Link Prediction in Temporal Network

Evolve GAT

By
Akhil (2K19/CO/038)
Anshuman Raj Chauhan (2K19/CO/067)

01

Motivation

A wide range of systems can be modelled on a graph.
LP in these graphs therefore becomes a prolific task.

It has applications like
- Recommender systems
- Traffic prediction in networks,
- Biological interactions
- Terrorist network detection

Definition

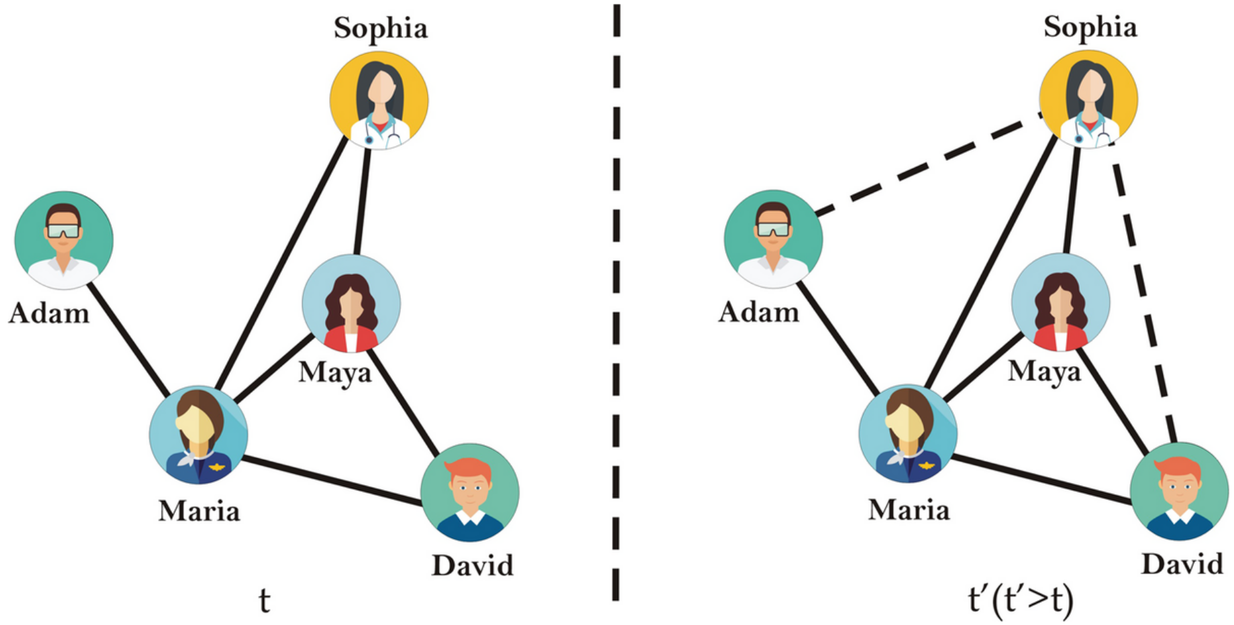
Link Prediction

Link Prediction is defined as the task of prediction of future edges in a Graph given that current/ current and previous states (Vertices and Edges) of the graph are provided.

Types of Networks

Static Networks

Prediction of the future State of graph provided current state G(V, E) is known.



LP in Static Network

Temporal (Dynamic) Networks

Prediction of future State of Graph provided state of Graph $G(V, E)$ with its evolutionary information is known.

- Snapshots

  The network is represented as a series of static networks, one for each time step.

- Contact Sequences

  If the duration of interaction is negligible, the network is represented as a triplet (i, j, t), where i and j are entities and t is the time of interaction.

- Interval Graph

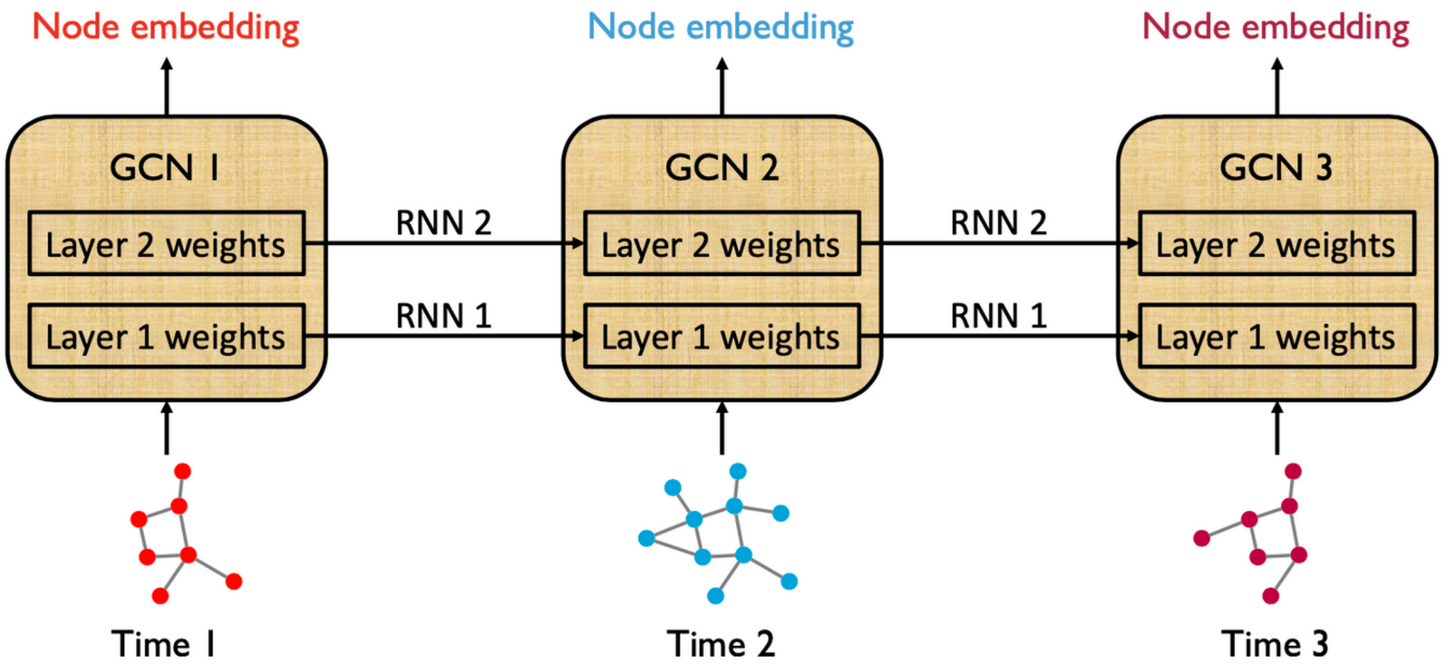
  If the duration of interaction is not negligible, $T(e) = \{(t1, t1'), \ldots, (tn, tn')\}$ over which the edge e is active.

Research Papers

EvolveGCN

EvolveGCN

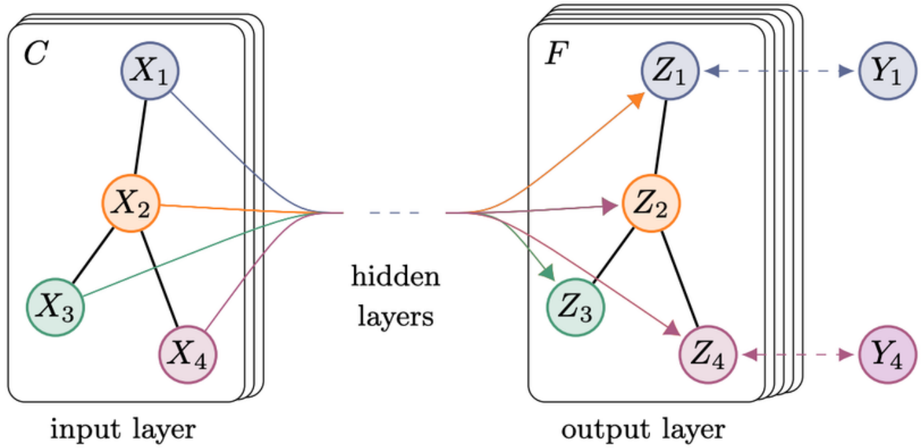
Evolve GCN combines Graph Convolution with LSTM to capture the network evolution through time.



Graph Convolution

Graph Convolution is very similar to Image convolution, but with the adaptiveness to work on a more complex data structure.

$$H_t^{(l+1)} = \mathrm{GCONV}(A_t, H_t^{(l)}, W_t^{(l)})$$
$$:= \sigma(\widehat{A}_t H_t^{(l)} W_t^{(l)}),$$




```
| 1 0 1 |     | 0.2 0.3 0.4 |     | 0.3 0.4 0.5 |
| 1 1 0 | @   | 0.3 0.1 0.5 | =   | 0.5 0.4 0.9 |
| 1 1 1 |     | 0.1 0.1 0.1 |     | 0.6 0.5 1.0 |
    A                 W                 H
```

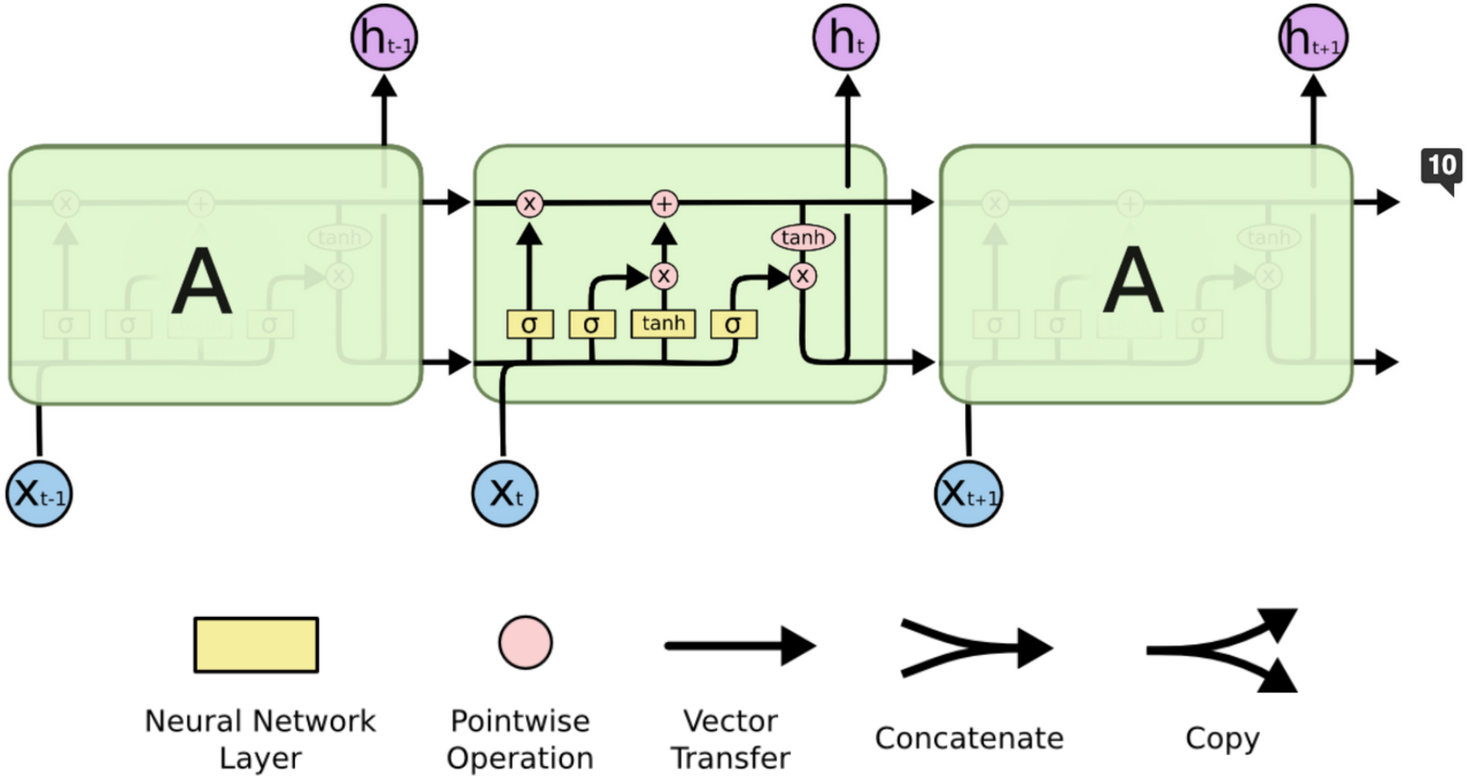
Simple Graph Convolution Operation

LSTM

LSTM(Long Short term Memory) cells are a variation of RNN's with the ability to capture long term dependencies in sequences.

COMPONENTS

- Input Gate -
- Output Gate -
- Forget Gate -
- Cell State -



Graph Attention Network (GAT)

GAT

Attention Mechanism helps in selectively concentrating on a few relevant things, while ignoring others in a deep neural networks.

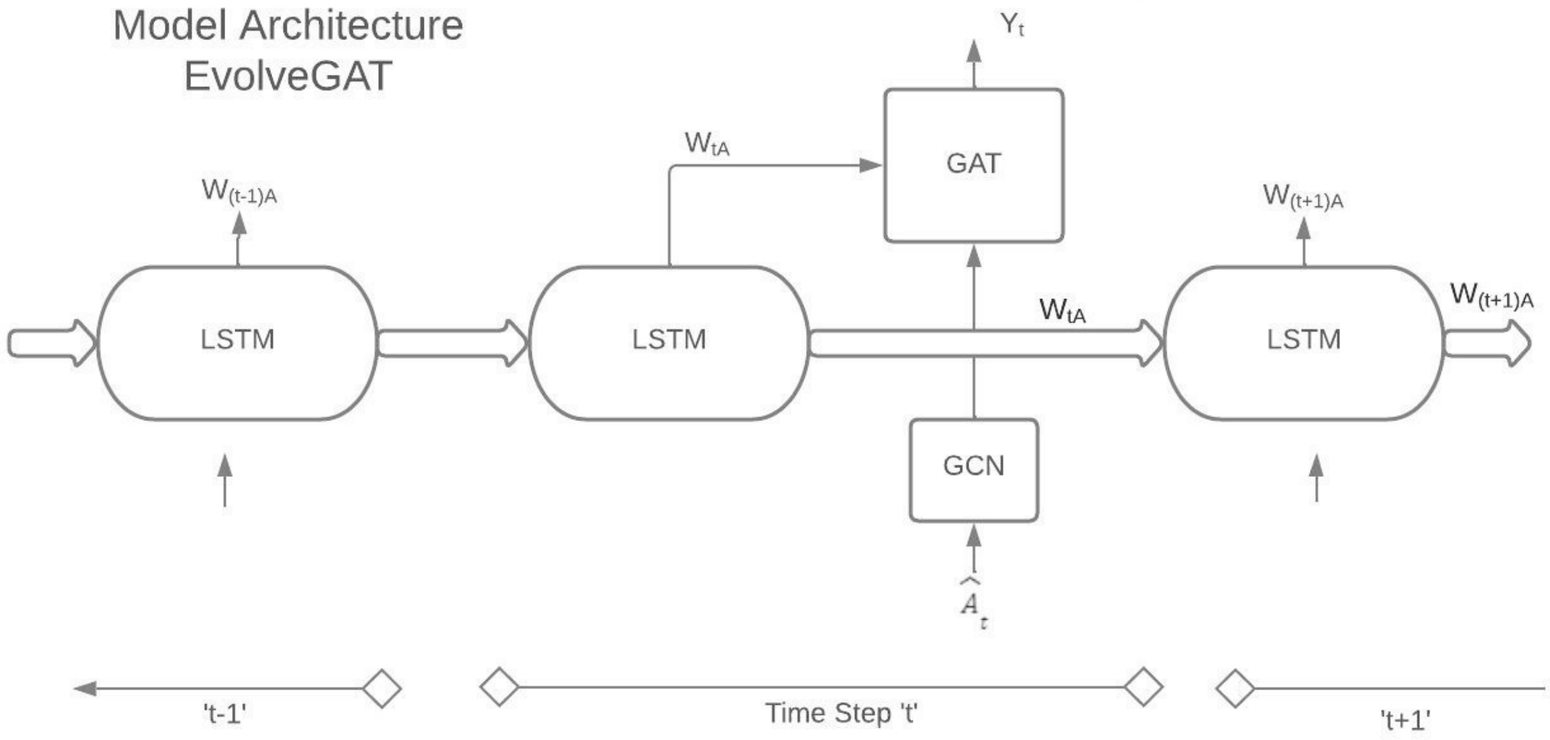
Graph Attention Network employs self attention features over node features to improve the node embeddings of a graph by giving higher weightage to embedding of nodes that are more important.

$$e_{ij} = a(\vec{h}_i, \vec{h}_j) \qquad \alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k \in \mathcal{N}_i} \exp(e_{ik})} \qquad \vec{h}_i' = \Big\|_{k=1}^{K} \sigma\left(\sum_{j \in \mathcal{N}_i} \alpha_{ij}^k \mathbf{W}^k \vec{h}_j\right)$$

EvolveGAT Model

EvolveGAT Model

Node Embedding - Graph Attention Network for node embedding with its parameters being generated by the LSTM at each time stamp. (Note - Only LSTM parameters are trained while GAT param. are generated by the LSTM.)



Edge Classification - Similarity index using inner join b/w embeddings.

Problems Faced

- **Excess GAT Parameters**

  The excess parameters were significantly increasing the training time. So to reduce the training time, we

  1. Used LSTM to generate and train GCN layer's weight.
  2. Removed some learnable parameters from GAT's layer.

- **Imbalanced Dataset**

  Sparse nature of graph made the dataset highly imbalanced. We balanced the dataset using **down-sampling**.

- **Large Graph**

  Due to the large number of nodes, the GAT weight matrix was too large to be generated by LSTM, so a Linear layer was added before GAT to reduce the dimensions.

Novelty

- **GAT** integration for better embedding generation.

- Developed a **simpler attention mechanism** to reduce training parameters.

- **Many to Many LSTM architecture** was used instead of many to one, in order to facilitate training at each time step.

Model Training

Model Summary

Dataset Used - BitCoin OTC

https://pytorch-geometric.readthedocs.io/en/latest/modules/datasets.html#torch_geometric.datasets.BitcoinOTC

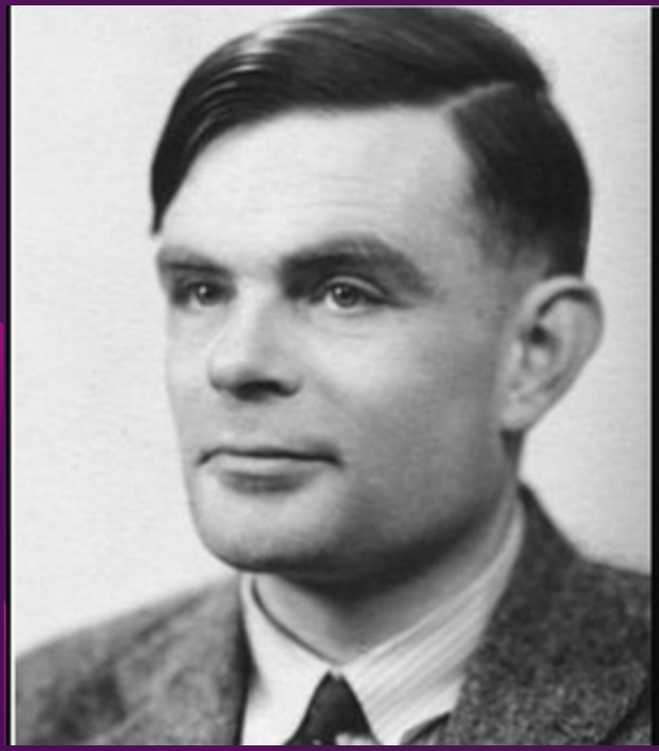
The dataset consisting of 138 who-trusts-whom networks of sequential time steps.

Avg. Nodes - 6000
Avg. Edges - 35600
Training Time Steps - 40
Testing Time Steps - 10
Training Time - 1.5 hrs

Final **Loss** on training Data - **0.36**
AUC Score on Test Data - **0.61**

Conclusion

We developed and implemented a novel approach for link prediction in temporal networks that has a good scope for further improvement and the potential to replace the current static link prediction methods with the dynamic ones.



Sometimes it is the people no one imagines anything of who do the things that no-one can imagine

— Alan Turing —