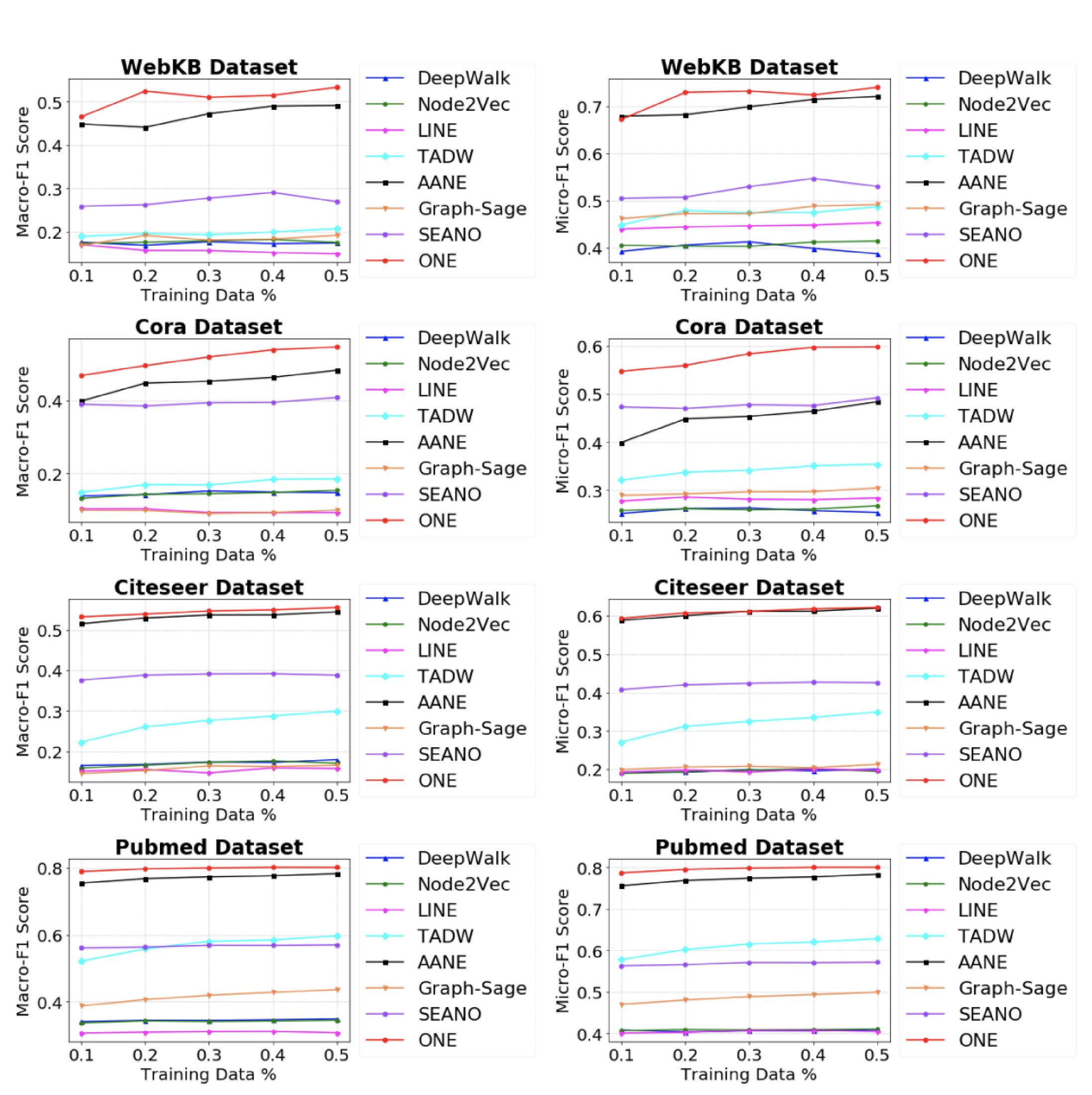



Figure 4: Performance of different embedding algorithms for Classification with Random Forest

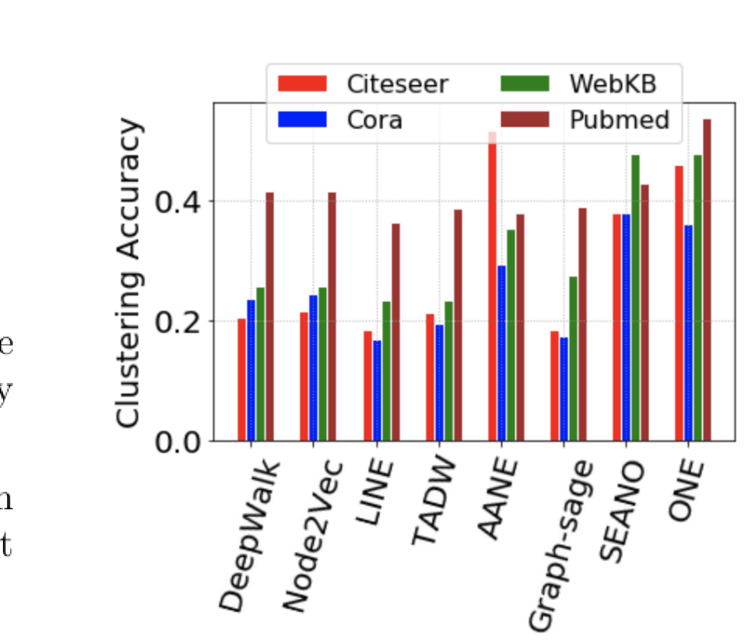



Figure 5: Clustering Accuracy of KMeans++ on the embeddings generated by different algorithms.

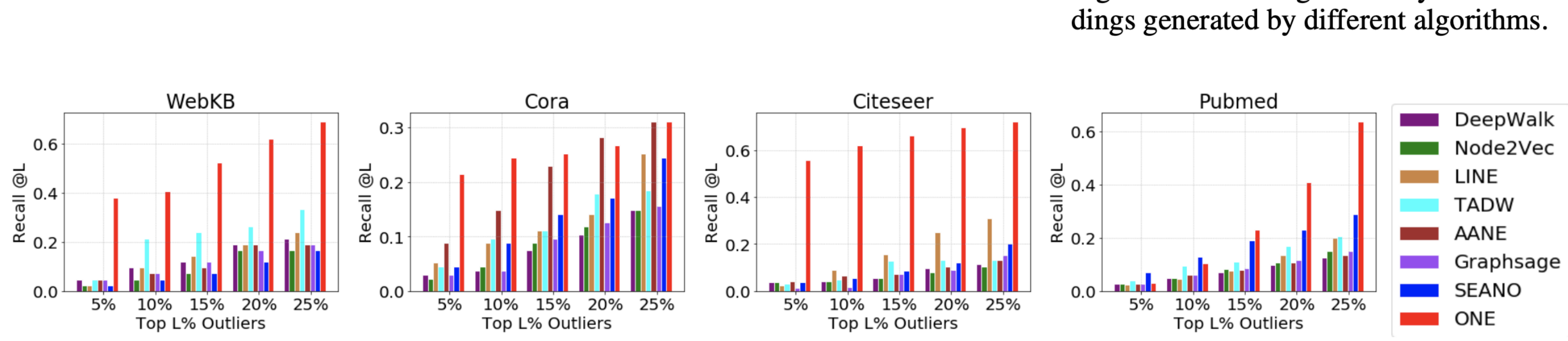



Figure 3: Outlier Recall at top L% from the ranked list of outliers for all the datasets. ONE, though it is an unsupervised algorithm, outperforms all the baseline algorithms in most of the cases. SEANO uses 20% labeled data for training.