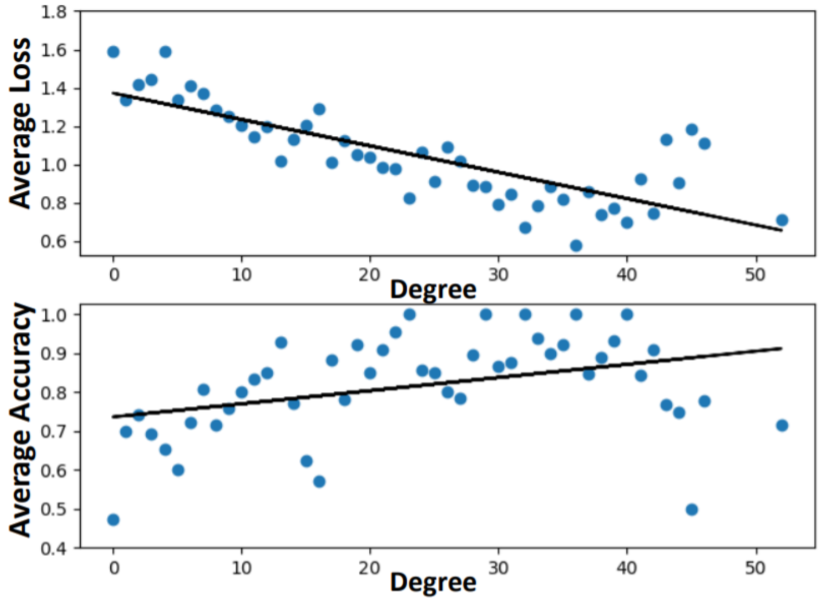
Towards Expressive Tail Node Embeddings for Link Prediction on Networks

Yijun Ma

Renmin University of China

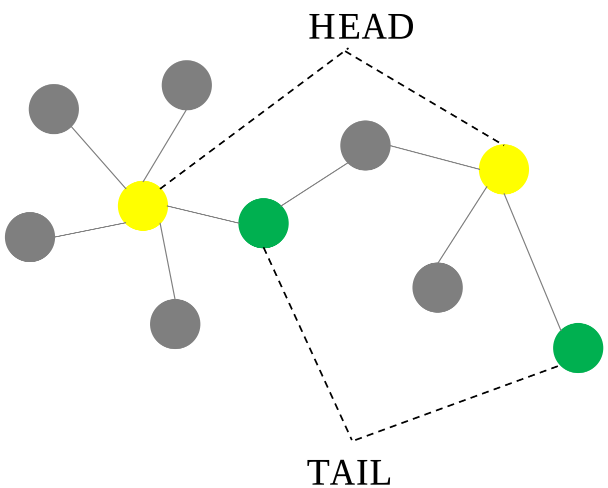
2022.08

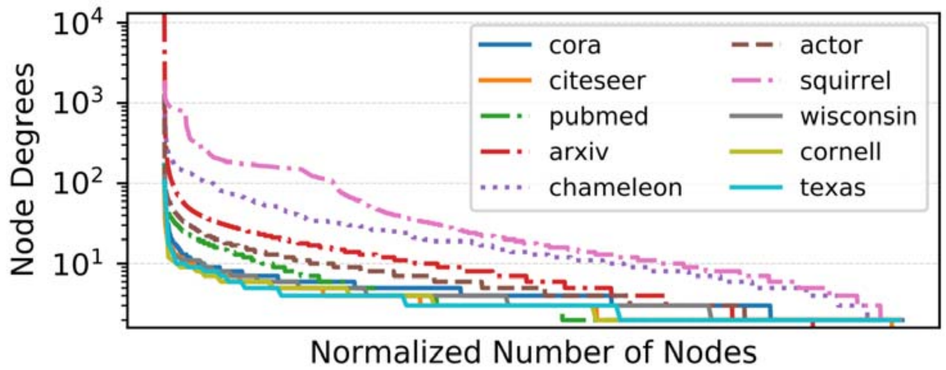
- Background
  - Advancing graph embedding techniques has shown promising results in various downstream tasks, such as node classification, link prediction and recommendation
  - Despite all the progress we've made, there's some recent work revealing that low degree nodes are suffering higher loss and lower predictive accuracy, which causes a degree-related fairness concern, or denoted as **low degree bias**



Reference: Jian Kang, et.al. *RawlsGCN: Towards Rawlsian Difference Principle on Graph Convolutional Network*. In WWW 2022

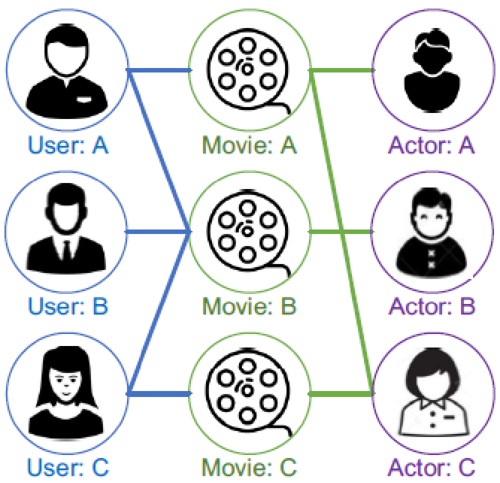
- Background
  - Highly qualified node embedding learning in graph needs rich and high-quality neighborhood information
  - Due to the long-tailed degree distribution in real-world networks, tail nodes accounting for vast majority of nodes usually have few and even noisy links, which will lead to partially inferior performance if all the nodes are treated equally

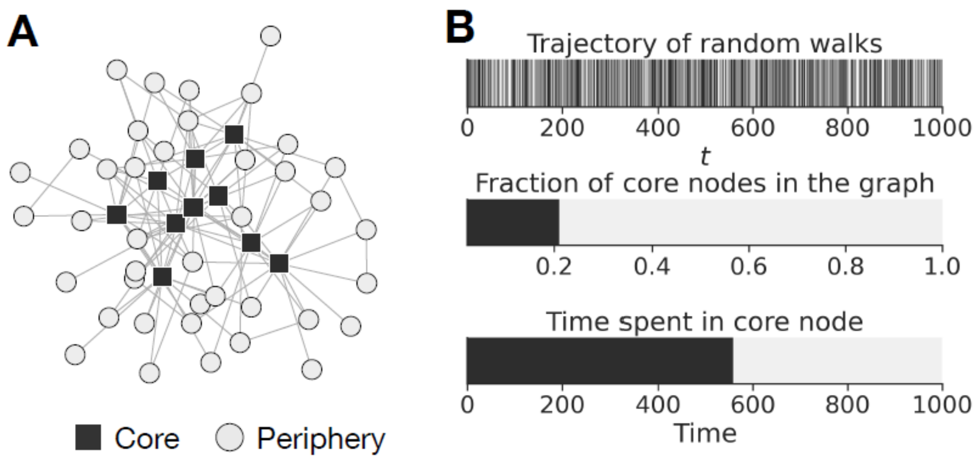




Reference: Wenqing Zheng, et.al. *Cold Brew: Distilling Graph Node Representations with Incomplete or Missing Neighborhoods*. In ICLR 2022

- Background
  - Existing strategy
    - Debias with heterogeneous information
    - Debias without heterogeneous information: Two-sided bias
      - Sampling-level debias
      - **Model-level debias (Mainly discussed today)**





Reference:
Sadamori Kojaku, et.al. *Residual2Vec: Debiasing graph embedding with random graphs*. In NIPS 2021

- Outline
  - Mainstream strategy: Train a competitive base model on head nodes first, then generalize it to tail nodes
    - Head nodes have abundant structural information, thus can encode less error in the base model
  - Different methods adopt different generalization strategy
    - Self-supervised learning based
      - B. Hao, et.al. *Pre-Training Graph Neural Networks for Cold-Start Users and Items Representation.* WSDM 2021
    - Meta-learning based
      - Z. Liu, et.al. *Towards Locality-Aware Meta-Learning of Tail Node Embeddings on Networks*. CIKM 2020
    - Missing neighborhood imputation
      - Z. Liu, et.al. *Tail-GNN: Tail-Node Graph Neural Networks*. KDD 2021
    - Knowledge distillation
      - W. Zheng, et.al. *Cold Brew: Distilling Graph Node Representations with Incomplete or Missing Neighborhoods.* ICLR 2022

Pre-Training Graph Neural Networks for Cold-Start Users and Items Representation

Bowen Hao
Renmin University of China
jeremyhao@ruc.edu.cn

Jing Zhang*
Renmin University of China
zhang-jing@ruc.edu.cn

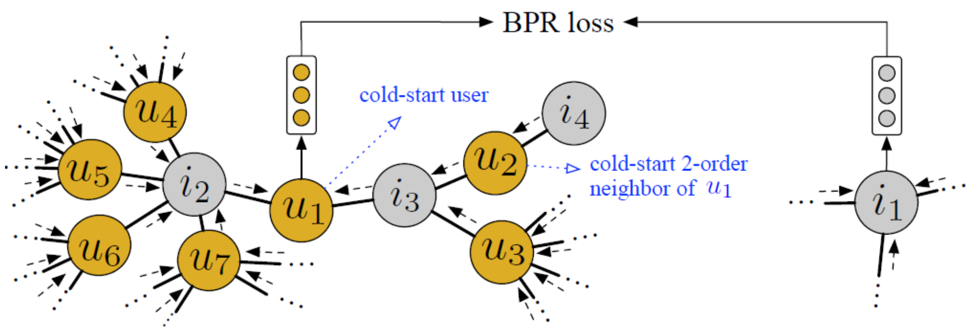
Hongzhi Yin
The University of Queensland
h.yin1@uq.edu.au

Cuiping Li
Renmin University of China
licuiping@ruc.edu.cn

Hong Chen
Renmin University of China
chong@ruc.edu.cn

WSDM 2021

- PT-GNN
  - Intuition
    - Despite advancing GNN models can incorporate higher-order collaborative signal to alleviate cold-start problem, they don’t explicitly optimize the tail(cold-start) users/items or deal with cold-start neighbors
    - Existing work, such as FastGCN and GraphSAGE, tend to do neighbor filtering randomly
  - Motivation & Contribution
    - *How to learn more accurate embeddings for cold-start users/items by GNN?*
      - A **reconstruction-based pre-training GNN model** is proposed to learn high-quality embeddings for cold-start users/items
    - *How to explicitly deal with high-order cold-start neighbors when performing graph convolution*?
      - A **meta aggregator** and an MDP-based **neighbor sampler** are proposed to enhance aggregation ability for each graph convolution step

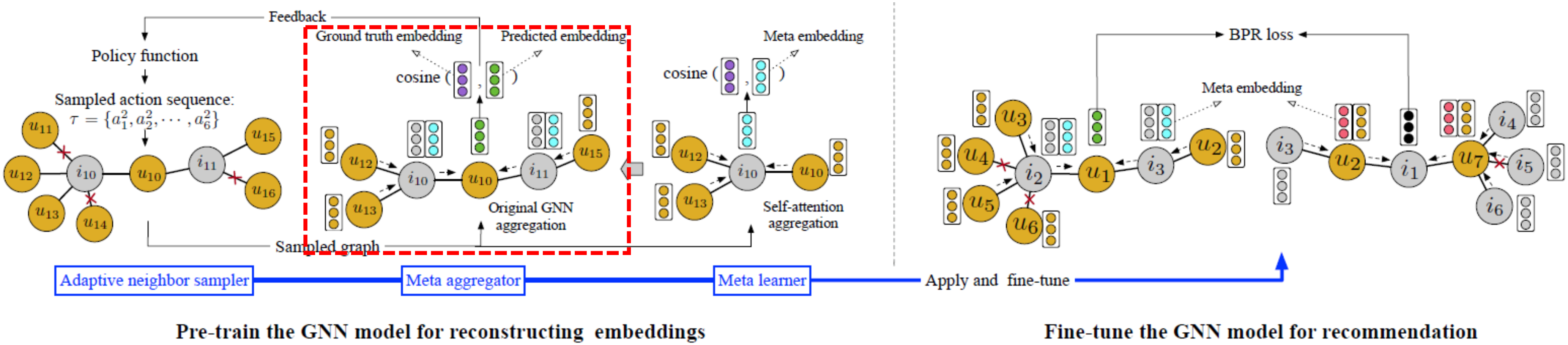


- PT-GNN
  - Basic pre-training GNN model
    - Pretext task: reconstruct the cold-start user/item embeddings
    - Training data: head nodes with abundant interactions
    - Ground truth: NCF embeddings learned upon full observed interactions
    - Synthetic cold-start users/items generation: Randomly sample K neighbors in each layer

$$
\begin{aligned}
\mathbf{h}^l_{\mathcal{N}(u)} &= \text{AGGREGATE}(\{\mathbf{h}^{l-1}_i, \forall i \in \mathcal{N}(u)\}), \\
\mathbf{h}^l_u &= \sigma(\mathbf{W}^l \cdot \text{CONCAT}(\mathbf{h}^{l-1}_u, \mathbf{h}^l_{\mathcal{N}(u)}),
\end{aligned}
$$

$$
\begin{aligned}
\mathbf{h}^L_{\mathcal{N}(u)} &= \text{AGGREGATE}(\{\mathbf{h}^{L-1}_i, \forall i \in \mathcal{N}(u)\}), \\
\mathbf{h}^L_u &= \sigma(\mathbf{W}^L \cdot \mathbf{h}^L_{\mathcal{N}(u)}).
\end{aligned}
$$

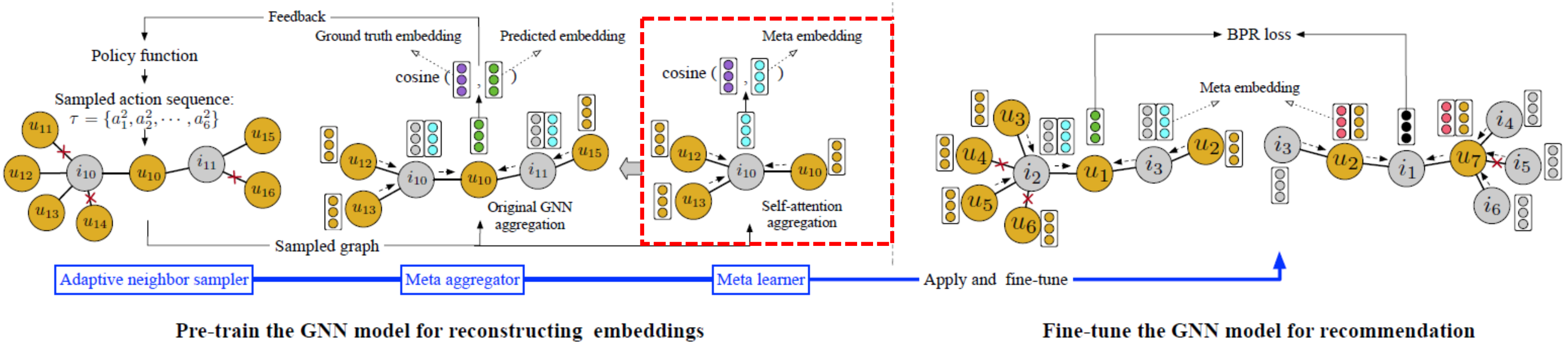
$$
\Theta^*_f = \arg\max_{\Theta_f} \sum_u \cos(\mathbf{h}^L_u, \mathbf{h}_u),
$$



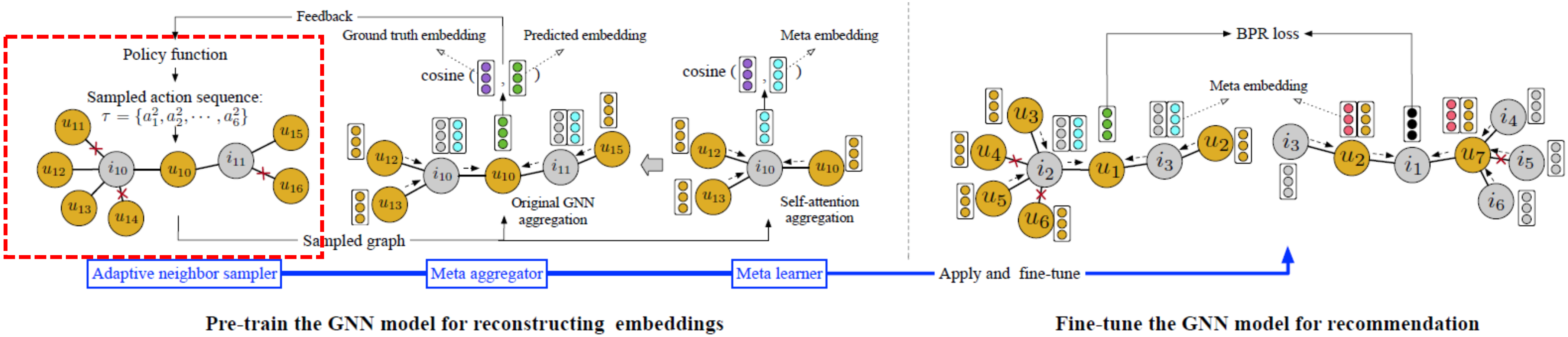
- PT-GNN
  - Meta aggregator
    - Cold-start neighbors will harm the embedding quality due to their inaccuracy
    - A self-attention based meta-learner g is designed to learn an additional embedding for each node based on first-order neighbors only
      - Pre-training GNN f is designed to deal with cold-start target nodes, while ***g* is designed for enhancing representation of cold-start neighbors**
      - Meta aggregator is an extension of pre-training GNN model

$$\{\mathbf{h}_1, \cdots, \mathbf{h}_K\} \leftarrow \mathrm{SELF\_ATTENTION}(\{\mathbf{h}_1^0, \cdots, \mathbf{h}_K^0\}),$$
$$\tilde{\mathbf{h}}_u = \mathrm{AVERAGE}(\{\mathbf{h}_1, \cdots, \mathbf{h}_K\}).$$

$$\mathbf{h}_u^l = \sigma(\mathbf{W}^l \cdot \mathrm{CONCAT}(\tilde{\mathbf{h}}_u, \mathbf{h}_u^{l-1}, \mathbf{h}_{N(u)}^l)),$$

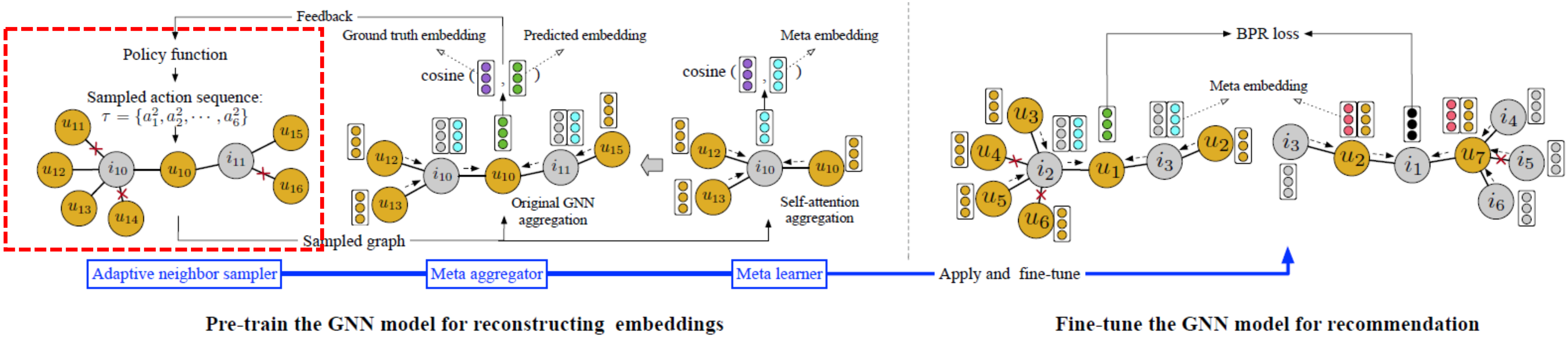


- PT-GNN
  - Adaptive neighbor sampler
    - Neighbor sampling as Markov Decision Process (MDP)
      - An adaptive sampling strategy can be learned based on feedbacks of pre-training GNN model
      - Neighbor sampler can be formulated as a $L-1$ MDP subtasks, where l_{th} subtask indicates sampling l-order neighbors
      - All the first-order neighbors are kept for a complete user profile, which means that the sampling will be conducted from second-order to L-order neighbors
      - The overall process will finish after the l-th subtask deletes all the neighbors or the L-th subtask is finished
      - **No assumption of what kind of neighbors are useful is made**



- PT-GNN
  - Adaptive neighbor sampler
    - Task formulation
      - State s_t^l : For the t-th l-order neighbor to be determined, state feature is defined as the cosine similarity between its initial embedding and target user u's initial embedding, and the element-wise product between its initial embedding of formerly selected neighbor by the l-1-th subtask and the average embedding of all the formerly selected neighbors, respectively
      - Action a_t^l : For the t-th l-order neighbor to be determined, a_t^l is defined as a binary value to represent whether to sample the neighbor or not
      - Policy P:

$$\begin{aligned} \mathbf{H}_t^l &= \mathrm{ReLU}(\mathbf{W}_1^l \mathbf{s}_t^l + \mathbf{b}^l), \\ P(a_t^l | \mathbf{s}_t^l, \Theta_s^l) &= a_t^l \sigma(\mathbf{W}_2^l \mathbf{H}_t^l) + (1 - a_t^l)(1 - \sigma(\mathbf{W}_2^l \mathbf{H}_t^l)), \end{aligned}$$



- PT-GNN
  - Adaptive neighbor sampler
    - Task formulation
      - Reward: Indicate whether the performed actions are reasonable or not, which is reflected by performance difference

$$R(a_t^l, \mathbf{s}_t^l) = \begin{cases} \cos(\hat{\mathbf{h}}_u^L, \mathbf{h}_u) - \cos(\mathbf{h}_u^L, \mathbf{h}_u) & \text{if } t = |\mathcal{N}^{l'}(u)| \wedge l = l'; \\ 0 & \text{otherwise,} \end{cases}$$

    - Objective function: **Monto-carlo policy gradient**

$$\sum_\tau P(\tau; \Theta_s) R(\tau)$$

$$\nabla_{\Theta_s} = \frac{1}{M} \sum_{m=1}^{M} \sum_{l=1}^{l'} \sum_{t=1}^{|\mathcal{N}^l(u)|} \nabla_{\Theta_s} \log P(a_t^{m,l} | \mathbf{s}_t^{m,l}, \Theta_s^l) R(a_t^{m,l}, s_t^{m,l}),$$

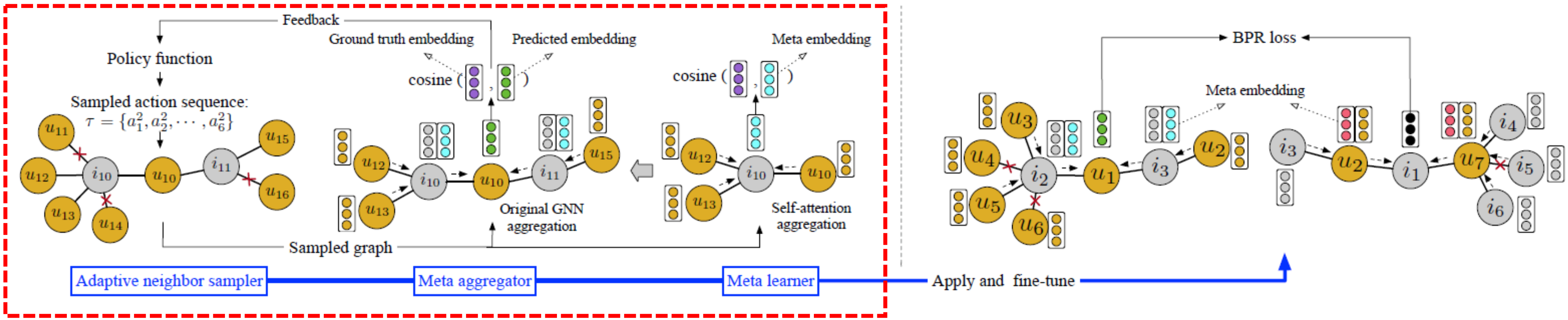
Algorithm 1: The Joint Training Process.

```
Input: Train_T = {(u_k, i_k)}, the ground truth embeddings
       {(h_u, h_i)}, a pre-trained meta learner with Θ_g^0, meta
       aggregator with Θ_f^0 and Θ_g^0 and neighbor sampler with Θ_s^0.
Initialize Θ_s = Θ_s^0, Θ_f = Θ_f^0, Θ_g = Θ_g^0 ;
for epoch from 1 to E do
    foreach u_k or i_k in Train_T do
        for l in {2, 3, ..., L} do
            Sample a sequence of actions
              τ^l = {a_1^l, ..., a_t^l, ..., a_{|N^l(u)|}^l} by Eq. (7);
            if ∀ a_t^l = 0 or l = L then
                Compute R(a_{|N^l(u)|}^l, s_{|N^l(u)|}^l) by Eq. (8);
                Compute gradients by Eq. (9);
                Break;
    Update Θ_s;
    if Jointly Training then
        Update Θ_g and Θ_f ;
```

- PT-GNN
  - Model training
    - During joint training in line 4, each parameter is updated by a linear combination of its old version and the new version for stable optimization, i.e. $\Theta_{new} = \lambda\Theta_{new} + (1-\lambda)\Theta_{old}$, where $\lambda \ll 1$

Algorithm 2: The Overall Training Process.

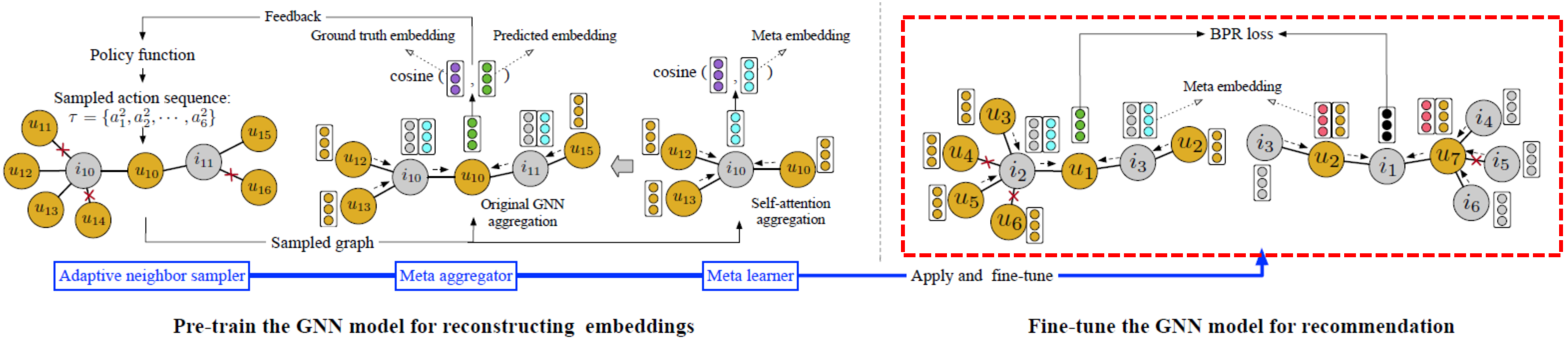
1 Pre-train the meta learner with parameter Θ_g;
2 Pre-train the meta aggregator with parameter Θ_f when fixing Θ_g;
3 Pre-train the neighbor sampler with parameter Θ_s by Algorithm 1 when fixing Θ_g and Θ_f;
4 Jointly train the three modules together with parameters Θ_g, Θ_f and Θ_s by running Algorithm 1;



Pre-train the GNN model for reconstructing embeddings
Fine-tune the GNN model for recommendation

- PT-GNN
  - Fine-tune for downstream recommendation task
    1. Neighbor sampling through pre-trained neighbor sampler
    2. Produce user/item embedding through pre-trained meta aggregator
    3. Obtain relevance score and perform fine-tune through BPR loss

$$\bar{y}(u, i) = \sigma(\mathbf{W} \cdot \mathbf{h}_u^L)^{\mathrm{T}} \sigma(\mathbf{W} \cdot \mathbf{h}_i^L)$$



- PT-GNN
  - Datasets

Table 1: Statistics of the Datasets.

Dataset	#Users	#Items	#Interactions	#Sparse Ratio
MovieLens-1M	6,040	3,706	1,000,209	4.47%
MOOCs	82,535	1,302	458,453	0.42%
Last.fm	992	1,084,866	19,150,868	1.78%

- PT-GNN
    - Intrinsic evaluation results (embedding inference)

Methods	Ml-1M (user)		MOOCs (user)		Last.fm (user)		Ml-1M (item)		MOOCs (item)		Last.fm (item)	
	3-shot	8-shot	3-shot	8-shot	3-shot	8-shot	3-shot	8-shot	3-shot	8-shot	3-shot	8-shot
NCF	-0.017	0.063	-0.098	-0.062	0.042	0.117	-0.118	-0.017	-0.036	0.027	-0.036	-0.018
GraphSAGE	0.035	0.105	0.085	0.128	0.104	0.134	0.113	0.156	0.116	0.182	0.112	0.198
Basic-GraphSAGE	0.076	0.198	0.103	0.152	0.132	0.184	0.145	0.172	0.172	0.196	0.166	0.208
Meta-GraphSAGE	0.258	0.271	0.298	0.320	0.186	0.209	0.434	0.448	0.288	0.258	0.312	0.333
NSampler-GraphSAGE	0.266	0.284	0.294	0.336	0.196	0.212	0.448	0.460	0.286	0.306	0.326	0.336
GraphSAGE*	**0.368**	**0.375**	**0.302**	**0.338**	**0.326**	**0.384**	**0.470**	**0.491**	**0.316**	**0.336**	**0.336**	**0.353**
GAT	0.020	0.049	0.092	0.138	0.092	0.125	0.116	0.126	0.108	0.118	0.106	0.114
Basic-GAT	0.046	0.158	0.104	0.168	0.158	0.180	0.134	0.168	0.112	0.126	0.209	0.243
Meta-GAT	0.224	0.282	0.284	0.288	0.206	0.212	0.438	0.462	0.294	0.308	0.314	0.340
NSampler-GAT	0.296	0.314	0.339	0.354	0.198	0.206	0.464	0.472	0.394	0.396	0.338	0.358
GAT*	**0.365**	**0.379**	**0.306**	**0.366**	**0.309**	**0.394**	**0.496**	**0.536**	**0.362**	**0.384**	**0.346**	**0.364**
FastGCN	0.009	0.012	0.063	0.095	0.082	0.114	0.002	0.036	0.007	0.018	0.007	0.013
Basic-FastGCN	0.082	0.146	0.083	0.146	0.104	0.149	0.088	0.113	0.099	0.121	0.159	0.182
Meta-FastGCN	0.181	0.192	0.282	0.280	0.224	0.274	0.216	0.266	0.248	0.278	0.230	0.258
NSampler-FastGCN	0.188	0.194	0.281	0.286	0.226	0.277	0.268	0.288	0.267	0.296	0.246	0.253
FastGCN*	**0.198**	**0.212**	**0.288**	**0.291**	**0.266**	**0.282**	**0.282**	**0.298**	**0.296**	**0.302**	**0.268**	**0.278**
FBNE	0.034	0.102	0.053	0.065	0.142	0.164	0.168	0.190	0.137	0.168	0.127	0.133
Basic-FBNE	0.162	0.190	0.162	0.185	0.135	0.180	0.176	0.209	0.157	0.180	0.167	0.173
Meta-FBNE	0.186	0.204	0.269	0.284	0.175	0.192	0.426	0.449	0.236	0.272	0.178	0.182
NSampler-FBNE	0.208	0.216	0.259	0.283	0.203	0.207	0.422	0.439	0.226	0.273	0.164	0.183
FBNE*	**0.242**	**0.265**	**0.306**	**0.321**	**0.206**	**0.219**	**0.481**	**0.490**	**0.301**	**0.382**	**0.182**	**0.199**
LightGCN	0.093	0.108	0.060	0.068	0.162	0.184	0.201	0.262	0.181	0.232	0.213	0.245
Basic-LightGCN	0.178	0.192	0.212	0.226	0.182	0.192	0.318	0.336	0.234	0.260	0.252	0.290
Meta-LightGCN	0.226	0.241	0.272	0.285	0.206	0.221	0.336	0.346	0.314	0.331	0.372	0.392
NSampler-LightGCN	0.238	0.256	0.286	0.294	0.204	0.212	0.348	0.384	0.296	0.314	0.356	0.401
LightGCN*	**0.270**	**0.286**	**0.292**	**0.309**	**0.229**	**0.234**	**0.382**	**0.408**	**0.334**	**0.353**	**0.386**	**0.403**

- PT-GNN
  - Extrinsic evaluation results (recommendation)

Methods	Ml-1M		MOOCs		Last.fm	
	Recall	NDCG	Recall	NDCG	Recall	NDCG
NCF	0.008	0.101	0.021	0.047	0.005	0.007
GraphSAGE	0.006	0.082	0.085	0.066	0.003	0.011
Basic-GraphSAGE	0.013	0.135	0.082	0.091	0.007	0.044
Meta-GraphSAGE	0.016	0.209	0.096	0.116	0.007	0.097
NSampler-GraphSAGE	0.021	0.221	0.101	0.122	0.008	0.088
GraphSAGE*	**0.024**	**0.235**	**0.110**	**0.129**	**0.008**	**0.131**
GAT	0.008	0.099	0.023	0.055	0.006	0.033
Basic-GAT	0.016	0.163	0.032	0.093	0.006	0.147
Meta-GAT	0.017	0.191	0.063	0.123	0.009	0.184
NSampler-GAT	0.012	0.188	0.084	0.132	0.010	0.199
GAT*	**0.014**	**0.208**	**0.100**	**0.139**	**0.018**	**0.232**
FastGCN	0.003	0.019	0.064	0.089	0.006	0.068
Basic-FastGCN	0.008	0.102	0.099	0.117	0.012	0.083
Meta-FastGCN	0.009	0.123	0.105	0.124	0.018	0.116
NSampler-FastGCN	0.011	0.118	0.108	0.128	0.020	0.136
FastGCN*	**0.012**	**0.123**	**0.119**	**0.140**	**0.023**	**0.186**
FBNE	0.002	0.088	0.048	0.041	0.009	0.013
Basic-FBNE	0.009	0.104	0.064	0.087	0.003	0.032
Meta-FBNE	0.012	0.101	0.088	0.101	0.006	0.087
NSampler-FBNE	0.013	0.118	0.102	0.117	0.006	0.099
FBNE*	**0.014**	**0.121**	**0.117**	**0.138**	**0.007**	**0.129**
LightGCN	0.014	0.207	0.102	0.112	0.001	0.083
Basic-LightGCN	0.012	0.211	0.112	0.121	0.005	0.097
Meta-LightGCN	0.018	0.221	0.120	0.139	0.005	0.101
NSampler-LightGCN	0.020	0.218	0.116	0.132	0.007	0.106
LightGCN*	**0.022**	**0.227**	**0.123**	**0.142**	**0.007**	**0.114**

Towards Locality-Aware Meta-Learning of Tail Node Embeddings on Networks

Zemin Liu[1*], Wentao Zhang[2*†], Yuan Fang[1], Xinming Zhang[2], Steven C.H. Hoi[1,3‡]
[1]Singapore Management University, [2]University of Science and Technology of China, [3]Salesforce Research Asia
{zmliu, yfang}@smu.edu.sg, zwt@mail.ustc.edu.cn, xinming@ustc.edu.cn, shoi@salesforce.com

CIKM 2020

- Meta-Tail2Vec
  - Intuition
    - Tail node embedding is a **few-shot learning task**
    - By using regression model trained by head nodes only, the quality of predicted tail node embeddings can approach which of head node embeddings, even with scarce structural information
  - Motivation & Contribution
    - *How to learn effective embeddings for tail nodes with scarce structural information?*
      - Formulate the tail node embedding learning problem as a regression task via **oracle reconstruction**
    - *How to ensure that the regression model can fit both head and tail nodes?*
      - A base regression model hinged on link dropout is proposed
    - *How to adapt the model to the unique locality of each node?*
      - **Locality-aware tasks** are formulated in a meta-learning framework, which allows for easy local adaptation of the base model

- Meta-Tail2Vec
  - Problem formulation
    - Given tail node set $V_{tail} = \{v \in V: |N_v| \leq k\}$ and head node set $V_{head} = \{v \in V: |N_v| > k\}$, where $|N_v|$ is the node degree
    - Denote embeddings of head nodes and tail nodes as $O = \{h_v: v \in V_{head}\}$ and $\{h_v: v \in V_{tail}\}$ respectively, which are actually node features in this work
    - A regression model $F(v; \Theta)$ is trained on the head nodes, treating O obtained by base embedding model ϕ as the oracle embeddings
      - $F(v; \Theta)$ is expected to output new predicted embeddings $\widehat{h_v}$ to reconstruct the oracle embeddings O
      - Optimization:

$$\arg\min_{\Theta} \sum_{v \in \mathcal{V}_{\text{head}}} \|F(v; \Theta) - \mathbf{h}_v\|^2$$

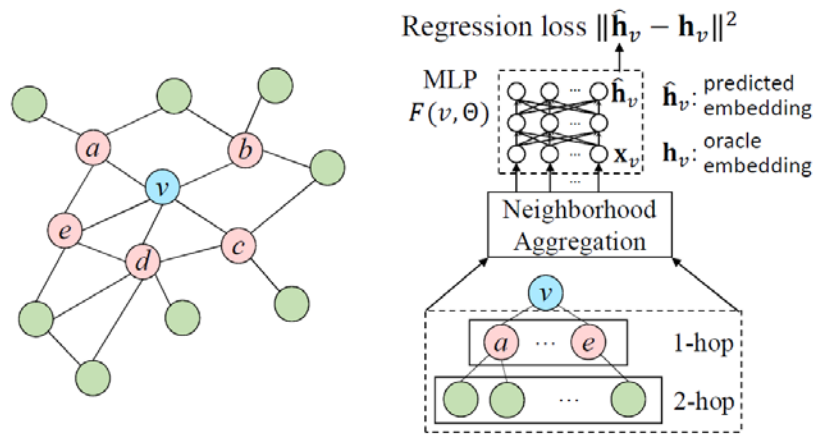
    - Goal: learn new embedding vectors for tail nodes $\{\widehat{h_v}: v \in V_{tail}\}$ with improved quality

- Meta-Learned Few-Shot Regression
  - Embedding regression model
    - m-hop neighborhood aggregation

$$\mathcal{N}_v^{(m)} = \bigcup_{i \in \mathcal{N}_v^{(m-1)}} \mathcal{N}_i \qquad \mathbf{x}_v = \textsc{Aggr}(\{\mathbf{h}_i : i \in \mathcal{N}_v^{(1)} \cup \mathcal{N}_v^{(2)} \cup \ldots \cup \mathcal{N}_v^{(m)}\})$$

    - x_v is leveraged as the input of MLP to reconstruct the embedding h_v

$$\hat{\mathbf{h}}_v = F(v; \Theta) = W_2 \cdot \sigma(W_1 \mathbf{x}_v + \mathbf{b}_1) + \mathbf{b}_2$$



(a) 2-hop neighbors of node v (b) Embedding regression model

- Meta-Learned Few-Shot Regression
  - Locality-aware few-shot regression tasks
    - Why not more extreme pre-train then fine-tune strategy? → Fine-tune can easily cause overfitting in tail nodes due to limited structural information
    - **Episodic meta-learning framework** is adopted to learn an adaptable prior regression model
    - Each task represents the unique locality of a node
      - Given query node, we aim to predict its embedding after training on a set of support nodes
      - The goal is to extract common prior knowledge of all tasks in an MAML manner
  - Task formulation
    - For each head node v, a meta-training task $T_v = (S_v, q_v)$ is defined, where $S_v = \{(i, h_i): i \in \widetilde{N_v}\}$ and $q_v = (v, h_v)$ are support set and query respectively, where $\widetilde{N_v}$ is the randomly sampled neighborhood with K neighbors
    - For each tail node u, a meta-testing task $T_u = (S_u, q_u)$ is defined, where $S_u = \{(i, h_i): i \in N_u\}$ and $q_u = (u, ?)$ are support set and query respectively, where the embedding of u is unknown and to be predicted
    - A prior regression model $F(v; \Theta)$ is learned, which can be quickly adapted to new tasks by performing just a few gradient updates on the support set of the new task

- Meta-Learned Few-Shot Regression
  - Task formulation
    - Given task $T_v = (S_v, q_v)$, the prior Θ will be adapted by S_v, generating local model Θ'_v

$$L_{S_v}(\Theta) = \sum_{(i,\mathbf{h}_i)\in S_v} \|F(i;\Theta) - \mathbf{h}_i)\|^2 \qquad \Theta'_v = \Theta - \alpha \frac{\partial L_{S_v}(\Theta)}{\partial \Theta}$$

    - Afterwards, local model Θ'_v will be applied to query node v to calculate the task loss, which is subsequently used to optimize prior Θ

$$L_{q_v}(\Theta'_v) = \|F(v;\Theta'_v) - \mathbf{h}_v\|^2 \qquad \arg\min_{\Theta} \sum_{T_v=(S_v,q_v)\in\mathcal{T}_{\text{train}}} L_{q_v}\left(\Theta - \alpha \frac{\partial L_{S_v}(\Theta)}{\partial \Theta}\right)$$

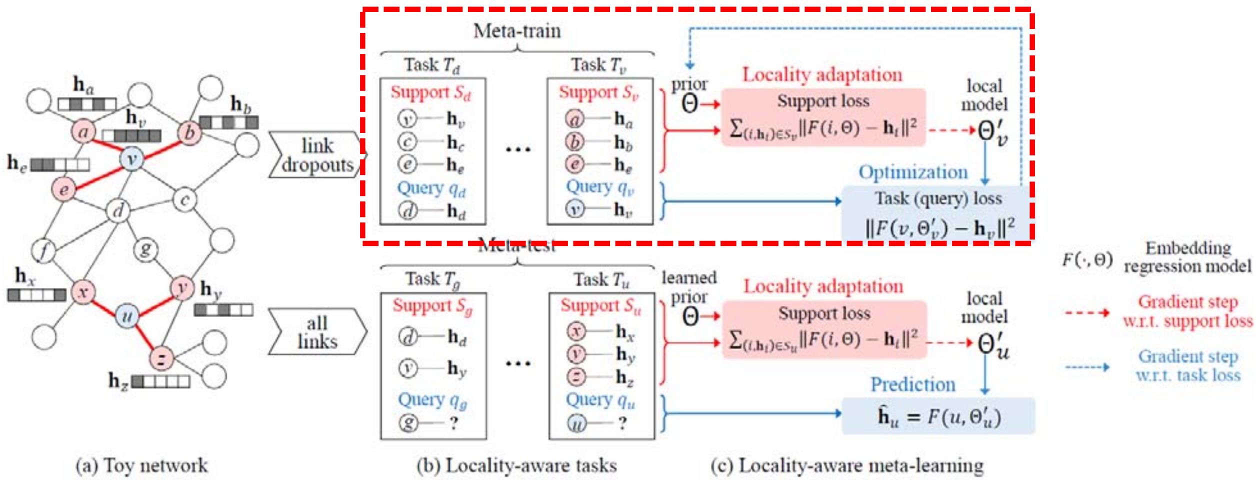


Figure 3: Overall framework of our locality-aware tail node embedding model meta-tail2vec. (Best viewed in color.)

- Meta-Learned Few-Shot Regression
  - More details
    - Why use neighbors of query node as support sets? → Randomly sampled support nodes aren't related to query node, thus can't reflect the locality of query node
    - Why randomly sample K neighbors as support set? → Transform each meta-training task as few-shot regression task, which is more similar to meta-testing tasks
    - Only head nodes are kept/sampled in support set: → Make sure they are all related to an oracle embedding for regression adaptation
      - For meta-testing tasks with no head nodes in support set, the original embedding will be directly used as output embedding

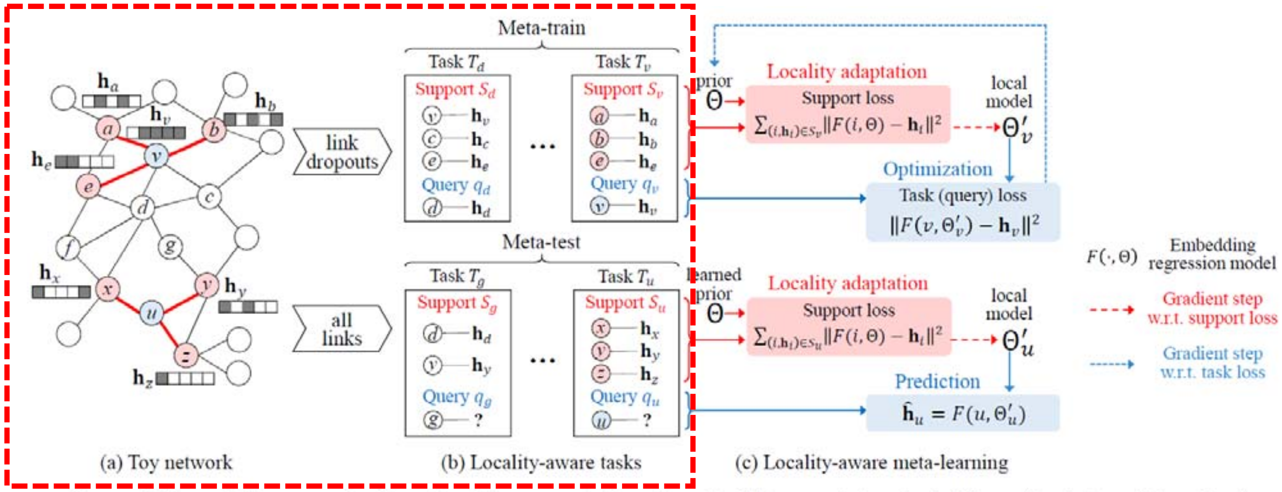


Figure 3: Overall framework of our locality-aware tail node embedding model meta-tail2vec. (Best viewed in color.)

- Meta-Tail2Vec
  - Datasets

Table 1: Summary of datasets.

	# nodes	# edges	# node classes	multi-label	# tail nodes
Wiki	2,405	17,981	19	No	1,069
Flickr	80,513	5,899,882	195	Yes	9,367
Email	1,005	25,571	42	No	235

- Meta-Tail2Vec
  - Experiment results

Table 2: Performance of node classification w.r.t. classic base embedding models.

		Base	Biased walk	Additive	Additive-2	a la carte	a la carte-2	Nonce2vec	Dropout	meta-tail2vec	Improv. over Base	Improv. over 2nd best
DeepWalk as the base embedding model												
Wiki	MicroF	44.27 ± 0.25	44.69 ± 0.31	45.32 ± 0.52	42.11 ± 0.76	23.65 ± 0.44	23.34 ± 0.47	44.97 ± 0.29	36.88 ± 0.65	**49.10** ± 0.23	+10.9%	+8.3%
	Accuracy	46.68 ± 0.31	47.05 ± 0.17	47.18 ± 0.29	44.73 ± 0.53	24.17 ± 0.49	24.48 ± 0.42	47.11 ± 0.22	38.13 ± 0.57	**50.70** ± 0.45	+8.6%	+7.5%
Flickr	MicroF	33.48 ± 0.26	33.61 ± 0.39	34.43 ± 0.41	32.59 ± 0.17	31.89 ± 0.47	32.25 ± 0.35	33.83 ± 0.28	33.91 ± 0.22	**36.31** ± 0.19	+8.5%	+5.5%
	Accuracy	32.44 ± 0.13	32.57 ± 0.19	33.29 ± 0.17	31.31 ± 0.24	32.13 ± 0.26	32.62 ± 0.31	33.01 ± 0.15	32.86 ± 0.09	**35.28** ± 0.25	+8.8%	+6.0%
Email	MicroF	51.32 ± 0.29	50.95 ± 0.24	52.50 ± 0.17	51.17 ± 0.23	17.88 ± 0.48	18.21 ± 0.52	51.84 ± 0.33	32.72 ± 0.45	**55.26** ± 0.18	+7.7%	+5.3%
	Accuracy	54.41 ± 0.34	54.13 ± 0.22	55.38 ± 0.43	53.82 ± 0.36	21.06 ± 0.45	21.13 ± 0.37	54.79 ± 0.19	33.85 ± 0.51	**57.78** ± 0.29	+6.2%	+4.3%
GraphSAGE as the base embedding model												
Wiki	MicroF	39.68 ± 0.24	40.07 ± 0.15	37.84 ± 0.31	35.96 ± 0.43	23.88 ± 0.47	22.52 ± 0.39	40.75 ± 0.33	19.78 ± 0.59	**44.29** ± 0.31	+11.6%	+8.7%
	Accuracy	41.22 ± 0.19	41.39 ± 0.06	39.31 ± 0.26	36.59 ± 0.25	25.71 ± 0.36	24.94 ± 0.62	41.65 ± 0.28	24.73 ± 0.42	**44.90** ± 0.12	+8.9%	+7.8%
Flickr	MicroF	29.38 ± 0.32	28.75 ± 0.31	27.86 ± 0.14	23.69 ± 0.44	30.02 ± 0.17	29.67 ± 0.20	29.85 ± 0.12	28.75 ± 0.11	**32.11** ± 0.41	+9.3%	+7.0%
	Accuracy	28.46 ± 0.08	27.52 ± 0.19	27.69 ± 0.31	22.82 ± 0.45	29.83 ± 0.22	28.18 ± 0.46	29.26 ± 0.31	28.78 ± 0.14	**31.96** ± 0.35	+12.3%	+7.1%
Email	MicroF	41.25 ± 0.17	41.07 ± 0.33	35.83 ± 0.31	34.19 ± 0.13	27.81 ± 0.44	26.97 ± 0.39	41.97 ± 0.24	23.47 ± 0.25	**46.73** ± 0.37	+13.3%	+11.3%
	Accuracy	42.61 ± 0.31	42.20 ± 0.31	37.25 ± 0.16	35.13 ± 0.35	29.41 ± 0.46	27.16 ± 0.34	43.23 ± 0.30	25.84 ± 0.18	**47.70** ± 0.46	+11.9%	+10.3%

- Meta-Tail2Vec
  - Experiment results

Table 3: Performance of link prediction w.r.t. classic base embedding models.

		Base	Biased walk	Additive	Additive-2	a la carte	a la carte-2	Nonce2vec	Dropout	meta-tail2vec	Improv. over Base	Improv. over 2^{nd} best
DeepWalk as the base embedding model												
Wiki	MRR	75.28 ± 0.37	75.13 ± 0.41	75.81 ± 0.62	74.89 ± 0.78	76.31 ± 0.25	76.14 ± 0.33	67.42 ± 0.87	77.06 ± 0.71	**79.18** ± 0.52	+5.2%	+2.8%
	Hit@1	51.83 ± 0.42	52.04 ± 0.57	52.51 ± 0.67	51.48 ± 0.39	53.70 ± 0.61	53.59 ± 0.32	53.34 ± 0.49	54.19 ± 0.30	**57.22** ± 0.46	+10.4%	+5.6%
Flickr	MRR	50.05 ± 0.30	49.57 ± 0.19	49.80 ± 0.45	49.72 ± 0.41	50.36 ± 0.55	50.71 ± 0.65	50.83 ± 0.48	50.25 ± 0.59	**52.18** ± 0.61	+4.3%	+2.7%
	Hit@1	25.32 ± 0.24	25.63 ± 0.55	26.10 ± 0.41	26.55 ± 0.62	26.07 ± 0.30	26.39 ± 0.58	26.67 ± 0.33	26.19 ± 0.44	**28.11** ± 0.40	+11.0%	+5.4%
Email	MRR	44.17 ± 0.35	44.58 ± 0.26	44.52 ± 0.68	44.96 ± 0.28	44.49 ± 0.50	45.11 ± 0.34	44.80 ± 0.15	45.33 ± 0.08	**48.42** ± 0.55	+9.6%	+6.8%
	Hit@1	19.47 ± 0.38	19.96 ± 0.27	21.38 ± 0.15	21.66 ± 0.40	22.45 ± 0.58	22.63 ± 0.31	20.90 ± 0.44	23.02 ± 0.33	**24.31** ± 0.46	+24.9%	+5.6%
GraphSAGE as the base embedding model												
Wiki	MRR	81.36 ± 0.14	82.01 ± 0.10	80.56 ± 0.45	80.39 ± 0.21	81.82 ± 0.53	80.94 ± 0.62	82.18 ± 0.64	82.52 ± 0.40	**84.38** ± 0.61	+3.7%	+2.3%
	Hit@1	58.87 ± 0.52	58.39 ± 0.15	58.43 ± 0.61	58.92 ± 0.30	59.56 ± 0.29	59.34 ± 0.44	59.70 ± 0.37	59.93 ± 0.56	**62.04** ± 0.68	+5.4%	+3.5%
Flickr	MRR	55.83 ± 0.29	56.17 ± 0.36	55.04 ± 0.25	55.40 ± 0.58	56.28 ± 0.49	56.76 ± 0.40	56.31 ± 0.32	56.85 ± 0.71	**58.15** ± 0.43	+4.2%	+2.3%
	Hit@1	34.59 ± 0.52	35.15 ± 0.47	33.79 ± 0.38	33.36 ± 0.40	35.22 ± 0.68	35.29 ± 0.64	34.97 ± 0.50	35.74 ± 0.31	**36.92** ± 0.39	+6.7%	+3.3%
Email	MRR	46.71 ± 0.45	46.24 ± 0.29	46.05 ± 0.25	46.68 ± 0.44	47.03 ± 0.53	46.92 ± 0.30	47.18 ± 0.19	46.37 ± 0.60	**48.15** ± 0.44	+3.1%	+2.1%
	Hit@1	23.02 ± 0.23	22.73 ± 0.41	22.91 ± 0.44	22.65 ± 0.52	23.19 ± 0.39	23.14 ± 0.61	23.28 ± 0.43	23.07 ± 0.56	**24.55** ± 0.70	+6.6%	+5.4%

- Meta-Tail2Vec
  - Experiment results

Table 4: Performance of node classification w.r.t. robust base embedding models (MiF for MicroF; Acc for accuracy).

		Base	Additive	Nonce2vec	Dropout	meta-tail2vec
SDNE as the base embedding model						
Wiki	MiF	31.38 ± 0.34	34.46 ± 0.63	32.14 ± 0.75	34.69 ± 0.48	**37.99** ± 0.83
	Acc	34.10 ± 0.71	35.62 ± 0.28	34.59 ± 0.12	36.46 ± 0.43	**38.80** ± 0.64
Flickr	MiF	34.74 ± 0.86	35.49 ± 0.47	34.73 ± 0.37	35.38 ± 0.35	**38.50** ± 0.78
	Acc	32.67 ± 0.32	34.72 ± 0.28	33.59 ± 0.73	34.64 ± 0.49	**38.03** ± 0.66
Email	MiF	29.85 ± 0.48	31.07 ± 0.21	31.83 ± 0.46	34.50 ± 0.25	**47.21** ± 0.72
	Acc	32.90 ± 0.62	34.37 ± 0.27	33.79 ± 0.60	37.85 ± 0.47	**51.70** ± 0.33
ARGA as the base embedding model						
Wiki	MiF	32.22 ± 0.46	31.19 ± 0.23	32.47 ± 0.19	32.85 ± 0.23	**33.51** ± 0.31
	Acc	34.47 ± 0.52	33.84 ± 0.10	34.79 ± 0.21	35.22 ± 0.49	**35.80** ± 0.23
Flickr	MiF	24.60 ± 0.15	23.69 ± 0.18	25.16 ± 0.20	24.71 ± 0.15	**25.93** ± 0.25
	Acc	22.81 ± 0.17	21.59 ± 0.11	24.26 ± 0.46	23.65 ± 0.39	**25.37** ± 0.16
Email	MiF	24.38 ± 0.31	23.94 ± 0.47	24.97 ± 0.35	25.48 ± 0.53	**26.11** ± 0.25
	Acc	26.57 ± 0.43	25.69 ± 0.30	27.02 ± 0.37	27.54 ± 0.21	**27.95** ± 0.16
DDGCN as the base embedding model						
Wiki	MiF	29.49 ± 0.18	27.68 ± 0.58	31.20 ± 0.44	30.37 ± 0.35	**33.02** ± 0.43
	Acc	31.39 ± 0.25	30.82 ± 0.21	33.87 ± 0.75	32.69 ± 0.40	**36.27** ± 0.41
Flickr	MiF	28.57 ± 0.47	26.92 ± 0.08	30.09 ± 0.35	29.17 ± 0.26	**31.03** ± 0.52
	Acc	25.90 ± 0.71	24.44 ± 0.12	26.76 ± 0.49	26.37 ± 0.25	**28.38** ± 0.54
Email	MiF	38.95 ± 0.67	38.73 ± 0.55	39.62 ± 0.43	39.15 ± 0.40	**41.83** ± 0.34
	Acc	39.81 ± 0.56	38.20 ± 0.35	42.32 ± 0.63	41.69 ± 0.41	**44.13** ± 0.73

Table 5: Performance of link prediction w.r.t. robust base embedding models (H@1 for hit@1).

		Base	Additive	Nonce2vec	Dropout	meta-tail2vec
SDNE as the base embedding model						
Wiki	MRR	72.25 ± 0.48	72.53 ± 0.30	74.44 ± 0.39	75.08 ± 0.65	**76.97** ± 0.61
	H@1	52.19 ± 0.27	51.94 ± 0.39	54.50 ± 0.61	55.21 ± 0.35	**57.58** ± 0.74
Flickr	MRR	46.82 ± 0.20	47.09 ± 0.44	48.35 ± 0.51	48.17 ± 0.29	**49.31** ± 0.46
	H@1	26.23 ± 0.16	27.00 ± 0.33	28.82 ± 0.61	28.39 ± 0.10	**29.26** ± 0.20
Email	MRR	34.02 ± 0.76	34.29 ± 0.51	36.87 ± 0.49	36.42 ± 0.55	**39.55** ± 0.50
	H@1	17.51 ± 0.24	18.65 ± 0.51	21.19 ± 0.40	20.88 ± 0.13	**22.86** ± 0.63
ARGA as the base embedding model						
Wiki	MRR	48.57 ± 0.40	46.49 ± 0.38	49.16 ± 0.45	50.27 ± 0.14	**51.08** ± 0.20
	H@1	41.40 ± 0.52	40.49 ± 0.07	41.67 ± 0.35	42.22 ± 0.10	**43.73** ± 0.65
Flickr	MRR	35.52 ± 0.32	35.41 ± 0.72	35.69 ± 0.63	36.31 ± 0.28	**36.87** ± 0.45
	H@1	29.73 ± 0.34	28.86 ± 0.40	29.89 ± 0.62	30.51 ± 0.77	**31.37** ± 0.29
Email	MRR	26.83 ± 0.29	25.89 ± 0.47	26.91 ± 0.18	26.22 ± 0.40	**27.26** ± 0.55
	H@1	16.51 ± 0.29	16.30 ± 0.42	17.22 ± 0.40	16.89 ± 0.31	**17.87** ± 0.35
DDGCN as the base embedding model						
Wiki	MRR	73.25 ± 0.49	74.10 ± 0.34	74.28 ± 0.15	74.92 ± 0.53	**75.31** ± 0.67
	H@1	51.28 ± 0.39	50.77 ± 0.21	51.86 ± 0.45	52.56 ± 0.32	**53.30** ± 0.61
Flickr	MRR	52.17 ± 0.40	50.74 ± 0.51	52.23 ± 0.42	51.79 ± 0.60	**52.49** ± 0.34
	H@1	37.15 ± 0.38	35.82 ± 0.85	37.53 ± 0.42	37.16 ± 0.60	**38.68** ± 0.63
Email	MRR	41.58 ± 0.45	40.83 ± 0.37	42.96 ± 0.39	42.81 ± 0.12	**43.47** ± 0.18
	H@1	27.35 ± 0.39	28.31 ± 0.63	28.58 ± 0.25	28.87 ± 0.30	**29.22** ± 0.36

- Meta-Tail2Vec
  - Experiment results

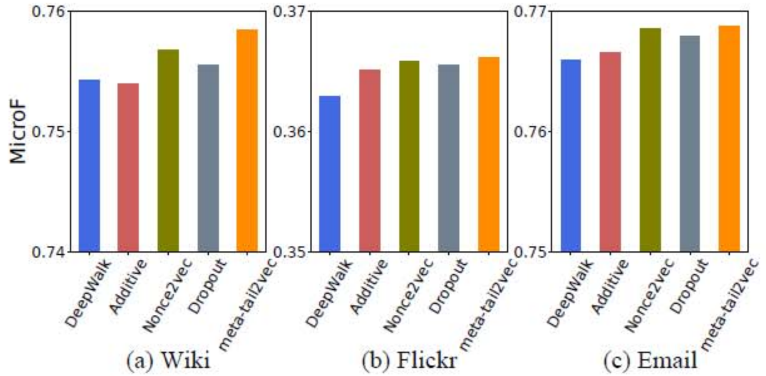


Figure 4: Performance of node classification on head nodes w.r.t. DeepWalk as the base embedding model.

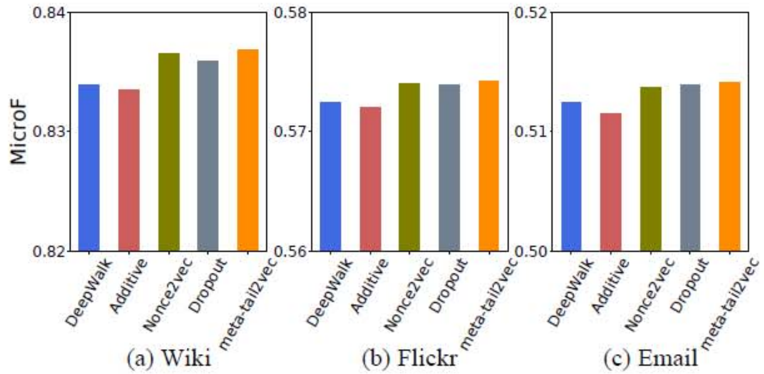


Figure 5: Performance of link prediction on head nodes w.r.t. DeepWalk as the base embedding model.

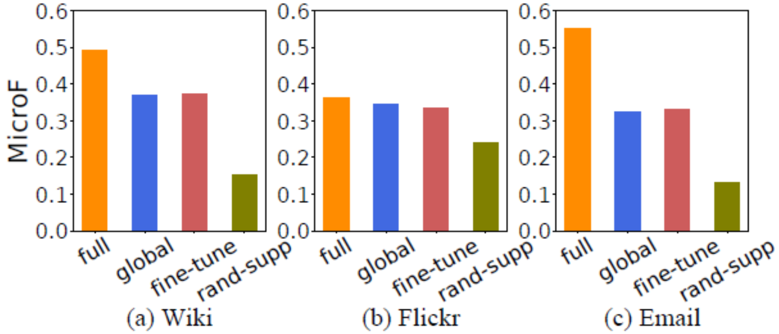


Figure 6: Ablation study of the meta-learning strategy on node classification w.r.t. DeepWalk as the base model.

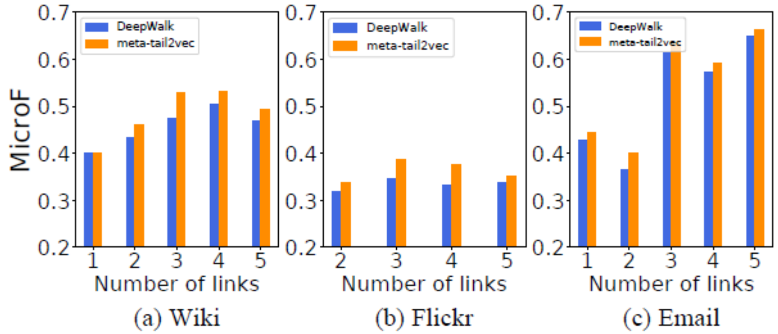


Figure 8: Impact of number of links on node classification w.r.t. DeepWalk as the base model.

Tail-GNN: Tail-Node Graph Neural Networks

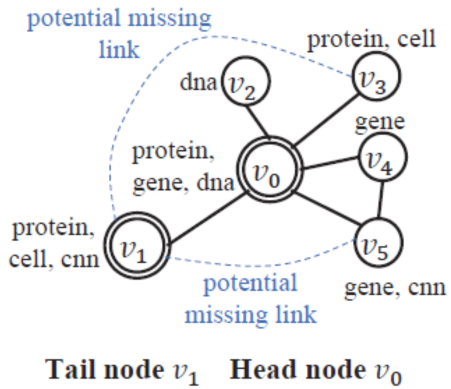
Zemin Liu
Singapore Management University
Singapore
zmliu@smu.edu.sg

Trung-Kien Nguyen
Singapore Management University
Singapore
tknguyen@smu.edu.sg

Yuan Fang
Singapore Management University
Singapore
yfang@smu.edu.sg

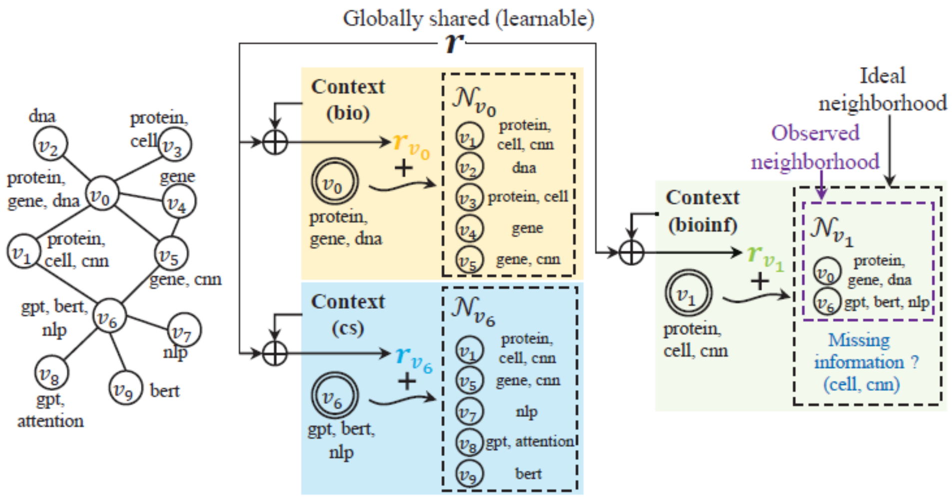
KDD 2021

- Tail-GNN
  - Intuition
    - Prior two-stage refinement model, such as meta-tail2vec, can't mitigate the bias and noise in base embedding model
    - Tail nodes can be directly transformed into heavy nodes through **missing neighborhood imputation**
  - Motivation and Contribution
    - *How to uncover missing information for tail nodes?*
      - Missing neighborhood can be imputed by exploring **potential weak links and relational tie** between head nodes and their neighborhood
    - *How to localize the missing information for individual tail node while maintaining the generality?*
      - A novel concept of **transferable neighborhood translation** can be exploited for missing neighborhood prediction
    - *How to utilize the predicted missing information in embedding model?*
      - A novel model Tail-GNN is proposed to perform aggregation with missing information for tail nodes



(b) Toy citation network

- Tail-GNN
  - Neighborhood translation and missing information prediction
    - Global neighborhood translation vector r_v : $h_v + r_v \approx h_{N_v}$, where h_{N_v} is the observed neighborhood representation obtained via pooling
    - Missing information definition for tail nodes: $m_v = h_{N_v^*} - h_{N_v}$, where $h_{N_v^*} = h_v + r_v$ is the ideal neighborhood representation



(a) Toy network (b) Neighborhood translation for head nodes (c) Neighborhood translation for tail nodes

- Tail-GNN
  - Details for neighborhood translation
    - How to transfer the neighborhood translation vector to achieve localizing
      - Scaling + Shifting: based on global translation vector r_v and local context

$$\mathbf{r}_v^l = \phi(\mathbf{h}_v^l, \mathbf{h}_{\mathcal{N}_v}^l, \mathbf{r}^l; \theta_\phi^l) = (\gamma_v^l + 1) \odot \mathbf{r}^l + \beta_v^l$$

$$\gamma_v^l = \text{LeakyReLU}(\mathbf{W}_\gamma^{l,1}\mathbf{h}_v^l + \mathbf{W}_\gamma^{l,2}\mathbf{h}_{\mathcal{N}_v}^l)$$

$$\beta_v^l = \text{LeakyReLU}(\mathbf{W}_\beta^{l,1}\mathbf{h}_v^l + \mathbf{W}_\beta^{l,2}\mathbf{h}_{\mathcal{N}_v}^l)$$

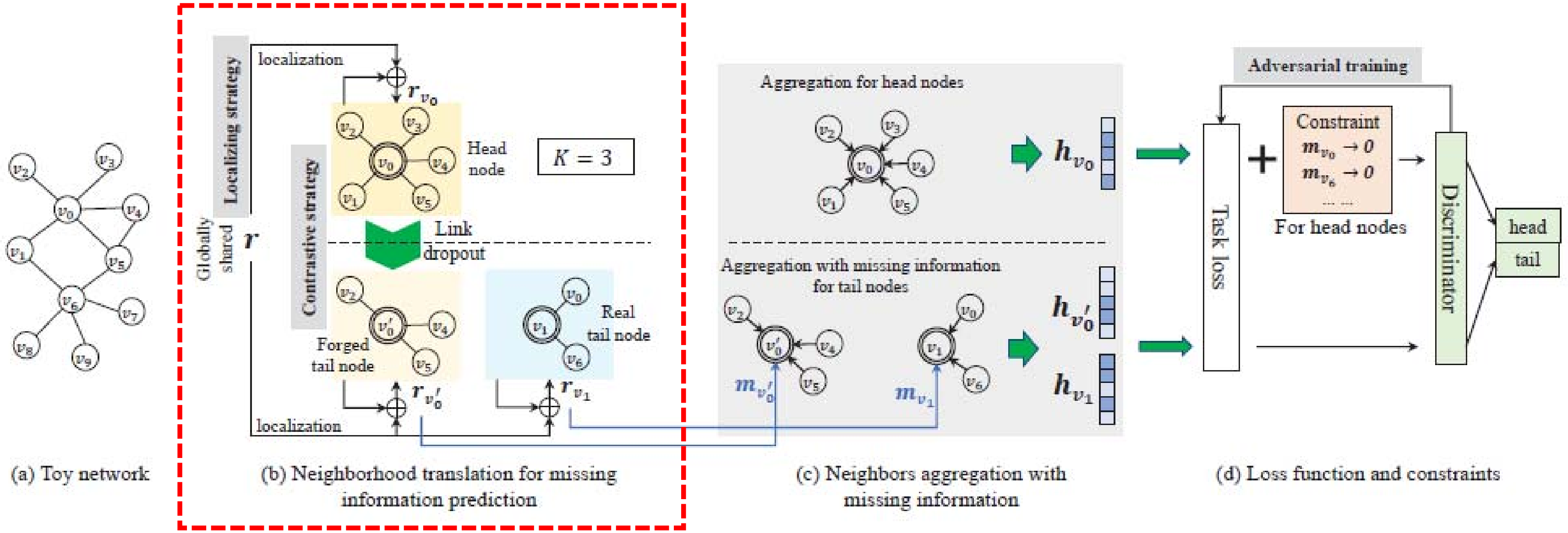


Figure 3: Overall framework of Tail-GNN.

- Tail-GNN
  - Details for proposed Tail-GNN method
    - Neighborhood aggregation with missing information

$$\mathrm{h}_v^{l+1} = \mathcal{M}(\mathrm{h}_v^l, \{\mathrm{m}_v^l\} \cup \{\mathrm{h}_i^l : i \in \mathcal{N}_v\}; \theta^{l+1})$$

    - Objective function

$$\mathcal{L}_m = \sum_{v \in \mathcal{V}_{\mathrm{tr}}} I_v \sum_{l=1}^{\ell} \|\mathrm{m}_v^{l-1}\|_2^2$$

$$\mathcal{L}_d = \sum_{v \in \mathcal{V}_{\mathrm{tr}}} \textsc{CrossEnt}(I_v, D(\mathrm{h}_v^{\ell}; \theta_d)) + \lambda_d \|\theta_d\|_2^2$$

$$\min_{\Theta} \max_{\theta_d} \; \mathcal{L}_t + \mu \mathcal{L}_m - \eta \mathcal{L}_d$$

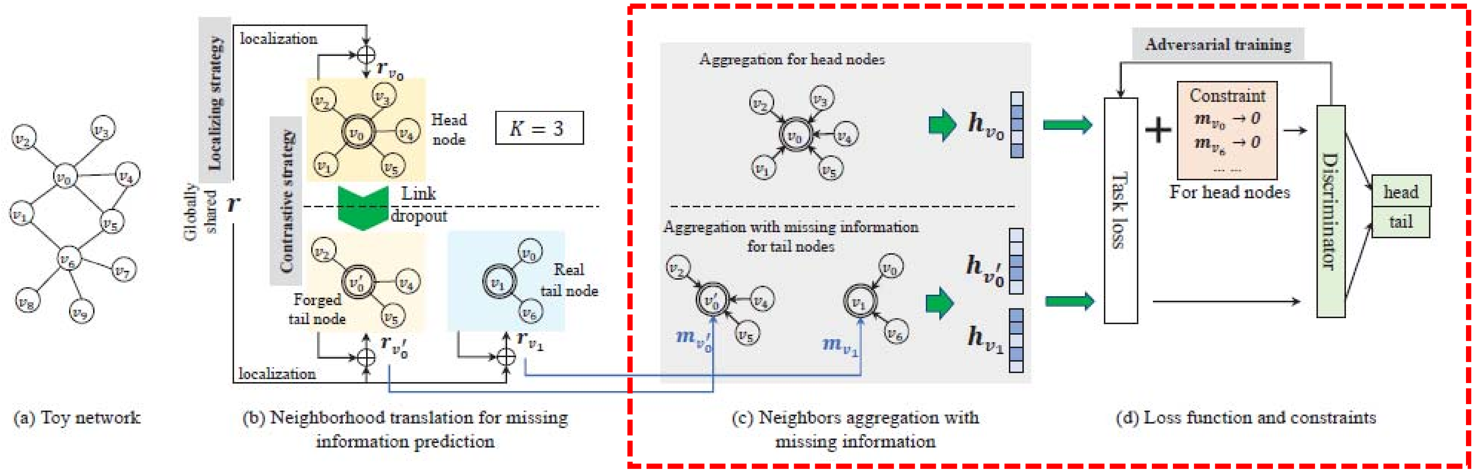


Figure 3: Overall framework of Tail-GNN.

- Tail-GNN
  - Other details for model training
    - Contrastive strategy: Generate more training data through head node link dropout to form correspondence
      - Help model to learn how to exactly impute missing neighborhood
    - Raw training set is composed of heavy nodes only, and tail nodes are only contained in test/valid sets

Algorithm 1 MODEL TRAINING FOR Tail-GNN

Input: graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathrm{X})$, task-related training data $\mathcal{V}_{\mathrm{tr}}$.
Output: Model parameters Θ.

initialize parameters Θ, θ_d;
while not converged **do**
    sample a batch of nodes from $\mathcal{V}_{\mathrm{tr}}$;
    for each node v in the batch **do** ▷ construct forged tail nodes
        if v is a head node **then**
            $v' \leftarrow \mathrm{LinkDropout}(v)$;
            add v' to the batch;
    for each node v in the batch **do**
        for each layer $l \in \{0, ..., \ell - 1\}$ **do**
            $\mathrm{r}_v^l = \phi(\mathrm{h}_v^l, \mathrm{h}_{\mathcal{N}_v}^l, \mathrm{r}^l; \theta_\phi^l)$; ▷ localization, Eq. (9)
            $\mathrm{m}_v^l \leftarrow \mathrm{h}_v^l + \mathrm{r}_v^l - \mathrm{h}_{\mathcal{N}_v}^l$; ▷ missing information, Eq. (7)
            if v is a head node **then** ▷ aggregation for head, Eq. (1)
                $\mathrm{h}_v^{l+1} \leftarrow \mathcal{M}(\mathrm{h}_v^l, \{\mathrm{h}_i^l : i \in \mathcal{N}_v\}; \theta^{l+1})$;
            else ▷ aggregation for tail, Eq. (12)
                $\mathrm{h}_v^{l+1} \leftarrow \mathcal{M}(\mathrm{h}_v^l, \{\mathrm{m}_v^l\} \cup \{\mathrm{h}_i^l : i \in \mathcal{N}_v\}; \theta^{l+1})$;
    update Θ by minimizing Eq. (16) with θ_d fixed;
    update θ_d by maximizing Eq. (16) with Θ fixed;
return Θ.

- Tail-GNN
  - Datasets

Table 1: Summary of datasets.

	# Nodes	# Edges	# Features	# Classes	# Tail ($K = 5$)
Email	1,005	25,571	128	42	235
Squirrel	5,201	217,073	2,089	5	942
Actor	7,600	33,391	931	5	4,823
CoauthorCS	18,333	327,576	6,805	15	8,037
Amazon	937,349	12,455,925	100	44	248,125

- Tail-GNN
  - Experimental results

Table 2: Evaluation on tail node classification using GCN as the base model.

Henceforth, tabular results are in percent; the best result is **bolded** and the runner-up is <u>underlined</u>; a dash (-) denotes no result reported for failing to work on a large dataset.

Methods	Email		Squirrel		Actor		CoauthorCS		Amazon	
	Accuracy	Micro-F	Accuracy	Micro-F	Accuracy	Micro-F	Accuracy	Micro-F	Accuracy	Micro-F
DeepWalk	54.4 ± 0.3	51.3 ± 0.3	<u>28.8</u> ± 1.6	<u>28.0</u> ± 2.3	21.8 ± 0.6	18.2 ± 0.9	84.1 ± 0.7	81.5 ± 0.7	83.7 ± 0.1	<u>74.3</u> ± 0.6
GCN	<u>57.9</u> ± 1.2	<u>57.7</u> ± 1.3	24.8 ± 1.3	23.2 ± 1.8	<u>29.7</u> ± 0.2	15.0 ± 0.9	88.4 ± 0.1	86.1 ± 0.1	82.3 ± 0.2	70.6 ± 0.1
Additive	55.4 ± 0.4	52.5 ± 0.2	27.0 ± 1.7	22.9 ± 1.6	28.1 ± 0.3	15.1 ± 1.3	89.5 ± 0.1	87.8 ± 0.1	<u>84.2</u> ± 0.2	73.2 ± 0.6
a la carte	21.1 ± 0.4	17.9 ± 0.5	22.5 ± 1.1	22.5 ± 0.7	28.0 ± 0.5	14.8 ± 1.4	88.7 ± 0.2	86.7 ± 0.3	81.1 ± 0.1	69.7 ± 0.7
meta-tail2vec	57.1 ± 0.1	55.3 ± 0.2	25.1 ± 0.5	21.5 ± 0.3	<u>29.7</u> ± 0.4	20.1 ± 0.7	89.3 ± 0.1	87.4 ± 0.1	81.9 ± 0.1	71.4 ± 0.4
SDNE	32.9 ± 0.6	29.8 ± 0.5	23.8 ± 3.2	16.6 ± 6.2	24.4 ± 0.8	12.6 ± 5.6	70.6 ± 0.9	64.5 ± 1.1	-	-
ARGA	45.1 ± 0.9	41.2 ± 1.0	22.4 ± 1.0	22.8 ± 1.9	25.9 ± 0.3	8.2 ± 0.6	74.6 ± 1.8	67.9 ± 2.5	-	-
DDGCN	39.8 ± 0.6	38.9 ± 0.7	26.3 ± 2.1	26.4 ± 3.3	24.0 ± 0.4	11.7 ± 0.7	73.6 ± 0.9	68.8 ± 1.0	-	-
DEMO-Net	56.9 ± 0.6	56.5 ± 0.7	28.3 ± 0.5	22.5 ± 2.2	28.4 ± 0.8	<u>22.0</u> ± 1.3	<u>90.8</u> ± 0.5	<u>88.9</u> ± 0.6	83.1 ± 0.1	72.0 ± 0.4
role2vec	44.9 ± 1.6	43.8 ± 2.4	26.3 ± 0.8	27.5 ± 1.7	23.1 ± 0.1	18.3 ± 0.6	62.7 ± 0.3	56.3 ± 0.3	77.1 ± 0.2	61.5 ± 0.5
Tail-GCN	**59.2** ± 0.8	**58.5** ± 1.3	**30.2** ± 1.1	**31.1** ± 1.1	**34.9** ± 0.5	**25.2** ± 0.6	**93.6** ± 0.1	**92.7** ± 0.1	**87.0** ± 0.1	**78.2** ± 0.2

Table 3: Evaluation on tail node classification using other GNNs as the base model.

Methods	Email		Squirrel		Actor		CoauthorCS		Amazon	
	Accuracy	Micro-F	Accuracy	Micro-F	Accuracy	Micro-F	Accuracy	Micro-F	Accuracy	Micro-F
GAT	57.9 ± 0.4	57.3 ± 0.2	24.1 ± 2.4	23.1 ± 2.6	29.8 ± 0.6	13.2 ± 2.7	88.6 ± 0.2	86.2 ± 0.2	-	-
Tail-GAT	**59.4** ± 0.9	**58.2** ± 1.2	**28.8** ± 2.1	**30.4** ± 2.6	**34.5** ± 1.3	**24.7** ± 2.0	**92.5** ± 0.1	**90.8** ± 0.1	-	-
GraphSAGE	52.0 ± 1.6	51.3 ± 1.7	27.1 ± 2.7	26.4 ± 4.9	33.1 ± 1.1	23.2 ± 2.4	89.8 ± 2.4	87.7 ± 1.1	79.1 ± 0.4	62.8 ± 0.6
Tail-GraphSAGE	**55.7** ± 0.6	**54.9** ± 0.7	**28.5** ± 1.6	**28.2** ± 2.4	**34.1** ± 1.7	**26.8** ± 1.8	**93.8** ± 0.7	**92.4** ± 1.4	**85.1** ± 0.2	**75.5** ± 0.3

- Tail-GNN
  - Experimental results

Table 6: Evaluation on head node classification.

Methods	Squirrel Acc.	Squirrel Micro-F	Actor Acc.	Actor Micro-F	CoauthorCS Acc.	CoauthorCS Micro-F
GCN	28.8±1.0	24.4±0.2	**29.7**±0.8	13.0±1.3	92.0±0.1	90.6±0.1
Tail-GCN	**30.1**±1.3	**24.7**±1.9	28.7±2.3	**24.3**±3.3	**94.2**±0.2	**93.1**±0.4

Table 4: Evaluation on tail link prediction.

Methods	Squirrel MAP	Squirrel NDCG	Actor MAP	Actor NDCG	CoauthorCS MAP	CoauthorCS NDCG
DeepWalk	29.5±0.9	45.8±0.8	30.0±1.6	46.1±1.3	32.5±1.5	48.2±1.2
GCN	38.5±0.9	53.2±0.4	33.2±1.2	48.9±0.9	77.7±1.2	83.9±0.8
Additive	36.4±0.7	51.5±0.5	32.1±0.9	48.1±0.8	77.8±0.2	83.9±0.2
meta-tail2vec	39.7±0.6	54.2±0.7	33.1±1.0	48.8±0.8	76.0±0.5	82.7±0.2
Tail-GCN	**41.8**±2.4	**55.9**±1.3	**35.8**±2.2	**51.1**±2.0	**81.0**±0.8	**86.1**±0.4

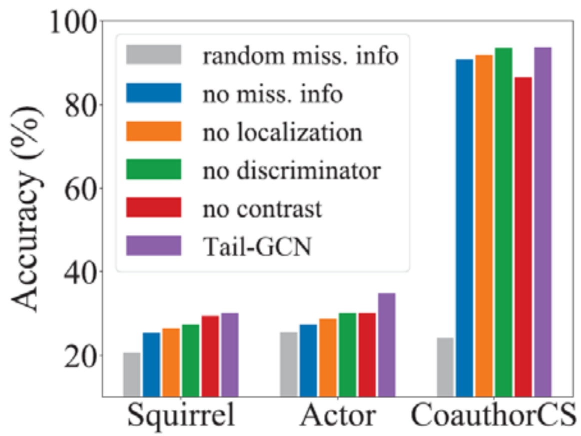


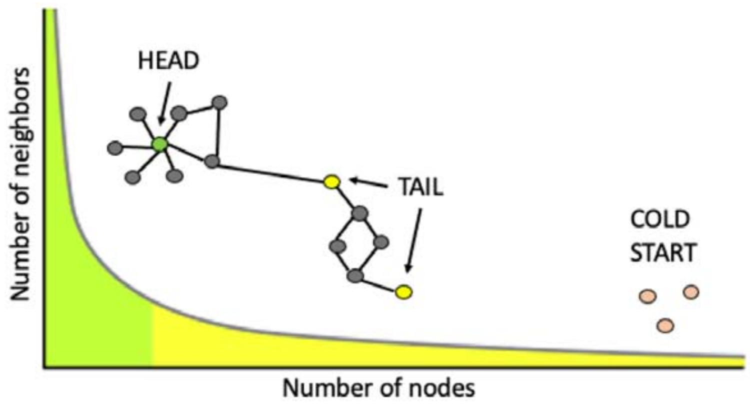
Figure 4: Ablation study.

COLD BREW: DISTILLING GRAPH NODE REPRESENTATIONS WITH INCOMPLETE OR MISSING NEIGHBORHOODS

Wenqing Zheng[1,2] **Edward W Huang**[1], **Nikhil Rao**[1], **Sumeet Katariya**[1],
Zhangyang Wang[1,2], **Karthik Subbian**[1]
[1]Amazon [2]The University of Texas at Austin
{wenqzhen,ewhuang,nikhilsr,katsumee,wzhangwa,ksubbian}@amazon.com

ICLR 2022

- Cold Brew
  - Intuition
    - Tail nodes have unreliable or absent neighborhood
  - Task definition: Strict Cold Start (SCS)
    - Most public datasets have 3% to 6% isolated nodes
    - Key to truly inductive GNN model
  - Motivation & Contribution
    - How to generalize better to tail and SCS nodes?
      - Cold Brew framework is proposed to distill the knowledge of a node-wise **Structural Embedding (SE) enhanced** GNN teacher model into a **MLP** student model
    - How to select cold-start-friendly model architecture?
      - A new metric **Feature-Contribution Ratio (FCR)** is defined to quantify the contribution of node features w.r.t. the adjacency structure in the dataset for a specific downstream task



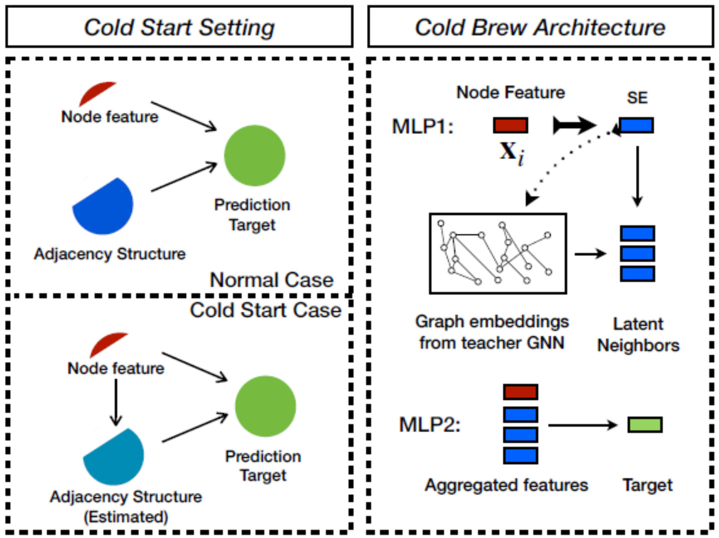
- Cold Brew
  - Strict Cold Start generalization
    - Prerequisites
      - Given node number N and node embedding dimension d
      - Due to $N \gg d$, the node embeddings form an **overcomplete set** in the d-dimensional space
      - It's possible that any node representation can be cast as a linear combination of $K \ll N$ existing node representations
    - Overall strategy
      - Naïve node-wise MLP is inferior due to its ignorance to graph information
      - Teacher-student knowledge distillation procedure
        - Teacher GNN model embed nodes onto a d-dimensional manifold with structural information
        - Student model learn a mapping from node features and *virtual neighborhood* to the manifold without access to structural information
        - **Note**: Unlike traditional knowledge distillation task, Cold Brew aims to train a student model better than the teacher under tail and SCS task setting, where GNN-based teacher model usually fails

- Cold Brew
  - Strict Cold Start generalization
    - Teacher model: Structural Embedding Enhanced GNN (GCN for example)
      - SE-GNN enables each node to encode self and neighbors' label information into its own embedding
      - Structural embedding can avoid over-smoothing

$$\mathbf{X}^{(l+1)} = \sigma\left(\tilde{A}\left(\mathbf{X}^{(l)}\mathbf{W}^{(l)} + \mathbf{E}^{(l)}\right)\right), \mathbf{X}^{(l)} \in \mathbb{R}^{N\times d_1}, \mathbf{W}^{(l)} \in \mathbb{R}^{d_1\times d_2}, \mathbf{E}^{(l)} \in \mathbb{R}^{N\times d_2}$$

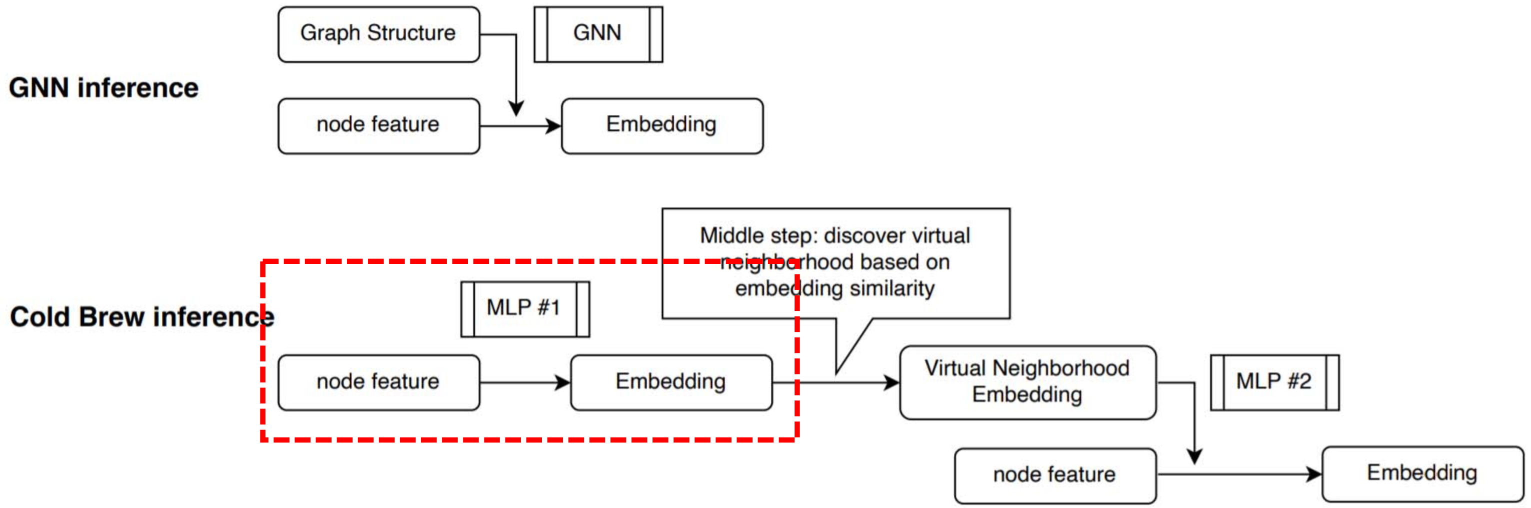
$$loss = CE(\mathbf{X}^{(L)}_{train}, \mathbf{Y}_{train}) + \eta\|\mathbf{E}\|_2^2$$

    - Student MLP model
      - The first MLP aims to mimic the node embedding generated by GNN teacher
      - The second MLP aims to transform both target node feature and virtual neighborhood into the embedding of interest



(a) The teacher-student knowledge distillation of the Cold Brew framework under the cold start setting.

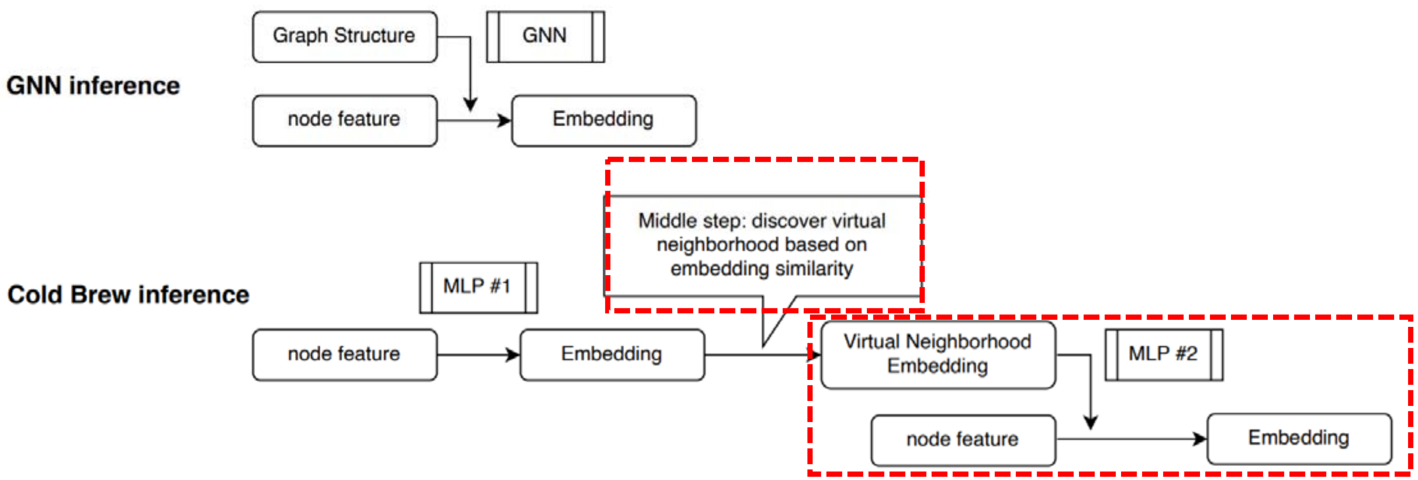
- Cold Brew
  - Strict Cold Start generalization
    - Student MLP model
      - Given embedding of interest $\bar{E}$, the first MLP will learn a mapping from input node feature to $\bar{E}$, s.t. $\xi_1: x_i^{(0)} \rightarrow e_i, e_i = \bar{E}[i,:]$
        - $\bar{E}$ can be the final output of teacher GNN, s.t. $\bar{E} \in R^{N \times d_{out}}$
        - $\bar{E}$ can also be the concatenation of all intermediate results of teacher GNN, s.t. $\bar{E} \in R^{N \times (d_{out} + d_{hidden} * (L-1))}$



- Cold Brew
  - Strict Cold Start generalization
    - Student MLP model
      - Virtual neighborhood discovery based on self-attention
        - $\Theta_K(\cdot)$ is the **top-K hard thresholding operator**
        - Every node embedding, whether or not seen previously, can be decomposed as a linear combination of the given overcomplete basis

$$\tilde{e}_i = softmax(\Theta_K(e_i \bar{\mathbf{E}}^\top))\bar{\mathbf{E}}$$

      - The second MLP will learn mapping ξ_2: $[x_i, e_i] \rightarrow y_i$
        - y_i is the ground truth or target of interest for node i.

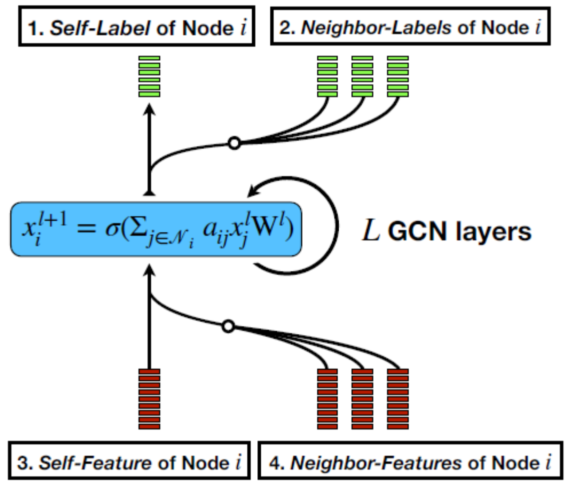


- Cold Brew
  - Model interpretation from label smoothing perspective

We cite Theorem 1 in [36]: *Suppose that the latent ground-truth mapping from node features to node labels is differentiable and L-Lipschitz. If the edge weights* a_{ij} *approximately smooth* $\mathbf{x}_i$ *over its immediate neighbors with error* ϵ_i*, i.e.,* $\mathbf{x}_i = \frac{1}{d_{ii}}\Sigma_{j\in\mathcal{N}} a_{ij}\mathbf{x}_j + \epsilon_i$*, then the* a_{ij} *also approximately smooth* y_i *to bound within error* $|y_i - \frac{1}{d_{ii}}\Sigma_{j\in\mathcal{N}_i} a_{ij}y_j| \leq L||\epsilon||_2 + o(\max_{j\in\mathcal{N}_i}(||\mathbf{x}_j - \mathbf{x}_i||_2))$*, where* $o(\cdot)$ *denotes a higher order infinitesimal.*

- Errors of label prediction are proportional to the difference of features after neighborhood aggregation
- For teacher model
  - Due to the additional structural embedding, the approximately smoothed x_i can be formalized as $\frac{1}{d_{ii}}\sum_{j\in N} a_{ij}x_j + \bar{E}[:, i] + \epsilon_i$
  - The error ϵ_i can be lowered if $\bar{E}$ is properly learned, allowing higher flexibility and expressiveness
- For student model
  - Without neighborhood aggregation, error ϵ_i will be non-negligible, leading to higher loss in the label prediction
  - Cold Brew mitigate the issue through virtual neighborhood aggregation

- Cold Brew
  - Feature-Contribution Ratio
    - Basic intuition: SCS generalization difficulty is proportional to the contribution ratio of *node features*
    - Two submodules
      - For node features only: an MLP is built to map the self-features to self-labels
      - For adjacency structure only: a Label Propagation (LP) method is adopted to learn representations from self- and neighbor-labels
    - Formalization
      - Given graph dataset G, the contribution of a submodule is defined to be residual performance of the submodule compared to a full-fledged GNN
      - Denote z_{MLP}, z_{LP}, z_{GNN} as the performance of MLP submodule, LP submodule and full GNN on the test set



(b) Four GNN Atomic Components in deciding GNN's output, which are used for FCR analysis.

$$\delta_{MLP} = z_{GNN} - z_{MLP}, \quad \delta_{LP} = z_{GNN} - z_{LP}$$

$$FCR(\mathcal{G}) = \begin{cases} \frac{\delta_{LP}}{\delta_{MLP}+\delta_{LP}} \times 100\% & z_{MLP} \leq z_{GNN} \\ 1 + \frac{|\delta_{MLP}|}{|\delta_{MLP}|+\delta_{LP}} \times 100\% & z_{MLP} > z_{GNN} \end{cases}$$

$$FCR(G) = \begin{cases} [0\%, 50\%) \rightarrow z_{GNN} > z_{LP} > z_{MLP} \\ [50\%, 100\%) \rightarrow z_{GNN} > z_{MLP} > z_{LP} \\ [100\%, +\infty) \rightarrow z_{MLP} > z_{GNN} > z_{LP} \end{cases}$$

- Cold Brew
  - Datasets and splits
    - Head data: top 10% highest degree nodes and induced subgraph
    - Isolation data: nodes correspond to the bottom 10% of the degree distribution, with all the edges artificially removed
    - Tail data: top 10% lowest degree nodes in the remaining graph
    - Overall data: without distinguishing head/tail/isolation

Stats.	Cora	Citeseer	Pubmed	Arxiv	Chameleon	E-comm1	E-comm2	E-comm3	E-comm4
Num. of Nodes	2708	3327	19717	169343	2277	4918	29352	319482	793194
Num. of Edges	13264	12431	108365	2315598	65019	104753	1415646	8689910	22368070
Max Degree	169	100	172	13161	733	277	1721	4925	12452
Mean Degree	4.90	3.74	5.50	13.67	28.55	21.30	48.23	27.20	28.19
Median Degree	4	3	3	6	13	10	21	15	14
Isolated Nodes %	3%	3%	3%	3%	3%	6%	5%	5%	6%

Table 1: The statistics of datasets selected for evaluations.

- Cold Brew
  - Experiment results (FCR evaluation)

	Cora	Citeseer	Pubmed	Arxiv	Cham.	Squ.	Actor	Cornell	Texas	Wisconsin
GNN	86.96	72.44	75.96	71.54	68.51	31.95	59.79	65.1	61.08	81.62
MLP	69.02	56.59	73.51	54.89	58.65	38.51	37.93	86.26	83.33	85.42
Label Propagation	78.18	45.00	67.8	68.26	41.01	22.85	29.69	32.06	52.08	40.62
FCR %	32.86 %	63.39 %	76.91%	16.45%	73.61%	141.91%	57.93%	139.04%	171.2 %	108.48 %
$\beta(\mathcal{G})$ %	83%	71%	79%	68%	25%	22%	24%	11%	6%	16%
$head - tail(GNN)$	4.44	23.98	11.71	5.9	0.24	-6.51	2.22	-4.37	-11.26	-33.92
$head - isolation(GNN)$	31.01	33.09	15.21	28.81	1.55	-4.85	22.61	-18.68	-24.62	-29.23

$$\beta(\mathcal{G}) = \frac{1}{|\mathcal{V}|} \sum_{v \in \mathcal{V}} \frac{\text{the number of } v\text{'s direct neighbors that have the same labels as } v}{\text{the number of } v\text{'s directly connected neighbors}} \times 100\%$$

- Cold Brew
  - Experiment results (Node classification)

Splits	Metrics/Models		Open-Source Datasets					Proprietary Datasets			
			Cora	Citeseer	Pubmed	Arxiv	Chameleon	E-comm1	E-comm2	E-comm3	E-comm4
Isolation	GNNs	GCN 2 layers	58.02	47.09	71.50	44.51	57.28	–	–	–	–
		GraphSAGE	66.02	51.46	69.87	47.32	59.83	+3.89	+4.81	+5.24	+0.52
	MLPs	Simple MLP	68.40	**53.26**	65.84	51.03	60.76	+5.89	+9.85	+5.83	+6.42
		GraphMLP	65.00	52.82	71.22	51.10	**63.54**	+6.27	+9.46	**+5.99**	+7.37
	Cold Brew	GCN + SE 2 layers	58.37	47.78	**73.85**	45.20	60.13	+0.27	+0.76	-0.50	+1.22
		Student MLP	**69.62**	53.17	72.33	**52.36**	62.28	**+7.56**	**+11.09**	+5.64	**+9.05**
Tail	GNNs	GCN 2 layers	84.54	**56.51**	74.95	67.74	58.33	–	–	–	–
		GraphSAGE	82.82	52.77	73.07	63.23	61.26	-3.82	-3.07	-2.87	-6.42
	MLPs	Simple MLP	70.76	54.85	67.21	52.14	50.12	-0.37	+1.74	-0.13	-0.45
		GraphMLP	70.09	55.56	71.45	52.40	52.84	-0.33	+1.64	**+1.27**	+0.80
	Cold Brew	GCN + SE 2 layers	**84.66**	56.32	**75.33**	**68.11**	**60.80**	**+0.85**	+0.44	-0.60	+1.10
		Student MLP	71.80	54.88	72.54	53.24	51.36	+0.32	**+3.09**	-0.18	**+2.09**
Head	GNNs	GCN 2 layers	88.68	80.37	85.79	73.35	67.49	–	–	–	–
		GraphSAGE	87.75	74.81	86.94	70.85	62.08	-4.26	-4.17	-3.50	-7.46
	MLPs	Simple MLP	74.33	72.00	89.00	56.34	60.82	-16.74	-18.10	-16.73	-16.51
		GraphMLP	72.45	69.83	89.00	56.65	62.44	-15.96	-18.08	-15.33	-15.41
	Cold Brew	GCN + SE 2 layers	**89.39**	**80.76**	87.83	**74.01**	70.56	**+1.11**	**+0.47**	-0.39	**+1.28**
		Student MLP	74.53	72.33	**90.33**	57.41	61.28	-15.28	-17.42	-17.02	-15.41
Overall	GNNs	GCN 2 layers	84.89	70.38	78.18	71.50	59.30	–	–	–	–
		GraphSAGE	80.90	66.21	76.73	68.33	70.02	-3.09	-3.86	-2.58	-5.48
	MLPs	Simple MLP	69.02	56.59	73.51	54.89	58.65	-12.69	-12.86	-12.68	-13.16
		GraphMLP	71.87	68.22	82.03	53.81	57.67	-12.26	-12.01	-10.80	-11.41
	Cold Brew	GCN + SE 2 layers	**86.96**	**72.44**	79.03	**71.92**	**68.51**	**+0.65**	-0.24	-0.77	**+1.43**
		Student MLP	72.36	67.54	**82.00**	54.94	59.07	-11.25	-11.51	-11.55	-11.21

- Cold Brew
  - Experiment results

Splits	Models	Datasets			
		Cora	Citeseer	Pubmed	E-comm1
Isolation	GCN 2 layers	34.10	50.41	51.52	–
	TailGCN	36.13	51.48	51.19	+2.18
	Meta-Tail2Vec	36.92	50.90	51.62	+2.34
	Cold Brew's MLP	**44.59**	**55.14**	**54.82**	**+5.39**

Splits	Models	Datasets			
		Cora	Citeseer	Pubmed	E-comm1
Isolation	GCN 2 layers	58.02	47.09	71.50	–
	TailGCN	62.04	51.87	72.10	+3.14
	Meta-Tail2Vec	61.16	50.46	71.80	+2.80
	Cold Brew's MLP	**69.62**	**53.17**	**72.33**	**+7.56**

Splits	Metrics/Models	Open-Source Datasets					Proprietary Datasets			
		Cora	Citeseer	Pubmed	Arxiv	Chameleon	E-comm1	E-comm2	E-comm3	E-comm4
Overall	GCN 64 layers	40.04	23.66	75.65	65.53	58.14	-5.49	-6.59	-6.13	-3.57
	GCN + SE 64 layers	**74.23**	**46.80**	**78.12**	**69.28**	**59.88**	**-1.71**	**-2.92**	**-3.29**	**-0.06**
Head	GCN 64 layers	46.46	49.84	85.89	67.53	67.16	-5.60	-6.24	-6.05	-3.16
	GCN + SE 64 layers	**87.38**	**71.18**	**86.81**	**71.35**	**69.63**	**-1.78**	**-2.17**	**-2.79**	**-0.35**
Tail	GCN 64 layers	45.14	24.42	71.89	63.91	56.48	-3.85	-3.62	-3.84	**-1.14**
	GCN + SE 64 layers	**79.56**	**36.52**	**74.88**	**65.19**	**61.73**	**-2.42**	**-2.52**	**-3.68**	-1.23
Isolation	GCN 64 layers	39.97	22.12	68.57	40.03	57.60	-4.66	-4.63	-4.93	**-1.89**
	GCN + SE 64 layers	**40.33**	**24.53**	**71.22**	**41.18**	**60.13**	**-3.08**	**-3.02**	**-4.00**	-2.32

- Other related work
  - Sampling bias in skip-gram based node embedding
    - S. Kojaku, et.al. *Residual2Vec: Debiasing graph embedding with random graphs*. NIPS 2021
      - Inspired by built-in debiasing mechanism in skip-gram negative sampling word2vec, Residual2Vec proposed to **debias from the noise distribution** based on dcSBM random graph to precisely reflect structural information
  - Introduce additional heterogeneous information
    - S. Wang, et.al. *Privileged Graph Distillation for Cold-Start Recommendation*. SIGIR 2021
      - Distill knowledge of user-item-attribute heterogeneous graph into **user/item-attribute bipartite graph** to generate effective cold-start representation only based on attribute
    - Z. Liu, et.al. *Learning Representations of Inactive Users: A Cross Domain Approach with Graph Neural Networks*. CIKM 2021
      - With auxiliary **social network signal**, transfer learning is conducted to make cold-start users able to benefit from knowledge of head users
    - J. Zheng, et.al. *Multi-view Denoising Graph Auto-Encoders on Heterogeneous Information Networks for Cold-start Recommendation*. KDD 2021
      - A **meta-path based DGAE** is proposed to help tail users make better use of limited semantic information from scarce meta-paths

- Conclusion
  - Background of low degree bias: observation and intuitive causes
  - Debias Strategy
    - Debias with heterogeneous information: (mainstream strategy)
      - HIN-based: MvDGAE
      - Content-based recommendation: Privileged Graph Distillation
      - Social network/KG enhanced: CD-GNN
    - Debias from sampling-level: Residual2vec
    - **Debias from model-level**: (On the rise, under-explored)
      - Self-supervised learning based: PT-GNN
      - Meta-learning based: Meta-Tail2Vec
      - Missing information imputation based: Tail-GNN
      - Knowledge distillation based: Cold Brew

Thanks!

2022.08