Temporal Graph Learning for Financial World:

Algorithms, Scalability, Explainability & Fairness

KDD 2022 Tutorial

Nitendra Rajput

VP & Head – AI Garage, Mastercard

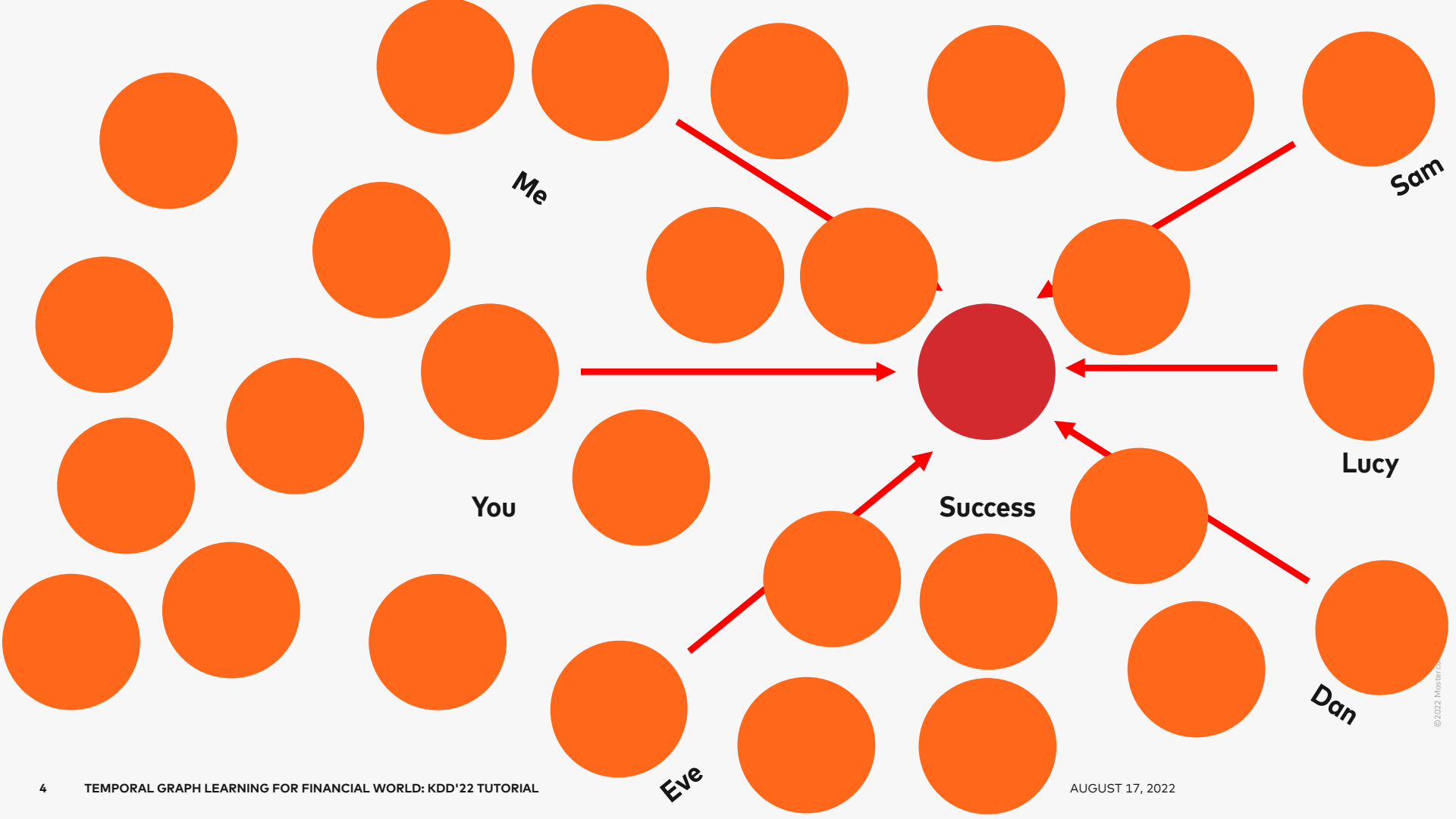
Karamjit Singh

Director – AI Garage, Mastercard

Acknowledgements: 20+ Data Scientists & Engineers at AI Garage who work in this area.

Why is the Mona Lisa the most successful painting of all time?

https://www.cbc.ca/player/play/2016438339762



Rules: Defined Decisions

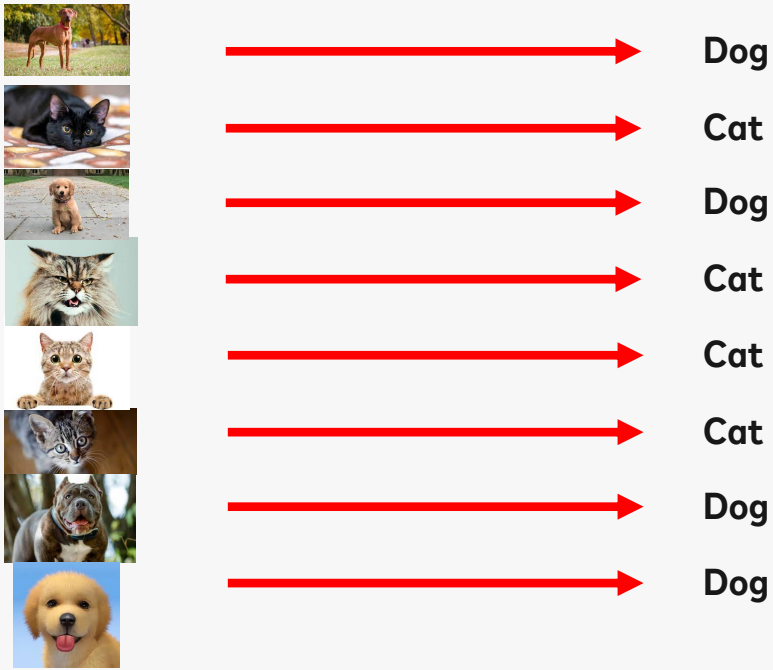
Rules: Defined Decisions

Cat

Rules: Defined Decisions

Dog

Learn the “rules”: Defined Decisions



Rules: Socially Accepted Decisions

You

Stock Price

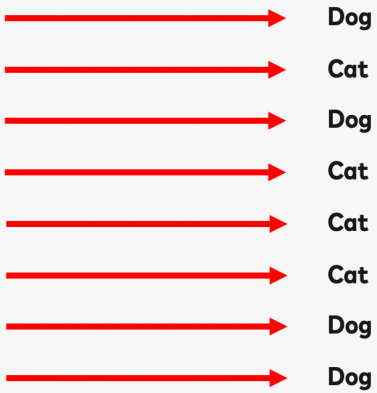
Credit Decisioning

Fraudulent Transaction

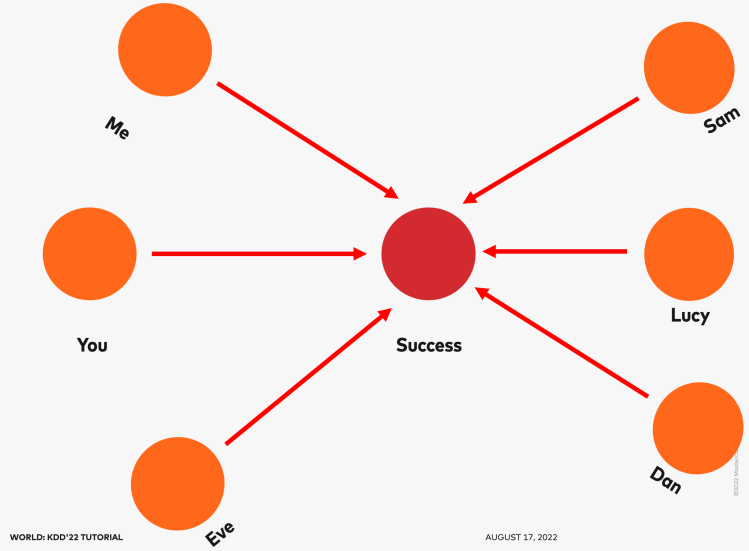
Delinquency

Money Laundering

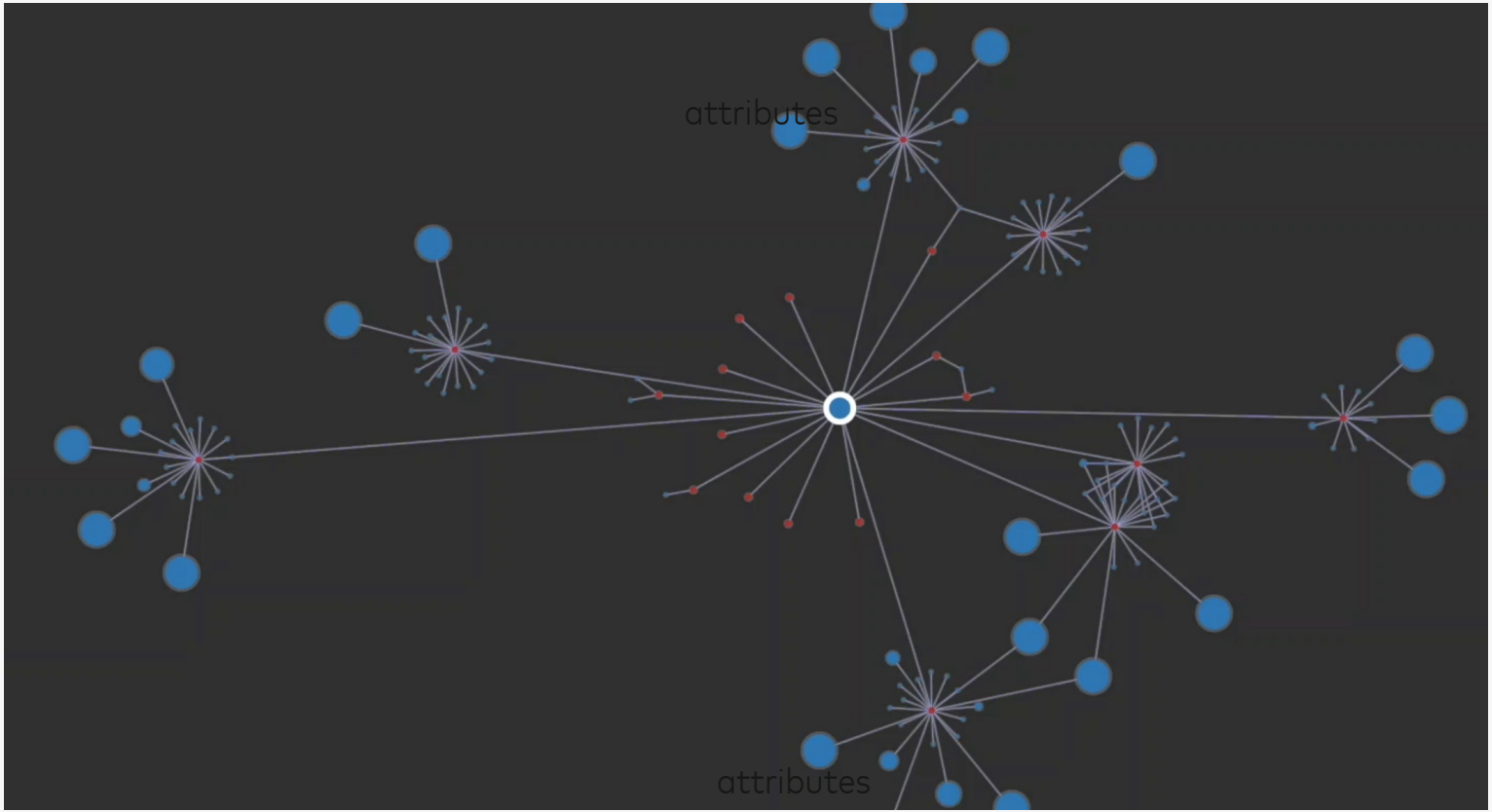
Learn the “rules”: Defined Decisions



Learn to make Socially Accepted Decisions



What defines a Graph?



Machine Learning on Transaction Graph

Predict the next edge

Next Transaction

Predict the properties of the next edge

Next Merchant

Amount

Time

CardPresent / CardNotPresent / Contactless

Machine Learning on Transaction Graph

Predict the dying node

Attiring cardholder/merchant

Predict the properties of the nodes

Non-sufficient Funds

Crypto Merchant

Breached Merchant

Breached Card

Agenda

1. **Foundation**
   - What is Graph Neural Networks
   - Graph Embeddings
   - Application of Graph neural networks
   - Graphs in Financial world
2. **Transactions as a Graph**
   - Applications of transactional Graph
     - Node level, Edge level, and Graph level
3. **Requirements and Challenges of GNNs for Financial world**
4. **State-of-the-art in GNNs**:
   a) Scalable
   b) Temporal
   c) Bi-Partite
   d) Fairness and Universal
   e) Transfer Learning in GNNs
5. **Open Research Problems**
6. **Q&A**

What is Embedding and Why?

5+3-2 = ?

KING – MAN + WOMAN = ???

BIKING - TODAY + YESTERDAY = ???

BROTHER - BOY + GIRL = ???

What is Embedding and Why?

KING – MAN + WOMAN = QUEEN

BIKING - TODAY + YESTERDAY = BIKED

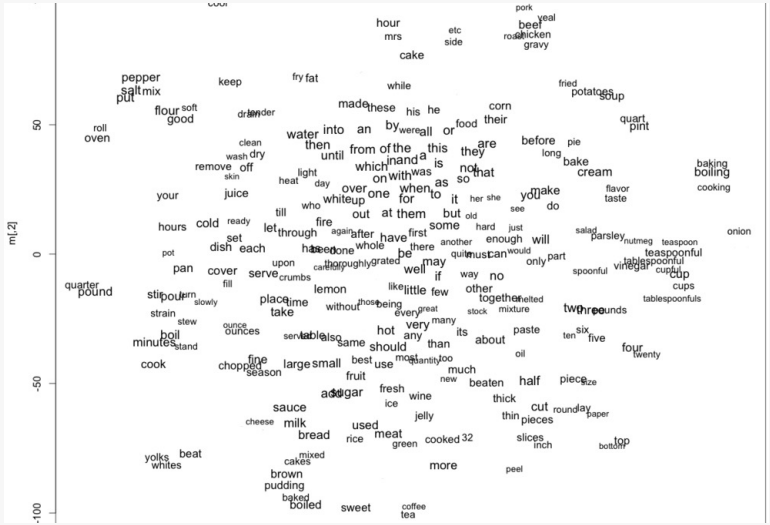
BROTHER - BOY + GIRL = SISTER

What is Embedding and Why?

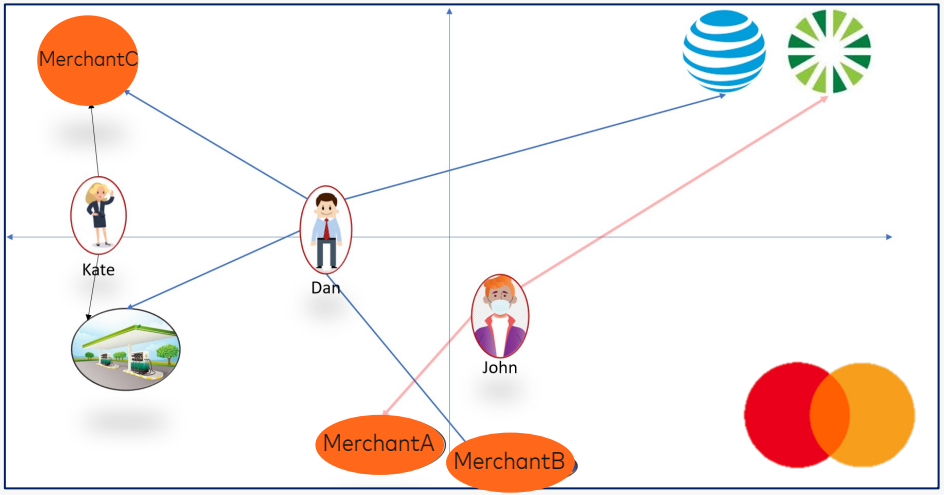
WORD → AI : f(WORD) → 0 | 3 | 7 | -2

< Embedding >

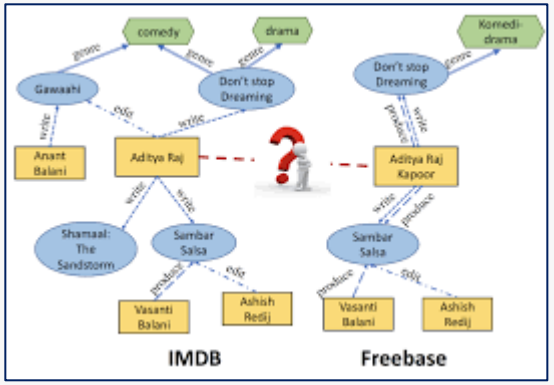
Words	Numeric Representation
KING	[1, 3, 5, 2, 0]
QUEEN	[1, 3, 8, 2, 5]
MAN	[4, 5, 8, 9, 2]
WOMAN	[4, 4, 8, 7, 6]

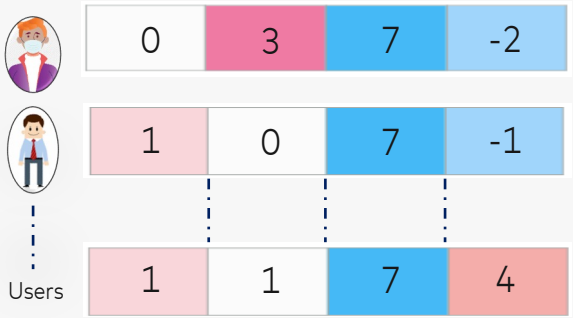


Richness unique to Mastercard

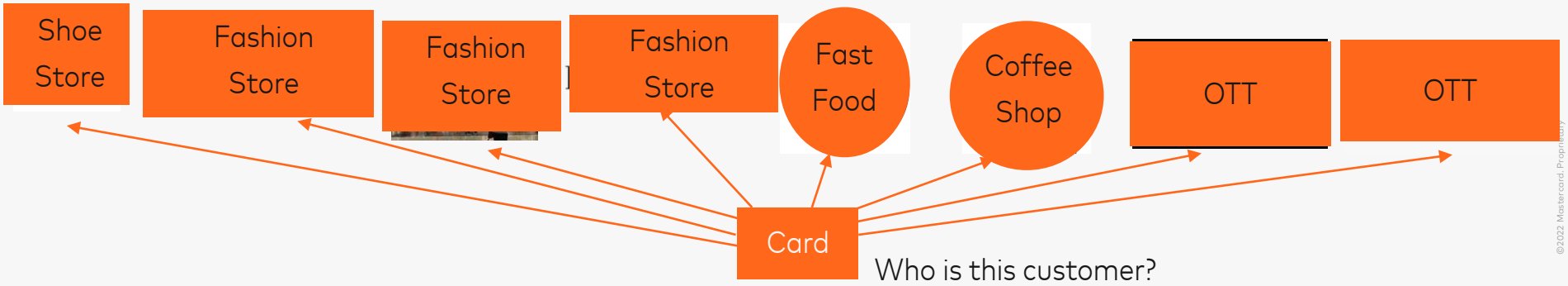


eCommerce Graph

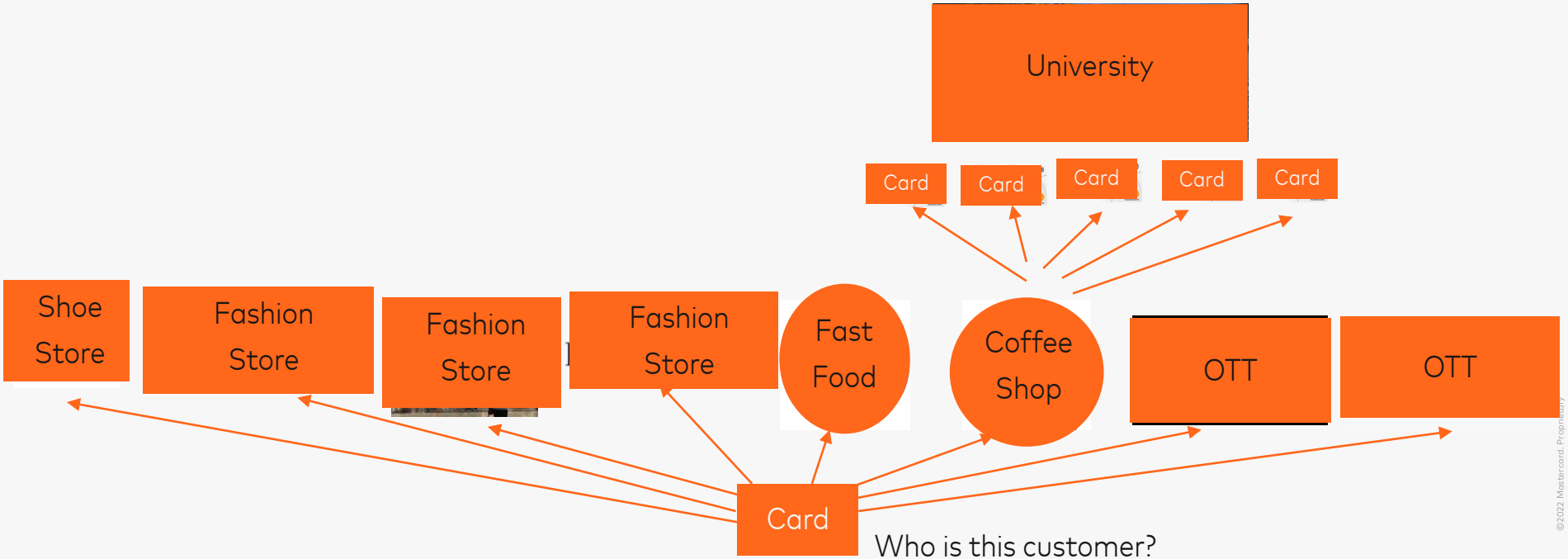




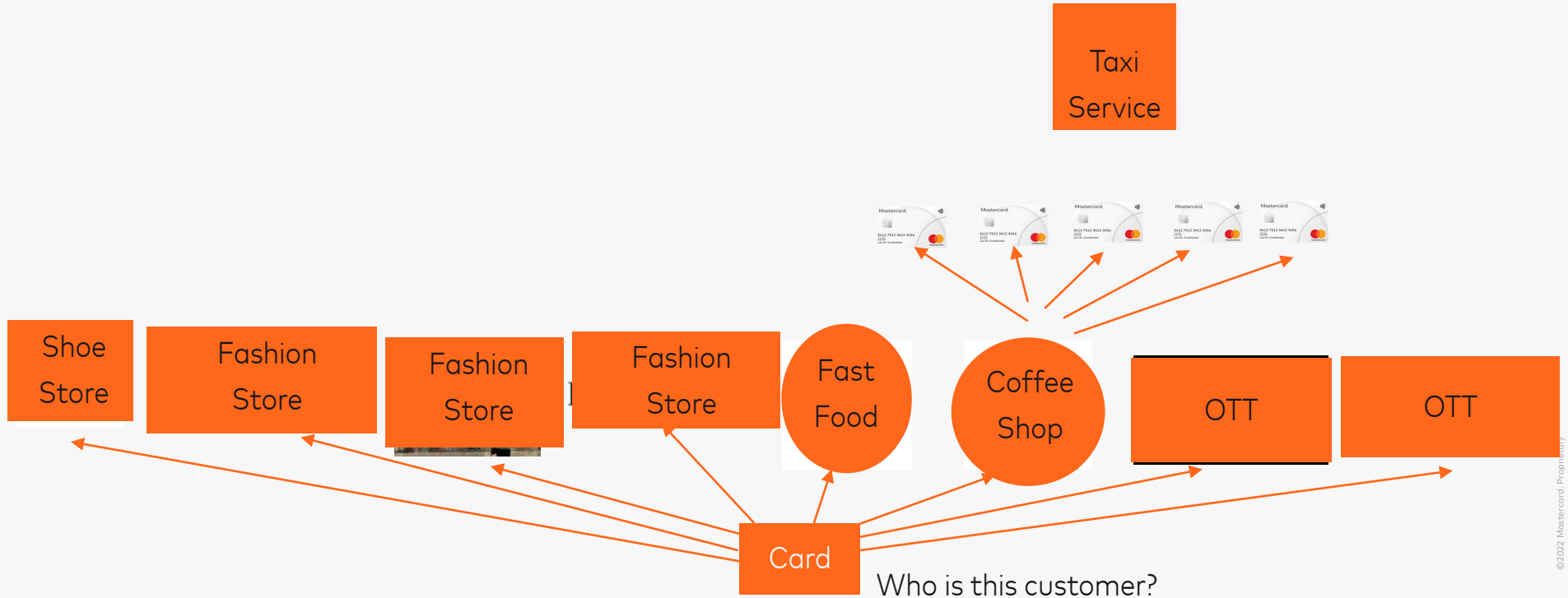
How we learn from Graphs



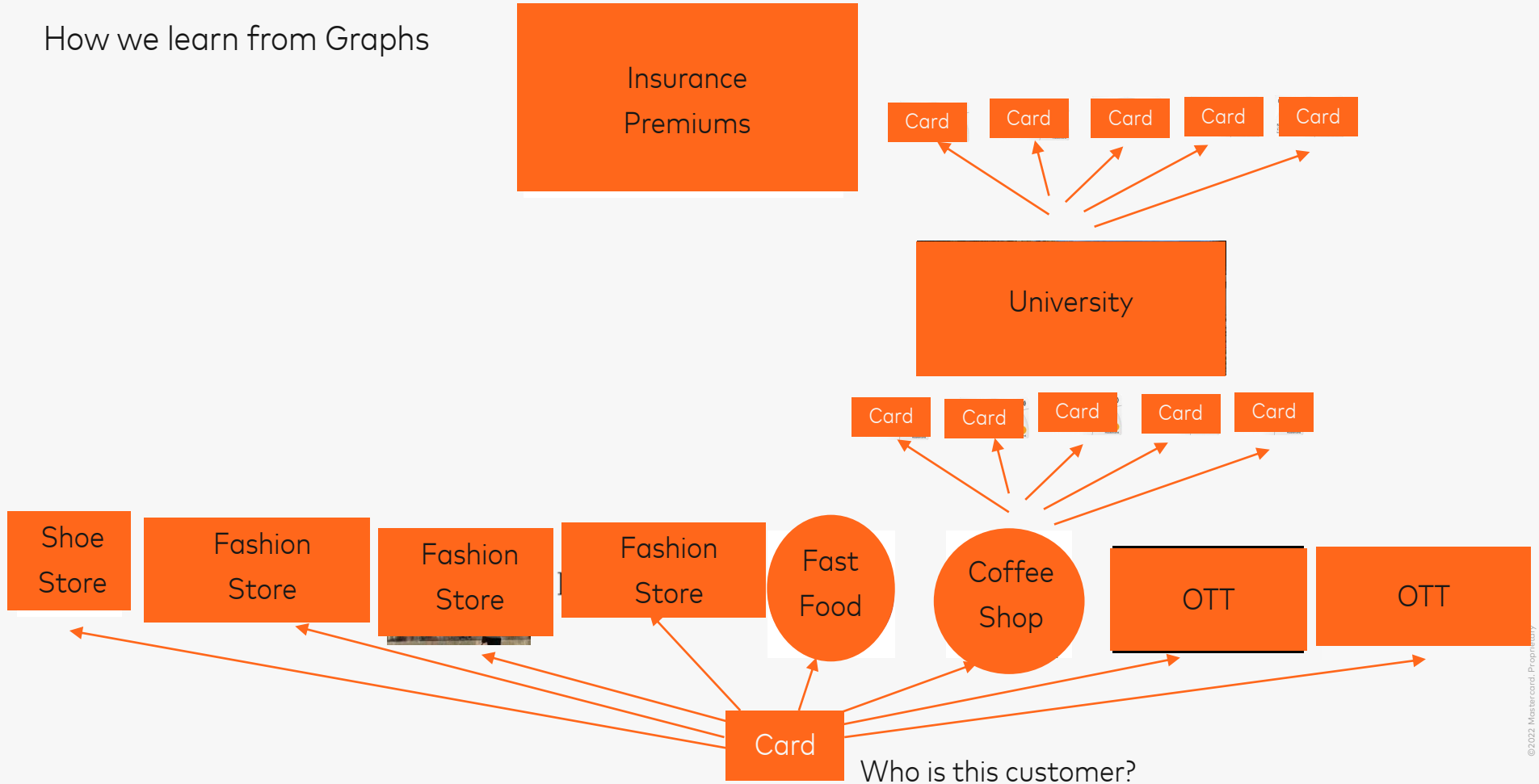
How we learn from Graphs



How we learn from Graphs



How we learn from Graphs



How we learn from Graphs

Insurance Premiums

Card Card Card Card Card

University

Card Card Card Card Card

Shoe Store | Fashion Store | Fashion Store | Fashion Store | Fast Food | Coffee Shop | OTT | OTT

Card

Who is this customer?

How we learn from Graphs

Insurance Premiums

Card Card Card Card Card

University

Card Card Card Card Card

Shoe Store

Fashion Store

Fashion Store

Fashion Store

Fast Food

Coffee Shop

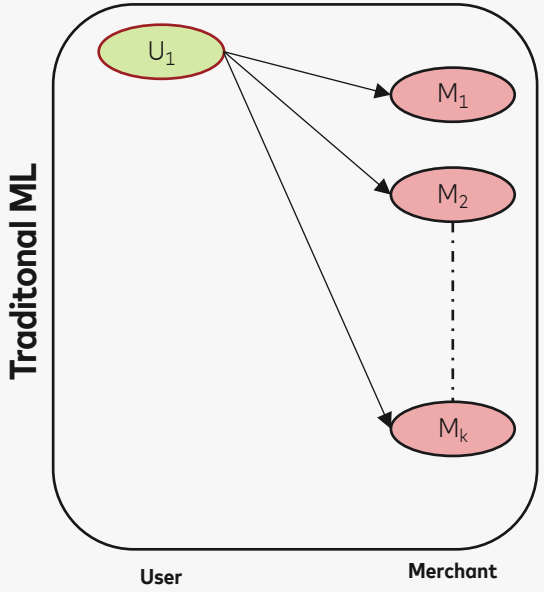
OTT

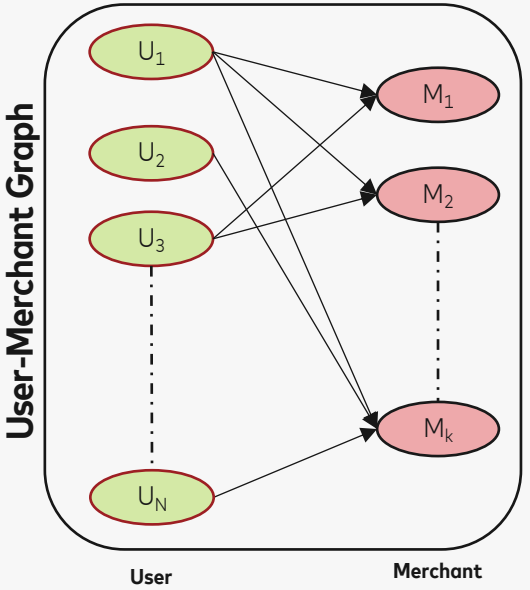
OTT

Card

Who is this customer?

Why Graphs??





T-ML:

1. No fixed length vector, could be very large
2. Information is limited to 1st order
3. Does not capture the context of other users/merchants (unique richness of Mastercard)

Graph:

1. Fixed length (compressed)
2. Enriched representations
3. Captures the context of other users and merchants

Graph Neural Network

Representing Graphs

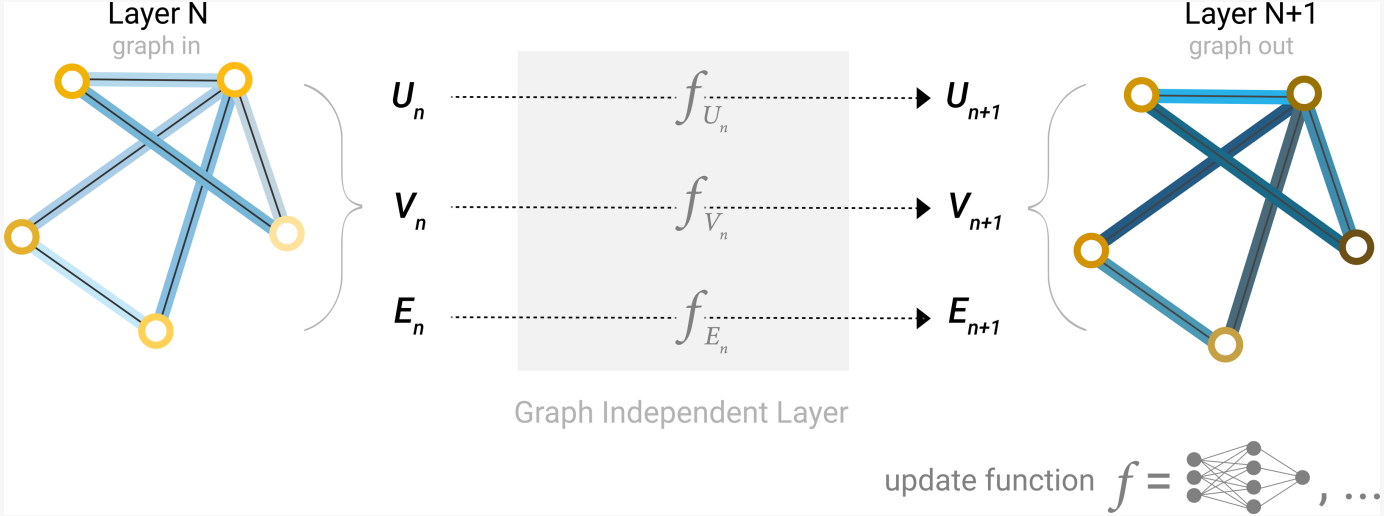
Nodes: Form a node feature matrix N by assigning each node an index i and storing the feature for $node\ i$ in N.

Edges: Form an edge feature matrix E by assigning each edge an index i and storing the feature for $node\ i$ in E.

Connectivity: Adjacency matrix is typically huge and sparse.

Connectivity in a Graph is best represented through an Adjacency list which has each edge e_k between node n_i and n_j being represented as a tuple *(i,j)* in the adjacency list.

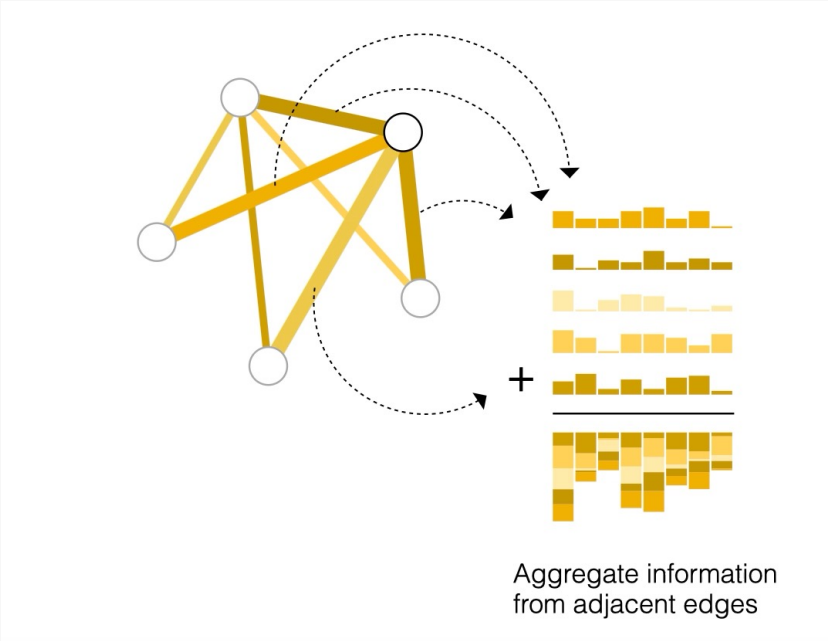
A Basic Graph Neural Network



From: https://distill.pub/2021/gnn-intro/

New embeddings, same structure

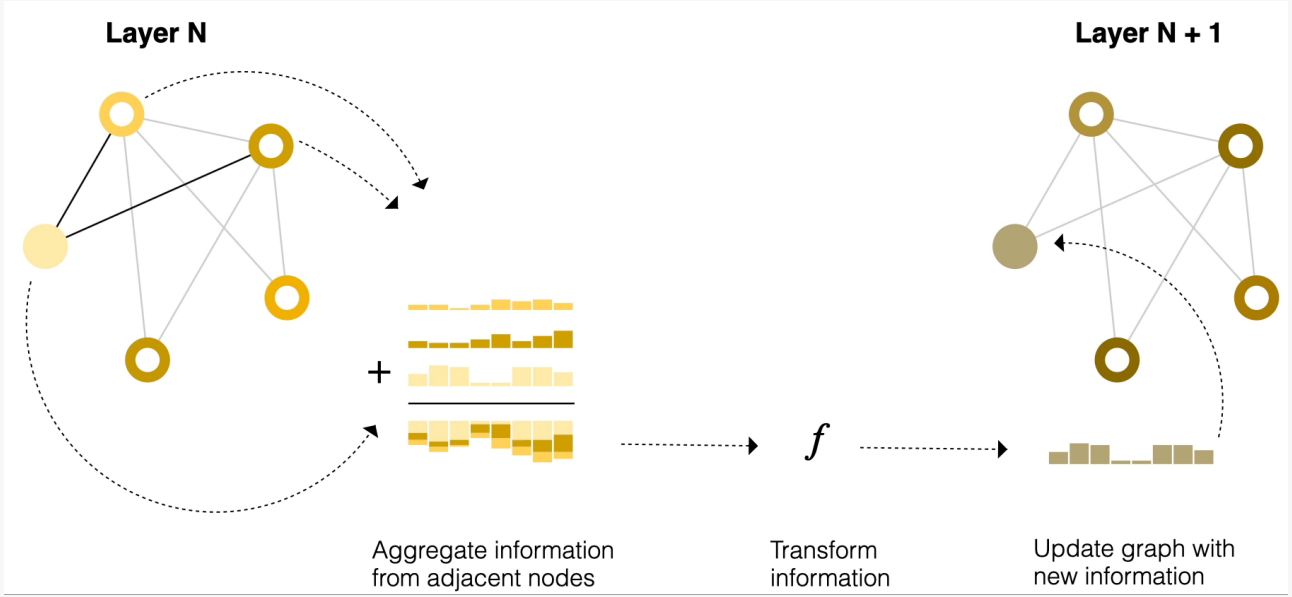
Pooling: To route information between different part of Graph



From: https://distill.pub/2021/gnn-intro/

Perform prediction tasks

Message-Passing: Influence representations based on the neighbours

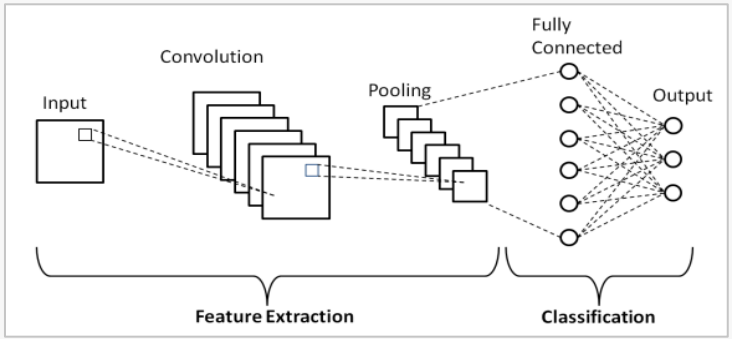


From: https://distill.pub/2021/gnn-intro/

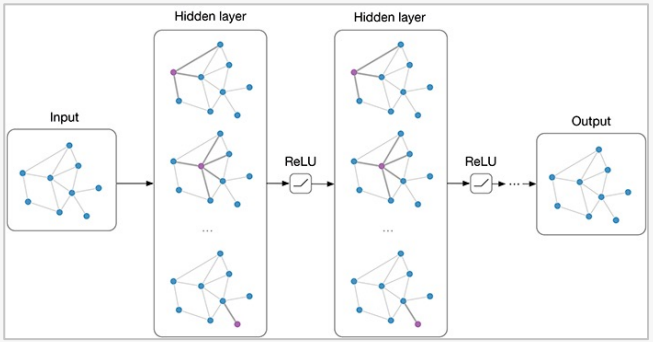
Gather neighbour node embeddings, aggregate function, update function

Analogy between CNN and GCN

- CNNs were developed for grid-like data – essentially image data
- In Image data pixels are arranged in a grid-like structure
- An Image can be thought of as a graph where each pixel is a node and is connected to all of its neighbouring pixels



- Graph structure present challenges to apply usual CNNs architecture because –
  - They are irregular
  - There is no ordering to the neighbours nodes and even if ordered, they don't have any semantic meaning



- General graphs do not have a canonical spatial representation; ConvGNNs perform convolution on the spectrum of the graph, given by eigenvectors of its Laplacian

- Traditional ConvGNNs calculate eigenvalues of Laplacian which can be expensive hence first order approximation of convolution was proposed as Graph Convolution network

Challenges with Graphs in the Financial World

Scalability

100s of millions of nodes, billions of edges

Temporal

Rich information in time-domain, multiple edges across same nodes

Heterogenous

Many entities: user, merchant, bank, location, industry

Inductive

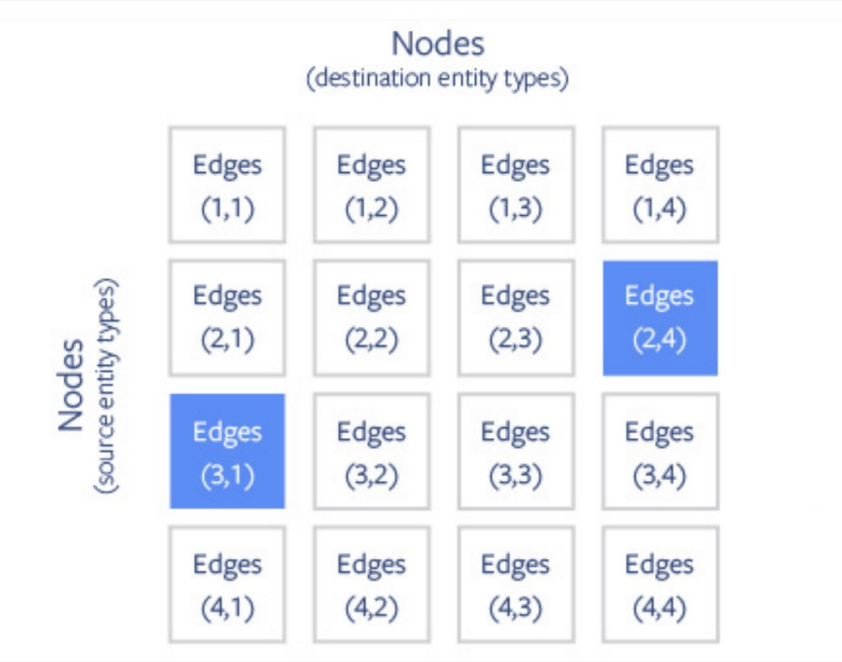
Addition and deletion of nodes and edges

Scalable Graphs

Scalable Graphs - Overview

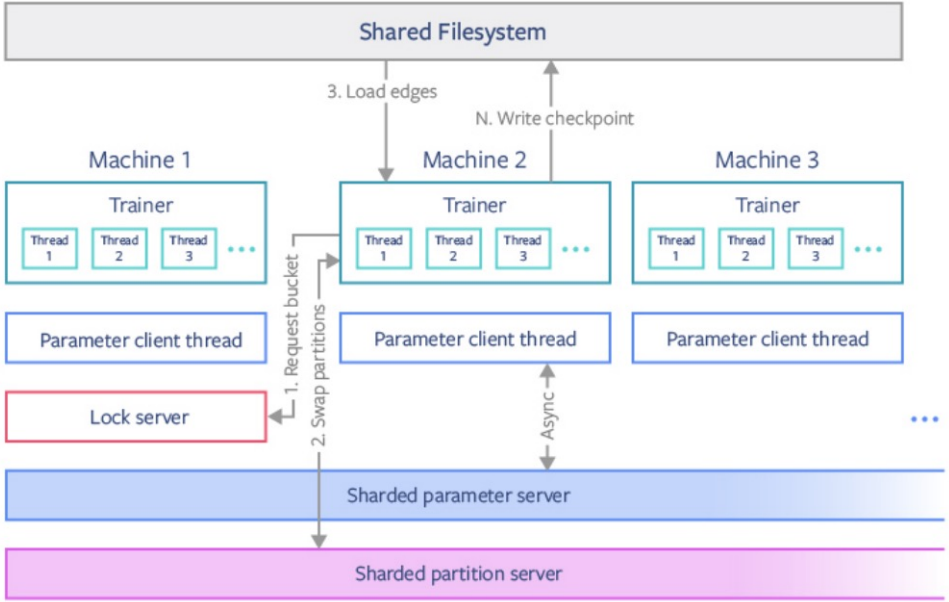
Model Type	Conference
PyTorch BigGraph	Machine Learning and Systems 1 (MLSys 2019)
Cleoa	ICONIP 2021
SIGN	Graph Representation Learning and Beyond (GRL+) Workshop at, ICML 2020

Pytorch BigGraph: Creating embeddings of graphs with billions of nodes



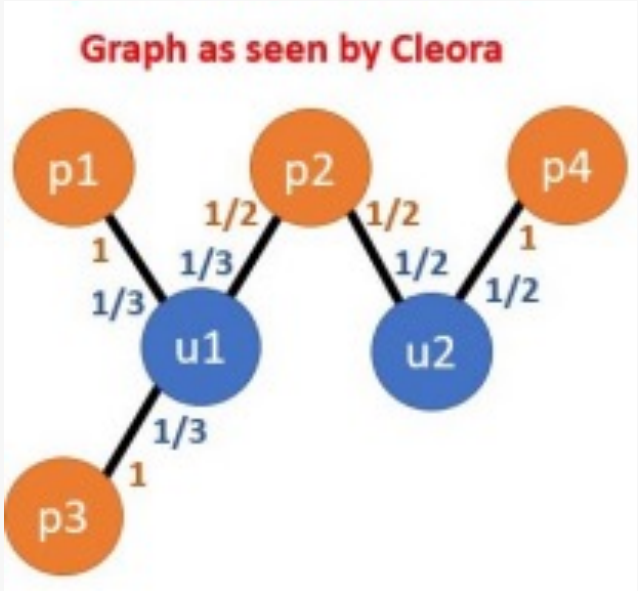
- Nodes are divided into P partitions that are sized to fit in memory. Edges are divided into buckets based on the partition of their source and destination node
- Once the nodes and edges are partitioned, training can be performed on one bucket at a time **or** in distributed mode

Distributed training



- PyTorch parallelization primitives is used for distributed training. A classical parameter server model
- Machines coordinate to train on disjoint buckets using a lock server
- Partitioned model parameters are exchanged via a sharded partition server, and shared parameters are updated asynchronously via a sharded parameter server

Cleora: Overview

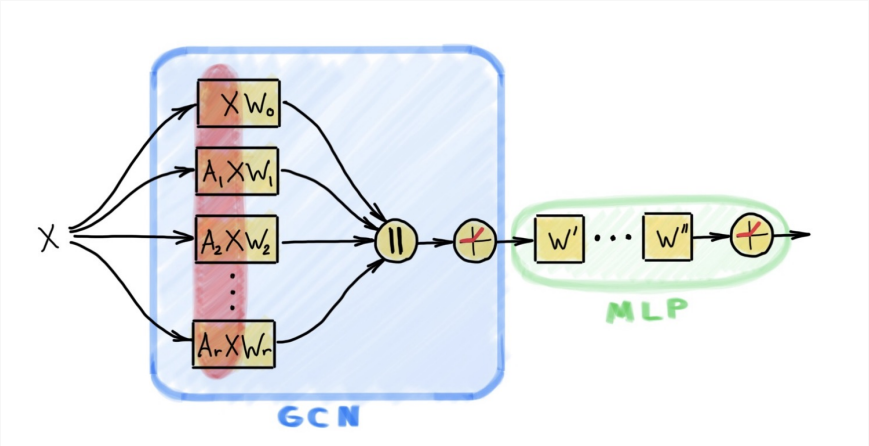


- Incoming edge weight is inversely proportional to the degree of the node
- Iterative averaging of neighbouring node feature is performed to ensure, nodes with similar neighbours are close to each other
- Avoiding the expensive negative sampling or objective function maximization/minimization makes this approach highly scalable

SIGN: Scalable Inception Graph Neural Networks

- . The key building block of our architecture is a set of linear diffusion operators represented as $n \times n$ matrices $A_1 \ldots A_r$ whose application to the node-wise features can be pre-computed.

$$Z=\sigma([X\,\Theta_0, A_1 X\Theta_1, \ldots, A_n\, X\Theta_n)$$
$$Y = \xi\,(Z\Omega)$$



SIGN: Scalable Inception Graph Neural Networks

- **Inception-like module Within the SIGN framework**
  - It is possible to choose one specific operator B and to define $A_k = B_k$ for k = 1, . . . , r.
  - In this setting the proposed model is analogous to the popular Inception module for classic CNN architectures: it consists of convolutional filters of different sizes determined by the parameter r, where r = 0 corresponds to 1 × 1 convolutions in the inception
  - It is also easy to observe that various graph convolutional layers can be obtained as particular settings of SIGN. In particular, by setting the σ non-linearity to PReLU , ChebNet, GCN, and S-GCN can be automatically learnt if suitable diffusion operator B and activation ξ are used.

- **Choice of the operators**
  - The choice of the diffusion operators jointly depends on the task, graph structure, and the features.
  - In complex networks such as social graphs, operators induced by triangles or cliques might help distinguishing edges representing weak or strong ties .In graphs with noisy connectivity, it was shown that diffusion operators based on Personalized PageRank (PPR) or Heat Kernel can boost performance .
  - We denote by SIGN(p,s,t) with r = p + s + t the configuration using up to the p-th, s-th, and t-th power of simple GCN-normalized, PPR-based, and triangle-based adjacency matrices, respectively.

Temporal Graphs

Temporal Graphs - Overview

Model Type	Model
GNNs with Discrete Time	Evolve GGCN
	GCLSTM
	GCN-GRU
GNNs with Continuous Time	TGAT
	TGNN
	SIGN
GNNs with Temporal Point Process	TREND
	TeGraf

Temporal Graphs – Discrete Time

GCN-GRU : Motivation

How to Model Uncertainty of a binary prediction ?

Binomial Opinion in Subjective Logic(SL)

A binomial opinion ω is represented by

$\omega = (b, d, u, a)$

Where:

- b: belief (e.g., agree)
- d: disbelief (e.g., disagree)
- u: uncertainty (i.e., ignorance, vacuity, or lack of evidence)
- a: base rate (i.e., a prior knowledge)

$b + d + u = 1$

Subjective Logic(SL) limitations

1. Lack of Scalability to deal with Large Networks.
2. Limited capability to handle heterogeneous topological and temporal dependencies among node- level opinions.
3. A high sensitivity with conflicting evidence that may generate counterintuitive opinions derived from the evidence.

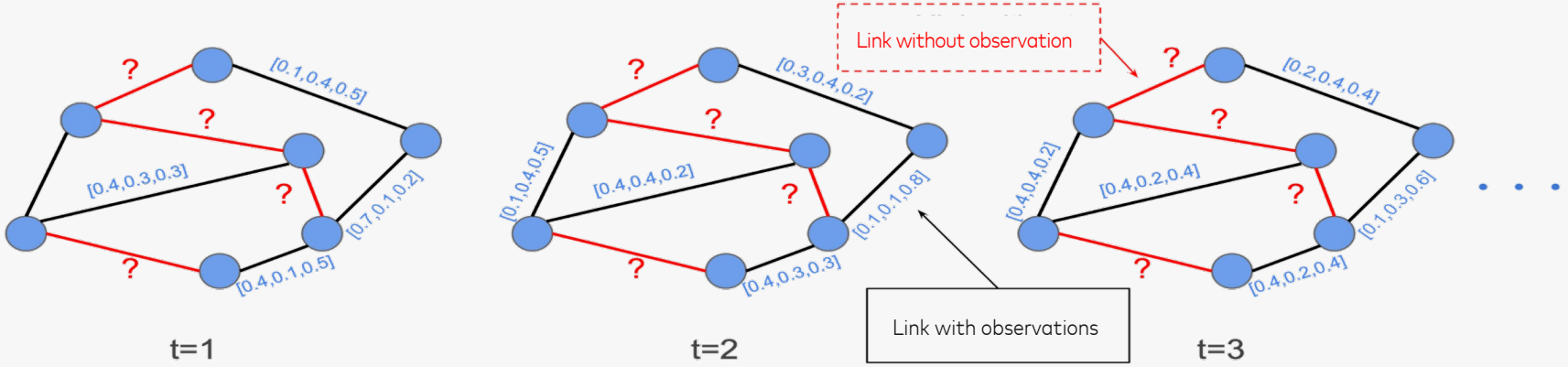
How to Solve These Challenges

Graph Convolutional Network and Gated Recurrent Units can provide solutions for

- ❖ dealing with graph network data
- ❖ modelling topological and temporal-heterogeneous dependency
- ❖ processing large-scale data (i.e., scalability)

Problem & Contributions

Given a Network G=(V,E), some of the edges in E have unknown uncertain opinions, the goal is to predict uncertain opinion of other edges.



Key Contributions:

- Considered GCN and GRU for modelling the **topological** and **temporal** heterogeneous dependency information.
- Modelled **conflicting** opinions based on robust statistics.
- Developed a highly scalable inference algorithm to predict dynamic, uncertain opinions in a **linear** computation time.

Proposed Approach : GCN-GRU-based Opinion

The Observed Beliefs and Uncertainties

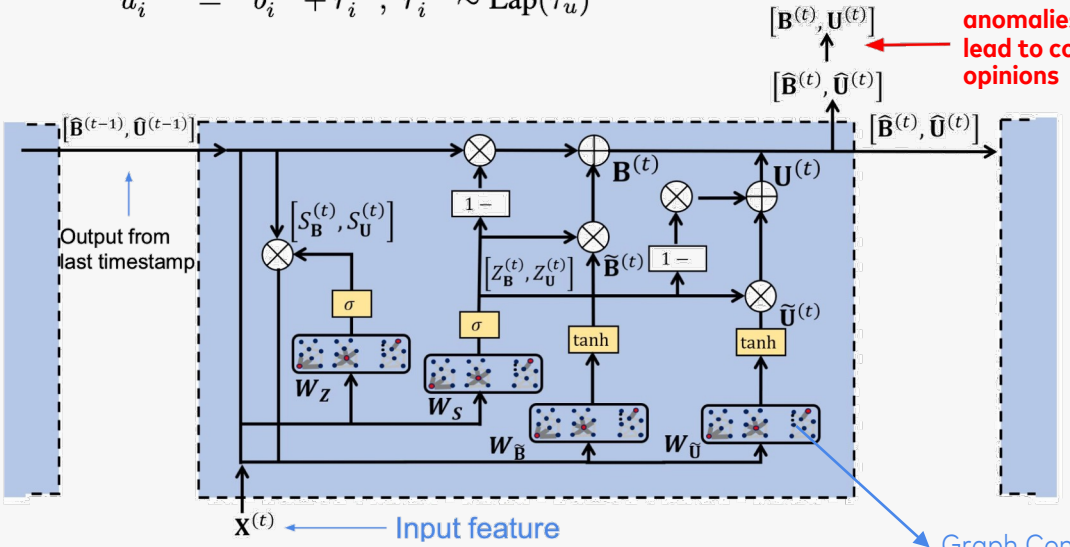
are considered noise version of their true value

$$b_i^{(t)} = \hat{b}_i^{(t)} + e_i^{(t)},\ e_i^{(t)} \sim \text{Lap}(\tau_b)$$
$$u_i^{(t)} = \hat{b}_i^{(t)} + r_i^{(t)},\ r_i^{(t)} \sim \text{Lap}(\tau_u)$$

Laplacian Distribution

robust statistics to handle anomalies that lead to conflicting opinions

$$\begin{aligned}
b_i^{(t)} &= \hat{b}_i^{(t)} + e_i^{(t)},\ e_i^{(t)} \sim \text{Lap}(\tau_b), i \in \mathbb{L}_t, \\
u_i^{(t)} &= \hat{u}_i^{(t)} + r_i^{(t)},\ r_i^{(t)} \sim \text{Lap}(\tau_u), i \in \mathbb{L}_t, \\
\hat{\mathbf{B}}^{(t)} &= \mathbf{Z}_{\mathbf{B}}^{(t)} \odot \hat{\mathbf{B}}^{(t-1)} + (1 - \mathbf{Z}_{\mathbf{B}}^{(t)}) \odot \tilde{\mathbf{B}}^{(t)}, \\
\hat{\mathbf{U}}^{(t)} &= \mathbf{Z}_{\mathbf{U}}^{(t)} \odot \hat{\mathbf{U}}^{(t-1)} + (1 - \mathbf{Z}_{\mathbf{U}}^{(t)}) \odot \tilde{\mathbf{U}}^{(t)}, \\
\left[\mathbf{Z}_{\mathbf{B}}^{(t)}, \mathbf{Z}_{\mathbf{U}}^{(t)}\right] &= \sigma\left(\left[g_{\mathbf{W}_{\mathbf{Z}}} \star [\hat{\mathbf{B}}^{(t-1)}, \mathbf{X}^{(t)}], g_{\mathbf{W}_{\mathbf{Z}}} \star [\hat{\mathbf{U}}^{(t-1)}, \mathbf{X}^{(t)}]\right]\right), \\
\left[\mathbf{S}_{\mathbf{B}}^{(t)}, \mathbf{S}_{\mathbf{U}}^{(t)}\right] &= \sigma\left(\left[g_{\mathbf{W}_{\mathbf{S}}} \star [\hat{\mathbf{B}}^{(t-1)}, \mathbf{X}^{(t)}], g_{\mathbf{W}_{\mathbf{S}}} \star [\hat{\mathbf{U}}^{(t-1)}, \mathbf{X}^{(t)}]\right]\right), \\
\tilde{\mathbf{B}}^{(t)} &= \tanh\left(g_{\mathbf{W}_{\tilde{\mathbf{B}}}} \star [\mathbf{S}_{\mathbf{B}}^{(t)} \odot \hat{\mathbf{B}}^{(t-1)}, \mathbf{X}^{(t)}]\right), \\
\tilde{\mathbf{U}}^{(t)} &= \tanh\left(g_{\mathbf{W}_{\tilde{\mathbf{U}}}} \star [\mathbf{S}_{\mathbf{U}}^{(t)} \odot \hat{\mathbf{U}}^{(t-1)}, \mathbf{X}^{(t)}]\right).
\end{aligned}$$



- Maximize the log probability function

$$\mathcal{L}(\Theta, \{\hat{\mathbf{B}}_{\mathcal{L}_t}, \hat{\mathbf{U}}_{\mathcal{L}_t}\}_{t=1}^{T}) = \log \prod_{t=1}^{T} \prod_{i \in \mathbb{L}_t} \left(\text{Prob}(b_i^{(t)} | \hat{b}_i^{(t)}; \tau_b) \text{Prob}(u_i^{(t)} | \hat{u}_i^{(t)}; \tau_u)\right)$$
$$= -\sum_{t=1}^{T} \sum_{i \in \mathbb{L}_t} \left(\frac{|b_i^{(t)} - \hat{b}_i^{(t)}|}{\tau_b} + \frac{|u_i^{(t)} - \hat{u}_i^{(t)}|}{\tau_u}\right)$$

Evolve GCN - Motivation

How to model evolving Graph over time?

Current Approach & it's Limitations

- Existing approaches typically resort to node embeddings and use a recurrent neural network (RNN, broadly speaking) to regulate the embeddings and learn the temporal dynamics.
- These methods require the knowledge of a node in the full time span
- Less applicable to the frequent change of the node set.
- It is challenging for RNNs to learn these irregular behaviours.

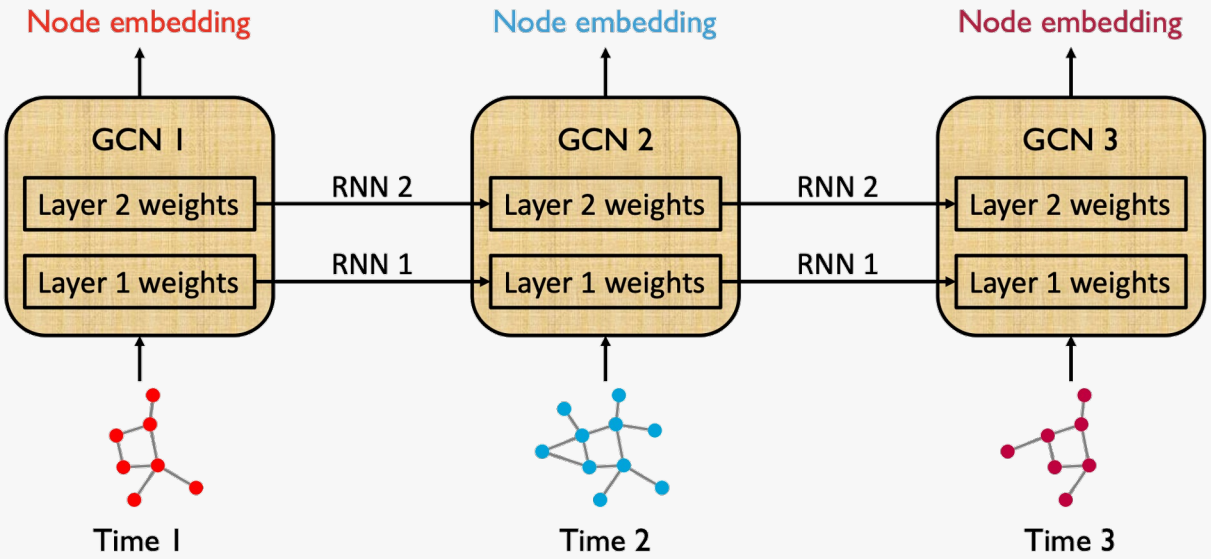
Examples:

- In financial networks, transactions naturally come with time stamps.
- Users of a social network develop friendship over time.

Key Contributions

- Evolve-GCN, which adapts the graph convolutional network (GCN) model along the temporal dimension without resorting to node embeddings
- Two architectures are considered for the parameter evolution.
- Less applicable to the frequent change of the node set.
- It is challenging for RNNs to learn these irregular behaviours.

EvolveGCN : An RNN Architecture



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

Updating the Weight Matrix

1) EvolveGCN-H

$$\underbrace{W_t^{(l)}}_{\text{hidden state}} = \text{GRU}(\underbrace{H_t^{(l)}}_{\text{input}}, \underbrace{W_{t-1}^{(l)}}_{\text{hidden state}})$$

(GCN weights = $W_t^{(l)}$; node embeddings = $H_t^{(l)}$; GCN weights = $W_{t-1}^{(l)}$)

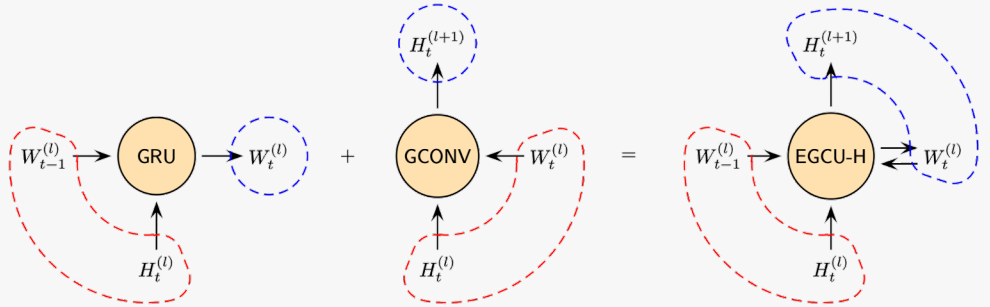
More Effective when node features are informative

2) EvolveGCN-O

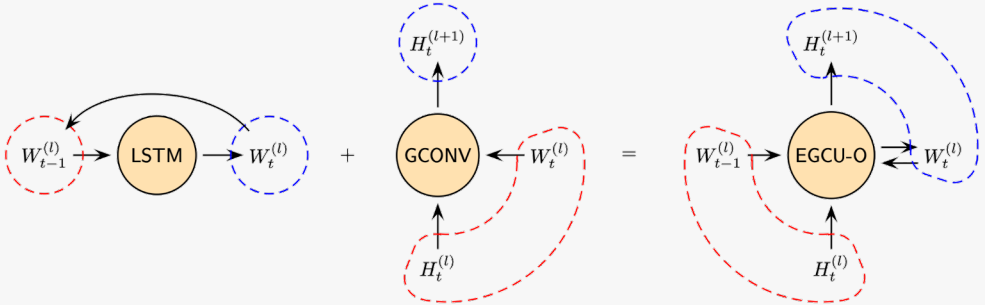
$$\underbrace{W_t^{(l)}}_{\text{output}} = \text{LSTM}(\underbrace{W_{t-1}^{(l)}}_{\text{input}})$$

(GCN weights = $W_t^{(l)}$; GCN weights = $W_{t-1}^{(l)}$)

More Effective when graph structure plays a vital role.



(a) EvolveGCN-H, where the GCN parameters are hidden states of a recurrent architecture that takes node embeddings as input.



(b) EvolveGCN-O, where the GCN parameters are input/outputs of a recurrent architecture.

GC-LSTM: Motivation

How to model Network structure which evolves with time?

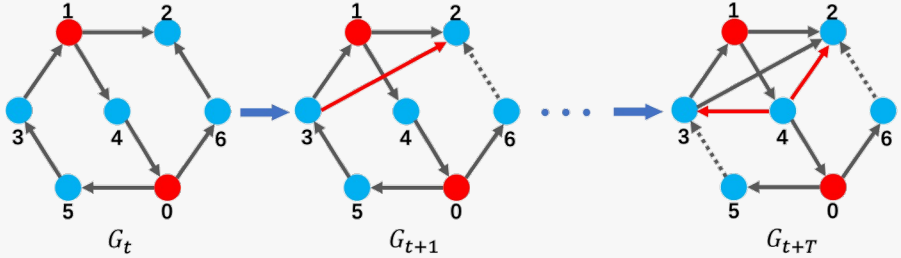
Current Approach & it's Limitations

Static Network

- Aims to predict the future links by the current network
- Focus on **structural feature** of the network
- Static network link prediction approaches, including similarity indexes and network embedding techniques, only make prediction based on a specific snapshot.

Dynamic network

- Need to learn the **temporal feature** of the network
- Most existing methods can only predict the added links in the network
- Most methods are not efficient at decoding deleted links over time



An example of dynamic networks.

Key Contributions

- Unified model capable of handling the spatio-temporal data.
- GC-LSTM, which could handle links that are going to appear or disappear.
- GC-LSTM can effectively handle high-dimensional, time-dependent, and sparse structural sequence data.

GC-LSTM :

Process

Step 1: Decide what info will be thrown away from previous cell state by **forget gate**

$$f_t = \sigma(A_t W_f + GCN_f^K(\tilde{A}_{t-1}, h_{t-1}) + b_f)$$

Step 2: Update the cell state

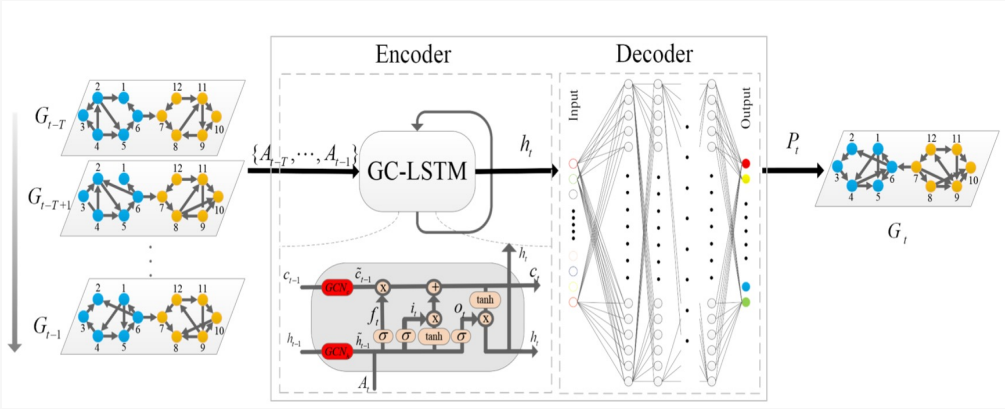
1. tanh layer generates a new candidate vector of the cell layers,
2. sigmoid layer which determines how many new candidate vector will be added to the cell state
3. update cell state (by (1) & (2))

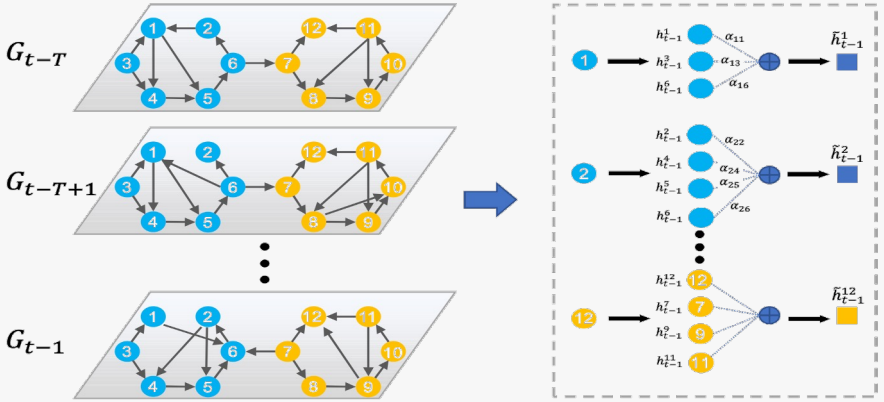
$$\bar{c}_t = tanh(A_t W_c + GCN_o^K(\tilde{A}_{t-1}, h_{t-1}) + b_c),$$
$$i_t = \sigma(A_t W_i + GCN_c^K(\tilde{A}_{t-1}, h_{t-1}) + b_i),$$
$$c_t = f_t \odot GCN_c^K c_{t-1} + i_t \cdot \bar{c}_t.$$

Step 3: Decide output (by **output gate**)

$$o_t = \sigma(A_t W_o + GCN_o^K(\tilde{A}_{t-1}, h_{t-1}) + b_0),$$
$$h_t = o_t \odot tanh(c_t).$$

Step 4: Decoder : MLP





The framework of 1^{st} order GCN hidden state h.

Temporal Graphs – Continuous Time

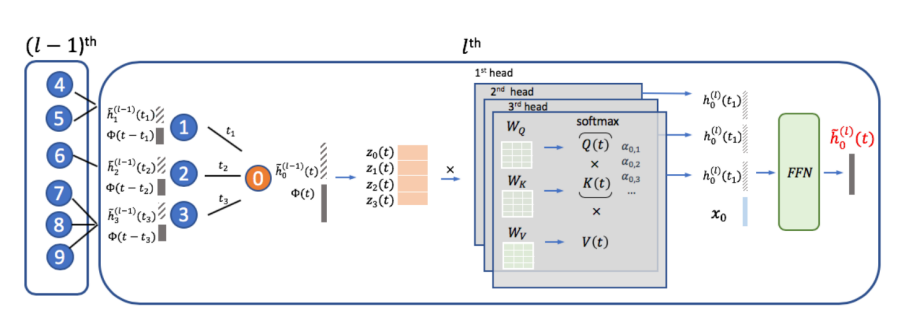
TGAT: Inductive Representation Learning on Temporal Graphs

- Current solutions rely on the discretized approximations
  - Taking graph snapshots
  - Aggregate by bins on the time axis
  - Non-smooth (suboptimal) interpolations

What do we need?

- A temporal walk / local neighbourhood construction approach
  - Extract & retrieve structural information that provide the dynamic system laws for learning;
  - Reveal the temporal-sequential signals;
- A continuous-time deep learning model
  - Aggregates contextual information under the topological construction;
  - Signal processing, learn the temporal patterns for extrapolation.

Temporal Graph Attention Layer:



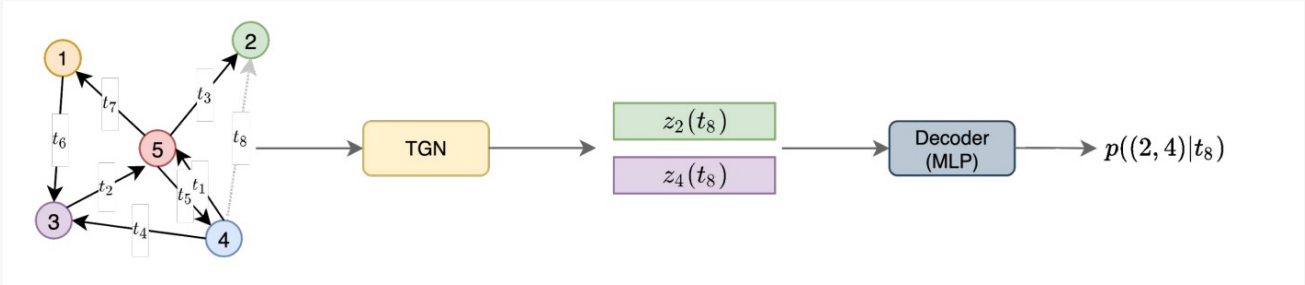
- Input : Neighbourhood hidden representation from the previous layer $(\mathrm{l}-1)$: $Z = \{\tilde{h}_1^{\,\mathrm{l}-1}(\mathrm{t}_1), \ldots, \tilde{h}_N^{\,\mathrm{l}-1}(\mathrm{t}_\mathrm{N})\}$ and the target node information with some time point $(\widetilde{h}_0^{l-1}(t), t)$
- Time-aware feature matrix for target node: $Z(t) = [\tilde{h}_0^{l-1}(t)|\phi_{D_T}(0), \tilde{h}_1^{l-1}(t_1)|\phi_{D_T}(t-t_1), \ldots]^T$
- Three different linear projections to obtain the 'query', 'key' and 'value' representations:

$$\mathrm{q}(t) = [Z(t)]_0 W_Q \qquad \mathrm{K}(t) = [Z(t)]_{1:N}\, W_K \qquad \mathrm{V}(t) = [Z(t)]_{1:N}\, W_V$$

- Hidden neighbourhood representations, i.e. $\mathrm{h}(t) = Attn\big(q(t), K(t), V(t)\big) \in \boldsymbol{R}_{\mathrm{h}}^{\mathrm{d}}$
- Output: target node's contextual-neighbourhood representation : $\widetilde{h}_0(t) = FFN(h(t)|x_0)$

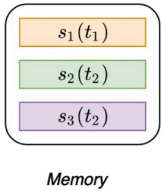
TGN : Temporal Graph Networks for Dynamic Graphs

- Model for dynamic graphs is an encoder-decoder pair - TGN is an encoder model which is able to generate temporal node embeddings for any node i and time t.
- Decoder is task-dependent, eg. MLP from two node embeddings to edge probability
- General theoretical framework, which consists of 5 different modules



5-Modules

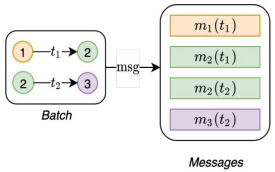
- State (vector) for each node the model has seen so far
- **Compressed representation** of all past interactions of a node
- **Not a parameter**, but the modules to update it are
- Updated also at test time
- Initialized at 0, it can handle new nodes (inductive)



1. Memory

- **Given an interaction** (i, j), **computes messages** for the source and the destination
- **Messages** will be used to **update the memory**

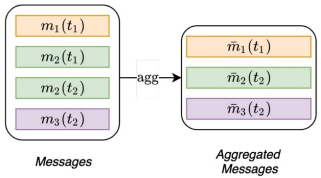
$$\mathbf{m}_i(t) = \mathrm{msg}_\mathrm{s}\left(\mathbf{s}_i(t^-), \mathbf{s}_j(t^-), t, \mathbf{e}_{ij}(t)\right),$$
$$\mathbf{m}_j(t) = \mathrm{msg}_\mathrm{d}\left(\mathbf{s}_j(t^-), \mathbf{s}_i(t^-), t, \mathbf{e}_{ij}(t)\right)$$



2. Message Function

Aggregates multiple **messages** for the same node in a batch

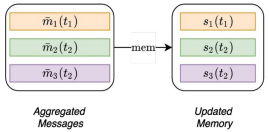
$$\bar{\mathbf{m}}_i(t) = \mathrm{agg}\left(\mathbf{m}_i(t_1), \ldots, \mathbf{m}_i(t_b)\right)$$



3. Message Aggregator

Updates memory using new messages

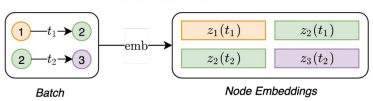
$$\mathbf{s}_i(t) = \mathrm{mem}\left(\bar{\mathbf{m}}_i(t), \mathbf{s}_i(t^-)\right)$$



4. Memory Updater

- **Computes** the **temporal embedding** of a node (which can be then used for prediction) using the graph
- **Solves** the **staleness problem** (memory becoming out of date)

$$\mathbf{z}_i(t) = \mathrm{emb}(i,t) = \sum_{j \in \mathcal{N}_i^k([0,t])} h\left(\mathbf{s}_i(t), \mathbf{s}_j(t), \mathbf{e}_{ij}, \mathbf{v}_i(t), \mathbf{v}_j(t)\right),$$



5.Graph Embedding

Temporal point process and Hawkes Process

$$\lambda(t|\mathcal{H}(t)) = \lim_{\Delta t \to 0} \frac{\mathbb{E}[N(t+\Delta t)|\mathcal{H}_t]}{\Delta t}$$

$$\lambda(t) = \mu(t) + \int_{-\infty}^{t} \kappa(t-s)\mathrm{d}N(s)$$

Equations for Hawkes Process

Point process models **discrete sequential** events, assuming that **historical** events can influence the **current** event.

TPPs (esp. Hawkes Process) is desirable for modeling temporal link formation !

For current event is influenced **more** by

recent events, **less** by **previous** events

Temporal Graphs – Limitations

Take discrete snapshots

- Dynamic network embedding: an extended approach for skip-gram based network embedding. IJCAI-2018.
- DynGEM: Deep embedding method for dynamic graphs. arxiv-2018.
- Dynamic network embedding by modeling triadic closure process. AAAI-2018.
- Evolvegcn: Evolving graph convolutional networks for dynamic graphs. AAAI-2020.

Not inductive to new nodes

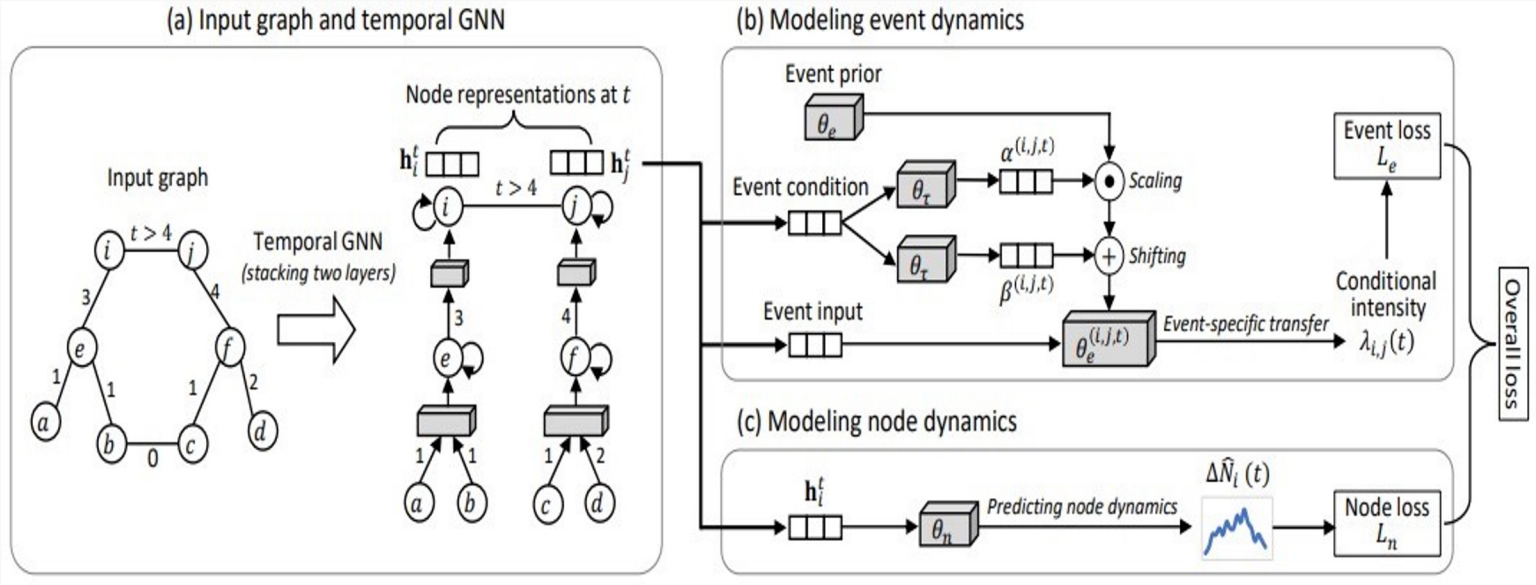
- Embedding temporal network via neighborhood formation. KDD-2018.
- Temporal network embedding with micro-and macro-dynamics. CIKM-2019.
- Dynamic Heterogeneous Graph Embedding via Heterogeneous Hawkes Process. ECML-PKDD 2021.

Not modelling the excitation effects

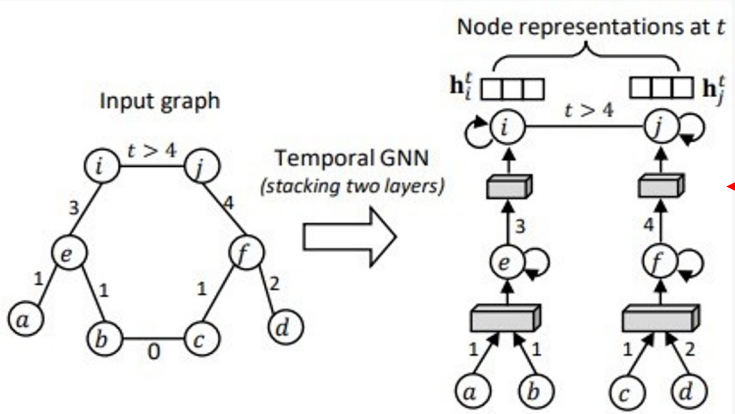
- DyRep: Learning representations over dynamic graphs.
- ICLR-2019. Inductive representation learning on temporal graphs. ICLR-2020.

TREND *(The Web Conference 2022)*

- Presents Adoption of Hawkes Process in GNN -> Temporal GNN Layer
- Integrates event and node dynamics in the model to model distinct and collective behaviour of events



TREND - Hawkes Process on temporal graph

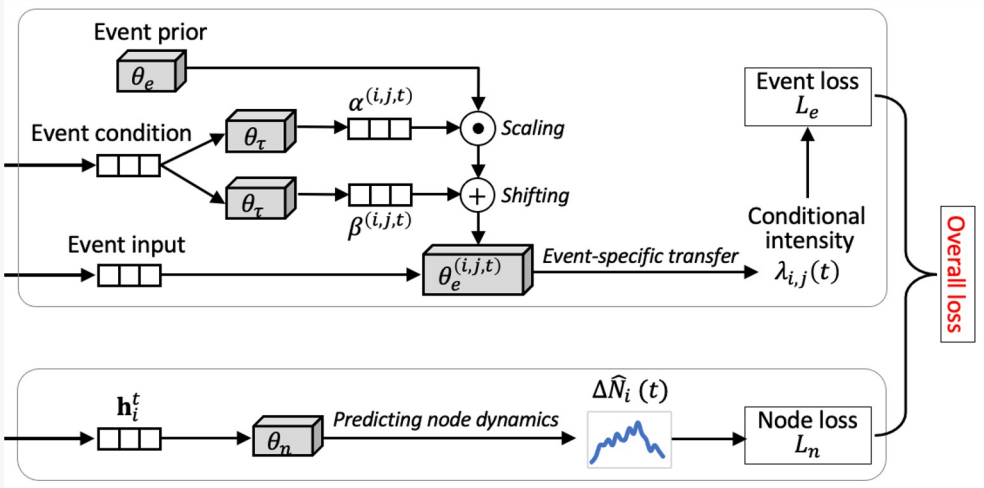


$$\lambda_{i,j}(t) = \mu_{i,j}(t) + \sum_{(i,j',t') \in \mathcal{H}_i(t)} \gamma_{j'}(t')\kappa(t-t') + \sum_{(i',j,t') \in \mathcal{H}_j(t)} \gamma_{i'}(t')\kappa(t-t')$$

$$\lambda_{i,j}(t) = f(\mathbf{h}_i^t, \mathbf{h}_j^t)$$

$$\mathbf{h}_i^{t,l} = \sigma\Big(\underbrace{\mathbf{h}_i^{t,l-1}\mathbf{W}_{\text{self}}^l}_{\text{self-information (for base intensity)}} + \underbrace{\sum_{(i,j',t') \in \mathcal{H}_i(t)} \mathbf{h}_{j'}^{t',l-1}\mathbf{W}_{\text{hist}}^l \tilde{\kappa}_i(t-t')}_{\text{historical neighbors' information (for excitement by historical events)}} \Big)$$

TREND – Overall Model



Edge Dynamics

$$\lambda_{i,j}(t) = f(\mathbf{h}_i^t, \mathbf{h}_j^t) = \mathrm{FCL}_e((\mathbf{h}_i^t - \mathbf{h}_j^t)^{\circ 2}; \theta_e)$$

$$\theta_e^{(i,j,t)} = \tau(\theta_e, \mathbf{h}_i^t \| \mathbf{h}_j^t; \theta_\tau) = (\alpha^{(i,j,t)} + 1) \odot \theta_e + \beta^{(i,j,t)}$$

$$\alpha^{(i,j,t)} = \sigma\left((\mathbf{h}_i^t \| \mathbf{h}_j^t)\mathbf{W}_\alpha + \mathbf{b}_\alpha\right) \qquad \beta^{(i,j,t)} = \sigma\left((\mathbf{h}_i^t \| \mathbf{h}_j^t)\mathbf{W}_\beta + \mathbf{b}_\beta\right)$$

$$L_e(i,j,t) = -\log(\lambda_{i,j}(t)) - Q \cdot \mathbb{E}_{k \sim P_n} \log(1 - \lambda_{i,k}(t))$$

Node Dynamics

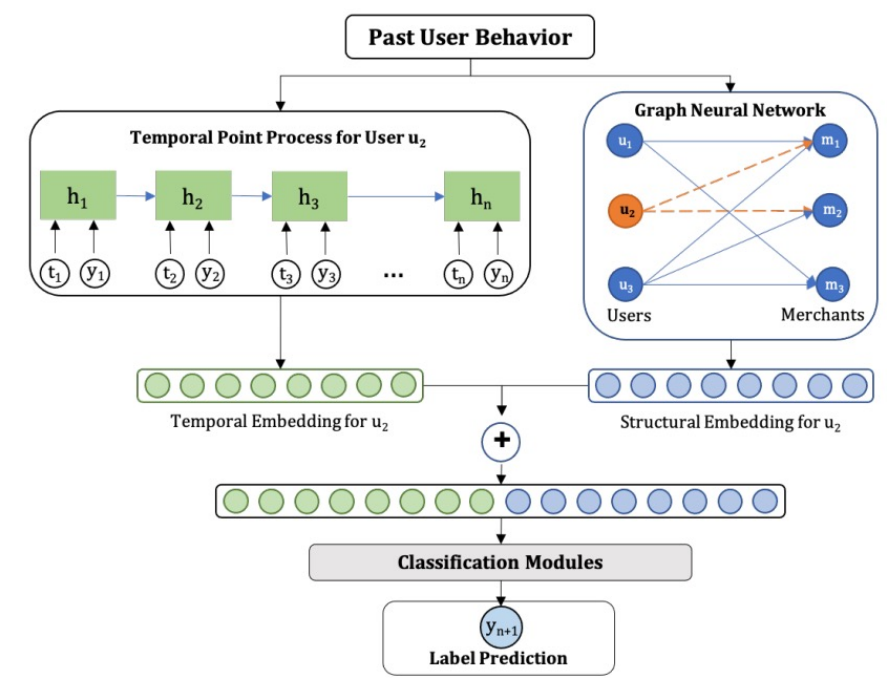
$$\Delta\hat{N}_i(t) = \mathrm{FCL}_n(\mathbf{h}_i^t; \theta_n)$$

Overall Loss

$$\arg\min_{\Theta} \sum_{(i,j,t) \in \mathcal{I}^{\mathrm{tr}}} L_e + \eta_1 L_n + \eta_2 (\|\alpha^{(i,j,t)}\|_2^2 + \|\beta^{(i,j,t)}\|_2^2)$$

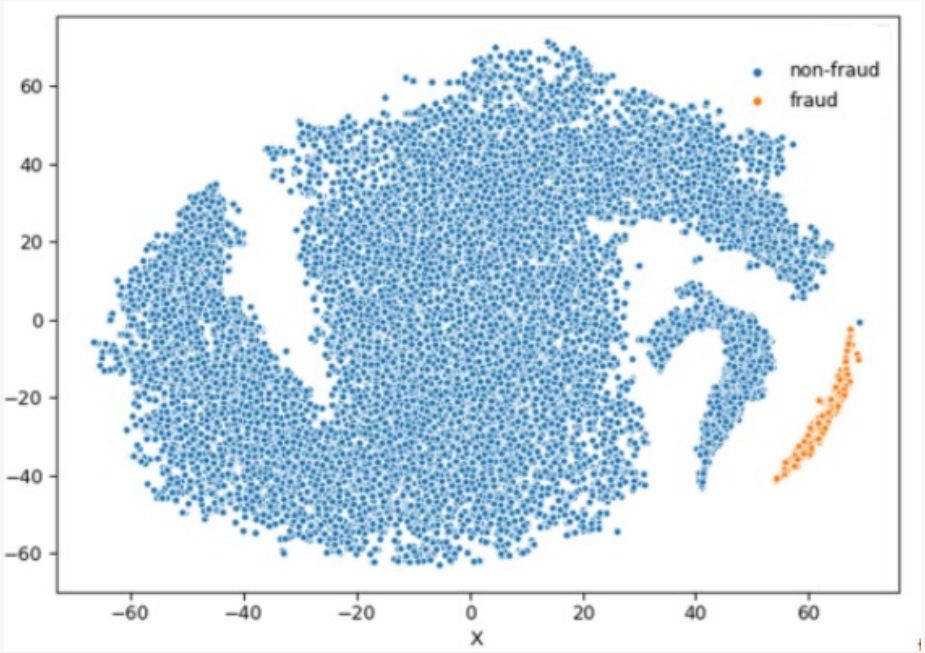
TeGraF: Temporal and Graph based Fraudulent Transaction Detection Framework

TeGraF: Architecture



- For the TPP based component, the dataset is modelled as a set of asynchronous sequences of the transaction times and the fraud label where for graph based component, the dataset is modelled as a Bipartite Graph between user and merchant
- TPP is helpful for capturing short term fraudulent behaviour, If an anomalous pattern is observed in a user's transaction behaviour very recently, there is a high chance of fraud transaction
- Graphs help in capturing the long term patterns of interacting with fraudulent merchants

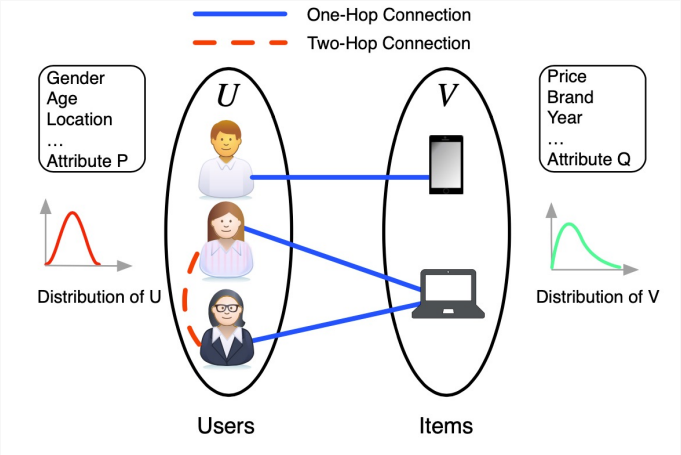
TeGraF: Results



- TSNE analysis of RMTPP+BiNE embedding shows the models is able to learn a clear distinction between the true positives (fraud transactions) and true negatives (nonfraud transactions)

BiPartite Graphs

Bipartite Graph



Challenges

- Although edges exists between nodes of two different domains only, implicit relations also exits between nodes of same domain. >> **BiNE**
- Each partition of a bipartite graph may follow different distributions in distinct feature domains. >> **BGNN**
- Scaling up to a million node Bipartite graph. >> **Cascade BGNN**
- Global properties of bipartite graph, including community structures of homogeneous nodes and long-range dependencies of heterogeneous nodes, are not well preserved >> **BiGI**

BiNE: Bipartite Network Embedding

Learn Vertex Embeddings by Joint Optimization over Explicit Edges and Implicit Relations

Modelling Explicit Relations

- The joint probability between vertices u_i and v_j is defined as:

$$P(i, j) = \frac{w_{ij}}{\sum_{e_{ij} \in E} w_{ij}}.$$

where w_{ij} is the weight of edge e_{ij} .

Modelling Implicit Relations

- The 2nd-order proximity between two vertices as:

$$w_{ij}^U = \sum_{k \in V} w_{ik} w_{jk}; \quad w_{ij}^V = \sum_{k \in U} w_{ki} w_{kj}.$$

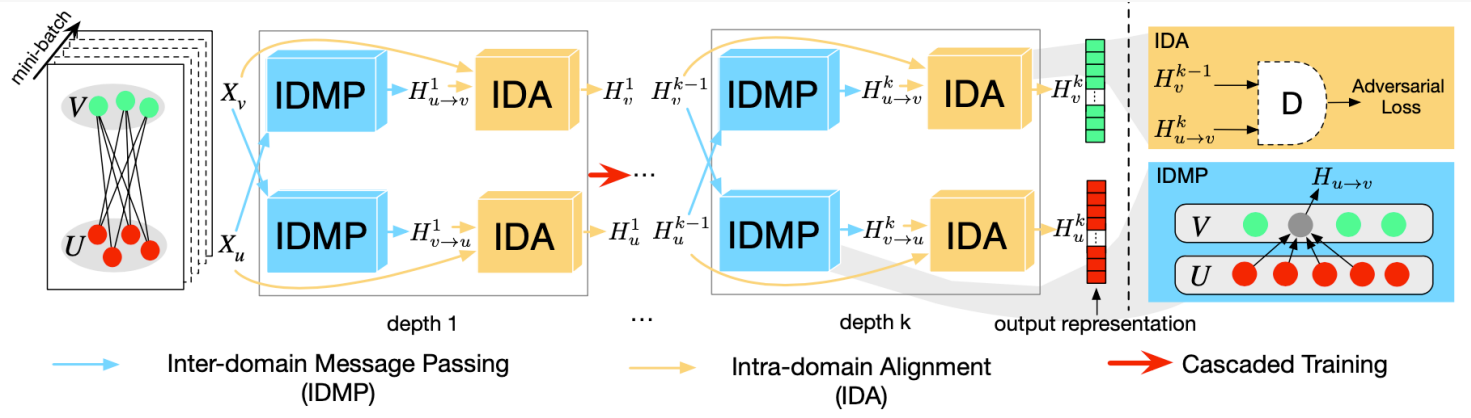
where w_{ij} is the weight of edge e_{ij} .

Cascade-BGNN

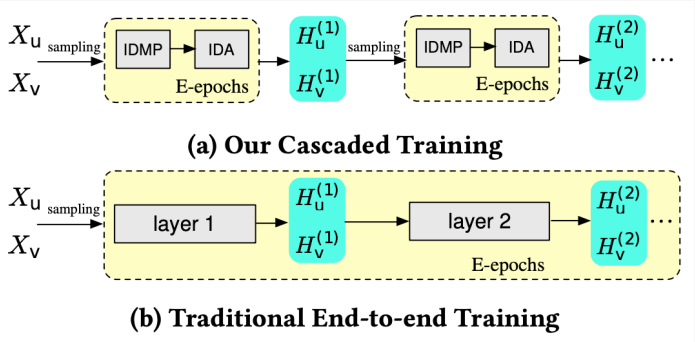
Efficient aggregation of information across and within two partitions of a bipartite graph.

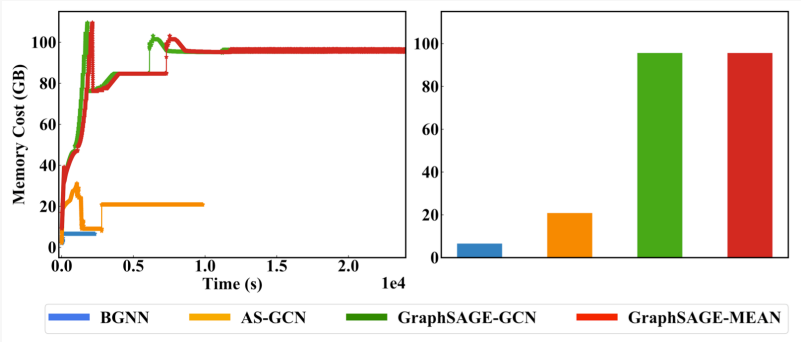
Domain Consistent: Given the inputs of two domains Xu and Xv , their self-supervised representation is obtained as Hu and Hv via inter-domain message passing and intra-domain distribution alignment.

Efficient at Large Scale: To enable multi-hop neighbour information aggregation, multiple layers are stacked to formulate a deep BGNN whose layers are trained in a cascading manner. That is, the training of the upper layer (depth k + 1 in Fig) begins only after the lower one (depth k in Fig. 2) has been trained completely.



Cascade-BGNN





- Cascaded training:
  - One depth (layer) training is regarded as training on a basic BGNN block.
  - Each depth completes its training and used as the input for later training.
- Traditional training:
  - Propagation through multiple depths for a fixed number of hops to train the final embedding.

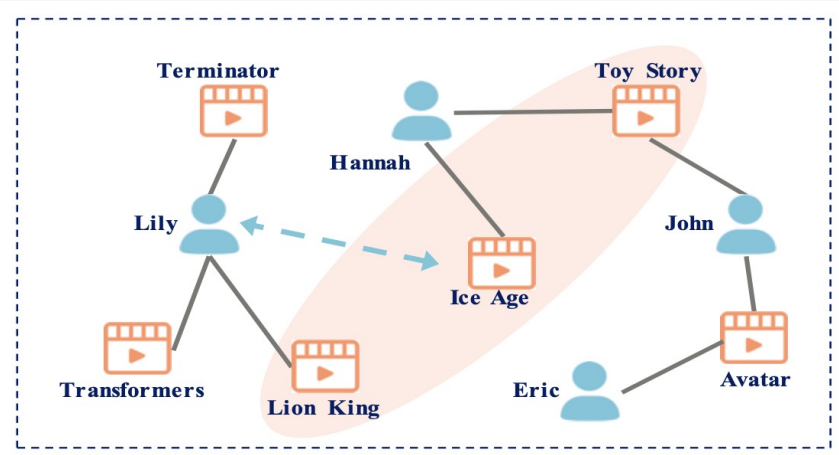
- Memory cost and training time on Tencent data.
  - The short blue line of BGNN and orange line of AS-GCN mean the training has finished
  - The training time of GraphSAGE is too long to be shown.

BiGI: Bipartite Graph Embedding via Mutual Information Maximization

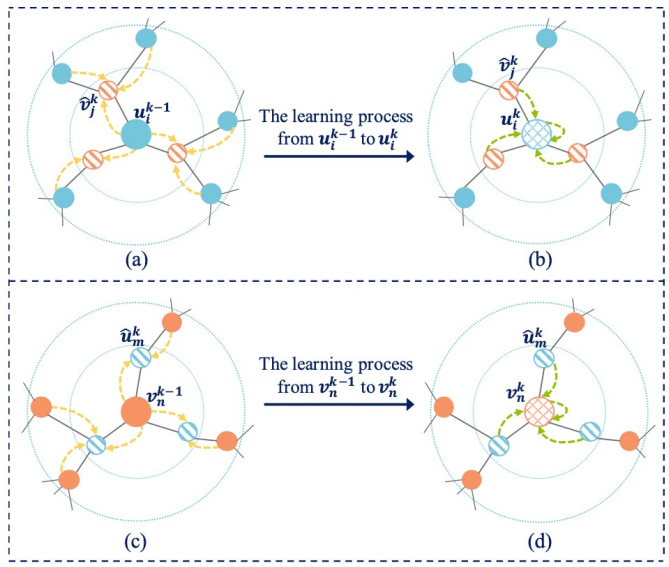
- Capture such global properties by introducing a novel local-global infomax objective
- Enables nodes in bipartite graph to be globally relevant

An example of user-movie bipartite graph. The orange shaded area represents a underlying community structure where three movies may share similar genres. The blue dotted lines denote the long-range dependency between

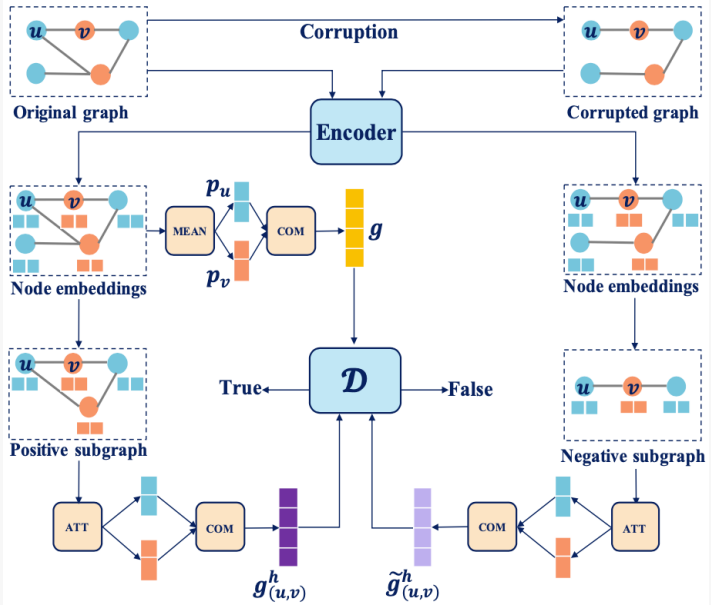
"Lily" and "Ice Age". However, these global properties are hard to be well learned from local graph structures



BiGI: Bipartite Graph Embedding via Mutual Information Maximization



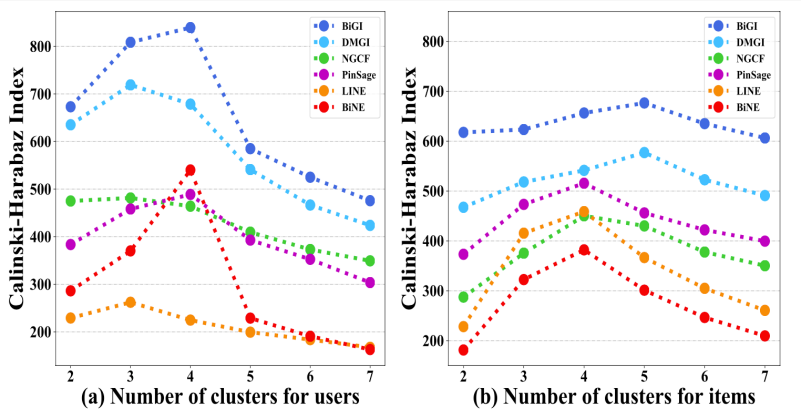
A simple illustration of the proposed encoder. In k-th layer, (a) and (b) show the learning process of $\boldsymbol{u_i^{k-1}}$. (c) and (d) show the learning process of $\boldsymbol{v_n^{k-1}}$ in a similar way. The yellow dotted lines and the green dotted lines demonstrate how to derive node embeddings $\boldsymbol{u_i^k}$ and $\boldsymbol{v_n^k}$.



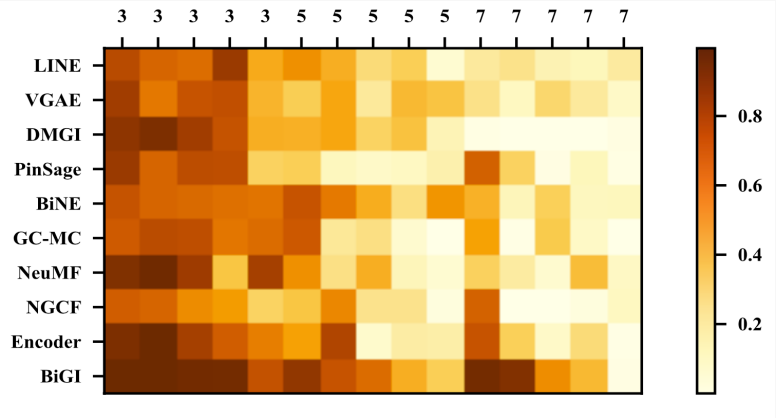
An overview of BiGI

"ATT", "MEAN" and "COM" denote the subgraph-level attention mechanism, mean operation and composition function, respectively. $\boldsymbol{p_u}$ and $\boldsymbol{p_v}$ are two prototype representations. $\boldsymbol{g}$ is the global representation and $\boldsymbol{g^h_{(u,v)}}$ is local representation.

BiGI: Bipartite Graph Embedding via Mutual Information Maximization



Results of clustering analysis. BiGI achieves the best clustering results (A higher score is preferred). It demonstrates that BiGI can better capture community structures of users and items simultaneously.



Visualization of prediction scores. BiGI achieves the better results compared with other baselines.

- When the distance of target node pair is relative short, e.g. 3, all baselines and BiGI can learn the latent interaction of node pair.
- With the increase of distance, the observable relation between ui and $v\ j$ is gradually weakened. Compared with state-of-the-art baselines, BiGI still maintains promising results.

Fairness in GNNs

Fairness in AI

What is Fairness in AI ?

Sampling Bias/Imbalance representation of every group

Pattern of group bias gets learnt by the model while training

Algorithmic prediction becomes biased towards a certain group

Fairness in Graph based Models

- Machine learning algorithms are known to exhibit algorithmic biases such as women being discriminated in a job-recommendation system or African-Americans being subjected to higher-interest credit cards.
- GNNs are widely used to learn node representation based on their neighbourhood structure. Although these representations capture the neighbourhood structure very well, they tend to become discriminative towards sensitive attributes since like interacts with like making similar embeddings for similar people.
- In this workshop we discuss about 2 State of art algorithms –
  - FairGNN
  - FLiB

We evaluate model fairness in terms of the two most widely used fairness metrics:

- Demographic Parity (DP) quantifies the degree of independence between the model outputs and the sensitive attributes. For binary-valued target outcome and sensitive attributes, demographic parity seeks to achieve.

$$P(\hat{y} = 1|a = 0) = P(\hat{y} = 1|a = 1)$$

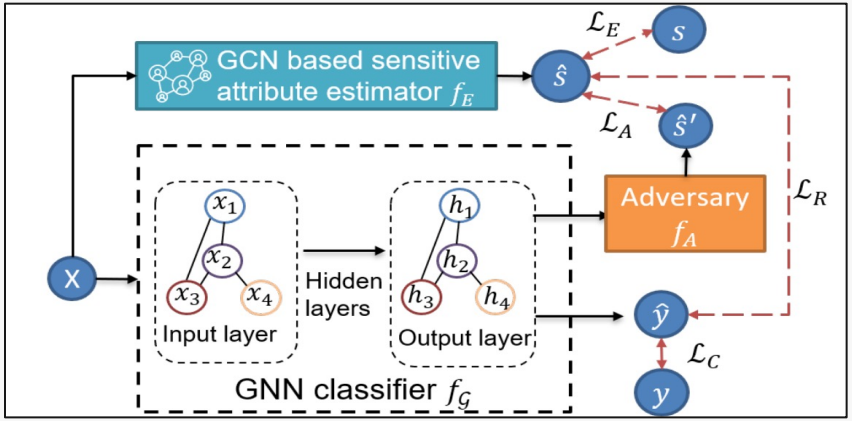
- Equal Opportunity (EO) requires that the probability of an instance in a positive class being assigned to a positive outcome be independent of its sensitive attribute. It is defined as:

$$P(\hat{y} = 1|a = 0, y = 1) = P(\hat{y} = 1|a = 1, y = 1)$$

FAIRGNN

Problem Definition- Given a graph G = (V, E, X), learn a fair GNN for fair node classification. Predictions should maintain high accuracy whilst satisfying the fairness criteria such as statistical parity.

Introduction-*FAIRGNN introduces adversarial debiasing to ensure the GNN classifier make predictions independent with the estimated sensitive attributes*



The overall framework of FairGNN.

Architecture of FAIRGNN

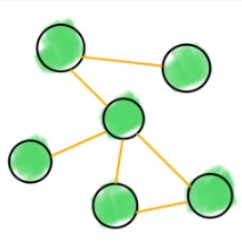
An illustration of the proposed framework is shown in Figure alongside, which is composed of-

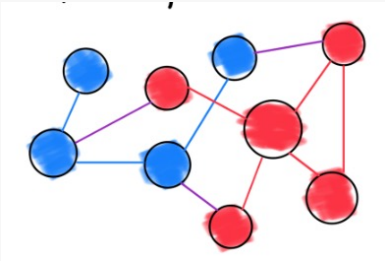
- The GNN classifier f_G-The GNN classifier f_G takes the graph G as input for node classification
- GCN based sensitive attribute estimator f_E-GCN based sensitive attribute estimator f_E uses the node embeddings learnt by f_G and predicts the sensitive attributes for nodes whose sensitive attributes are unknown
- An adversary f_A-An adversary f_A aims to predict the known or estimated sensitive attributes by f_E from the node representation learned by f_G; while f_G aims to learn fair node representations that can fool the adversary f_A to make wrong predictions.

FLiB: Fair Link Prediction in Bipartite Network

Issues with existing work

- Much of the research on Fairness in GNNs is focused on homogeneous networks and is limited for heterogeneous networks.
- Due to the cross-interaction between various types of nodes, most of the existing work in fair graph learning cannot invariably be extended for heterogeneous graphs.





Introduction to FLiB

- FLiB work assumes bias from one node type can also propagate to the other types, making a task like link prediction involving different types of nodes unfair.
- It becomes imperative to ensure that bias from user nodes is blocked from propagating to other types of nodes for keeping the task of link prediction fair

Key Contribution

For The task of link prediction in bipartite networks Our goal is twofold:

- To make this task between the two types of nodes fair (in terms of EO & DP) with respect to the binary sensitive attribute user nodes.
- Learn fair representations for both types of nodes such that those embeddings can be used for any further downstream task.

FLiB: Fair Link Prediction in Bipartite Network

Architectural components

The four major components of FLiB are -

- A GNN based link-predictor,
- Multi-layer Adversarial Debiasing,
- Multi-Nodal Adversarial Debiasing via Pseudo sensitive attributes,
- Fairness Aware Regularization Loss (FARL).

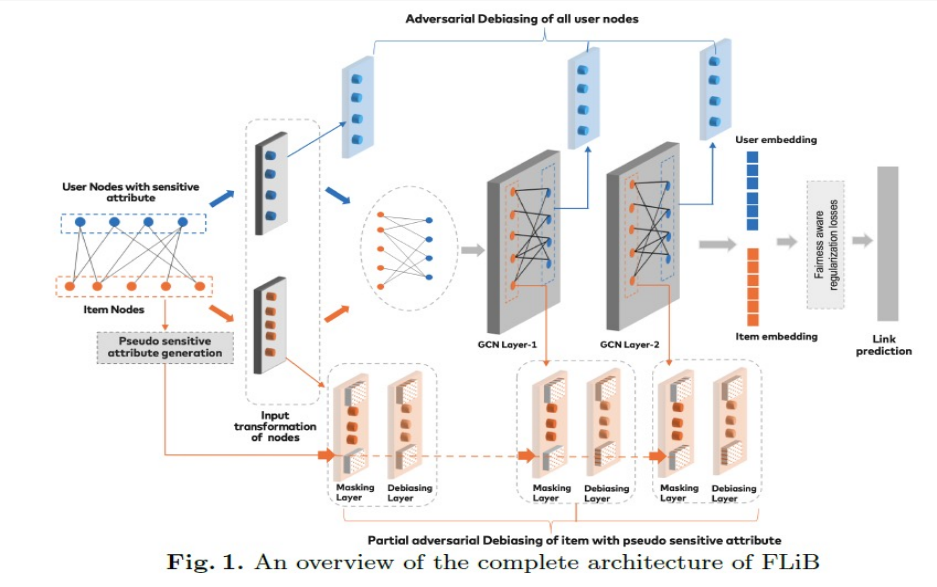


Fig. 1. An overview of the complete architecture of FLiB

Methodology

- FLiB adopts a multi-layer adversarial debiasing approach that debiases user nodes and promotes fairness in item nodes by blocking bias from propagating to them.
- FLiB further determines and explicitly debiases item nodes that are highly susceptible to bias using a susceptibility factor" (SF) calculated from the graph structure. The subset of nodes with high SF are assigned a pseudo-sensitive label which allows the integration of a parallel debiasing framework for item nodes as well.
- FLiB debiases these item nodes similar to user nodes via multi-layer adversarial debiasing using their pseudo-sensitive attributes. FLiB integrates metric-specific regularization losses to optimize for fairness in the node predictions.

FLiB: Fair Link Prediction in Bipartite Network

RESULTS

The table presents the results of FLiB and compares it against the state-of-the-art Fairness models on two fairness metrics, DP and EO.

Table 1. Comparative results of FLiB on MovieLens, and LastFM datasets

	Dataset	Movie Lens					LastFM				
GNN	Model	Acc	AR	$\|DP\|$	$\|EO\|$	AAc	Acc	AR	$\|DP\|$	$\|EO\|$	AAc
GCN	GNN	83.57	0.915	5.79	4.91	62.58	67.45	0.723	8.53	6.03	56.60
	FAIR-HIN	83.64	0.914	0.85	0.79	72.45	67.55	0.726	**0.19**	**1.44**	90.57
	FairGNN	83.98	0.918	5.33	4.40	**51.40**	67.90	0.730	1.79	4.06	**51.14**
	FLiB	83.40	0.914	**0.32**	**0.12**	55.06	67.92	0.740	1.53	2.56	56.48
GAT	GNN	78.04	0.852	5.73	7.02	62.33	70.57	0.776	9.32	7.19	52.83
	FAIR-HIN	77.48	0.829	**0.52**	0.67	61.75	67.31	0.725	0.63	**0.76**	84.91
	FairGNN	77.00	0.869	0.71	2.61	53.55	66.93	0.730	1.29	4.19	**51.14**
	FLiB	70.84	0.802	1.15	**0.54**	**50.74**	67.19	0.745	**0.16**	0.83	51.52

FLiB maintains the accuracy and AUCROC of link prediction task as that of other GNN models

FLiB reduces both DP and EO to achieve fair link predictions as seen by a

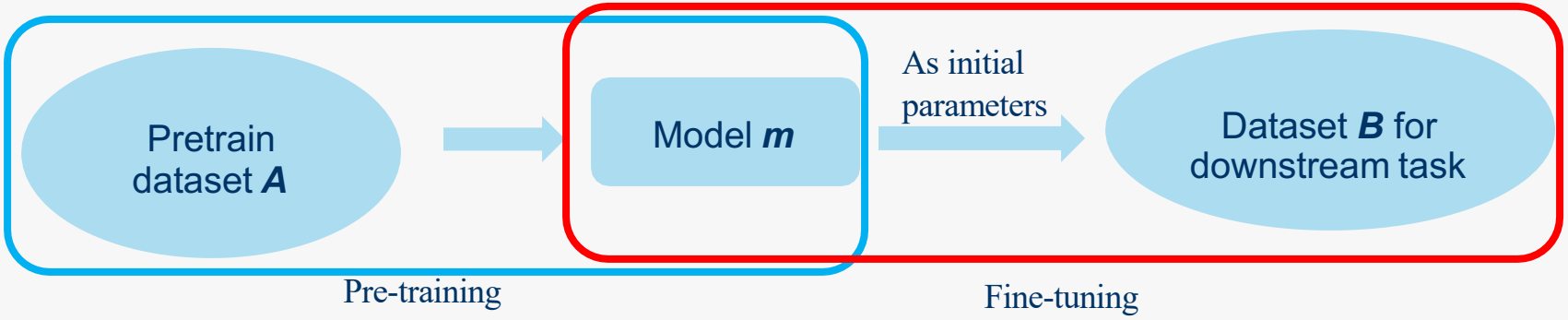
FLiB learn fair representations for both datasets as seen by adversarial accuracies which is close to 50%

Pretarined GNNs

Procedure of pre-training

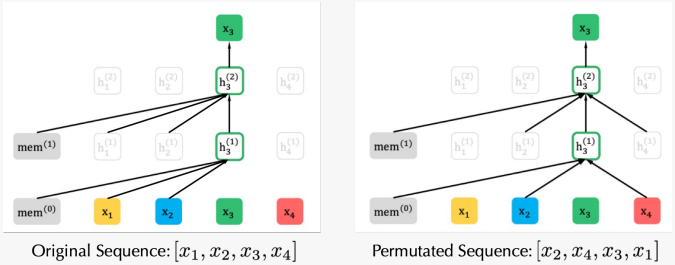
Pre-training:
- Step 1: Train model ***m*** on dataset ***A***

- Fine-tuning
  - Step 2: Use ***m***'s parameters as initial parameters
  - Step 3: Train model ***m*** to do downstream tasks on dataset ***B***

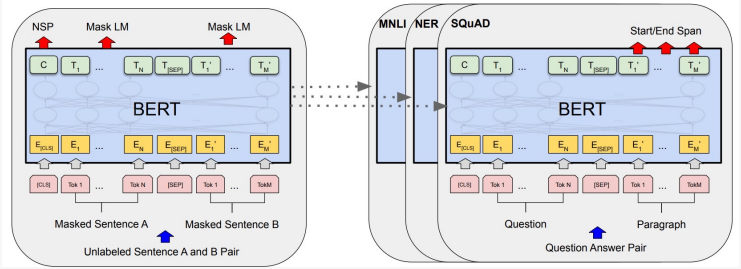


Development of pre-training

- **Development of pre-training models in NLP**
  - Unsupervised pre-training model:
    - Pre-trained Word Embeddings(e.g. Word2vec(Mikolov et. al. 2013))
  - Self-supervised pre-training model:
    - Pre-trained Contextual Encoders(e.g. Bert, XLNet)



XLNet(Yang et. al. 2020)



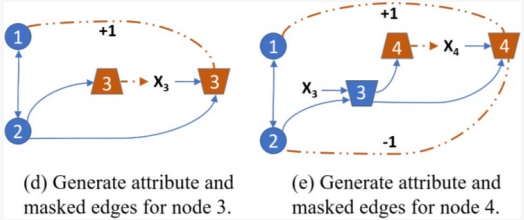
Bert(Devlin et. al. 2019)

Graph Pre-Training

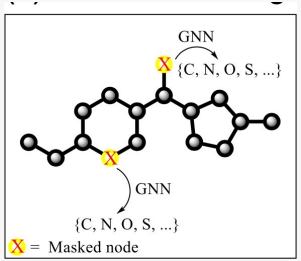
- Graph pre-train is a solution
  - Pre-trained model can quickly adapt to different downstream tasks to handle the out-of-distribution issue
  - Downstream tasks are no more highly depend on labeled data and thus model can learn general graph structure or property

Classification of Graph Pretraining

- Classifications of pre-train models: **Contrastive** and **Generative** methods
- **Generative method based models:**
  - Strategies for pre-training graph neural networks (Hu et al. ICLR 2019)
  - GPT-GNN: Generative Pre-Training of Graph Neural Networks (Hu et al. KDD 2020)



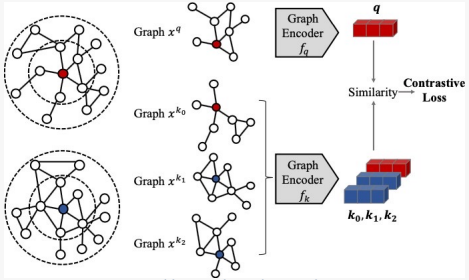
Edge/Attributes generation



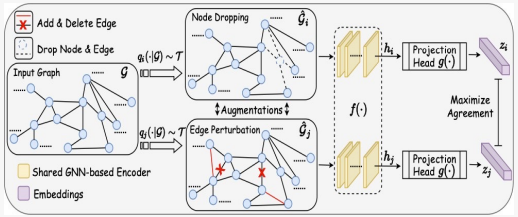
Attributes masking

Classification of Graph Pretraining

- Classifications of pre-train models: **Contrastive** and **Generative** methods
- **Contrastive method based model:**
  - GCC: Graph Contrastive Coding for Graph Neural Network Pre-Training (Qiu et al. KDD2020)
  - Graph Contrastive Learning with Augmentations (You et al. NIPS2020)



Instance discrimination

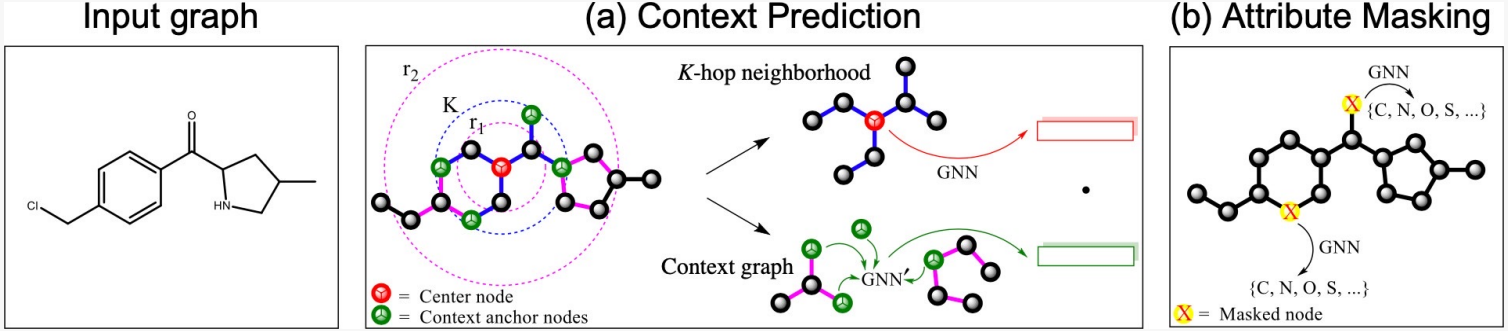


Contrast on augmented graphs

Strategies for Pre-Training

Develop some strategies to relieve the scarce task-specific labels problem

- Context prediction: Nodes in similar contexts obtain nearby embeddings
- Attribute masking: Capture domain knowledge from masking node attributes

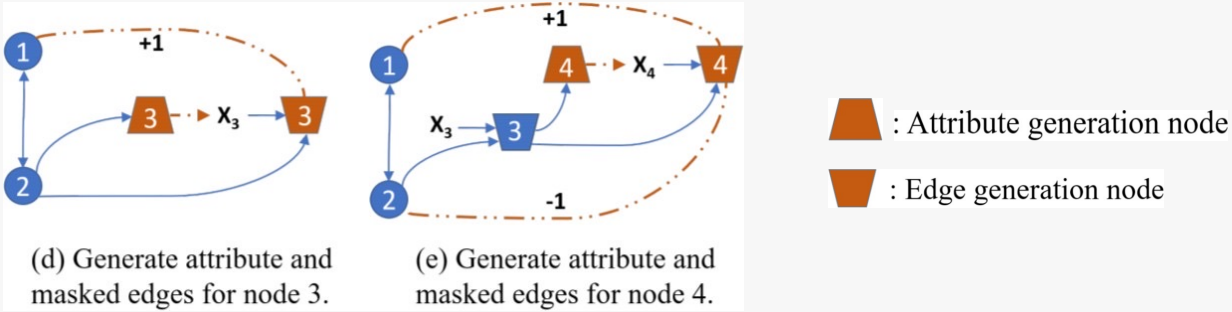


Hu W, Liu B, Gomes J, et al. Strategies for Pre-training Graph Neural Networks. ICLR2020.

GPT-GNN

Pre-train large-scale graph with reconstructing the input graph, which utilizes the unlabeled data for label scarcity issue. Decompose the reconstruction process into two coupled steps:

- Attribute generation：given observed edges, generate node attributes
- Edge generation: given observed edges and generated attributes, generate masked edges



Hu Z, Dong Y, Wang K, et al. GPT-GNN: Generative pre-training of graph neural networks. KDD2020.

GCC

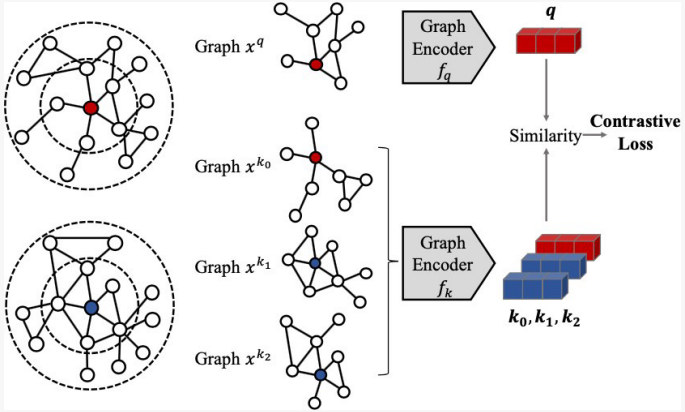
Leverage instance discrimination as the pretext task for structural information pre-training, which learns the transferable knowledge across graphs for tackling out-of-domain data

Instance Discrimination

- InfoNCE Loss for instance representations

that capture the similarities between instances

$$\mathcal{L} = -\log \frac{\exp\left(\boldsymbol{q}^\top \boldsymbol{k}_+ / \tau\right)}{\sum_{i=0}^{K} \exp\left(\boldsymbol{q}^\top \boldsymbol{k}_i / \tau\right)}$$

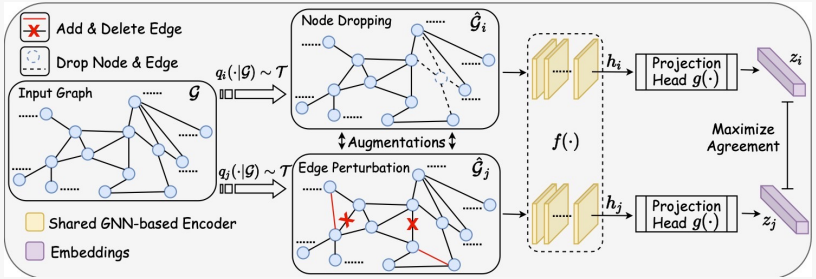


Instances sampled from same node close, while different apart

Qiu J, Chen Q, Dong Y, et al. GCC: Graph contrastive coding for graph neural network pre-training. KDD2020

GraphCL

Design four types of graph augmentations to incorporate various impacts in different settings: semi-supervised, unsupervised and transfer learning



$$\ell_n = -\log \frac{\exp(\text{sim}(\boldsymbol{z}_{n,i}, \boldsymbol{z}_{n,j})/\tau)}{\sum_{n'=1,n'\neq n}^{N} \exp(\text{sim}(\boldsymbol{z}_{n,i}, \boldsymbol{z}_{n',j})/\tau)}$$

Data augmentation	Type	Underlying Prior
Node dropping	Nodes, edges	Vertex missing does not alter semantics.
Edge perturbation	Edges	Semantic robustness against connectivity variations.
Attribute masking	Nodes	Semantic robustness against losing partial attributes.
Subgraph	Nodes, edges	Local structure can hint the full semantics.

Yuning You, Tianlong Chen, Yongduo Sui, et al. Graph Contrastive Learning with Augmentations. NIPS2020

Open Problems in Temporal Graphs for the Financial World

Elegant representation of the dynamics, heterogeneity and temporal behavior of nodes

New modeling techniques as opposed to applying defined machine learning based solutions

Handling the cold-start problem

Computational Complexity

Temporal Graph Learning for Financial World:

Algorithms, Scalability, Explainability & Fairness

KDD 2022 Tutorial

Nitendra Rajput

VP & Head – AI Garage, Mastercard

Karamjit Singh

Director – AI Garage, Mastercard

Acknowledgements: 20+ Data Scientists & Engineers at AI Garage who work in this area.