Anomaly Detection on Attributed Networks

Dylan Walker, Shengnan Miao, Bao Pham, Max Schwartz

Rensselaer Polytechnic Institute

December 9, 2021

Outline

Anomaly detection

- Anomaly Detection is the process of determining elements in a dataset that have a behavior that deviates from the rest of the dataset.
- Challenges remain for anomaly detection on attributed networks:
  (1) Network sparsity - the network structure could be very sparse on real-world attributed networks.
  (2) Data nonlinearity - the node interactions and nodal attributes are highly non-linear in nature while existing anomaly detectors mainly model the attributed networks with linear mechanisms.
  (3) Complex modality interactions - attributed networks usually have complex interactions for anomaly detection.

Attributed networks

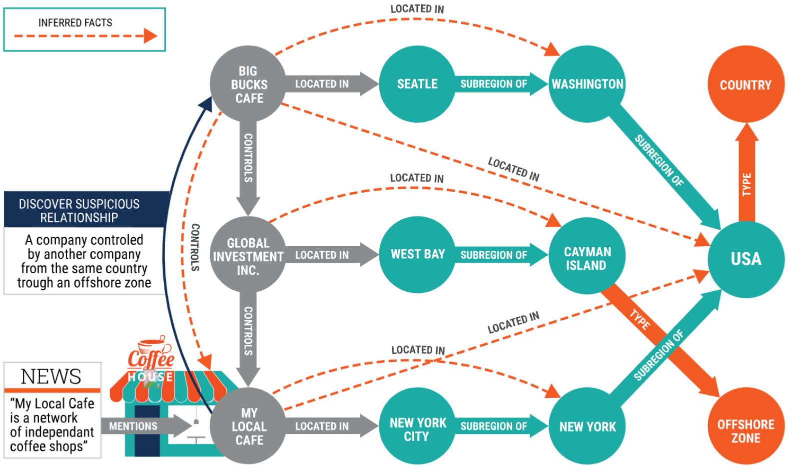
- An **attributed network** $\mathcal{G} = (\mathcal{V}, \mathcal{E}, X)$ is a graph with vertex set V, edge set E, and node feature matrix X.
- Conventionally, we let $N = |\mathcal{V}|$ denote the number of vertices and $m = |\mathcal{E}|$ denote the number of edges.
- Each node has a corresponding feature vector $x \in \mathbb{R}^k$, where k denotes the number of node features.
- Each edge in the network belongs to one of d different classes, where d denotes the number of distinct edge types.

Attributed networks (continued)

- The node feature matrix $X \in \mathbb{R}^{N \times k}$ compactly stores the node features for the entire network.
- The adjacency tensor $A \in \{0,1\}^{d \times N \times N}$ stores an adjacency matrix for each of the d different edge types in the network.

Knowledge Graph

- A knowledge graph is a type of directed attributed network that models semantic data;
- Nodes represent real-world entities;
- Edges capture the relationships between entities.



Importance of Anomaly Detection in Knowledge Graphs

- Anomaly detection on knowledge graphs allows us to discover entities within a system that have suspicious behavior.
- Effective anomaly detection on large networks can be used in security efforts by highlighting the abnormal networks entities.
- For example, in a financial network anomaly detection can be applied to detect fraudulent accounts by analyzing their transaction patterns.

Problem statement

- Given an attributed network $\mathcal{G} = (V, E.X)$, our goal is to rank the vertices of $\mathcal{G}$ by how likely they are to be anomalous within the overall context of the network $\mathcal{G}$.
- The goal of our model is to learn a threshold value λ and a scoring function $f : v_i \to \mathbb{R}$ for each vertex $v_i \in V$ such that we can classify each node as anomalous or normal.
- Let y_i denote the output classification for node v_i under our model where $y_i = 1$ if v_i is anomalous and $y_i = 0$ if v_i is normal. Our goal is to learn f and λ such that:

$$y_i = \begin{cases} 1 & f(v_i) \geq \lambda \\ 0 & \text{otherwise} \end{cases}$$

Graph Convolutional Networks (GCN)

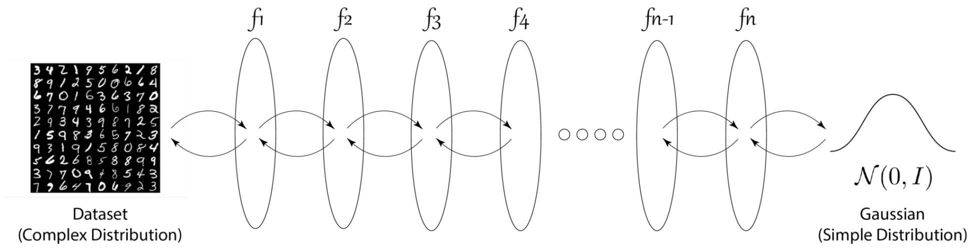
- Given an attributed network $\mathcal{G} = (V, E, X)$, we can use GCN's to learn embeddings $\{H^{(0)}, H^{(1)}, \ldots, H^{(L)}\}$

$$H^{(l+1)} = \sigma(\hat{D}^{-1/2}\hat{A}\hat{D}^{-1/2}H^{(l)}W^{(l)})$$

  with the following parameters:
  - $\hat{A} = A + I_N$ - Neighborhood adjacency matrix with self connections
  - $\hat{D} \in \mathbb{R}^{N \times N}$, the diagonal degree matrix of $\hat{A}$
  - $W^{(l)}$ - Weight matrix for layer l
  - σ - Nonlinear activation function
- Captures the critical inter-dependencies of network-structured data
- Node embeddings dependent on the local structure

Baseline Architecture

- Baseline: Auto-regressive Normalizing Flow Model:
  - An implementation based on graphAF
  - Normalizing flows is a generative modeling architecture that learns an invertible mapping from the data space to a latent probability space.
  - Auto-regressive normalizing flows learns a probability distribution that is used to sequentially reconstruct the network structure and node attributes



Baseline Architecture (continued)

- Given a sampled network neighborhood $\mathcal{N}(v_i) = (X_i, A_i)$, we use a normalizing flow composed of GCN layers to learn the parameters of a probability distribution over the features $\mathbf{x}_i$ and connections $\mathbf{a}_i$ of the central node v_i:

$$p(\mathbf{x}_i|\mathcal{N}(v_i)) = \mathcal{N}(\mu_i^X, (\alpha_i^X)^2)$$
$$p(\mathbf{a}_{i,j}|\mathcal{N}(v_i), \mathbf{x}_i, \mathbf{a}_{i,1:j-1}) = \mathcal{N}(\mu_{ij}^A, (\alpha_{ij}^A)^2)$$

- We then use maximum likelihood estimation with the following loss function to train the model

$$\mathcal{L}(v_i) = -log(p(\mathbf{x}_i)) + \sum_{j=1}^{N} -log(p(\mathbf{a}_{ij}))$$

- Lastly, we can use the loss to evaluate the scoring function for node v_i:

$$f(v_i) = \mathcal{L}(v_i)$$

Graph Autoencoder Architecture

- Proposed Graph Autoencoder:
  - Preliminary
    In attributed network, we have a node feature matrix $\mathbf{X} \in \mathbb{R}^{N \times k}$ and an adjacency matrix $\mathbf{A} \in \{0, 1\}^{d \times N \times N}$.
    Given an input network neighborhood $\mathcal{N}(v_i) = (\mathbf{X}_i, \mathbf{A}_i)$, the encoder $\mathrm{Enc}(\cdot)$, the decoder $\mathrm{Dec}(\cdot)$, then the learning process can be described as minimizing a cost function:

    $$\min \mathbb{E}[\mathrm{dist}(\mathbf{X}_i, \mathrm{Dec}(\mathrm{Enc}(\mathbf{X}_i), \mathbf{A}_i, \mathrm{Dec}(\mathrm{Enc}(\mathbf{A}_i))]$$

    where $\mathrm{dist}(\cdot, \cdot)$ is a predefined distance metric.
  - Encoder
    A series of GCN layers are used to encode the graph neighborhoods into a latent embedding $\mathbf{Z}$.

Graph Autoencoder Architecture (continued)

- Structural Decoder
  The structural decoder learns an approximation of the adjacency tensor $\hat{\mathrm{A}}_i$

$$\hat{\mathbf{A}}_i = \sigma(\mathbf{Z}\mathbf{Z}^T)$$

  Where σ is the element-wise sigmoid function, $\sigma(x) = \frac{1}{1+e^{-x}}$
- Attribute Decoder
  The attribute decoder learns an approximation of the node feature matrix $\hat{\mathbf{X}}$

$$\hat{\mathbf{X}}_i = GCN(\mathbf{Z}, \mathbf{A}_i)$$

- Loss Function

$$\mathcal{L} = (1-\alpha)\|\mathbf{A}_i - \hat{\mathbf{A}}_i\|_F^2 + \alpha\|\mathbf{X}_i - \hat{\mathbf{X}}_i\|_F^2$$

- Anomaly Scoring

$$f(\mathbf{v}_i) = \mathcal{L}(\mathcal{N}(\mathbf{v}_i))$$

Anomaly detection for semantic network

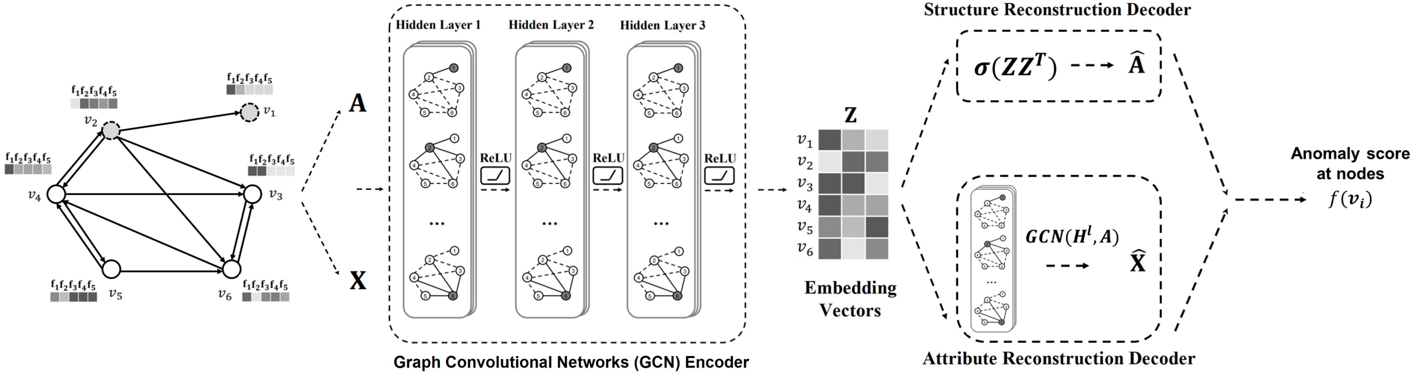


Figure 1: The overall framework of our proposed model for deep anomaly detection on semantic networks.

- Semantic networks are used in natural language processing applications such as semantic parsing and word-sense disambiguation.

Example of semantic network

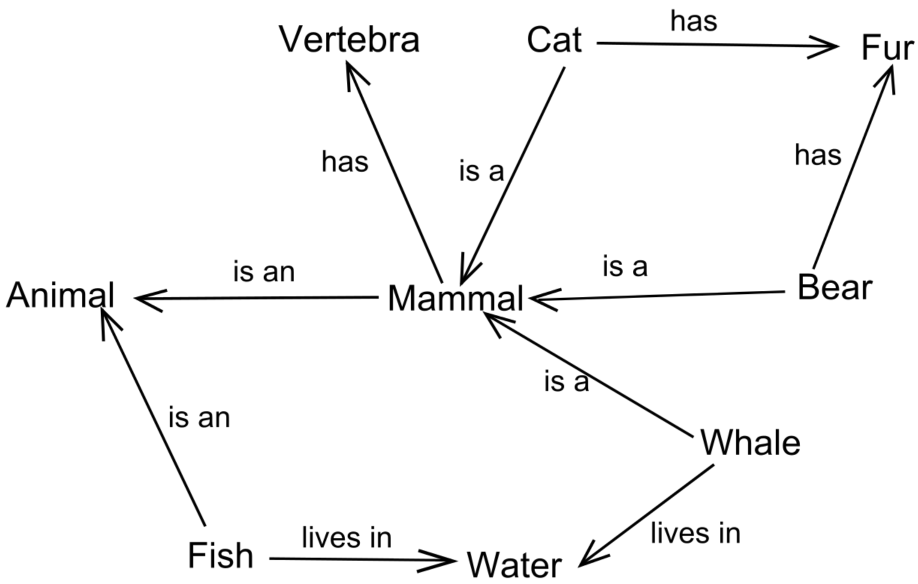


Figure 2: In this knowledge graph of semantic network, vertices represent concepts and edges represent semantic relations between concepts.

NELL Dataset

- Entity $\underset{relation}{\rightarrow}$ *value*
- Each node (entity) has a best query
- Best query $\leftarrow$ best-entity-query $+$ best-value-query
  - Query is embedded with Google pre-trained Universal Sentence Encoder

Entity	Relation	Value	Query
concept:company:limited_brands	concept:companyceo	concept:ceo:leslie_wexner	limited brands Leslie-Wexner
concept:company:limited_brands	generalizations	concept:retailstore	limited brands
concept:company:limited_brands	generalizations	concept:ceo:leslie_wexner	limited brands

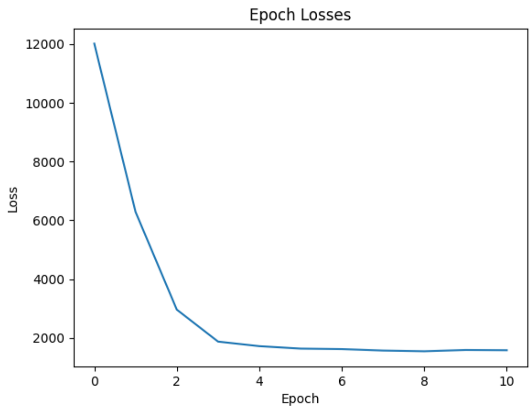
Train and Anomaly Data Formation

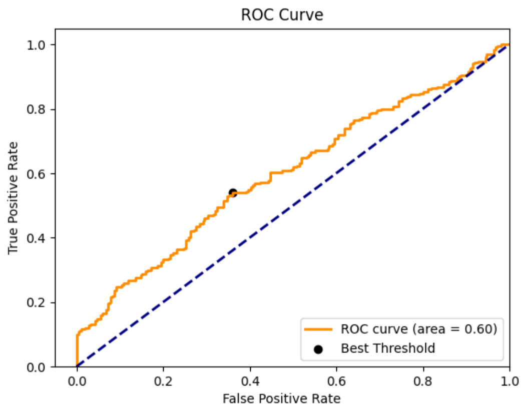
- Produce a small data set that has 6182 nodes and 9649 edges with 60 distinct edge types
- To train and test our model we sample small neighborhoods of the network as follows:
  - Randomly select a node v
  - Use breadth-first search to find a local neighborhood
  - Introduce artificial anomalies into the network
- We introduce dense unexpected relationships into the network in the form of cliques by:
  - Randomly select n nodes from the graph and form a clique amongst them.
  - Repeat m times to get a set of $m \times n$ anomalous samples.

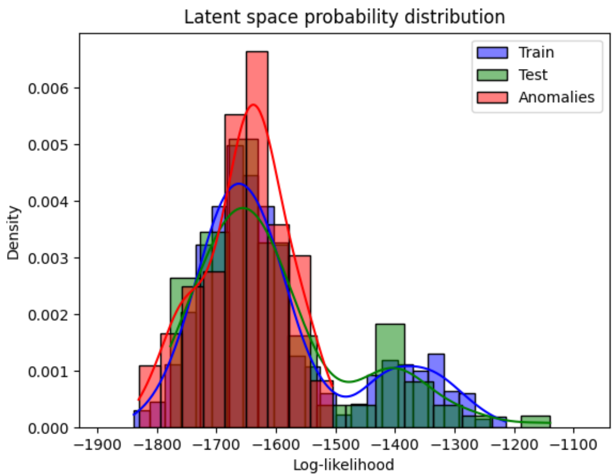
Numerical Experiments

- Evaluation Metrics
  - ROC-AUC
  - Precision
  - Recall
  - F1 score

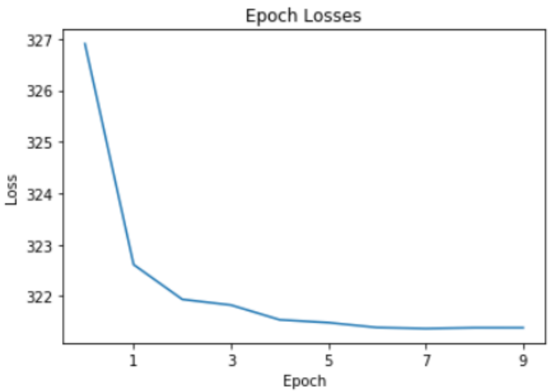
Results of the Flow Model

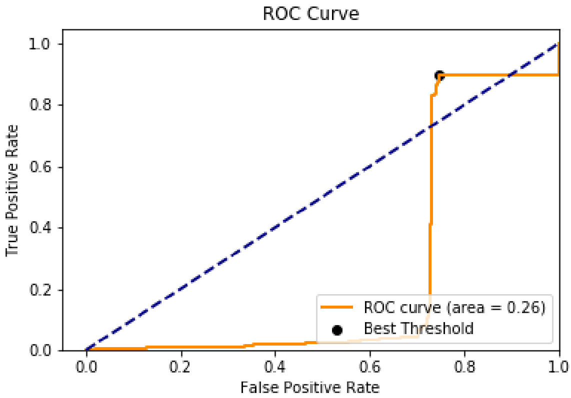


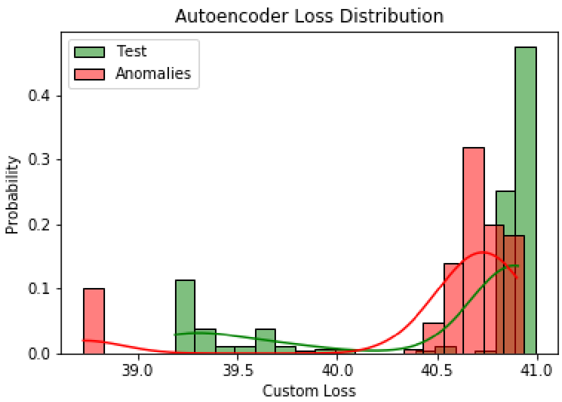




Results of Graph Autoencoder







Comparison

Model	Precision-50	Precision-100	Precision	Recall-50	Recall-100	Recall	F1-score	ROC-AUC
Flow model	0.740	0.670	0.598	0.148	0.268	0.536	0.565	0.601
Graph Autoencoder	0.500	0.284	0.284	0.100	0.100	0.100	0.148	0.257
% difference	-24.0%	-38.6%	-31.4%	-4.8%	-16.8%	-43.6%	-41.7%	-34.4%

Future work

- Test and extend the anomaly detectors on different network datasets (e.g., social networks, web-graph, or product co-purchasing networks) and more complex queries.
- Use stochastic optimization and distributed learning to accelerate the training process and deal with large network datasets.
- Investigate the robustness of the detectors in the presence of other types of anomaly.
- Train the network longer and experiment with deepening or widening the model.

Reference

1. Yong, Z. X., Torrent, T. T. . Semi-supervised deep embedded clustering with anomaly detection for semantic frame induction. In *Proceedings of The 12th Language Resources and Evaluation Conference* (pp. 3509-3519).
2. Ding, K., Li, J., Bhanushali, R., Liu, H. . Deep anomaly detection on attributed networks. In *Proceedings of the 2019 SIAM International Conference on Data Mining* (pp. 594-602).
3. Kipf, T.N. and Welling, M., 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907.*
4. Cer, D., Yang, Y., Kong, S.Y., et al., 2018, November. Universal sentence encoder for English. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations* (pp. 169-174).

Reference (continued)

1. https://towardsdatascience.com/introduction-to-normalizing-flows-d002af262a4b
2. https://www.ontotext.com/knowledgehub/fundamentals/what-is-a-knowledge-graph/