

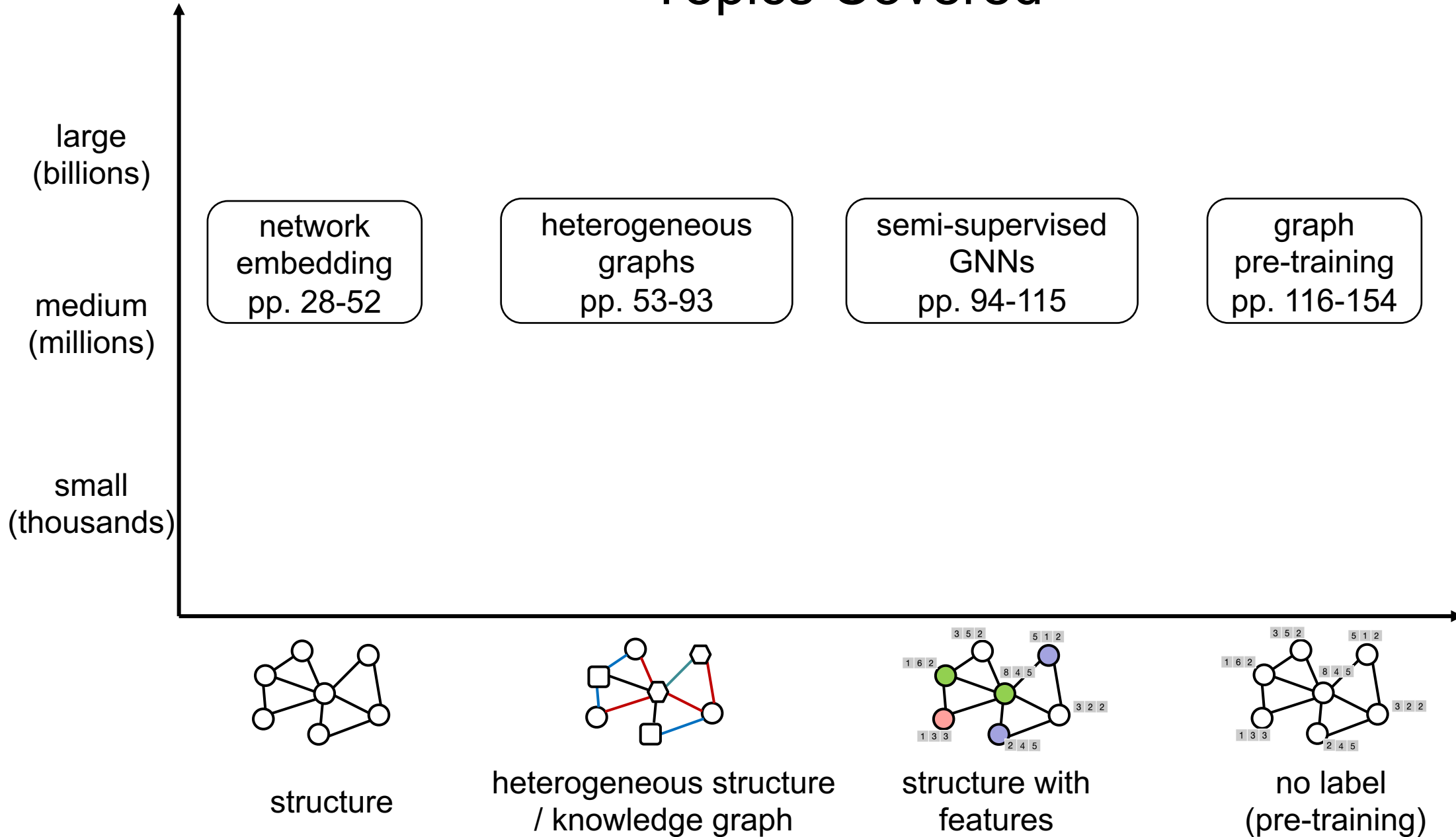
Graph Representation Learning and Pre-Training

Yuxiao Dong

Knowledge Engineering Group (KEG)
Department of Computer Science and Technology
Tsinghua University

<https://keg.cs.tsinghua.edu.cn/yuxiao/>
yuxiaod@tsinghua.edu.cn

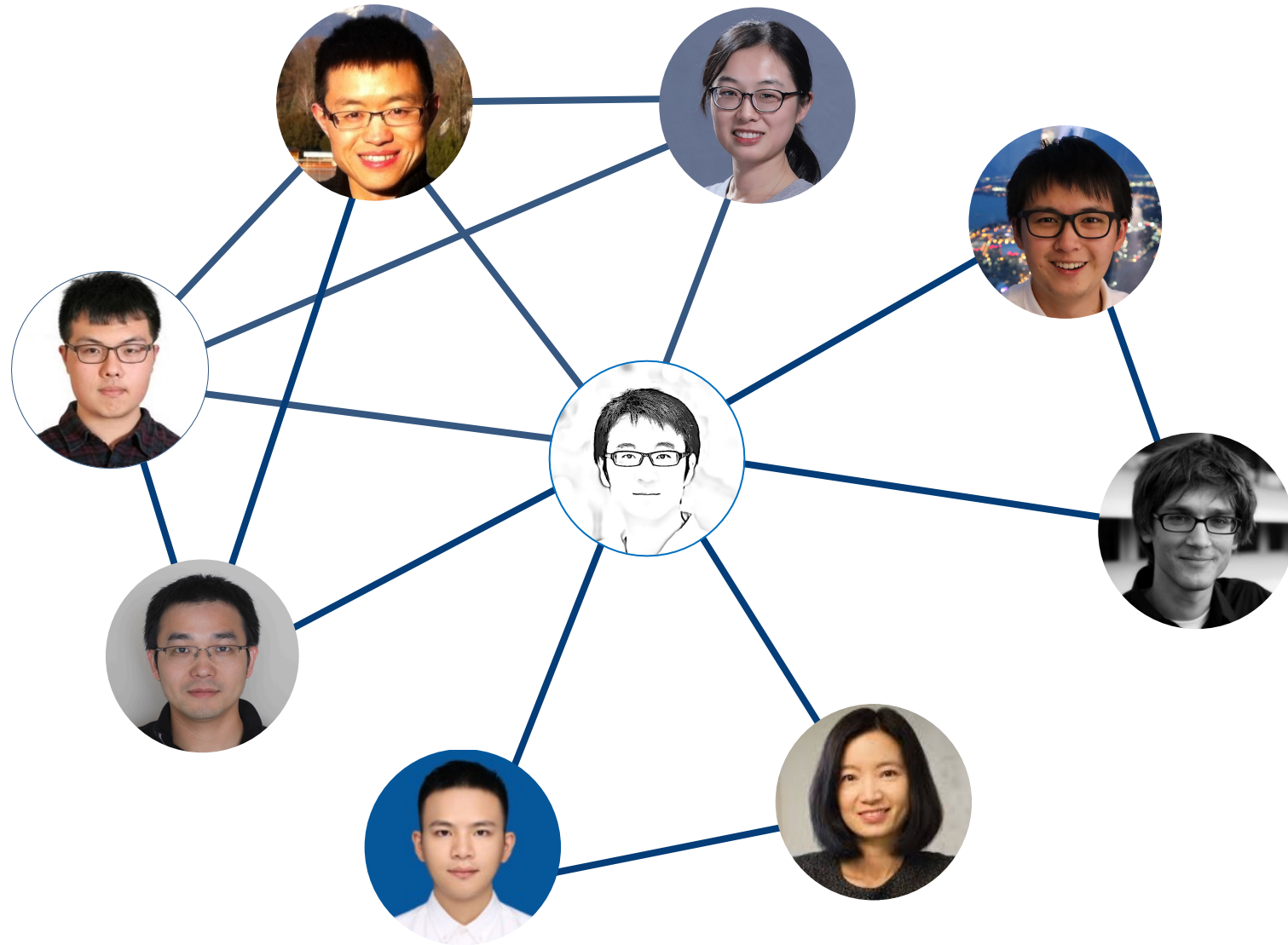
Topics Covered



Joint Work with

*Xiao Liu, Jiezhong Qiu, Ziniu Hu, Wenzheng Feng, Zhenyu Hou
Yukuo Cen, Weihua Hu, Jie Zhang, Chenhui Zhang, Yuyang Xie
Hao Ma, Kuansan Wang, Yizhou Sun, Jure Leskovec, Jie Tang*

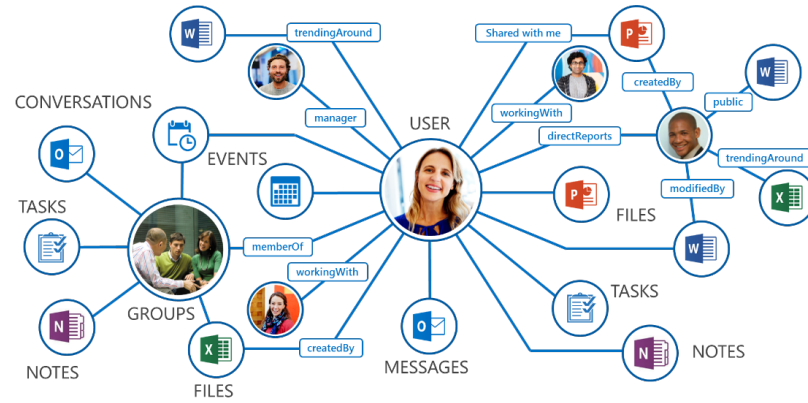
Why Graphs?



Graphs in Society



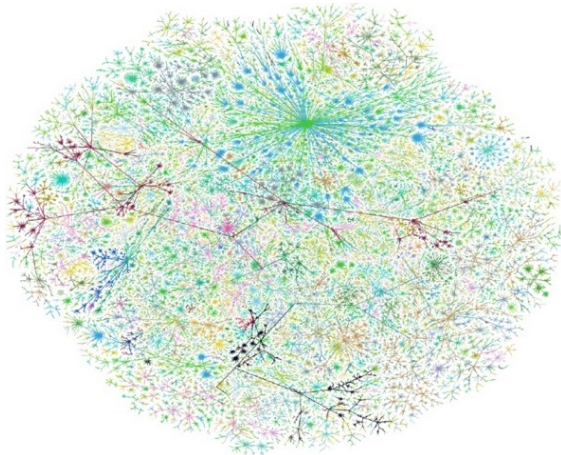
Academic Graph



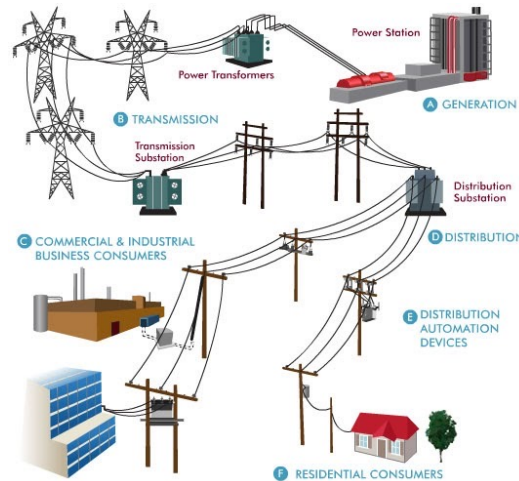
Social & Office Graph



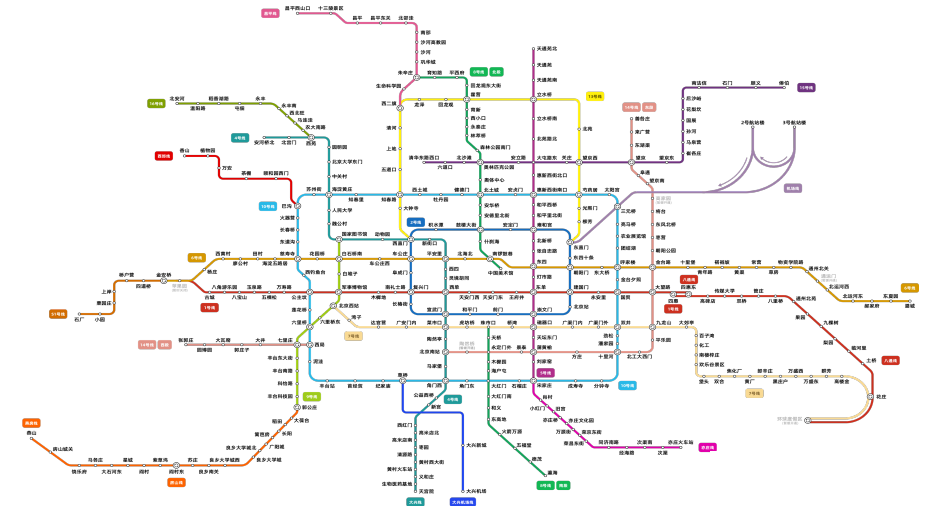
Knowledge Graph



Internet

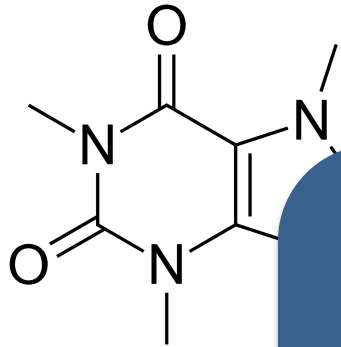


Electrical Grid Network

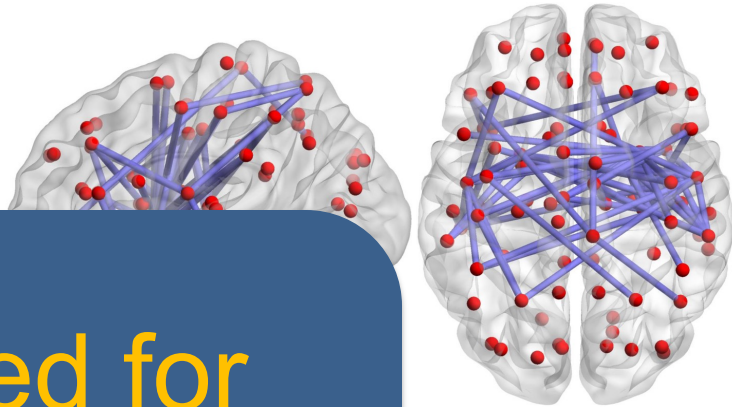
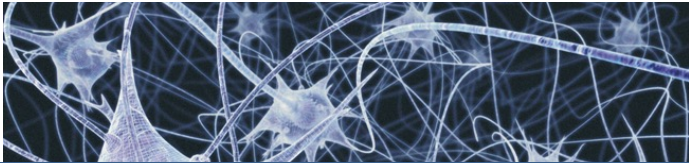


Transportation

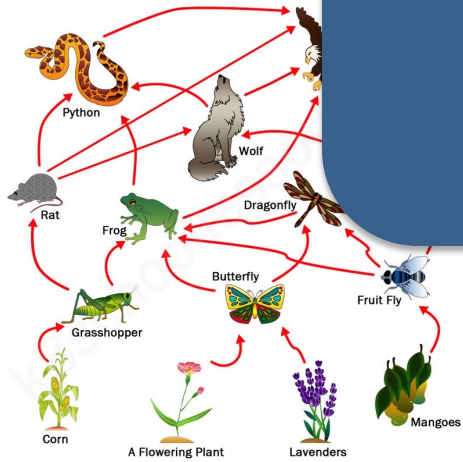
Graphs in Nature



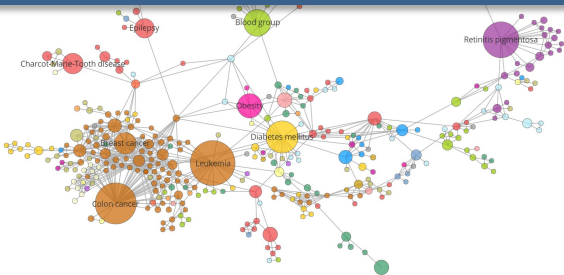
Molecules



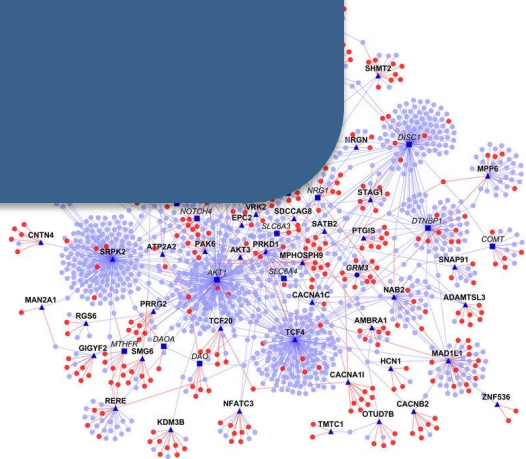
Graphs are widely used for abstracting complex systems of interacting objects!



Food Web



Human Disease Networks

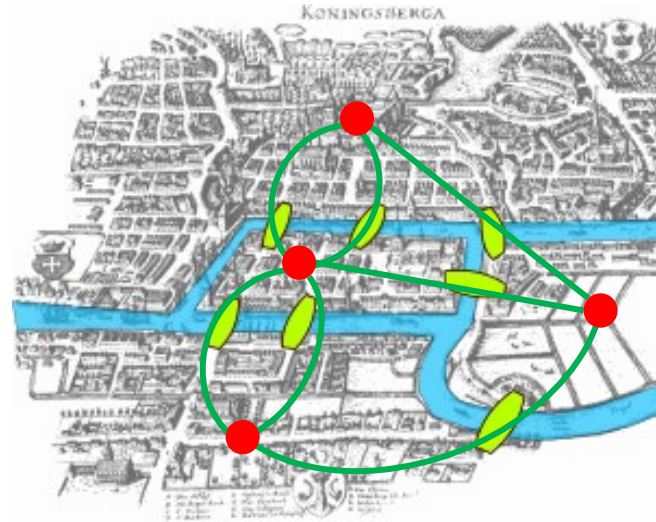


Protein-Protein Interactions

When Did the Mind of Graphs Start?



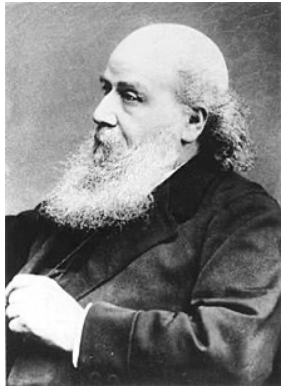
Leonhard Euler
(1707--1783)



Seven Bridges of Königsberg (1736)

Can we design a routine to walk through
each bridge once and only once?

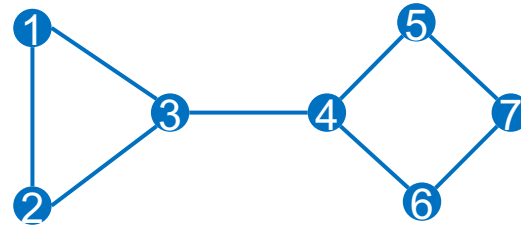
When Did the Term “graph” Start?



James J Sylvester
(1814--1897)

The term “graph” (1878)

The term “matrix” (1850)



This is a graph!

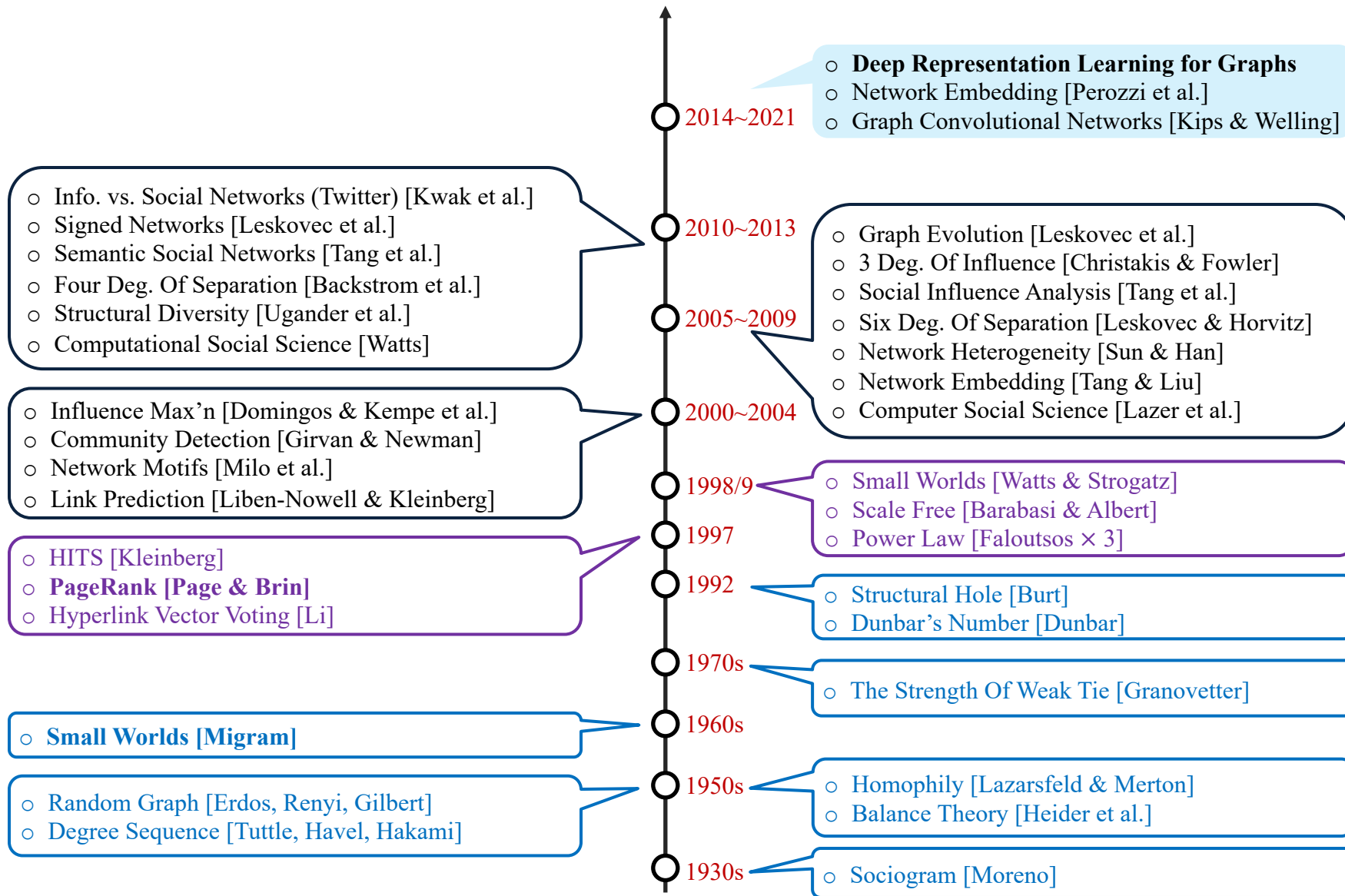
The graph G can be
represented as a matrix!

$G = (V, E)$, where V is the node set
and E denotes the edge set.

- $V: v_1, v_2, v_3, v_4, v_5, v_6, v_7$
- $E: e_{12}, e_{13}, e_{23}, e_{34}, e_{45}, e_{46}, e_{57}, e_{67}$
- $E \subseteq V \times V$

- #nodes: $n = |V| = 7$
 - The **order** of the graph G
- #edges: $m = |E| = 8$
 - The **size** of the graph G

Graph & Network Research



The past 10 years:
Deep/Representation Learning on Graphs

2000-2013
More CS & ML on Graphs

1996-2000
CS & Physics

The 20th Century:
Sociology & Anthropology

50 MOST APPEARED KEYWORDS

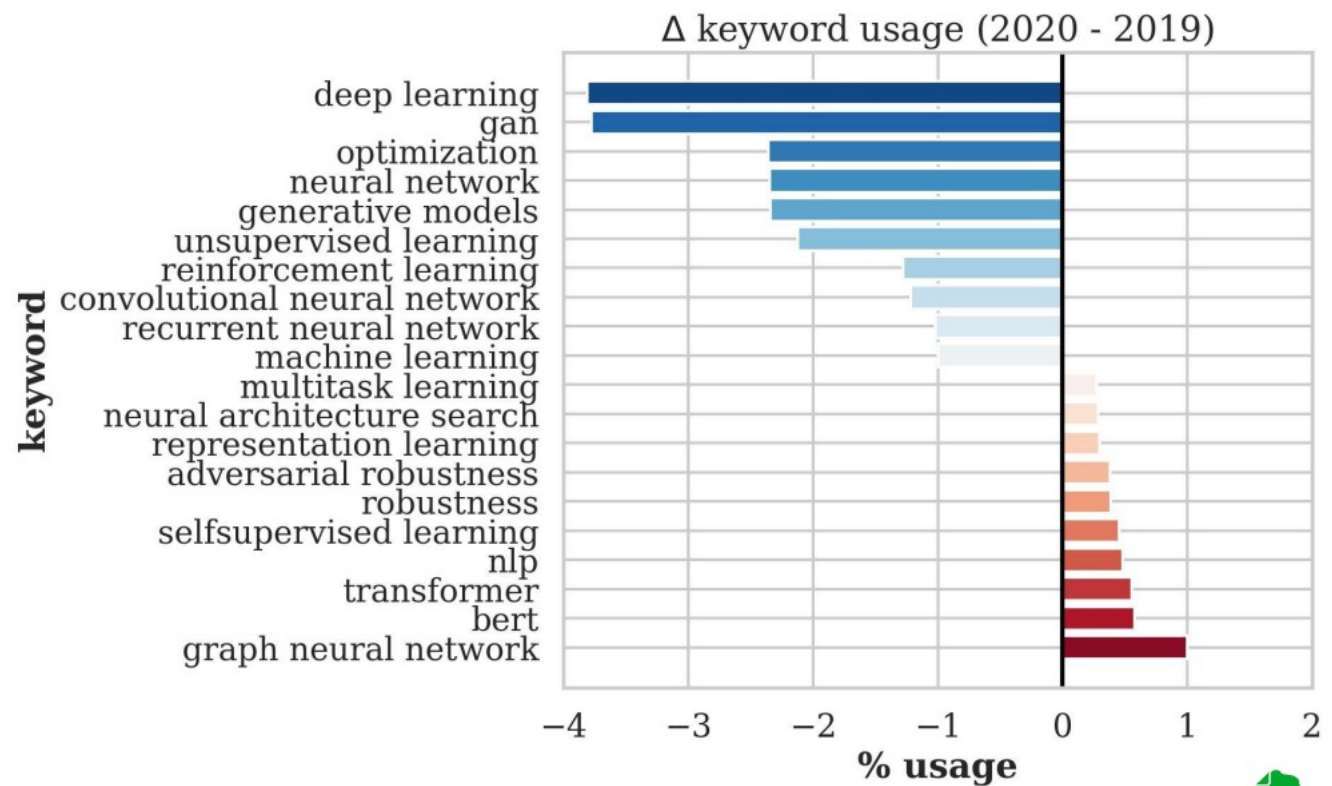
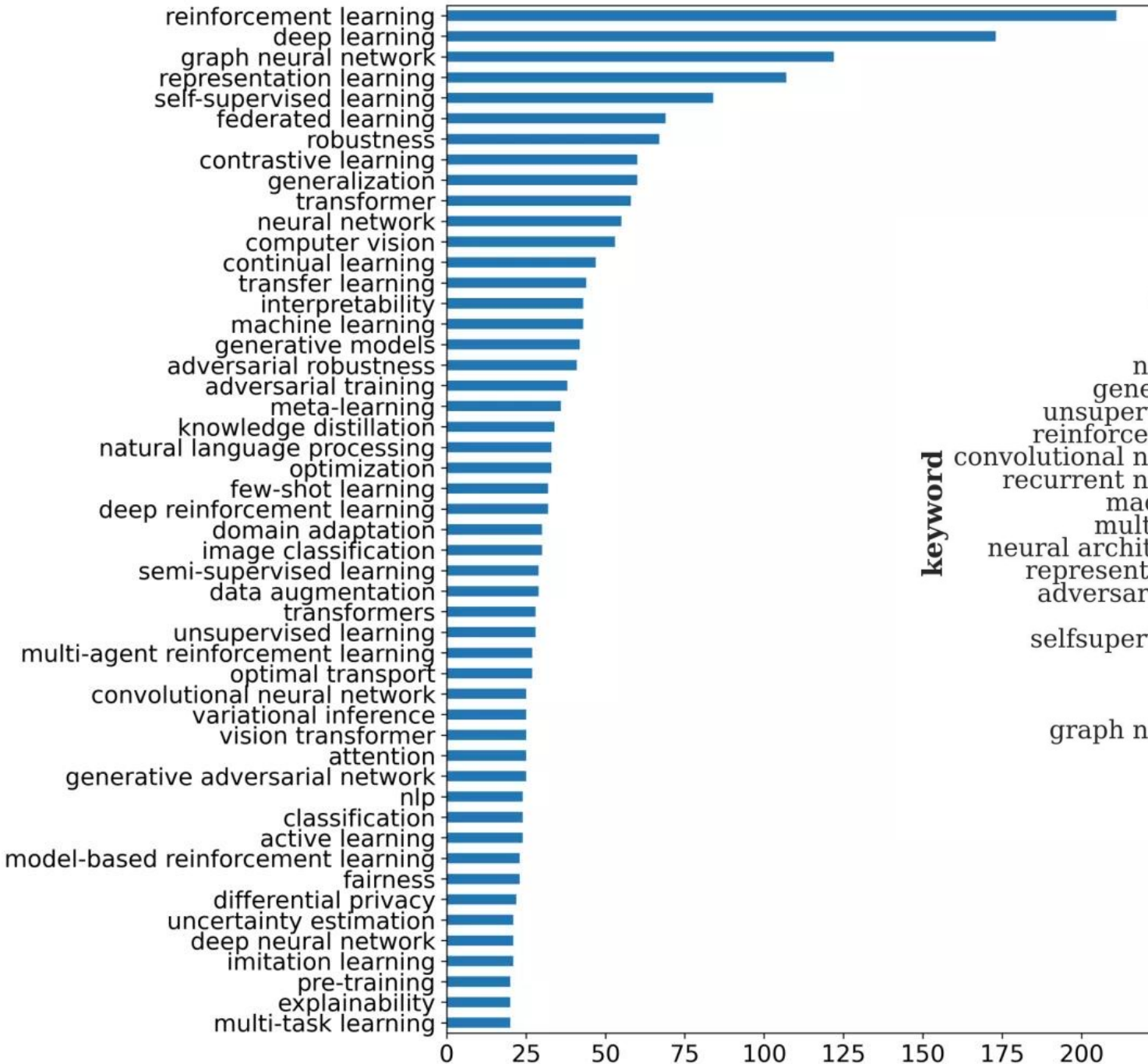
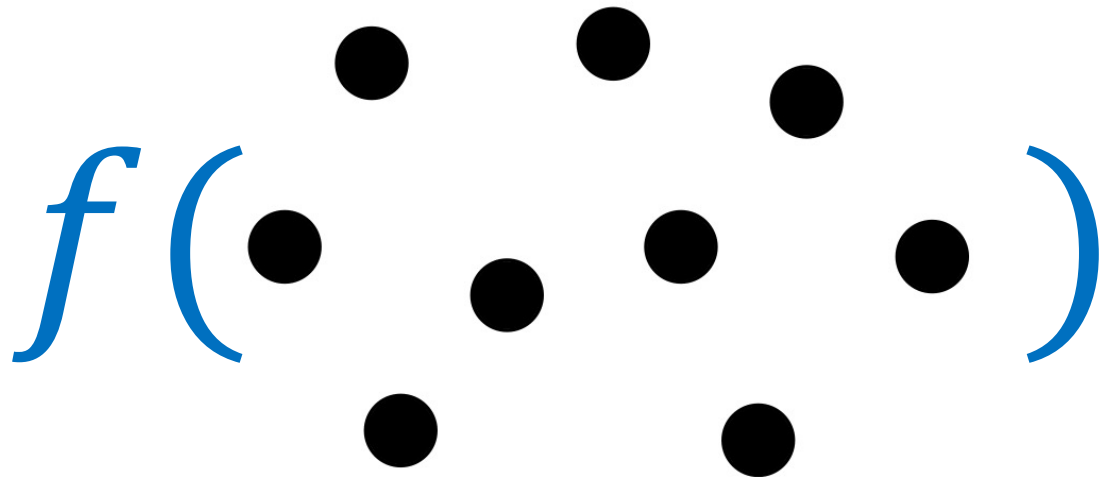
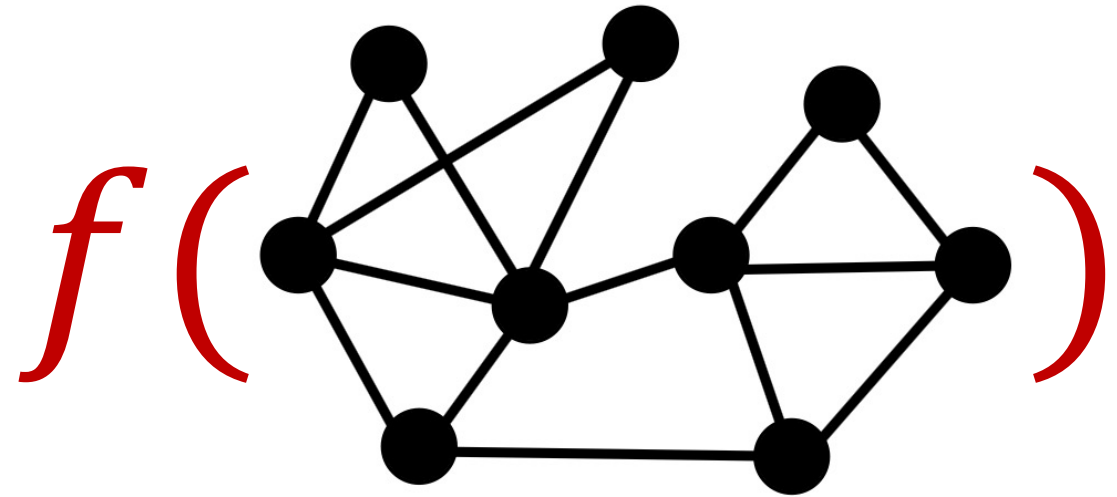


Figure Credit: Velickovic, ICLR Conferences

Graph Machine Learning

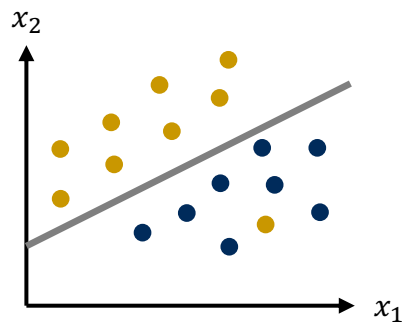
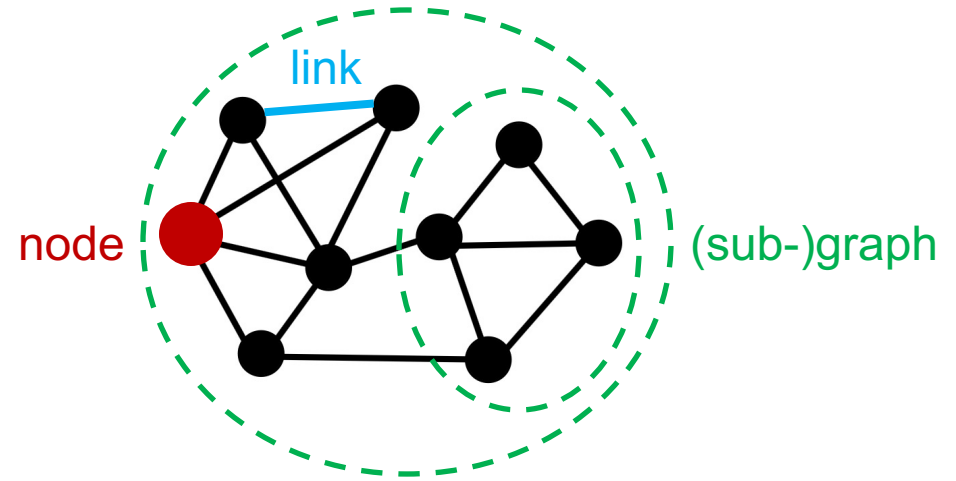
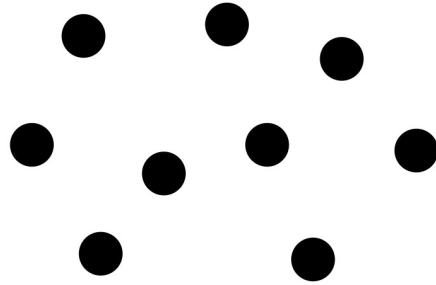


Machine Learning on
Data

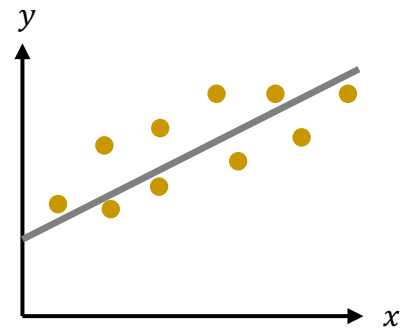


Machine Learning on
Graphs

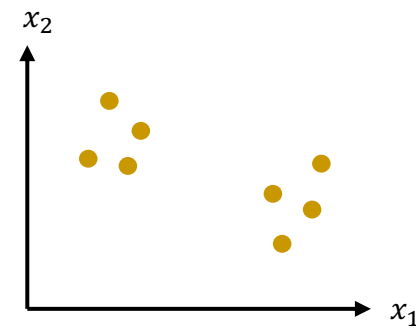
Graph Machine Learning



classification

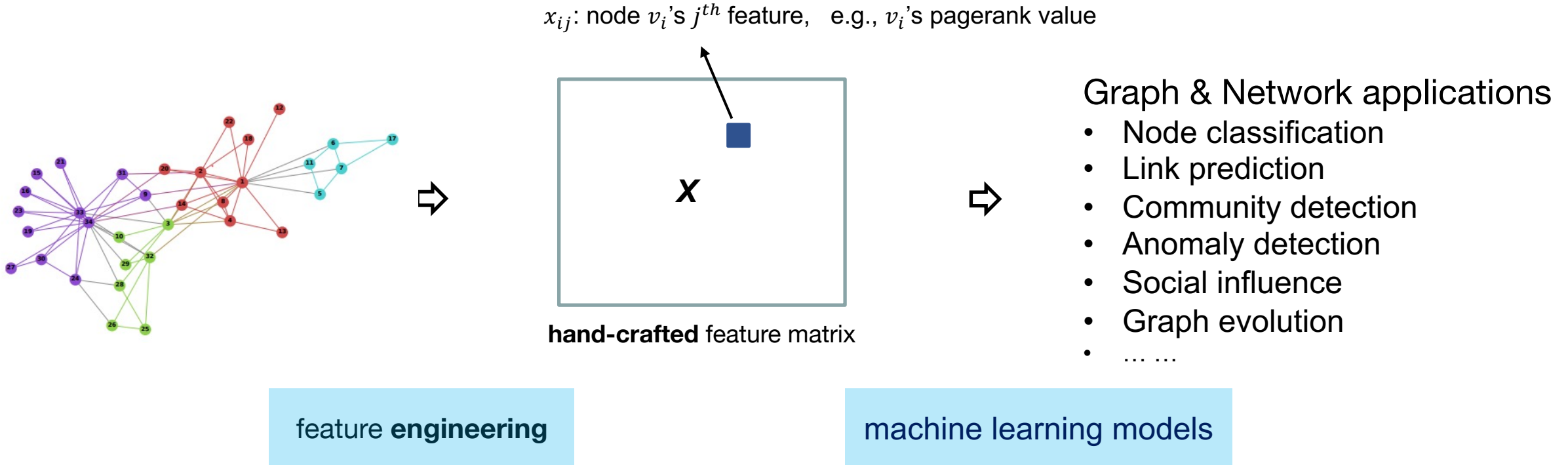


regression



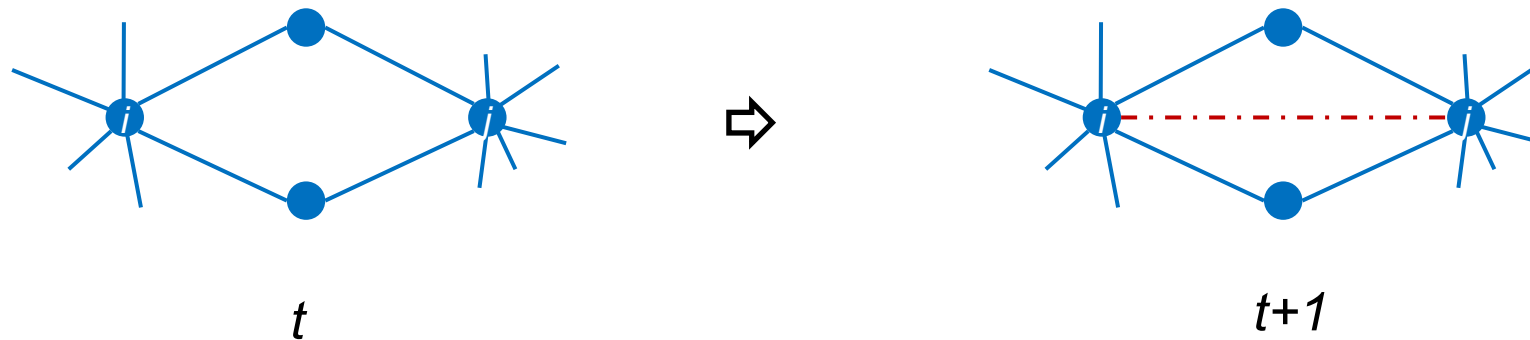
clustering

Before Deep Learning ...



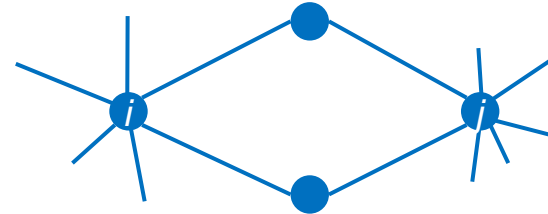
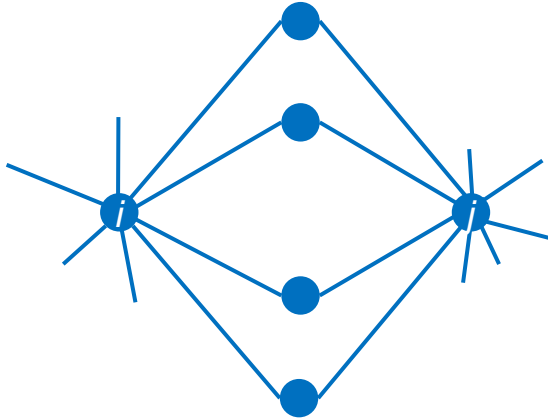
Before Deep Learning ...

- Given two nodes v_i and v_j that are not connected right now, we aim to infer whether a link will form between them.
 - Friend recommendation, e.g., “People you may know” on LinkedIn or Facebook, “Who to follow” on Twitter
 - Item recommendation, e.g., movies to watch in Netflix, books to buy in Amazon



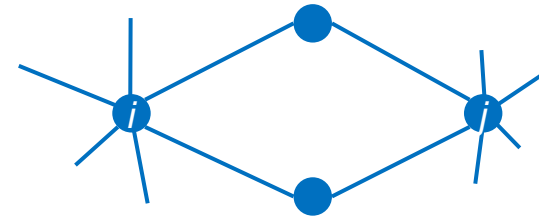
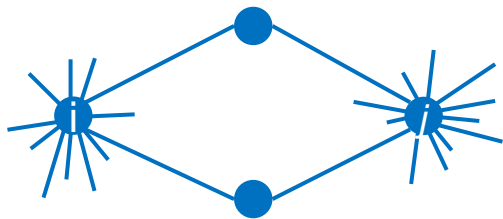
Before Deep Learning ...

- The number of common neighbors between two nodes
- $S_{ij} = |N(v_i) \cap N(v_j)|$, where $N(v_i)$ represents the neighbors of v_i .



Before Deep Learning ...

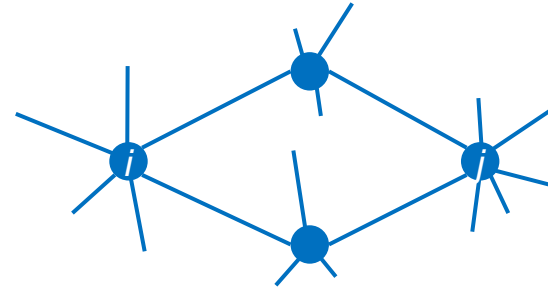
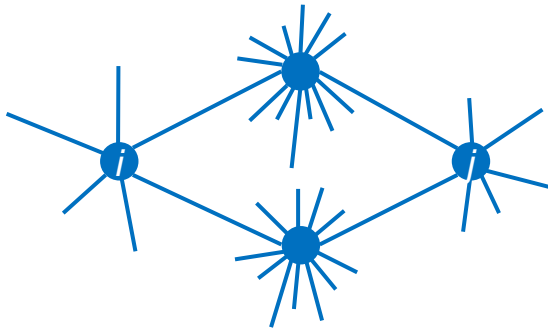
- The intersection of two's neighbors over the union of their neighbors
- $S_{ij} = \frac{|N(v_i) \cap N(v_j)|}{|N(v_i) \cup N(v_j)|}$, where $N(v_i)$ represents the neighbors of v_i .



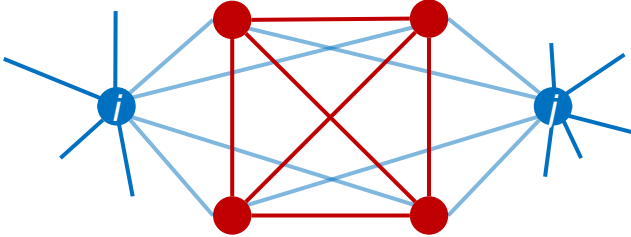
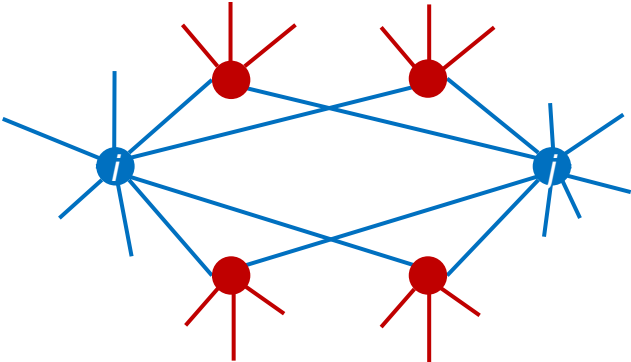
Before Deep Learning ...

- Adamic Adar

- $$S_{ij} = \sum_{v_p \in N(v_i) \cap N(v_j)} \frac{1}{\log|N(v_i)|}$$

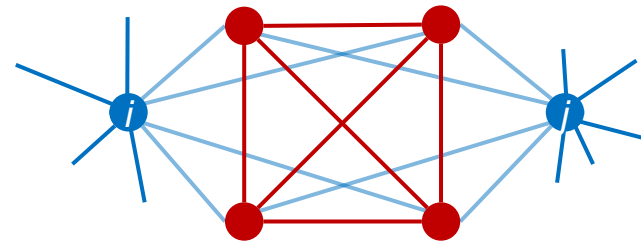
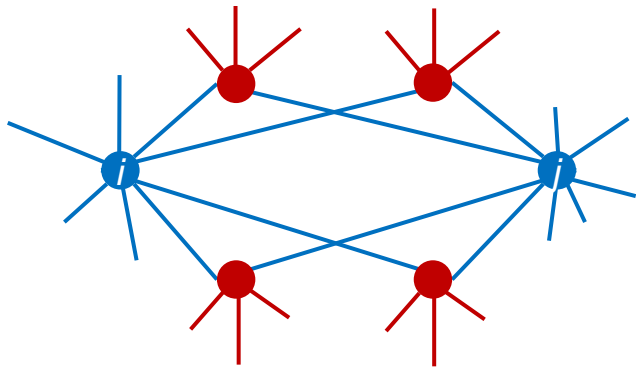


Before Deep Learning ...



Before Deep Learning ...

Structural Diversity



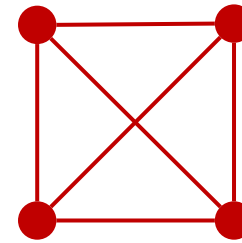
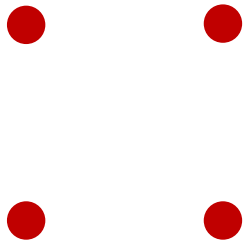
$$P_1 (\text{blue } i \text{ --- } \text{blue } j \mid \text{d6})$$

?

$$P_2 (\text{blue } i \text{ --- } \text{blue } j \mid \text{d4})$$

Before Deep Learning ...

Structural Diversity

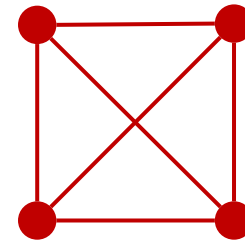
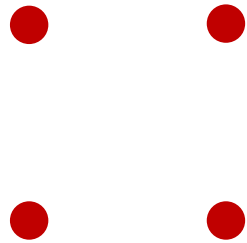


more diverse

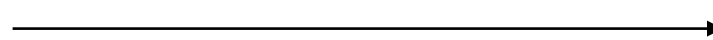
less diverse

Before Deep Learning ...

Structural Diversity

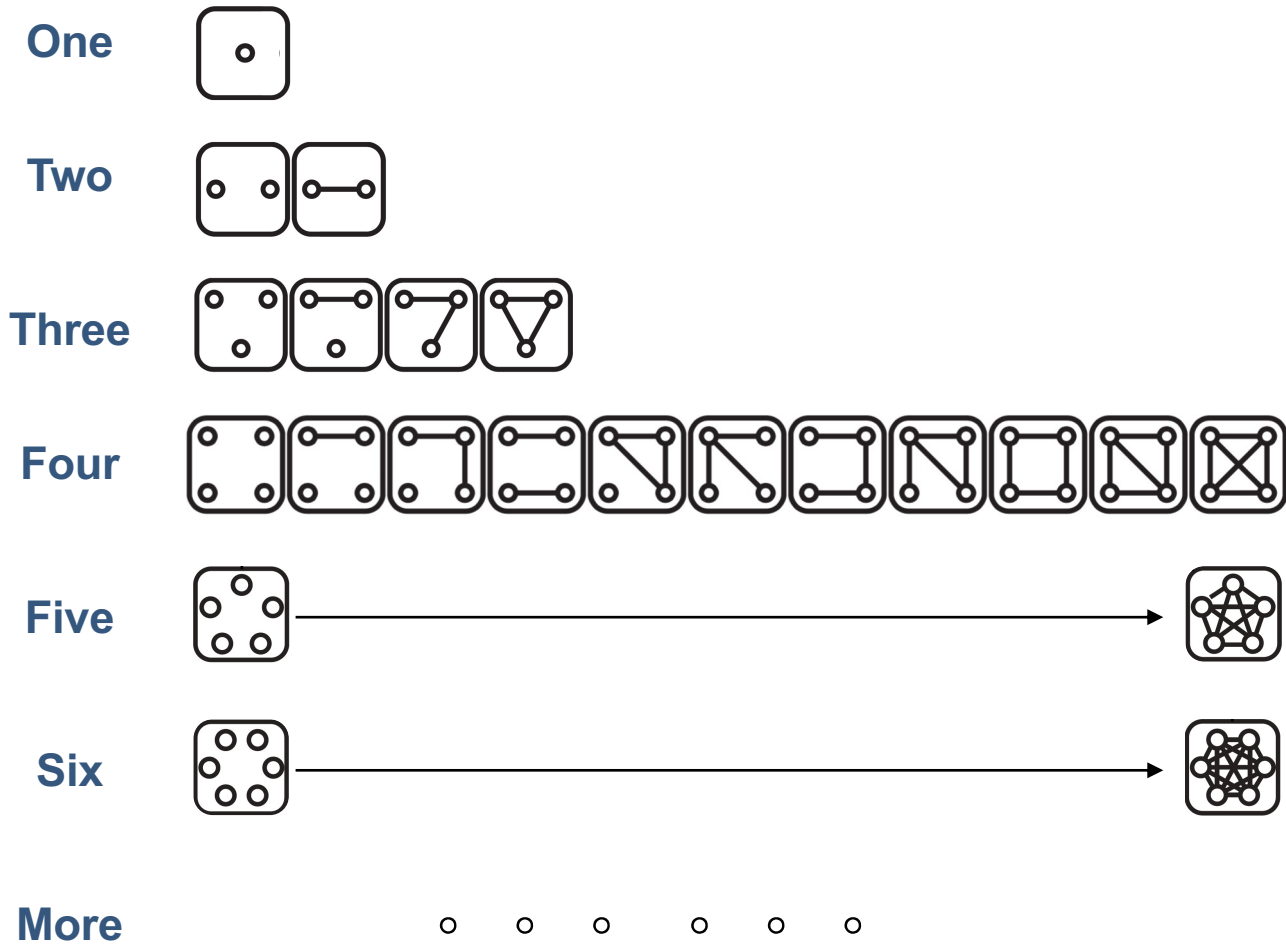


more diverse

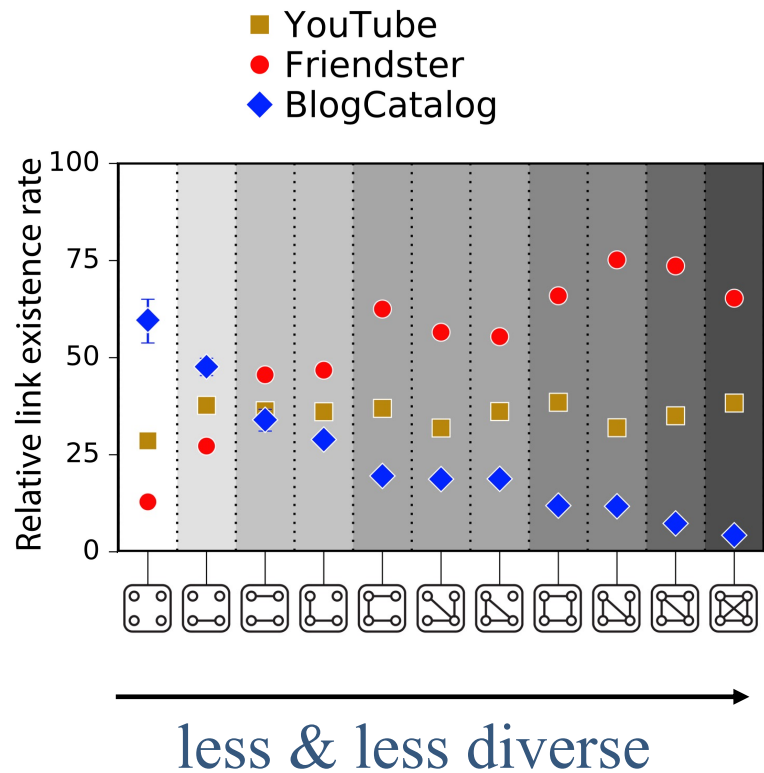


less diverse

Before Deep Learning ...



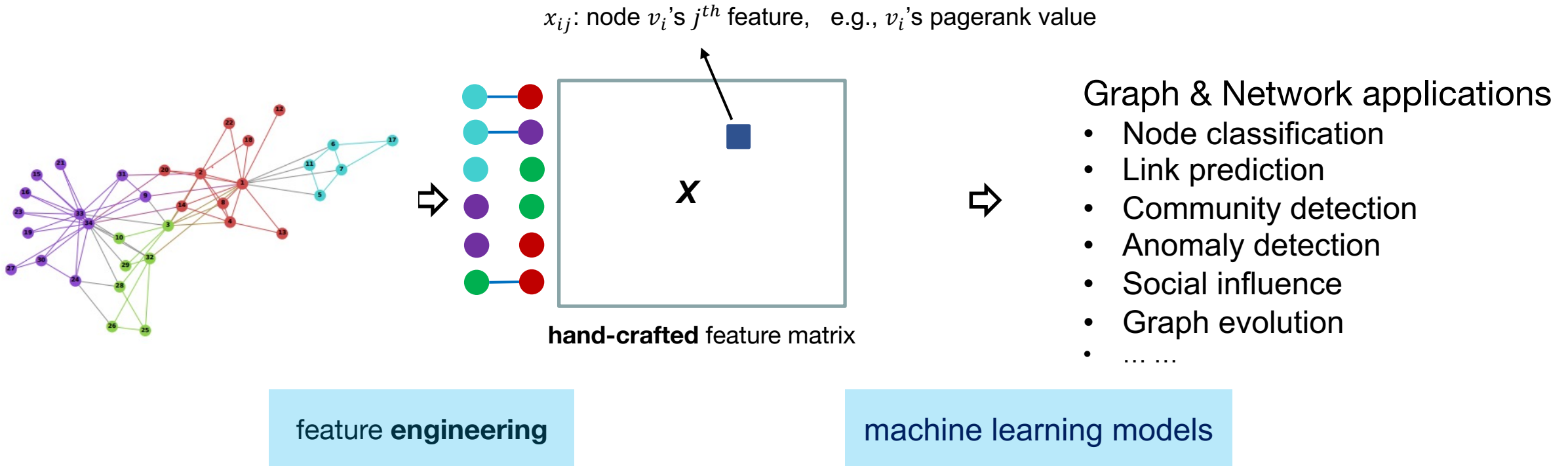
Before Deep Learning ...



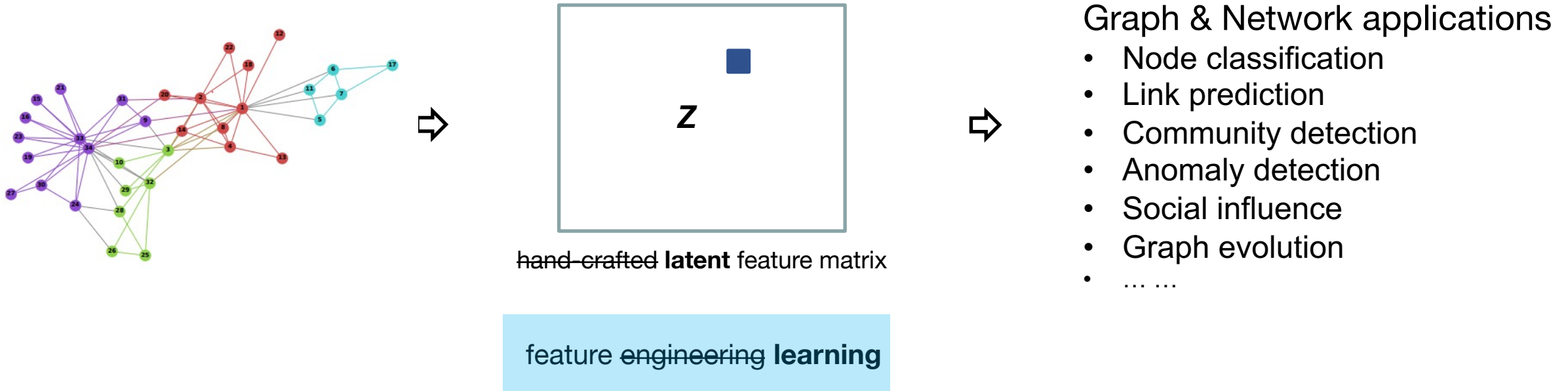
$$P_1(i-j | \text{simple motif}) < P_2(i-j | \text{complex motif})$$

$$P_1(i-j | \text{simple motif}) > P_2(i-j | \text{complex motif})$$

Before Deep Learning ...

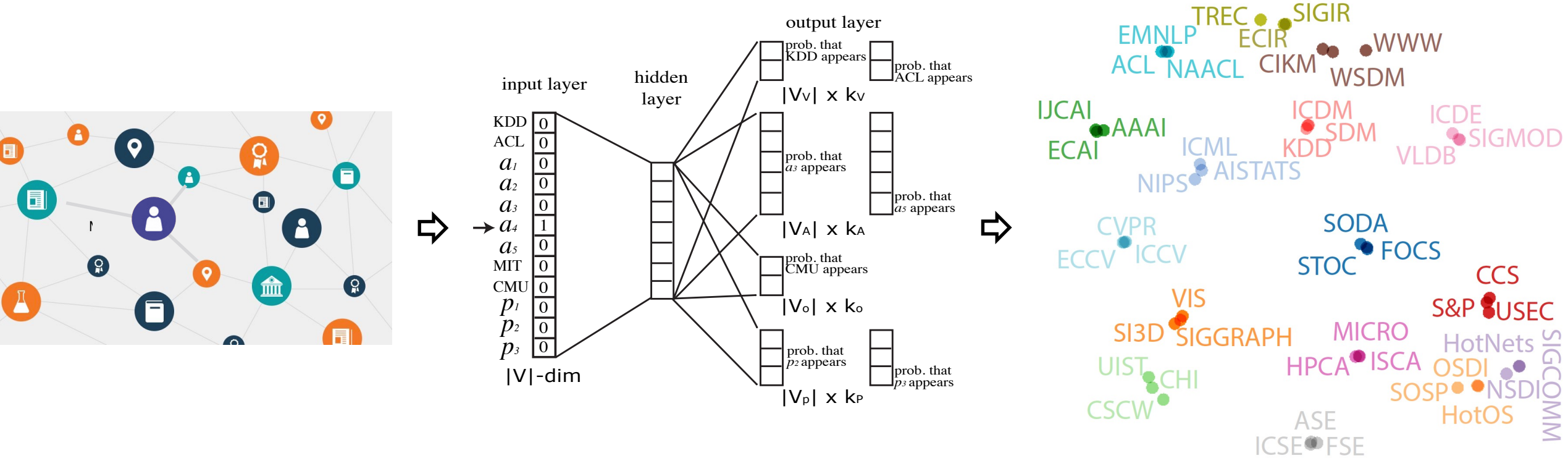


Graph Representation Learning



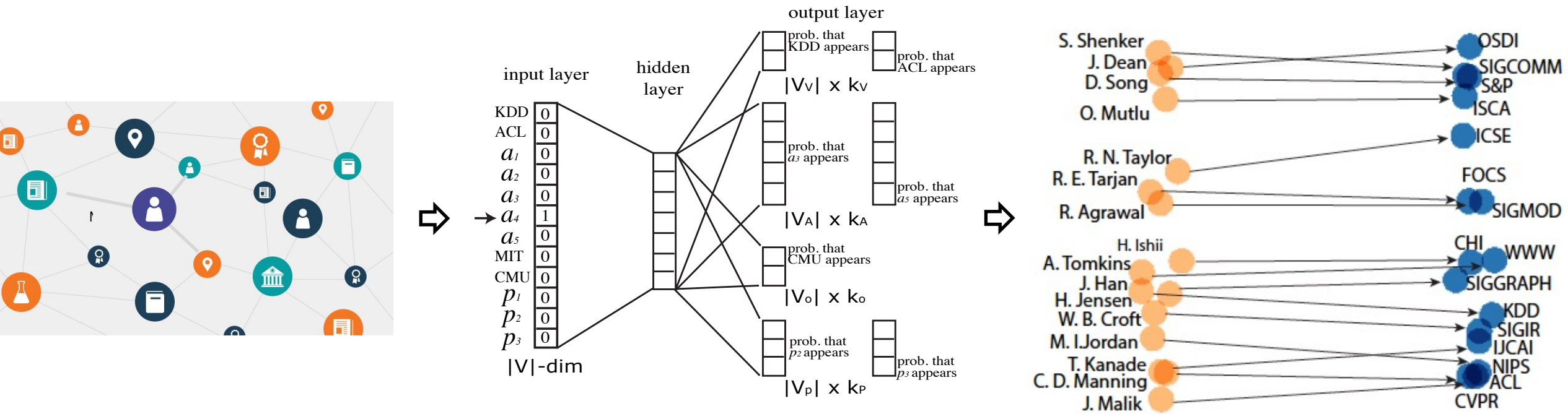
- Input: a network $G = (V, E)$
- Output: $Z \in R^{|V| \times k}$, $k \ll |V|$, k -dim vector Z_v for each node v .

Graph Representation Learning: An Example

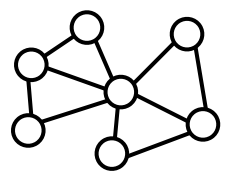
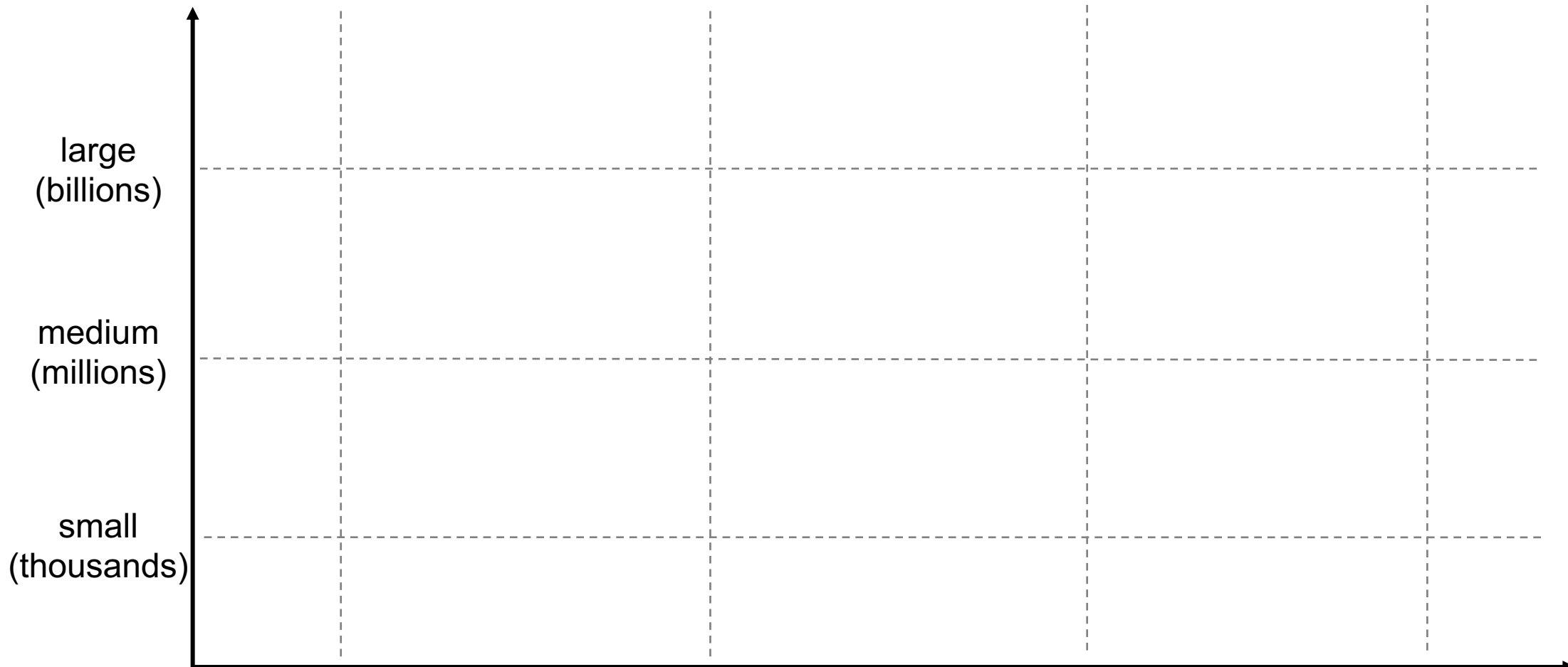


- Input: a graph $G = (V, E)$
- Output: $Z \in R^{|V| \times k}$, $k \ll |V|$, k -dim vector Z_v for each node v .

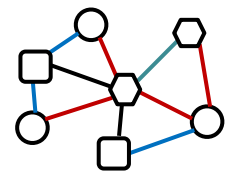
Graph Representation Learning: An Example



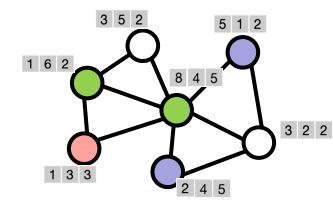
- Input: a graph $G = (V, E)$
- Output: $Z \in R^{|V| \times k}$, $k \ll |V|$, k -dim vector Z_v for each node v .



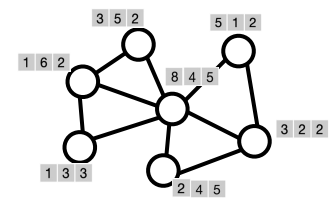
structure



heterogeneous structure / knowledge graph



structure with features



no label (pre-training)

Word Embeddings in NLP

- Input: a text corpus $D = \{W\}$
- Output: $X \in R^{|W| \times d}$, $d \ll |W|$, d -dim vector X_w for each word w .

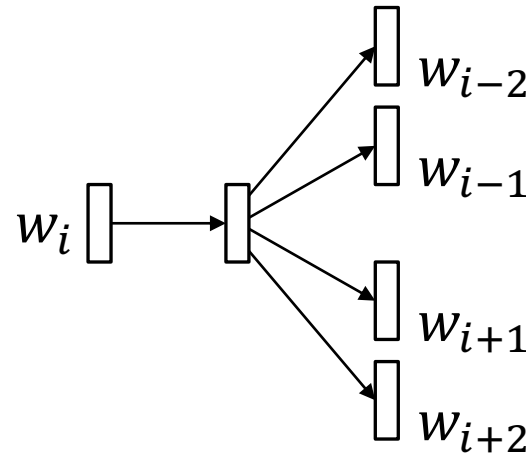
The connections between individuals form the structural backbone of human societies, which manifest as networks. In a network sense, individuals matter in the ways in which their unique demographic attributes and diverse interactions activate the emergence of new phenomena at larger, societal levels. Accordingly, this thesis develops computational models to investigating the ways that individuals are embedded in and interact within a wide range of over one hundred big networks—the biggest with over 60 million nodes and 1.8 billion edges—with an emphasis on two fundamental and interconnected directions: user demographics and network diversity.

Work in this thesis in the direction of demographics unveils the social strategies that are used to satisfy human social needs evolve across the lifespan, examines how males and females build and maintain similar or dissimilar social circles, and reveals how classical social theories—such as weak/strong ties, social balance, and small worlds—are influenced in the context of digitally recorded big networks coupled with socio-demographics. Our work on demographics also develops scalable graphical models that are capable of incorporating structured discoveries (features), facilitating conventional data mining tasks in networks. Work in this part demonstrates the predictability of user demographic attributes from networked systems, enabling the potential for precision marketing and business intelligence in social networking services. Work in this thesis in the direction of diversity examines how the

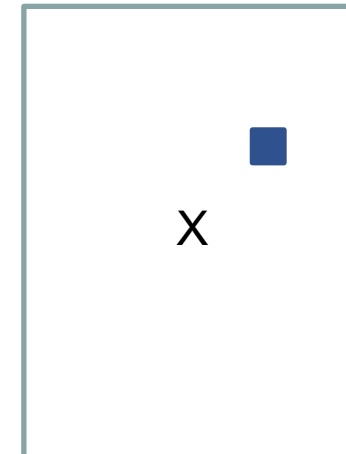


- Computational lens on big social and information networks.
- The connections between individuals form the structural ...
- In a network sense, individuals matters in the ways in which ...
- Accordingly, this thesis develops computational models to investigating the ways that ...
- We study two fundamental and interconnected directions: user demographics and network diversity
-

sentences

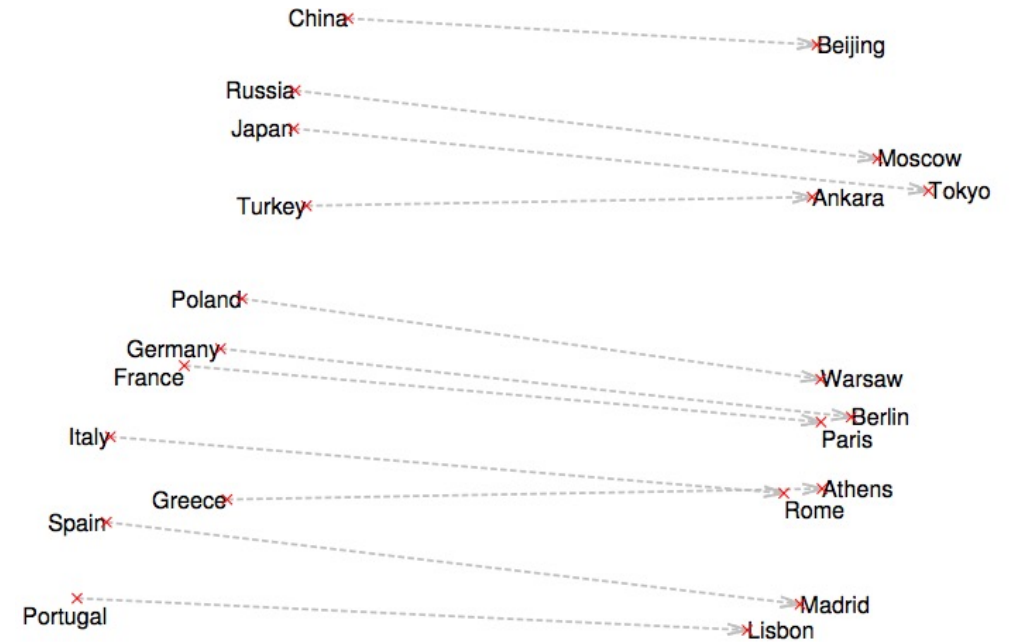
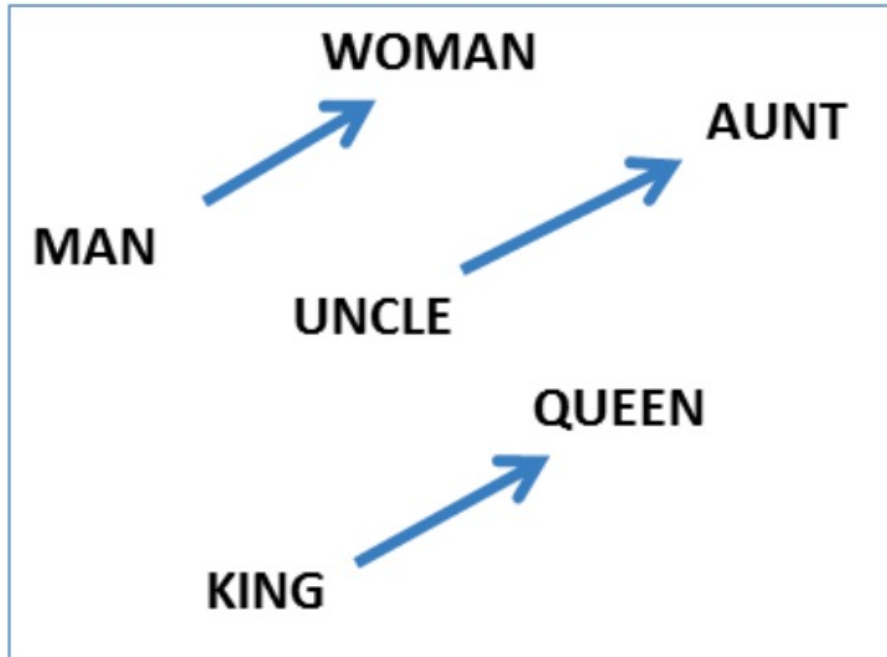


Word embedding models



latent feature matrix

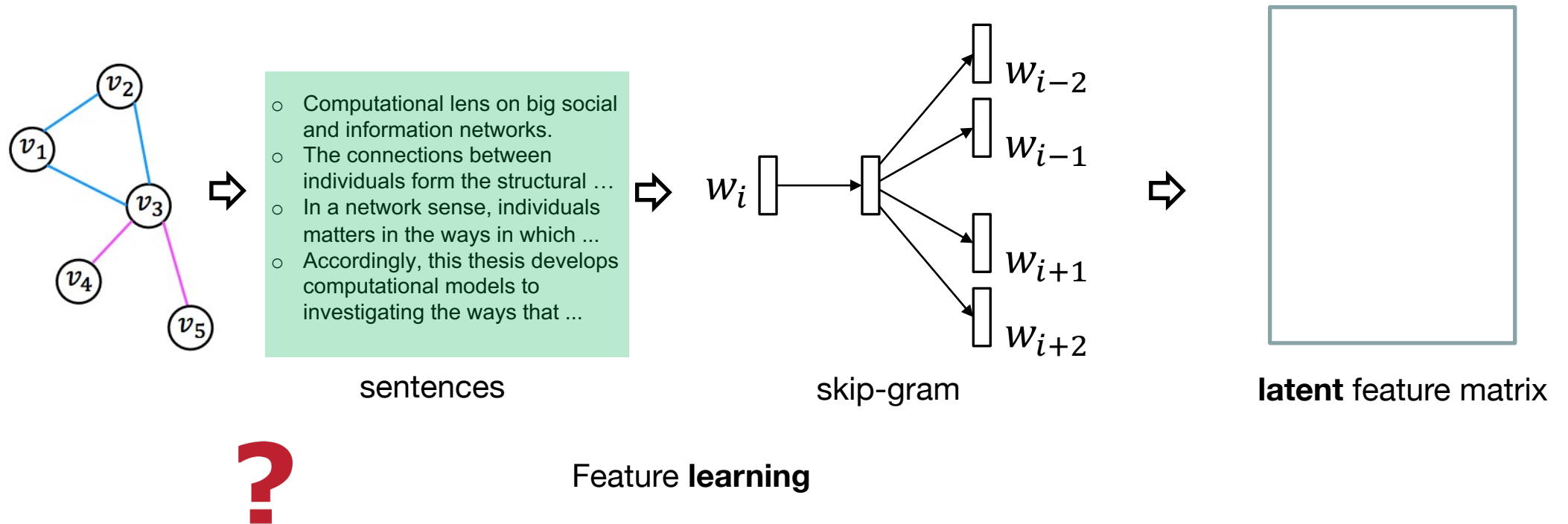
Word Embeddings in NLP



- Basic assumption: geographically close words---a word and its context words---in a sentence or document exhibit interrelations in human natural language.
- Key idea: try to predict the words that surrounding each one.

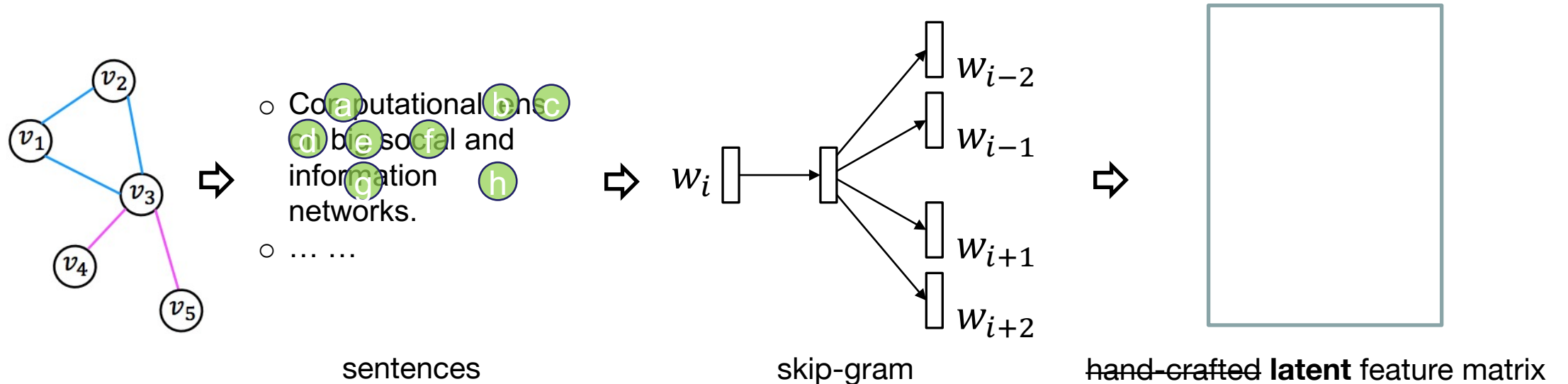
Network Embedding

- Input: a network $G = (V, E)$
- Output: $X \in R^{|V| \times k}$, $k \ll |V|$, k -dim vector X_v for each node v .



Network Embedding

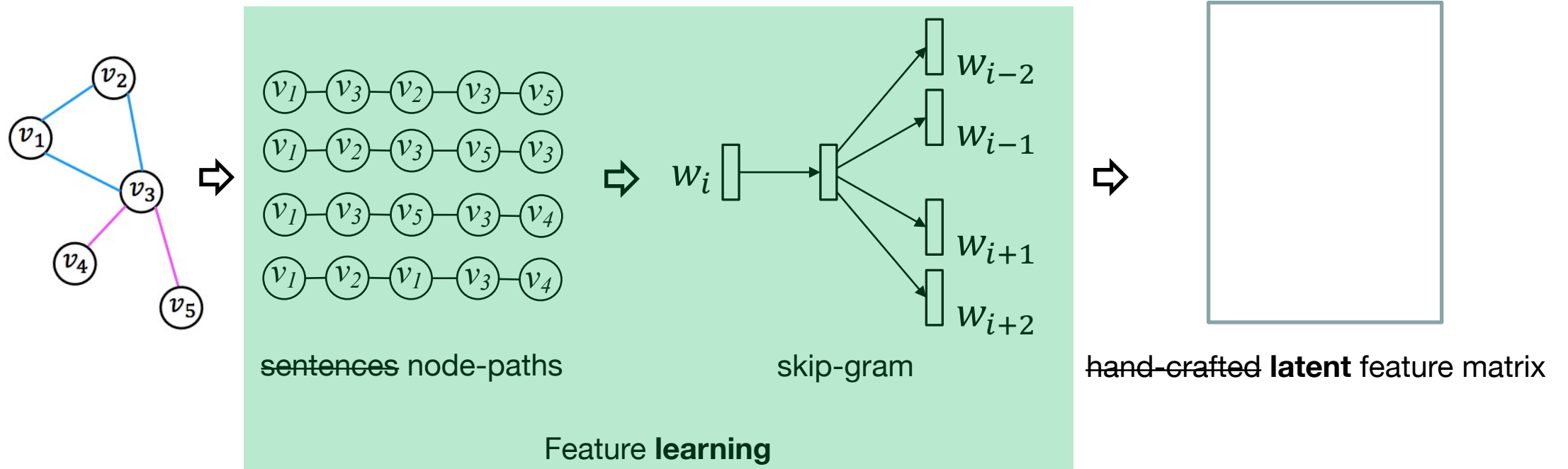
- Input: a network $G = (V, E)$
- Output: $X \in R^{|V| \times k}$, $k \ll |V|$, k -dim vector X_v for each node v .



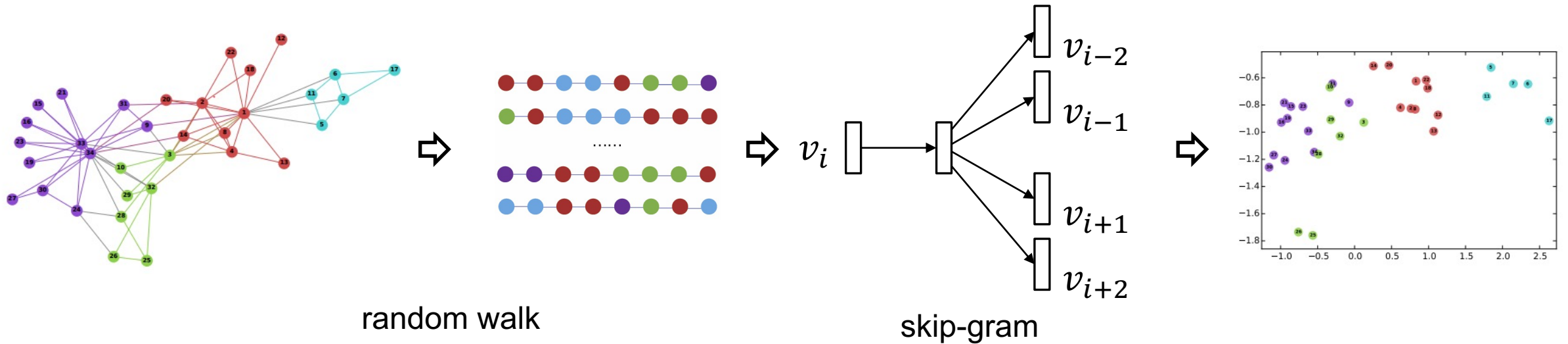
Feature engineering **learning**

Network Embedding: DeepWalk & node2vec

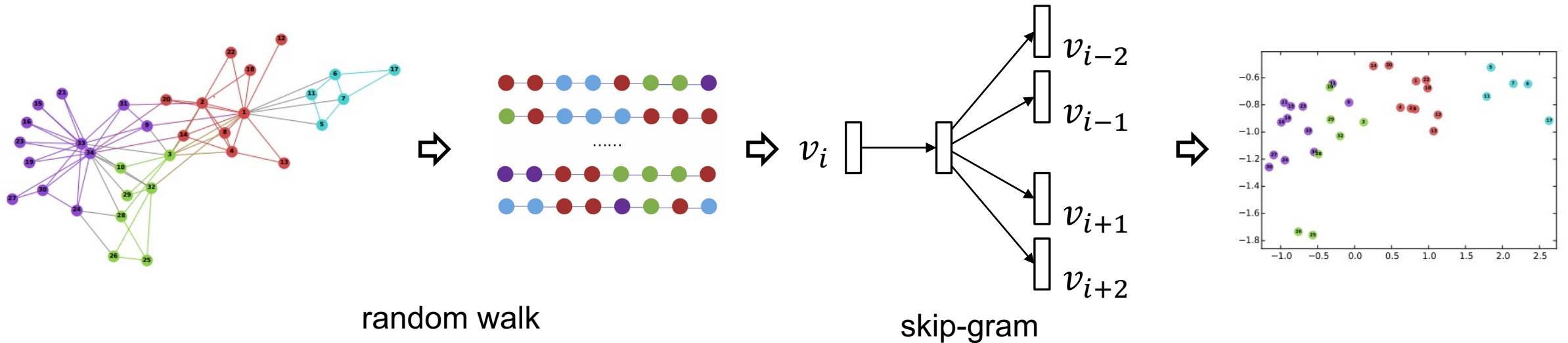
- Input: a network $G = (V, E)$
- Output: $X \in R^{|V| \times k}$, $k \ll |V|$, k -dim vector X_v for each node v .



Network Embedding: Random Walk + Skip Gram



Network Embedding: Random Walk + Skip Gram

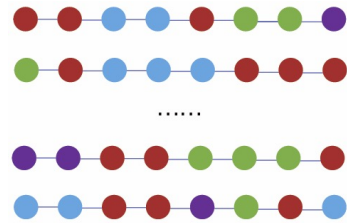
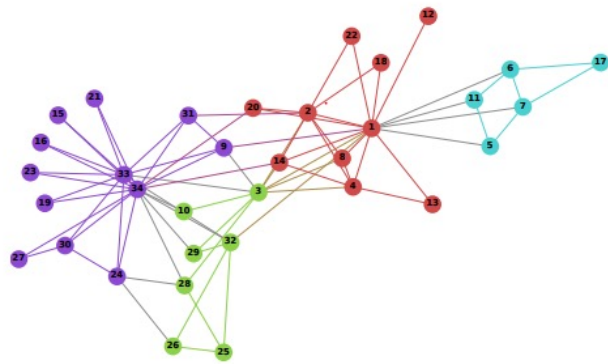


Random Walk Strategies:

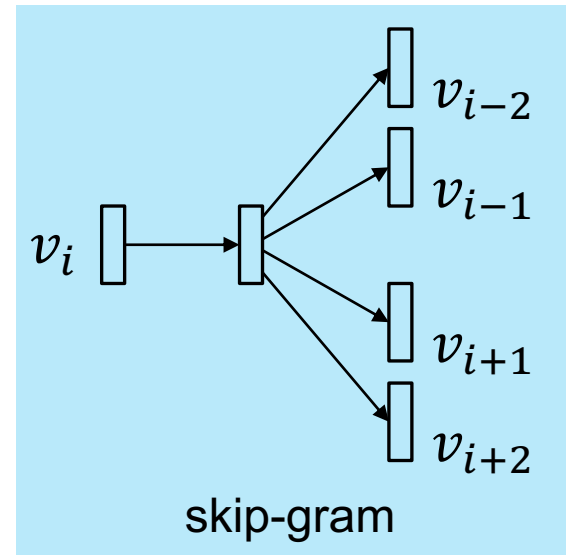
- DeepWalk (walk length > 1)
- LINE (walk length = 1)
- PTE (walk length = 1)
- node2vec (biased random walk)

1. Perozzi et al. DeepWalk: Online learning of social representations. In *KDD'14*. **Most Cited Paper in KDD'14**.
2. Tang et al. LINE: Large scale information network embedding. In *WWW'15*. **Most Cited Paper in WWW'15**.
3. Grover and Leskovec. node2vec: Scalable feature learning for networks. In *KDD'16*. **2nd Most Cited Paper in KDD'16**.

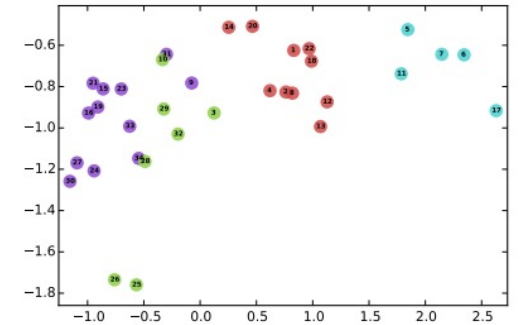
Understanding Random Walk + Skip Gram



random walk



skip-gram



Graph Language

- G : graph
- A : adjacency matrix
- D : degree matrix
- $vol(G)$: volume of G



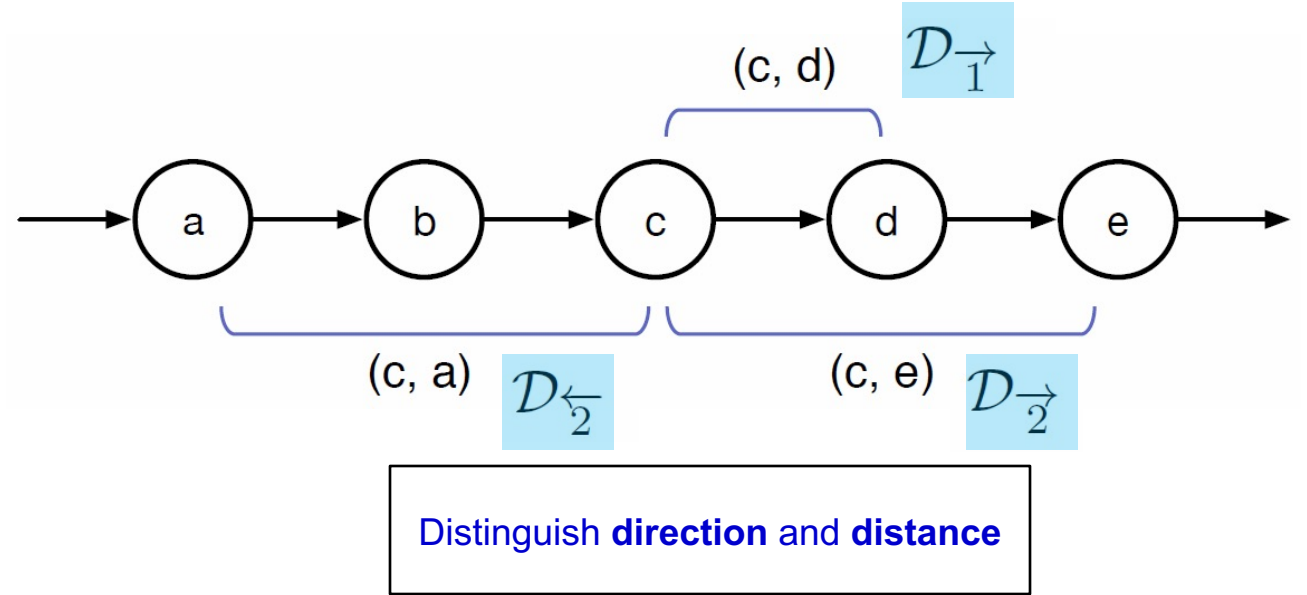
$$\log\left(\frac{\#(w, c)|\mathcal{D}|}{b\#(w)\#(c)}\right)$$

NLP Language

- $\#(w, c)$: co-occurrence of w & c
- $\#(w)$: occurrence of word w
- $\#(c)$: occurrence of context c
- \mathcal{D} : word-context pair (w, c) multi-set
- $|\mathcal{D}|$: number of word-context pairs

Understanding Random Walk + Skip Gram

$$\log \left(\frac{\#(w, c) |\mathcal{D}|}{b \#(w) \cdot \#(c)} \right) = \log \left(\frac{\frac{\#(w, c)}{|\mathcal{D}|}}{b \frac{\#(w)}{|\mathcal{D}|} \frac{\#(c)}{|\mathcal{D}|}} \right)$$



NLP Language

- $\#(w, c)$: co-occurrence of w & c
- $\#(w)$: occurrence of word w
- $\#(c)$: occurrence of context c
- \mathcal{D} : word-context pair (w, c) multi-set
- $|\mathcal{D}|$: number of word-context pairs

- Formally, for $r = 1, 2, \dots, T$, we define

$$\mathcal{D}_{\vec{r}} = \{ (w, c) : (w, c) \in \mathcal{D}, w = w_j^n, c = w_{j+r}^n \}$$

$$\mathcal{D}_{\overleftarrow{r}} = \{ (w, c) : (w, c) \in \mathcal{D}, w = w_{j+r}^n, c = w_j^n \}$$

Understanding Random Walk + Skip Gram

$$\log \left(\frac{\#(w, c) |\mathcal{D}|}{b \#(w) \cdot \#(c)} \right) = \log \left(\frac{\frac{\#(w, c)}{|\mathcal{D}|}}{b \frac{\#(w)}{|\mathcal{D}|} \frac{\#(c)}{|\mathcal{D}|}} \right)$$

$$\frac{\#(w, c)}{|\mathcal{D}|} = \frac{1}{2T} \sum_{r=1}^T \left(\frac{\#(w, c)_{\vec{r}}}{|\mathcal{D}_{\vec{r}}|} + \frac{\#(w, c)_{\overleftarrow{r}}}{|\mathcal{D}_{\overleftarrow{r}}|} \right)$$

the length of random walk $L \rightarrow \infty$ $P = D^{-1}A$

$$\frac{\#(w, c)_{\vec{r}}}{|\mathcal{D}_{\vec{r}}|} \xrightarrow{p} \frac{d_w}{\text{vol}(G)} (P^r)_{w,c} \quad \frac{\#(w, c)_{\overleftarrow{r}}}{|\mathcal{D}_{\overleftarrow{r}}|} \xrightarrow{p} \frac{d_c}{\text{vol}(G)} (P^r)_{c,w}$$

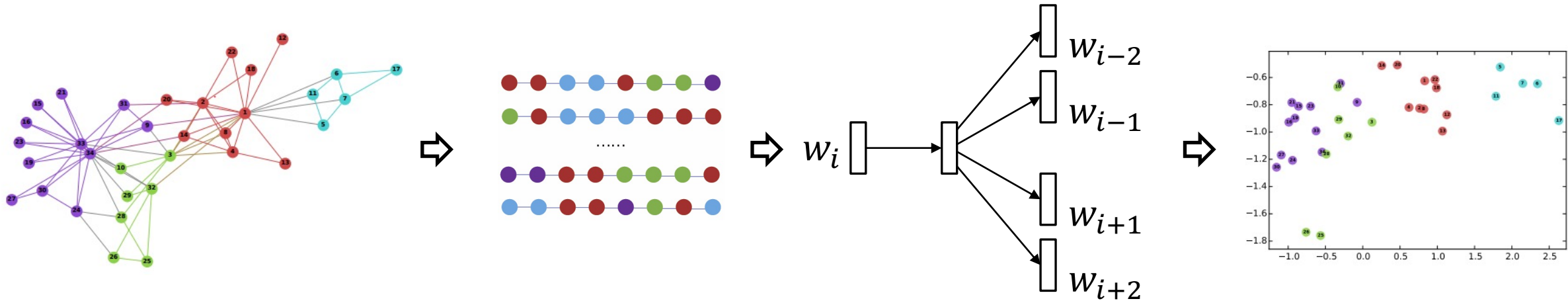
$$\frac{\#(w, c)}{|\mathcal{D}|} \xrightarrow{p} \frac{1}{2T} \sum_{r=1}^T \left(\frac{d_w}{\text{vol}(G)} (P^r)_{w,c} + \frac{d_c}{\text{vol}(G)} (P^r)_{c,w} \right)$$

$$\frac{\#(w)}{|\mathcal{D}|} \xrightarrow{p} \frac{d_w}{\text{vol}(G)} \quad \frac{\#(c)}{|\mathcal{D}|} \xrightarrow{p} \frac{d_c}{\text{vol}(G)}$$

$$\frac{\frac{\#(w, c)}{|\mathcal{D}|}}{\frac{\#(w)}{|\mathcal{D}|} \cdot \frac{\#(c)}{|\mathcal{D}|}} \xrightarrow{p} \frac{\frac{1}{2T} \sum_{r=1}^T \left(\frac{d_w}{\text{vol}(G)} (P^r)_{w,c} + \frac{d_c}{\text{vol}(G)} (P^r)_{c,w} \right)}{\frac{d_w}{\text{vol}(G)} \cdot \frac{d_c}{\text{vol}(G)}}$$

$$= \text{vol}(G) \left(\frac{1}{T} \sum_{r=1}^T P^r \right) D^{-1}$$

Understanding Random Walk + Skip Gram



DeepWalk is *asymptotically and implicitly* factorizing

$$\log \left(\frac{\text{vol}(G)}{b} \left(\frac{1}{T} \sum_{r=1}^T (\mathbf{D}^{-1} \mathbf{A})^r \right) \mathbf{D}^{-1} \right)$$

\mathbf{A} Adjacency matrix

\mathbf{D} Degree matrix

$$\text{vol}(G) = \sum_i \sum_j A_{ij}$$

b : #negative samples

T : context window size

Unifying DeepWalk, LINE, PTE, & node2vec as Matrix Factorization

- DeepWalk $\log \left(\frac{\text{vol}(G)}{b} \left(\frac{1}{T} \sum_{r=1}^T (\mathbf{D}^{-1} \mathbf{A})^r \right) \mathbf{D}^{-1} \right)$
- LINE $\log \left(\frac{\text{vol}(G)}{b} \mathbf{D}^{-1} \mathbf{A} \mathbf{D}^{-1} \right)$ $T = 1$
- PTE $\log \left(\begin{bmatrix} \alpha \text{vol}(G_{\text{ww}}) (\mathbf{D}_{\text{row}}^{\text{ww}})^{-1} \mathbf{A}_{\text{ww}} (\mathbf{D}_{\text{col}}^{\text{ww}})^{-1} \\ \beta \text{vol}(G_{\text{dw}}) (\mathbf{D}_{\text{row}}^{\text{dw}})^{-1} \mathbf{A}_{\text{dw}} (\mathbf{D}_{\text{col}}^{\text{dw}})^{-1} \\ \gamma \text{vol}(G_{\text{lw}}) (\mathbf{D}_{\text{row}}^{\text{lw}})^{-1} \mathbf{A}_{\text{lw}} (\mathbf{D}_{\text{col}}^{\text{lw}})^{-1} \end{bmatrix} \right) - \log b$
- node2vec $\log \left(\frac{\frac{1}{2T} \sum_{r=1}^T (\sum_u \mathbf{X}_{w,u} \mathbf{P}_{c,w,u}^r + \sum_u \mathbf{X}_{c,u} \mathbf{P}_{w,c,u}^r)}{b (\sum_u \mathbf{X}_{w,u}) (\sum_u \mathbf{X}_{c,u})} \right)$

\mathbf{A} Adjacency matrix

\mathbf{D} Degree matrix

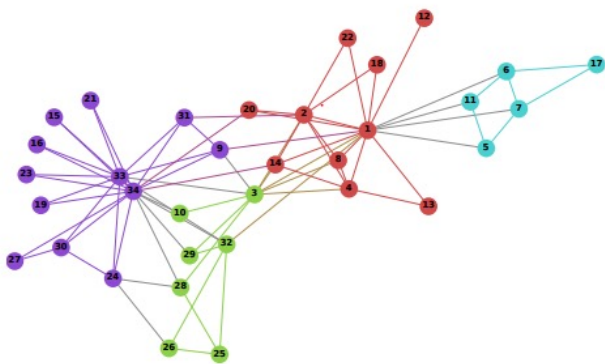
$$\text{vol}(G) = \sum_i \sum_j A_{ij}$$

b : #negative samples

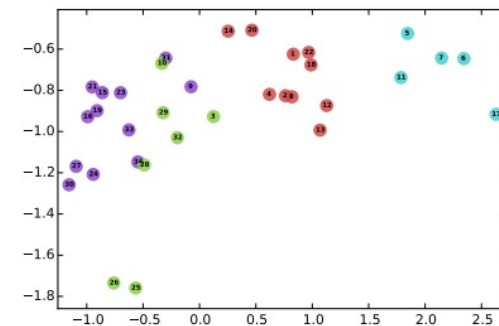
T : context window size

1. Perozzi et al. DeepWalk: Online learning of social representations. In *KDD'14*. **Most Cited Paper in KDD'14**.
2. Tang et al. LINE: Large scale information network embedding. In *WWW'15*. **Most Cited Paper in WWW'15**.
3. Grover and Leskovec. node2vec: Scalable feature learning for networks. In *KDD'16*. **2nd Most Cited Paper in KDD'16**.

NetMF: Explicitly Factorizing the Matrix



**Matrix
Factorization
(NetMF)**



DeepWalk is asymptotically and *implicitly* factorizing

$$\log \left(\frac{\text{vol}(G)}{b} \left(\frac{1}{T} \sum_{r=1}^T (\mathbf{D}^{-1} \mathbf{A})^r \right) \mathbf{D}^{-1} \right)$$

\mathbf{A} Adjacency matrix

\mathbf{D} Degree matrix

$$\text{vol}(G) = \sum_i \sum_j A_{ij}$$

b : #negative samples

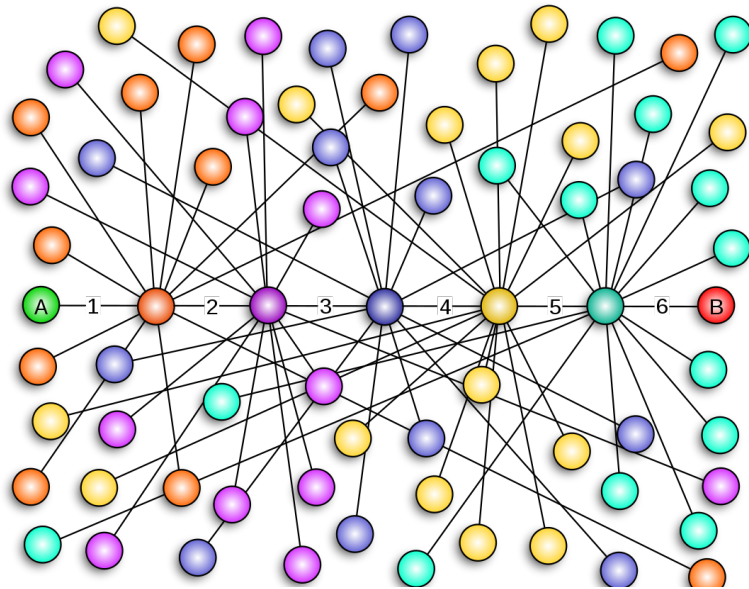
T : context window size

NetMF

1. Construction of \mathbf{S}
2. Factorization of \mathbf{S}

$$\mathbf{S} = \log \left(\frac{\text{vol}(G)}{b} \left(\frac{1}{T} \sum_{r=1}^T (\mathbf{D}^{-1} \mathbf{A})^r \right) \mathbf{D}^{-1} \right)$$

Challenge?



six (four) degrees of separation

$$\Rightarrow \mathbf{S} = \log \left(\frac{\text{vol}(G)}{b} \left(\frac{1}{T} \sum_{r=1}^T (\mathbf{D}^{-1} \mathbf{A})^r \right) \mathbf{D}^{-1} \right)$$

n^2 non-zeros
Dense!!

Time complexity
 $O(n^3)$

How to Solve it?

NetMF

1. Construction of \mathbf{S}
2. Factorization of \mathbf{S}

NetSMF—**S**parse

1. **S**parse Construction of \mathbf{S}
2. **S**parse Factorization of \mathbf{S}

$$\mathbf{S} = \log \left(\frac{\text{vol}(G)}{b} \left(\frac{1}{T} \sum_{r=1}^T (\mathbf{D}^{-1} \mathbf{A})^r \right) \mathbf{D}^{-1} \right)$$

Sparsify \mathcal{S}

For random-walk matrix polynomial $L = D - \sum_{r=1}^T \alpha_r D (D^{-1} A)^r$

where $\sum_{r=1}^T \alpha_r = 1$ and α_r non-negative

One can construct a $(1 + \epsilon)$ -spectral sparsifier \tilde{L} with $O(n \log n \epsilon^{-2})$ non-zeros

in time $O(T^2 m \epsilon^{-2} \log n)$ for undirected graphs

$$\alpha_1 = \dots = \alpha_T = \frac{1}{T} \quad \Rightarrow \quad \begin{aligned} \mathcal{S} &= \log^\circ \left(\frac{\text{vol}(G)}{b} \left(\frac{1}{T} \sum_{r=1}^T (D^{-1} A)^r \right) D^{-1} \right) \\ &= \log^\circ \left(\frac{\text{vol}(G)}{b} D^{-1} (D - L) D^{-1} \right) \\ &\approx \log^\circ \left(\frac{\text{vol}(G)}{b} D^{-1} (D - \tilde{L}) D^{-1} \right) \end{aligned}$$

NetSMF

- ▶ Construct a random walk matrix polynomial sparsifier, $\tilde{\mathbf{L}}$
- ▶ Construct a NetMF matrix sparsifier.

$$\text{trunc_log}^\circ \left(\frac{\text{vol}(G)}{b} \mathbf{D}^{-1} (\mathbf{D} - \tilde{\mathbf{L}}) \mathbf{D}^{-1} \right)$$

- ▶ Factorize the constructed matrix

NetSMF---bounded approximation error

$$\begin{aligned}
 & \log^\circ \left(\frac{\text{vol}(G)}{b} \left(\frac{1}{T} \sum_{r=1}^T (\mathbf{D}^{-1} \mathbf{A})^r \right) \mathbf{D}^{-1} \right) \\
 &= \log^\circ \left(\frac{\text{vol}(G)}{b} \mathbf{D}^{-1} (\mathbf{D} - \mathbf{L}) \mathbf{D}^{-1} \right) \longrightarrow \mathbf{M} \\
 &\approx \log^\circ \left(\frac{\text{vol}(G)}{b} \mathbf{D}^{-1} (\mathbf{D} - \tilde{\mathbf{L}}) \mathbf{D}^{-1} \right) \longrightarrow \tilde{\mathbf{M}}
 \end{aligned}$$

Theorem

The singular value of $\tilde{\mathbf{M}} - \mathbf{M}$ satisfies

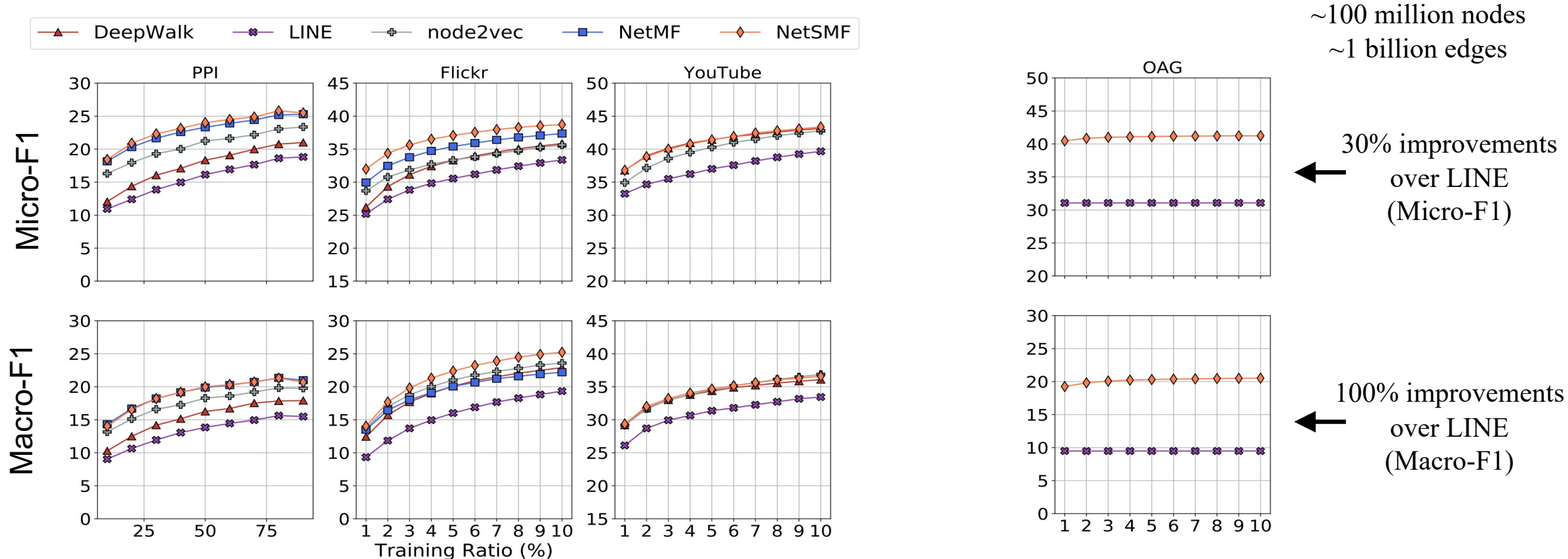
$$\sigma_i(\tilde{\mathbf{M}} - \mathbf{M}) \leq \frac{4\epsilon}{\sqrt{d_i d_{\min}}}, \forall i \in [n].$$

Theorem

Let $\|\cdot\|_F$ be the matrix Frobenius norm. Then

$$\left\| \text{trunc.log}^\circ \left(\frac{\text{vol}(G)}{b} \tilde{\mathbf{M}} \right) - \text{trunc.log}^\circ \left(\frac{\text{vol}(G)}{b} \mathbf{M} \right) \right\|_F \leq \frac{4\epsilon \text{vol}(G)}{b\sqrt{d_{\min}}} \sqrt{\sum_{i=1}^n \frac{1}{d_i}}.$$

Results



- **Effectiveness:** NetMF (explicit MF) \approx NetSMF (sparse MF) $>$ DeepWalk/LINE (implicit MF)
- **Scalability:** NetSMF can handle billion-scale network embedding

Microsoft researchers unlock the black box of network embedding

Published February 7, 2018

By [Kuansan Wang](#), Managing Director, MSR Outreach Academic Services



Research Area

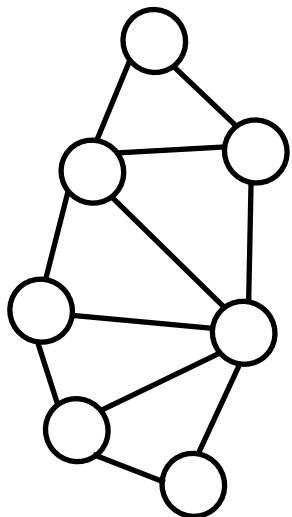
 [Data platforms and analytics](#)

 [Search and information retrieval](#)



At the [ACM Conference on Web Search and Data Mining 2018](#), my team will introduce research that, for the first time, provides a theoretical explanation of popular methods used to automatically map the structure and characteristics of networks, known as network embedding. We then use this theoretical explanation to present a new network embedding method that performs as well as or better than existing methods.

Networks are fundamental ways of representing knowledge and relating to the world. As humans, we think through association; we naturally draw links between concepts or entities. People are linked through family relationships or through collaborations. Diseases are linked to treatments. Works of art are linked to their creators. Wikipedia represents the interconnectedness of human knowledge.

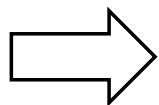


NetSMF

T: $O(T(T + d)m \log n + nd^2)$
 S: $O(Tm \log n + m + nd)$

1. Construct
sparse $f^o(\mathbf{M})$

2. Factorize
sparse $f^o(\mathbf{M})$



NetMF

T: $O(\beta mk + n^2 k)$
 S: $O(m + n^2)$

1. Eigen-decomp. to
 get low rank $\mathbf{M} \approx \mathbf{LR}$

2. Construct
 $f^o(\mathbf{LR})$

3. Factorize
 $f^o(\mathbf{LR})$

Goal:
 To factorize
 matrix $f^o(\mathbf{M})$

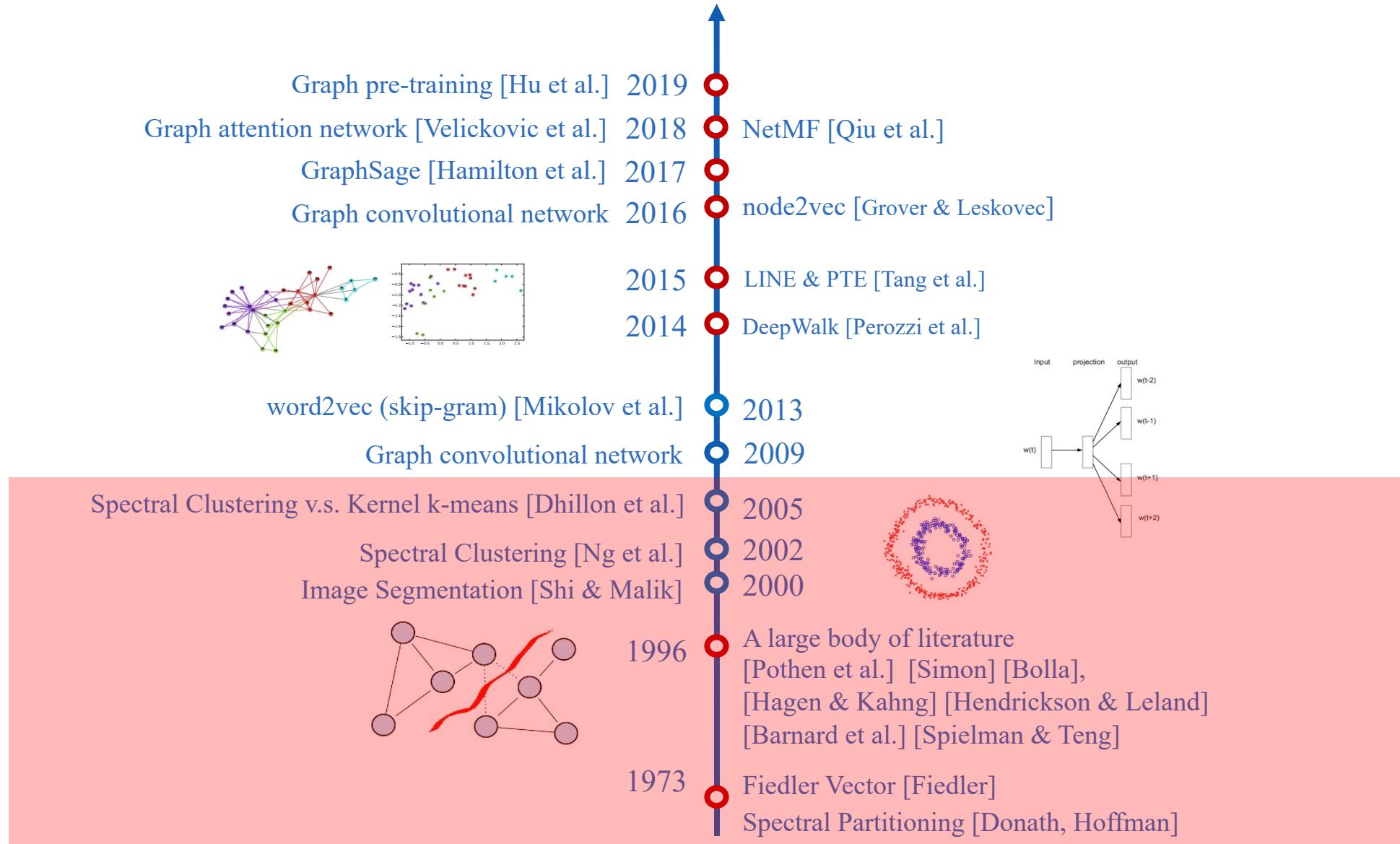
SketchNE

Time: $O(qmk + qnk^2)$
 Space: $O(m + nk)$

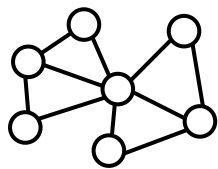
1. *Fast* eigen-
 decomp. to get low
 rank $\mathbf{M} \approx \mathbf{LR}$

2. *Sparse-sign randomized single pass*
SVD to avoid explicit construction &
 factorization of $f^o(\mathbf{LR})$

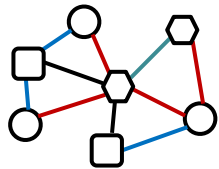
A Brief History of Network/Graph Embedding



Network Embedding



structure



heterogeneous structure
/ knowledge graph

Heterogeneous Graphs?

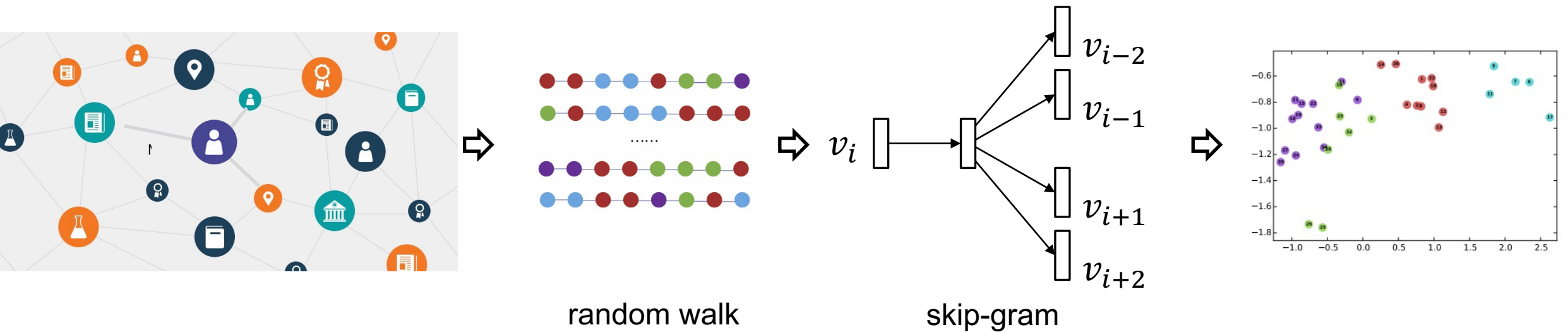


Academic Graph



Microsoft Office Graph

Homogeneous Network Embedding

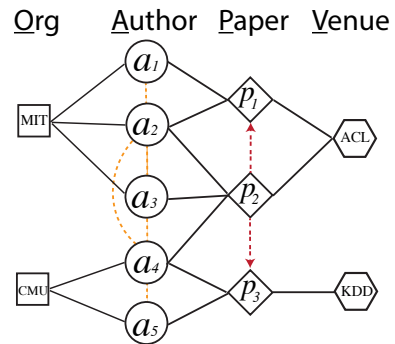


- How do we random walk over **different types of edges**?
- How do we apply skip-gram over **different types of nodes**?

metapath-Based Random Walks

- Given a meta-path scheme

$$\mathcal{P}: V_1 \xrightarrow{R_1} V_2 \xrightarrow{R_2} \dots V_t \xrightarrow{R_t} V_{t+1} \dots \xrightarrow{R_{l-1}} V_l$$



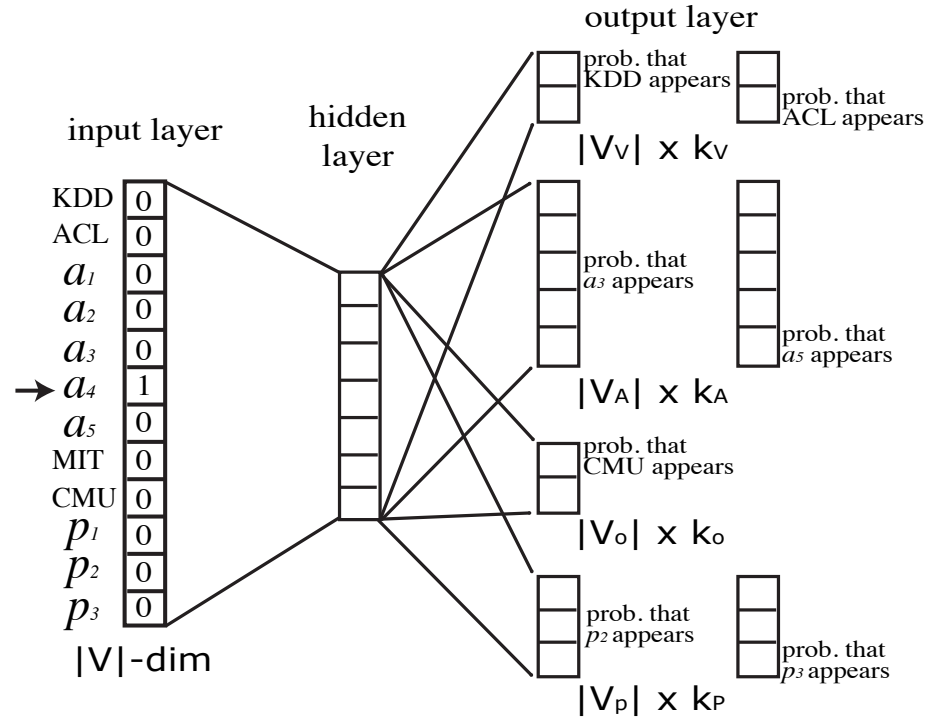
- The transition probability at step i is defined as

$$p(v^{i+1} | v_t^i, \mathcal{P}) = \begin{cases} \frac{1}{|N_{t+1}(v_t^i)|} & (v^{i+1}, v_t^i) \in E, \phi(v^{i+1}) = t+1 \\ 0 & (v^{i+1}, v_t^i) \in E, \phi(v^{i+1}) \neq t+1 \\ 0 & (v^{i+1}, v_t^i) \notin E \end{cases}$$

- Recursive guidance for random walkers, *i.e.*,

$$p(v^{i+1} | v_t^i) = p(v^{i+1} | v_1^i), \text{ if } t = l$$

Heterogeneous Skip-Gram



- objective function (heterogeneous negative sampling)

$$\mathcal{O}(\mathbf{X}) = \log \sigma(X_{c_t} \cdot X_v) + \sum_{k=1}^K \mathbb{E}_{u_t^k \sim P_t(u_t)} [\log \sigma(-X_{u_t^k} \cdot X_v)]$$

- softmax in *metapath2vec*

$$p(c_t|v; \theta) = \frac{e^{X_{c_t} \cdot X_v}}{\sum_{u \in V} e^{X_u \cdot X_v}}$$

- softmax in *metapath2vec++*

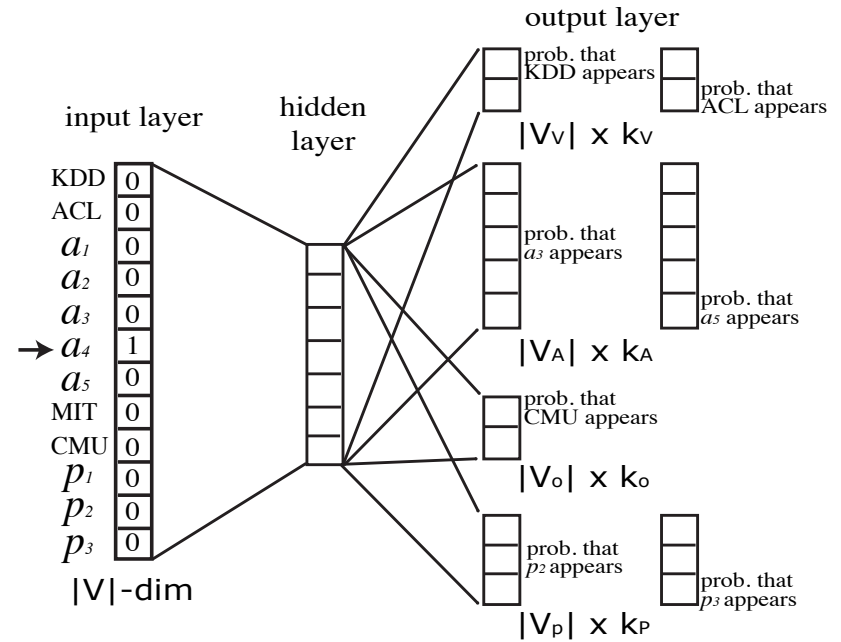
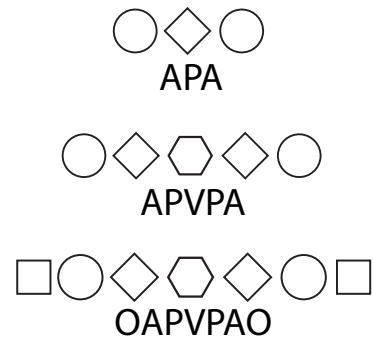
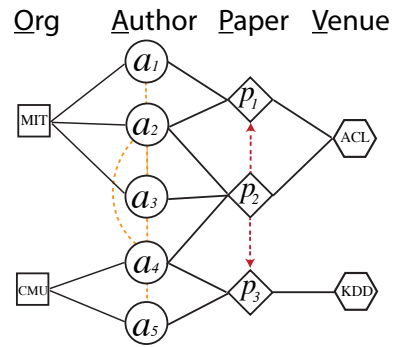
$$p(c_t|v; \theta) = \frac{e^{X_{c_t} \cdot X_v}}{\sum_{u_t \in V_t} e^{X_{u_t} \cdot X_v}}$$

- stochastic gradient descent

$$\frac{\partial \mathcal{O}(\mathbf{X})}{\partial X_{u_t^k}} = (\sigma(X_{u_t^k} \cdot X_v) - \mathbb{I}_{c_t}[u_t^k]) X_v$$

$$\frac{\partial \mathcal{O}(\mathbf{X})}{\partial X_v} = \sum_{k=0}^K (\sigma(X_{u_t^k} \cdot X_v) - \mathbb{I}_{c_t}[u_t^k]) X_{u_t^k}$$

Heterogeneous Network Embedding: *metapath2vec*



**metapath-based
random walks**

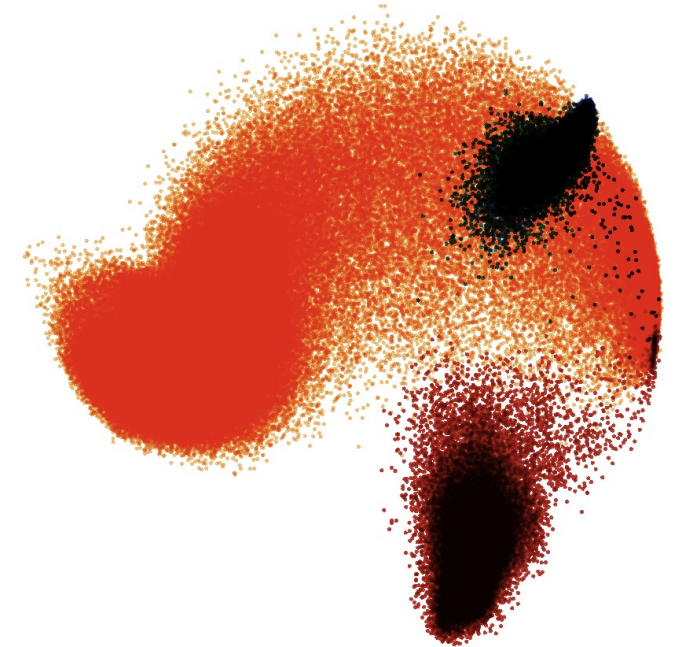
**heterogeneous
skip-gram**

Applications: Embedding Heterogeneous Academic Graph

	219,352,601 Papers
	239,952,453 Authors
	664,190 Topics
	4,388 Conferences
	48,731 Journals
	25,509 Institutions



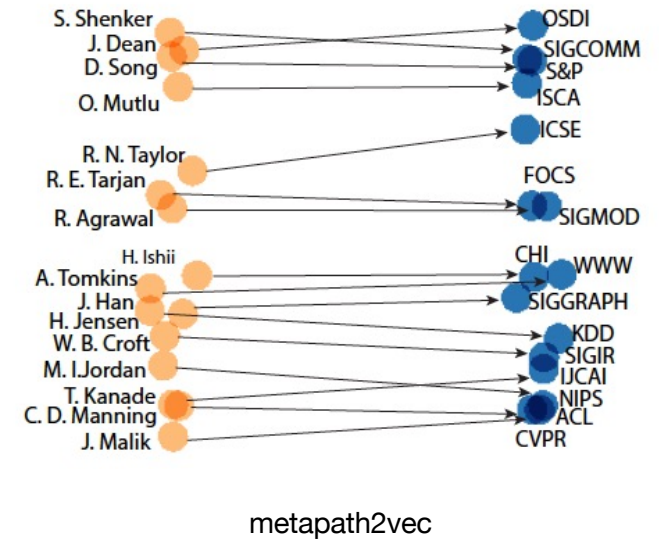
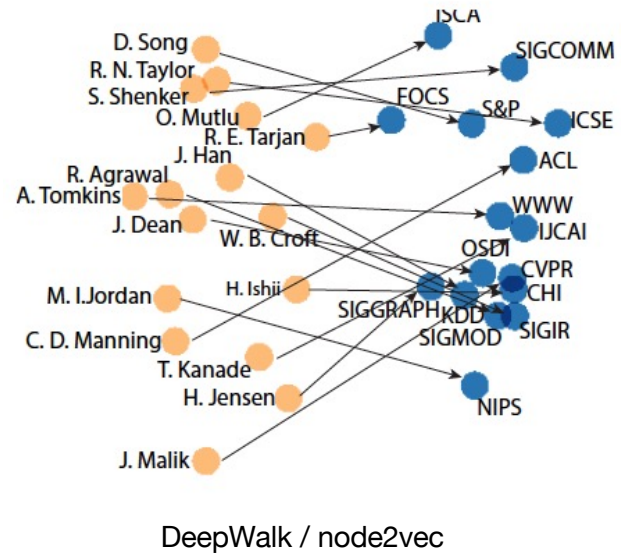
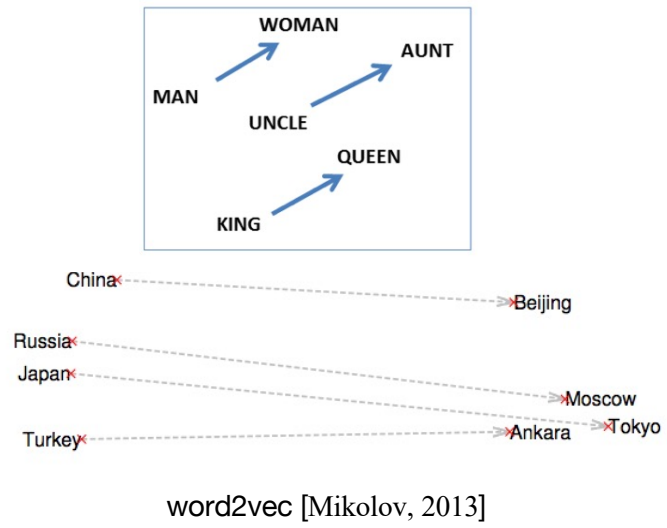
metapath2vec



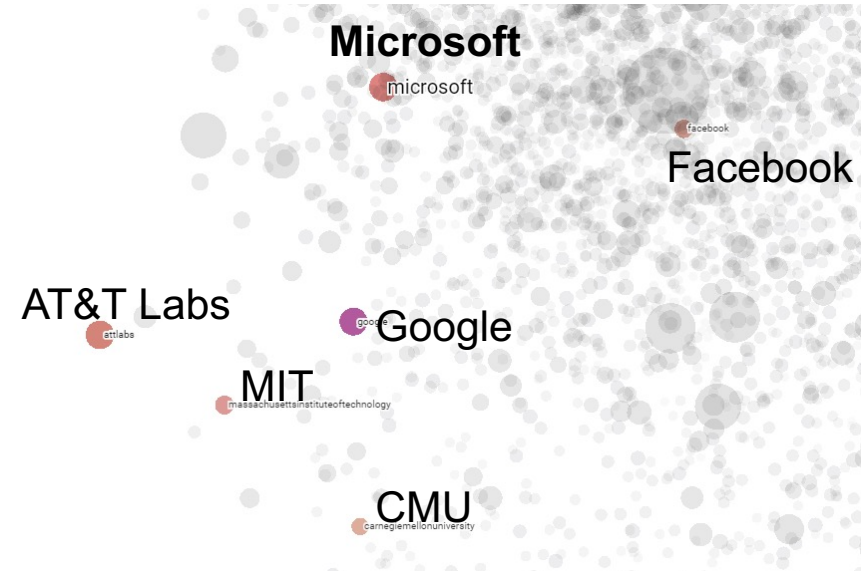
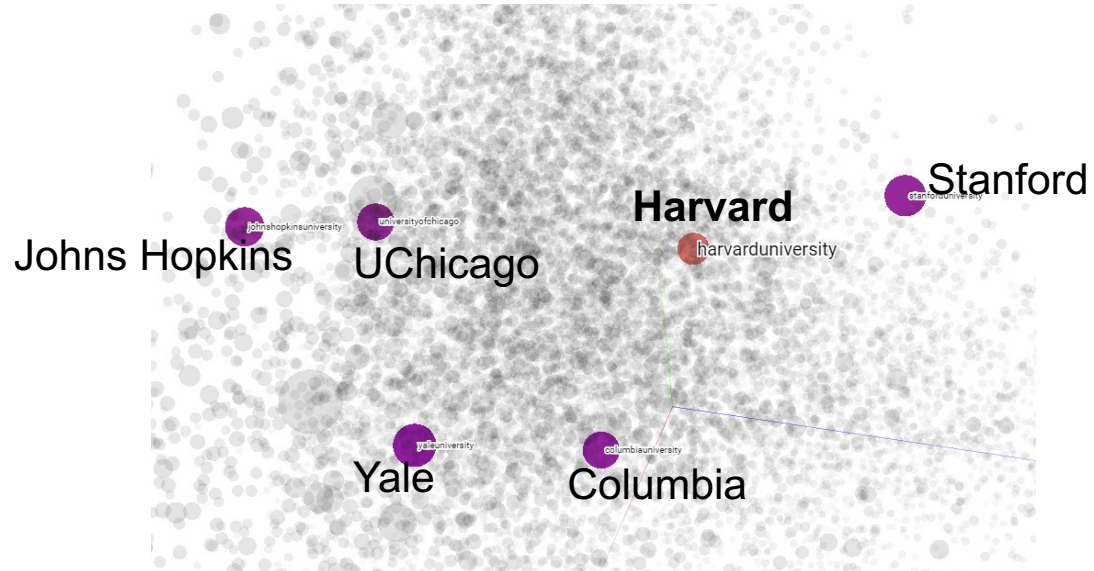
Open Academic Graph:

- *AMiner*
- *Microsoft Academic Graph*

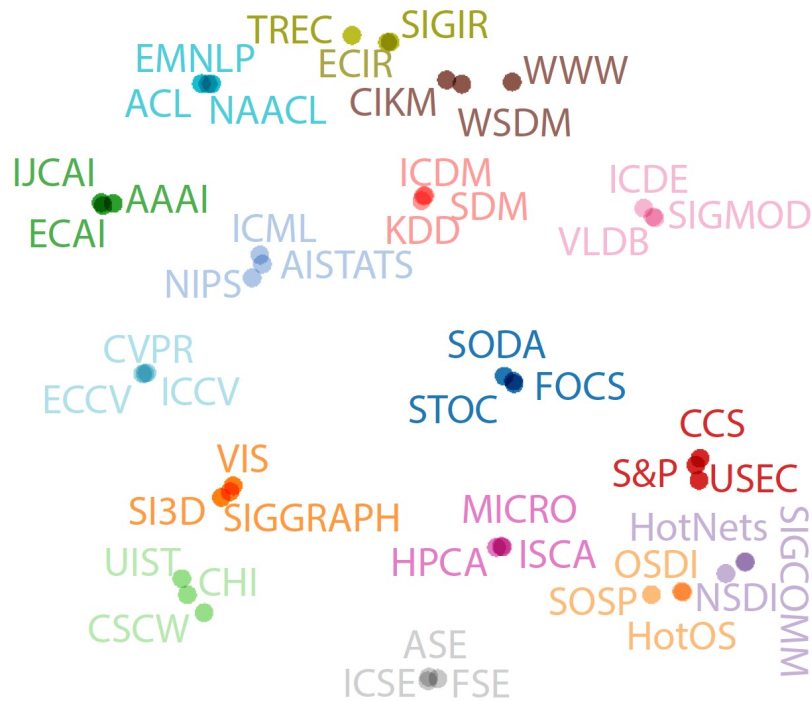
Applications



Applications



Applications



Microsoft Academic

Nature



Sign up /



Nature

257,560 Papers 23,827,633 Citations*

About

Nature is a British weekly scientific journal founded and based in London, England. As a multidisciplinary publication Nature features peer-reviewed research from a variety of academic disciplines, mainly in science, technology, and the natural sciences. It has core editorial offices across the United States, continental Europe, and Asia under the international scientific publishing company Springer Nature. Nature was one of the world's most cited scientific journals by the Science Edition of the 2019 Journal Citation Reports (with an ascribed impact factor of 42.778), making it one of the world's most-read and most prestigious academic journals. As of 2012, it claimed an online readership of about 3 million unique readers per month.

Website Links

nature.com | en.wikipedia.org

Related Journals

- Science
 - Proceedings of the National Academy of Sciences of the United States of America
 - Nature Communications
 - PLOS Biology
 - Philosophical Transactions of the Royal Society B
 - Current Biology
 - BioEssays
 - Nature Methods
 - EMBO Reports
 - PLOS ONE
 - PLOS Computational Biology
 - Cell
 - Nature Reviews Genetics
 - Trends in Genetics
 - Nature Biotechnology
 - eLife
 - Scientific Reports
 - Annals of the New York Academy of Sciences
 - Nature Genetics
 - Current Opinion in Genetics & Development
- View Less ^

Related Topics

- Biology
 - Genetics
 - Cell biology
- View More (17+) v

Multi-Sense Network Representation Learning in Microsoft Academic Graph

Established: February 22, 2016

Microsoft Academic \ Multi-Sense Network Representation Learning in Microsoft Academic Graph

Multi-Sense Network Representation Learning in Microsoft Academic Graph

March 5, 2020

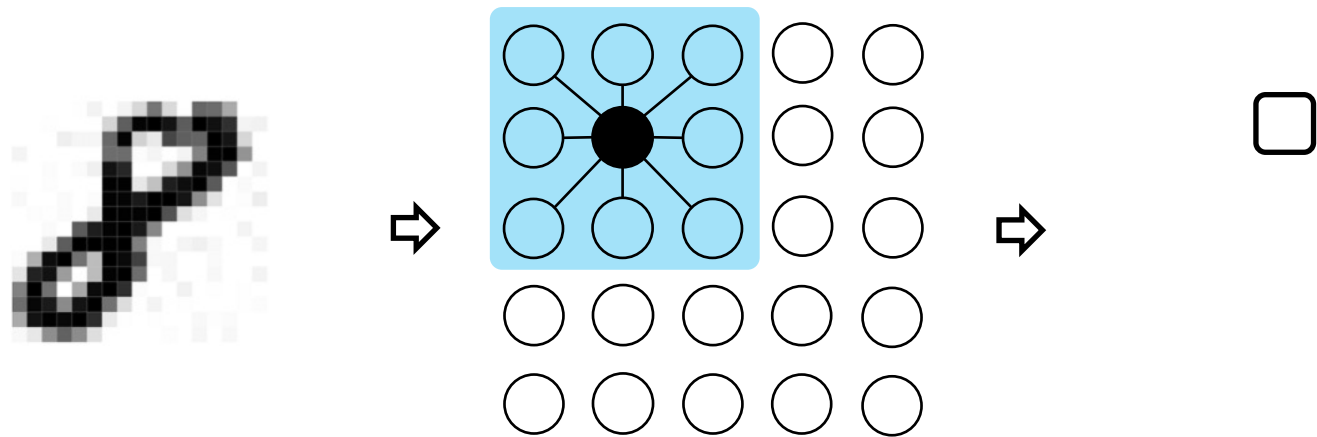


Over the past few years [deep representation learning](#) has revolutionized the developments of various domains, including [natural language processing \(NLP\)](#), [computer vision](#), and [speech](#). For example, in the NLP domain representation learning aims to learn [contextual embeddings for tokens/words](#) such that “words that occur in the same contexts tend to have similar meanings”. The distributional hypothesis that was first proposed by [Harris in 1954](#). The representation learning idea has also been extended to [networks](#), in which vertices that have the same structural contexts have the tendency to be similar.

Existing representation learning techniques use only one embedding vector for each token/node that may actually have different meanings under different contexts. This fundamental issue leads to the need of using more complicated models, such as [ELMo](#) and [Transformers](#), to try to recapture the contextual information for each customized context because one single vector is not enough to capture the contextual differences in both natural language and network structures. This issue could get worse when the network structures are organized in a heterogeneous way, which is the nature of the [Microsoft Academic Graph \(MAG\)](#), in which the structural contexts are naturally diverse in observation of different types of entities and their relationships.

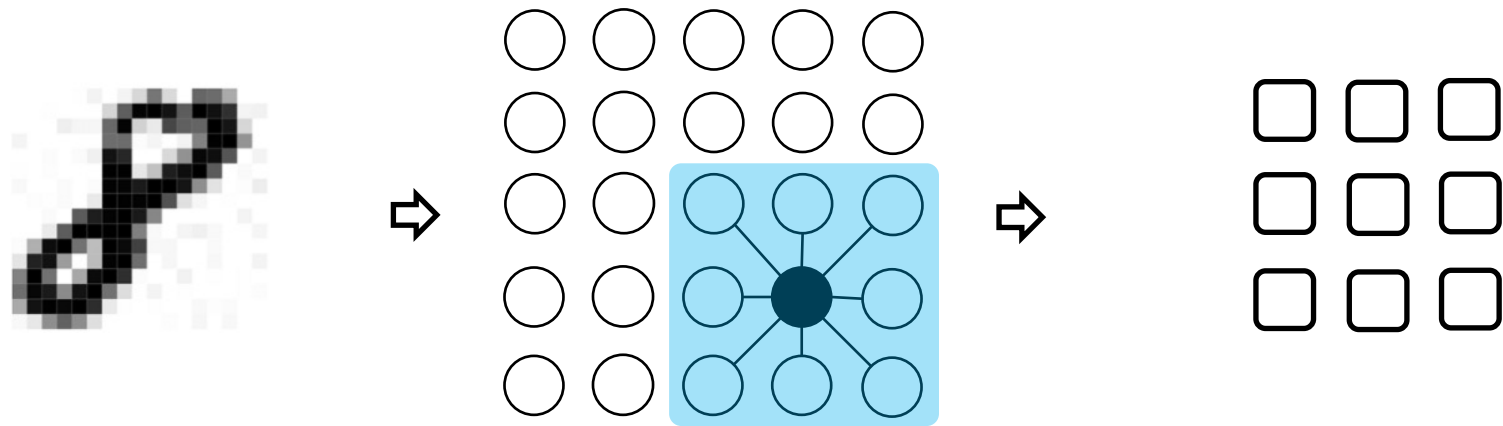
For additional context it's important to review how representation learning has shaped [network mining](#) and to demonstrate why one embedding vector is not enough to model different structural contexts in MAG. The traditional paradigm of [mining and learning with networks](#) usually begins with the discovery of networks' structural properties. With these structural properties extracted as features, [machine learning](#) algorithms can be applied for various of applications. Often, however, the characterization of these features involves domain knowledge and expensive computation. The emergence of [representation learning on networks](#) offers new perspectives to address this issue by translating discrete and structural symbols into continuous representations such as low-dimensional vectors, that computers can “understand” and process algebraically.

Neural Networks



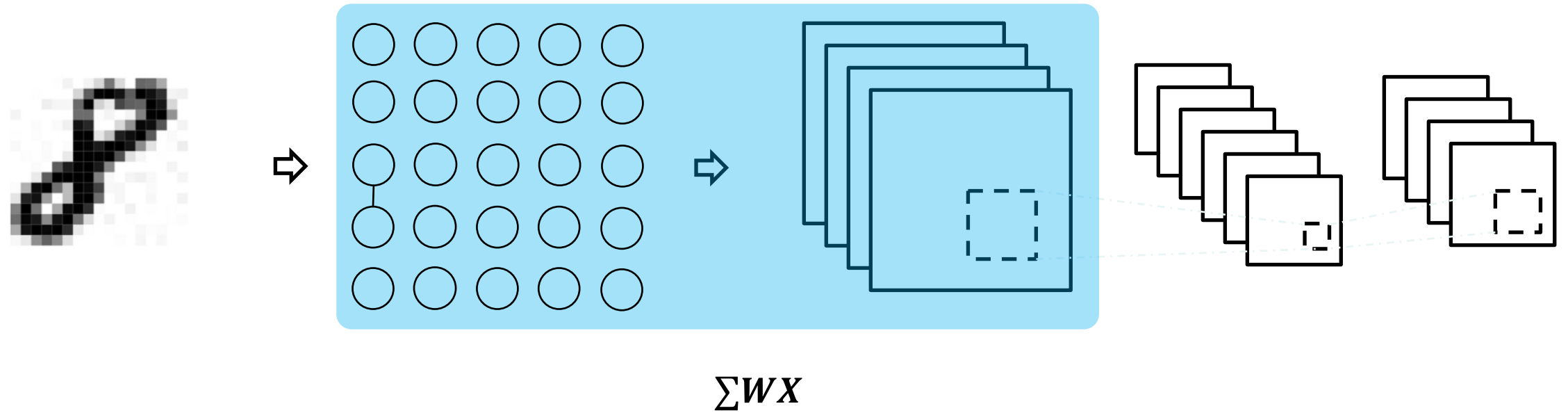
$$\sum W X$$

Neural Networks



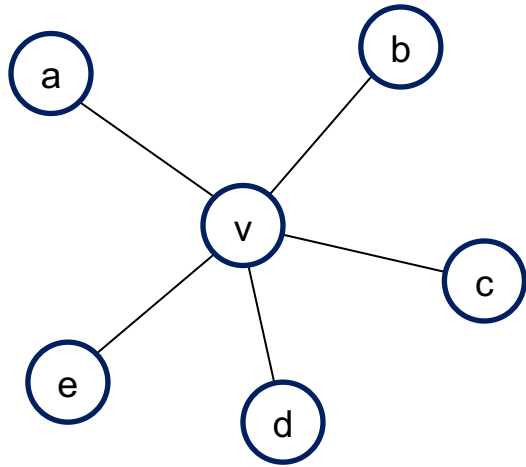
$$\sum W X$$

Neural Networks



It is straightforward to define convolutions over images with fixed 2D structures

Graph Neural Networks



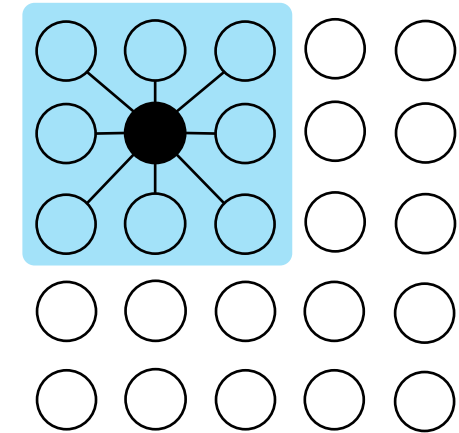
1. Choose neighborhood
2. Determine the order of selected neighbors
3. Parameter sharing

Graph Convolution

Neighborhood Aggregation:

- Iteratively aggregate neighbor information and pass into a neural network
- It can be viewed as a center-surround filter in CNN---graph convolutions!

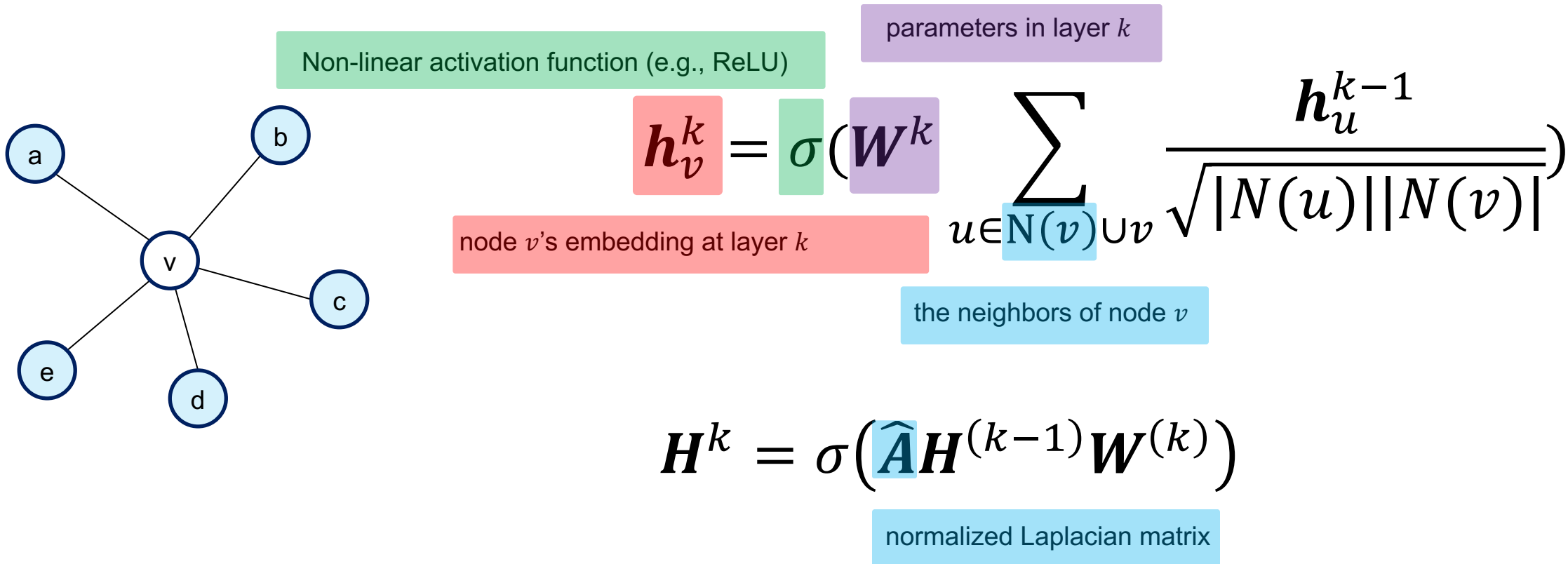
$$H^l[t] \leftarrow \underset{\forall s \in N(t), \forall e \in E(s,t)}{\text{Aggregate}} \left(\text{Extract} \left(H^{l-1}[s]; H^{l-1}[t], e \right) \right)$$



CNN

1. Niepert et al. Learning Convolutional Neural Networks for Graphs. In **ICML 2016**
2. Defferrard et al. Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. In **NIPS 2016**

Graph Convolutional Networks



Aggregate info from neighborhood via the normalized Laplacian matrix

Graph Attention

$$\text{GCN } \mathbf{h}_v^k = \sigma \left(\mathbf{W}^k \sum_{u \in N(v) \cup v} \frac{\mathbf{h}_u^{k-1}}{\sqrt{|N(u)||N(v)|}} \right)$$

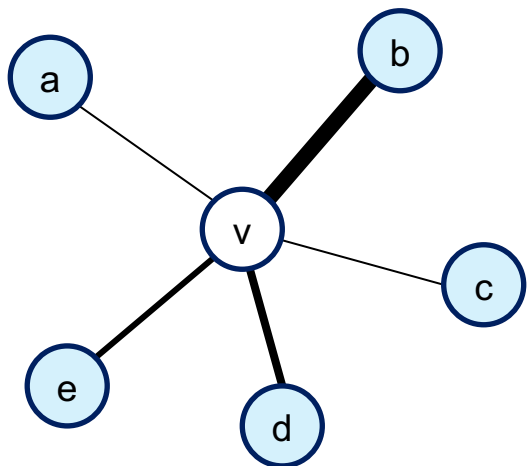
Aggregate info from neighborhood via the normalized Laplacian matrix

Graph Attention

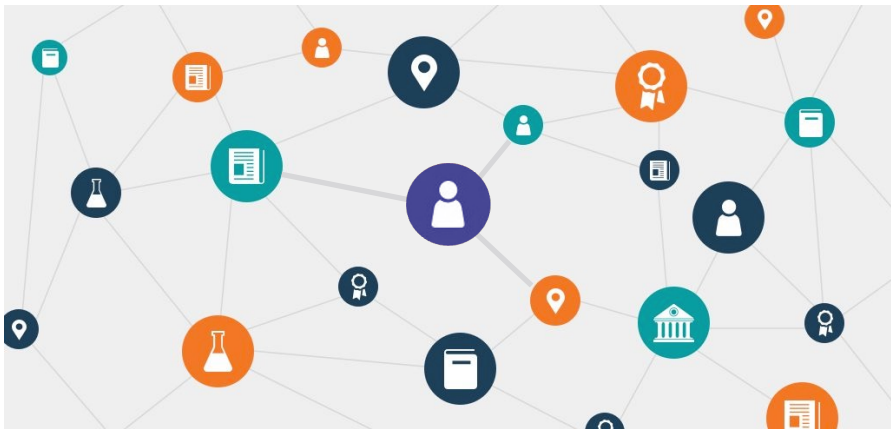
$$\mathbf{h}_v^k = \sigma \left(\sum_{u \in N(v) \cup v} \alpha_{v,u} \mathbf{W}^k \mathbf{h}_u^{k-1} \right)$$

$$\alpha_{v,u} = \frac{\exp(\text{LeakyReLU}(\mathbf{a}^\top [\mathbf{Q}\mathbf{h}_v, \mathbf{Q}\mathbf{h}_u]))}{\sum_{u' \in N(v) \cup \{v\}} \exp(\text{LeakyReLU}(\mathbf{a}^\top [\mathbf{Q}\mathbf{h}_v, \mathbf{Q}\mathbf{h}_{u'}]))}$$

Aggregate info from neighborhood via the learned attention



Heterogeneous Graphs?



Academic Graph



Microsoft Office Graph



LinkedIn Economic Graph



Facebook Entity Graph

Heterogeneous Graph Attention?

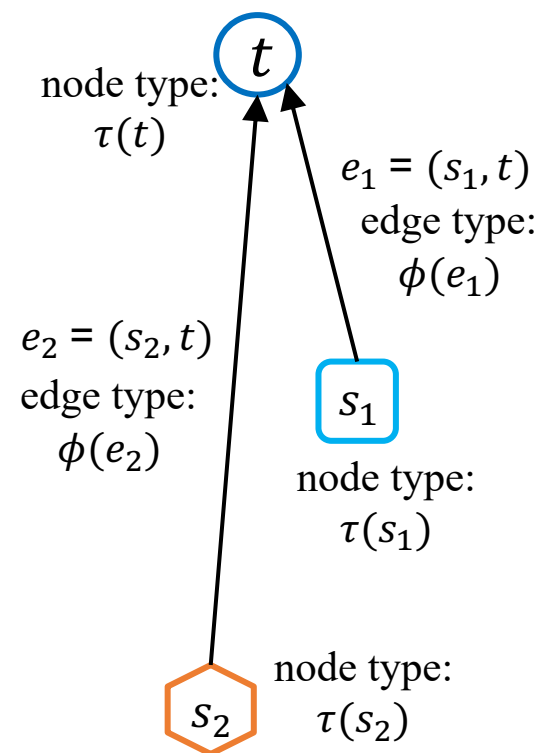
- Define unique parameters for each type of nodes & edges
- Parameterize attention & message passing weights according to the meta relation of each edge

- meta relation of an edge $e = (s, t)$

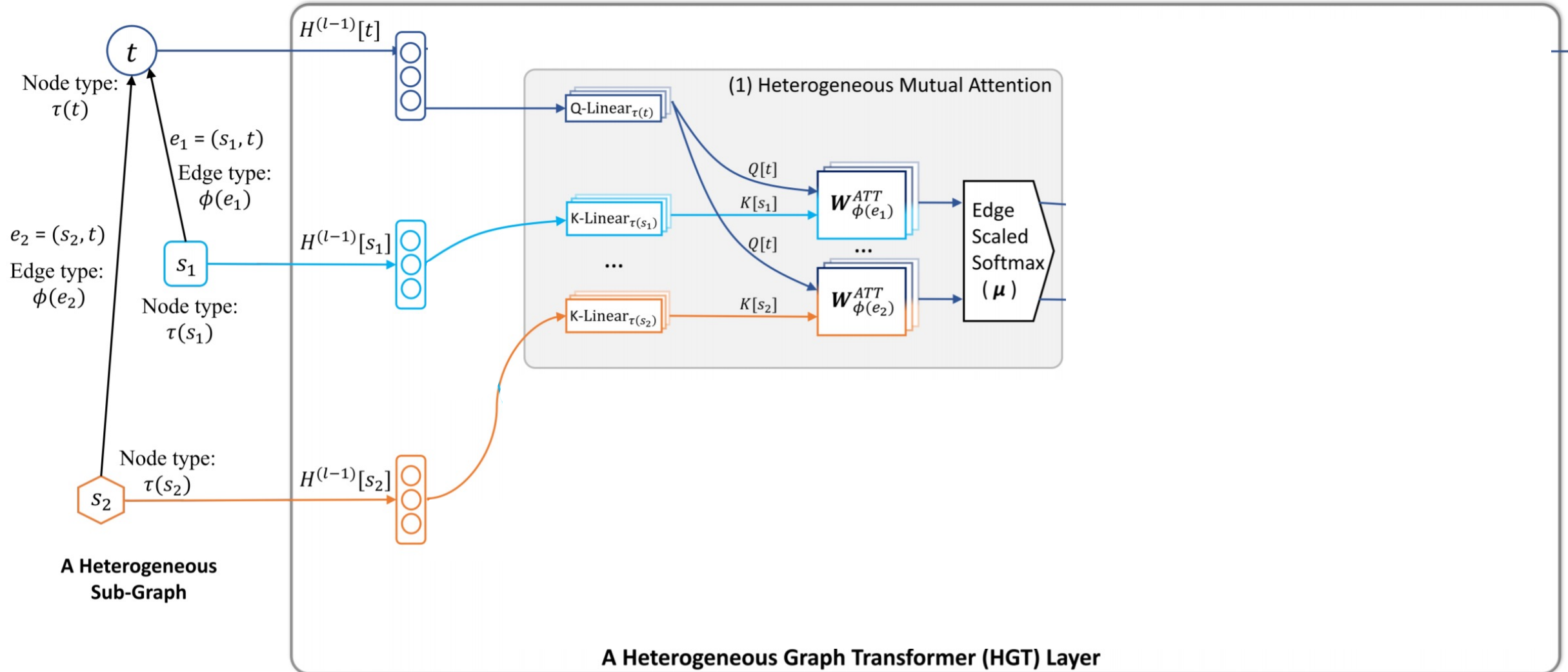
$$\langle \tau(s), \phi(e), \tau(t) \rangle$$



- $\langle \text{author, write, paper} \rangle$



Heterogeneous Graph Transformer (HGT)



Heterogeneous Graph Transformer (HGT)



meta relation of $e = (s, t)$

$$\langle \tau(s), \phi(e), \tau(t) \rangle$$

- **heterogeneous mutual attention**

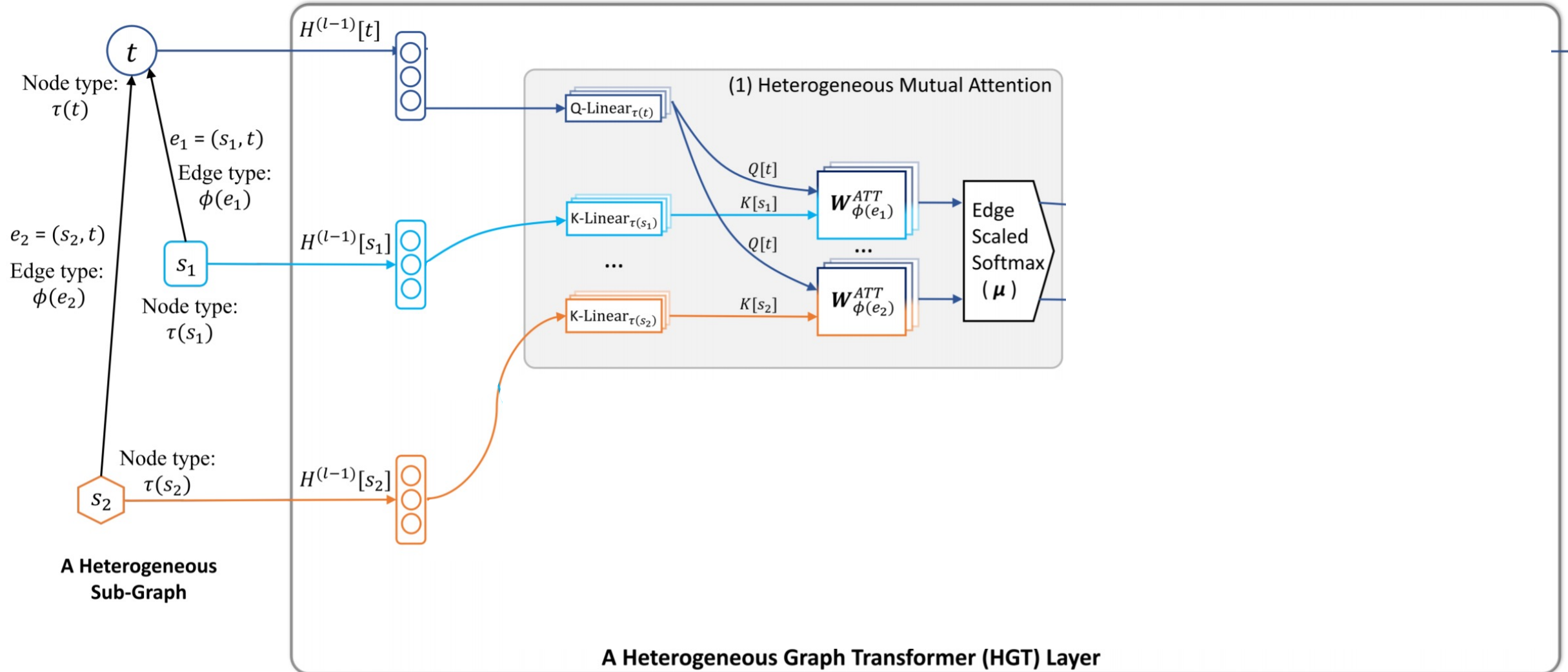
$$\mathbf{Attention}_{HGT}(s, e, t) = \text{Softmax}_{\forall s \in N(t)} \left(\parallel_{i \in [1, h]} \mathbf{ATT-head}^i(s, e, t) \right) \quad (3)$$

$$\mathbf{ATT-head}^i(s, e, t) = \left(K^i(s) W_{\phi(e)}^{ATT} Q^i(t)^T \right) \cdot \frac{\mu \langle \tau(s), \phi(e), \tau(t) \rangle}{\sqrt{d}}$$

$$K^i(s) = \text{K-Linear}_{\tau(s)}^i \left(H^{(l-1)}[s] \right)$$

$$Q^i(t) = \text{Q-Linear}_{\tau(t)}^i \left(H^{(l-1)}[t] \right)$$

Heterogeneous Graph Transformer (HGT)



Heterogeneous Graph Transformer (HGT)



meta relation of $e = (s, t)$

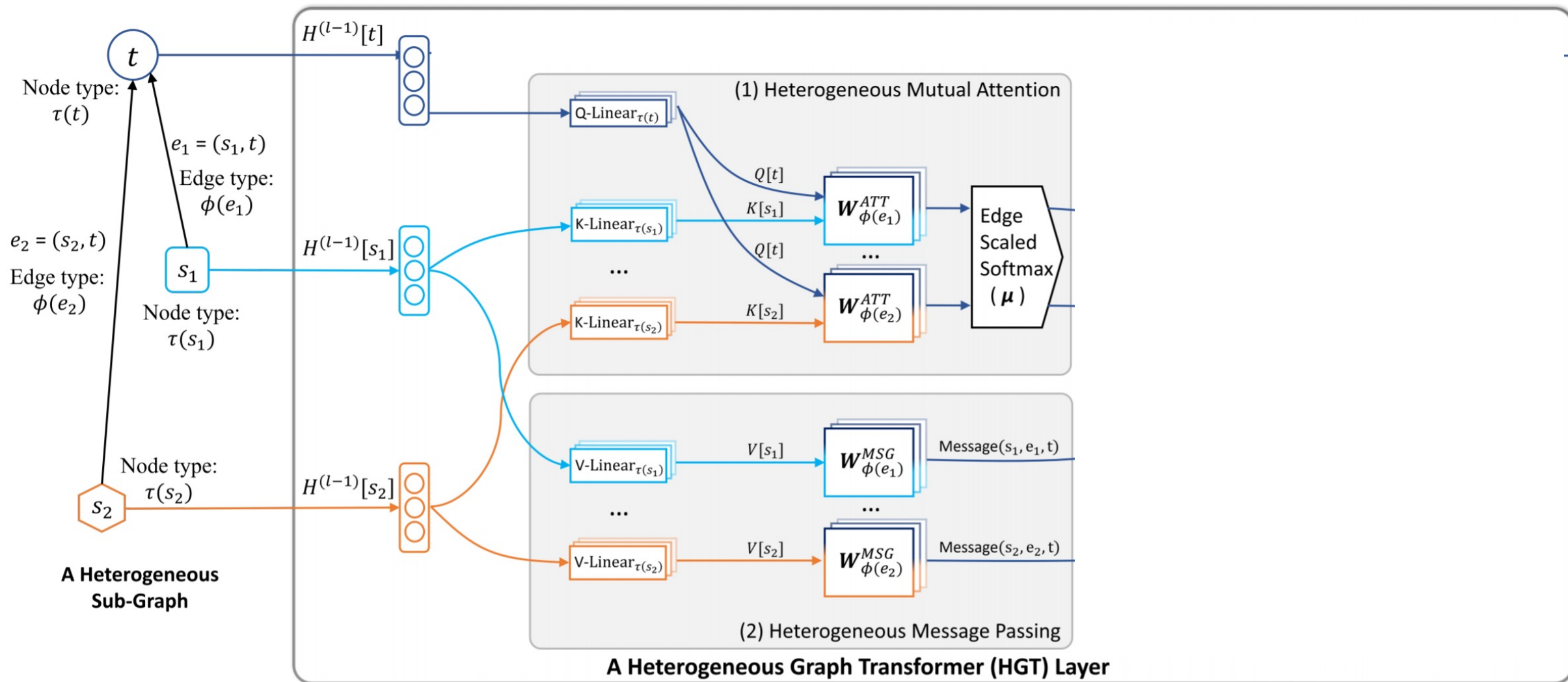
$\langle \tau(s), \phi(e), \tau(t) \rangle$

- **heterogeneous message passing**

$$\mathbf{Message}_{HGT}(s, e, t) = \parallel_{i \in [1, h]} \text{MSG-head}^i(s, e, t)$$

$$\text{MSG-head}^i(s, e, t) = \text{V-Linear}_{\tau(s)}^i \left(H^{(l-1)}[s] \right) W_{\phi(e)}^{MSG}$$

Heterogeneous Graph Transformer (HGT)



Heterogeneous Graph Transformer (HGT)



meta relation of $e = (s, t)$

$\langle \tau(s), \phi(e), \tau(t) \rangle$

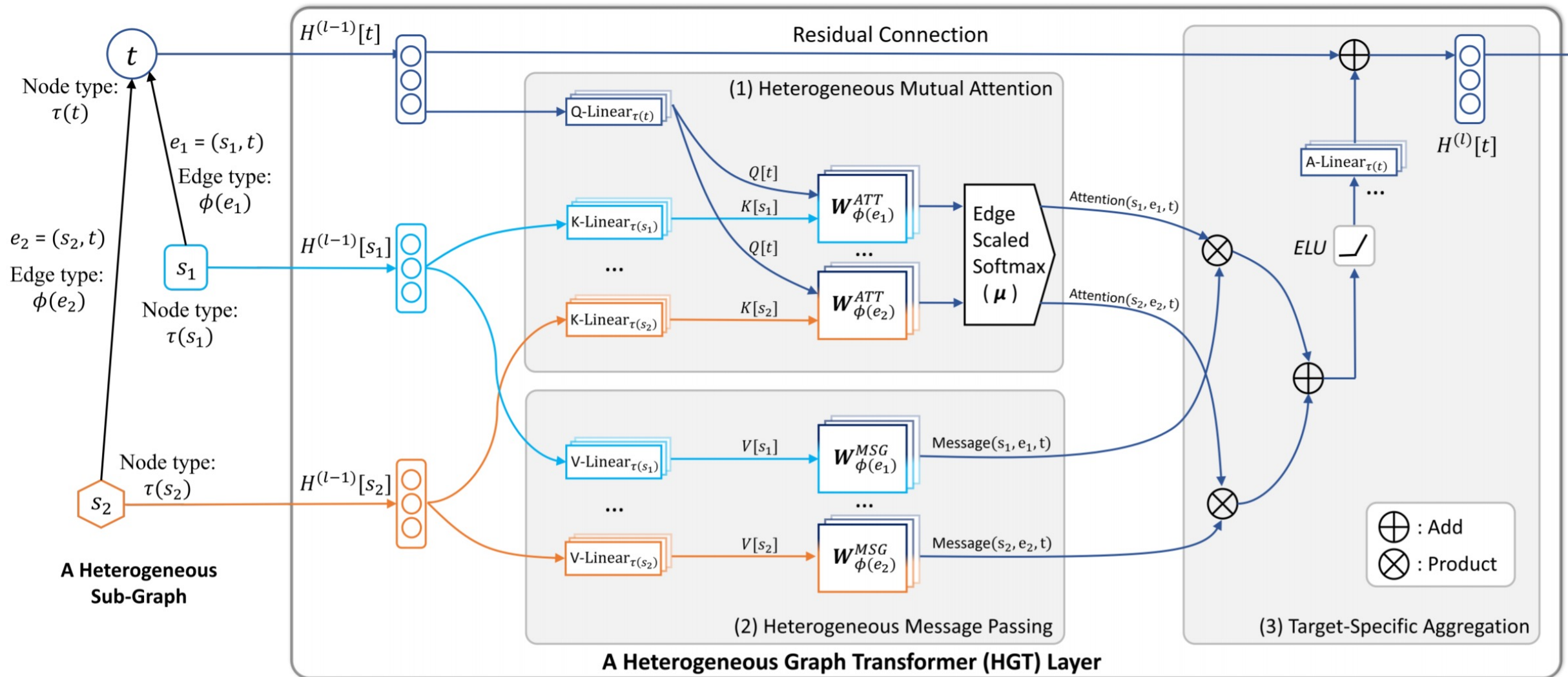
- **Target specific aggregation**

$$\tilde{H}^{(l)}[t] = \bigoplus_{\forall s \in N(t)} \left(\mathbf{Attention}_{HGT}(s, e, t) \cdot \mathbf{Message}_{HGT}(s, e, t) \right)$$

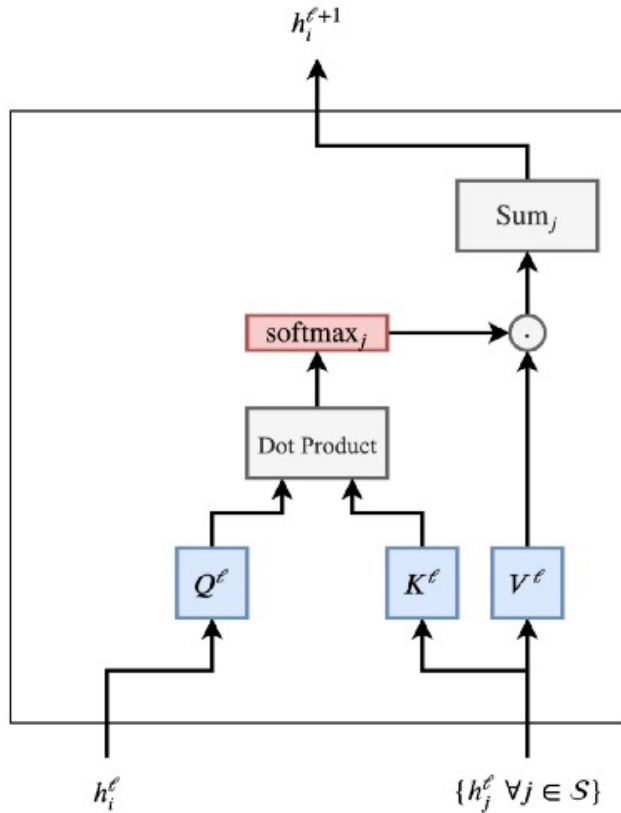
$$H^{(l)}[t] = \text{A-Linear}_{\tau(t)} \left(\sigma(\tilde{H}^{(l)}[t]) \right) + H^{(l-1)}[t]$$

$$\mathbf{Attention}_{HGT}(s, e, t) = \text{Softmax}_{\forall s \in N(t)} \left(\parallel_{i \in [1, h]} \mathbf{ATT-head}^i(s, e, t) \right)$$

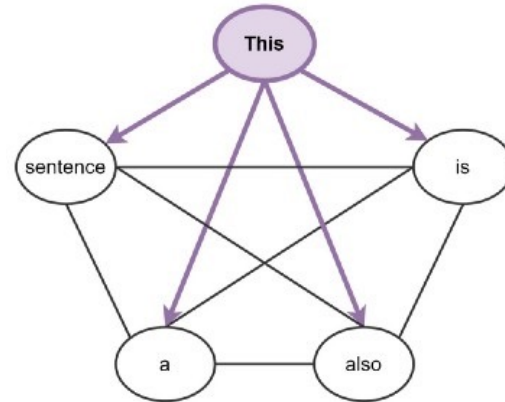
Heterogeneous Graph Transformer (HGT)



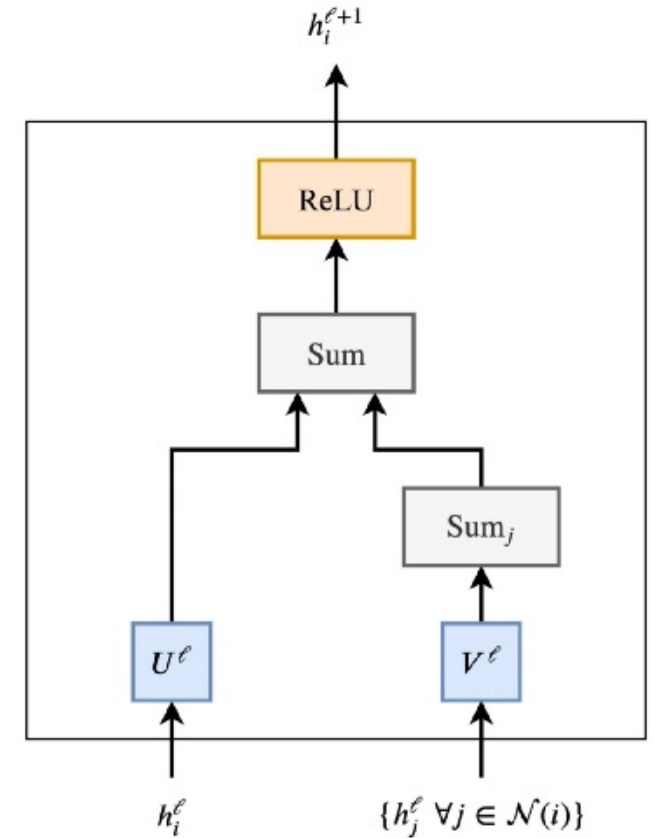
Transformer as GNNs



Transformer



- Translation?
- Sentiment?
- Next word?
- Part-of-speech tags?



Graph Neural Nets

Dynamic Heterogeneous Graphs?



Academic Graph



Microsoft Office Graph



LinkedIn Economic Graph



Facebook Entity Graph

Graph Dynamics

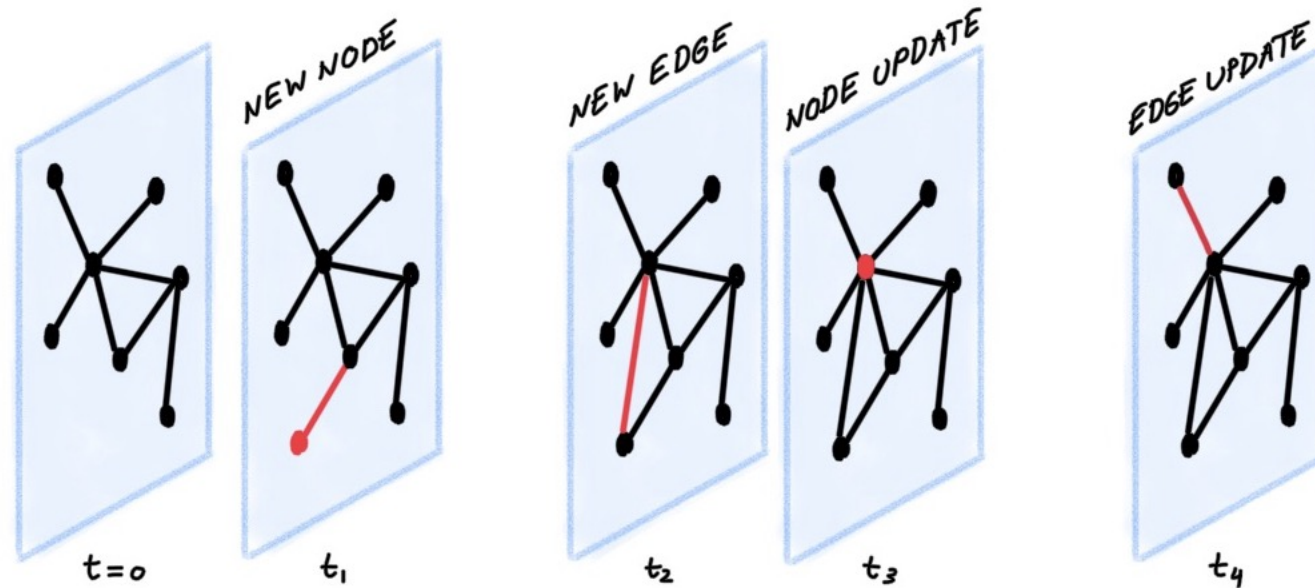
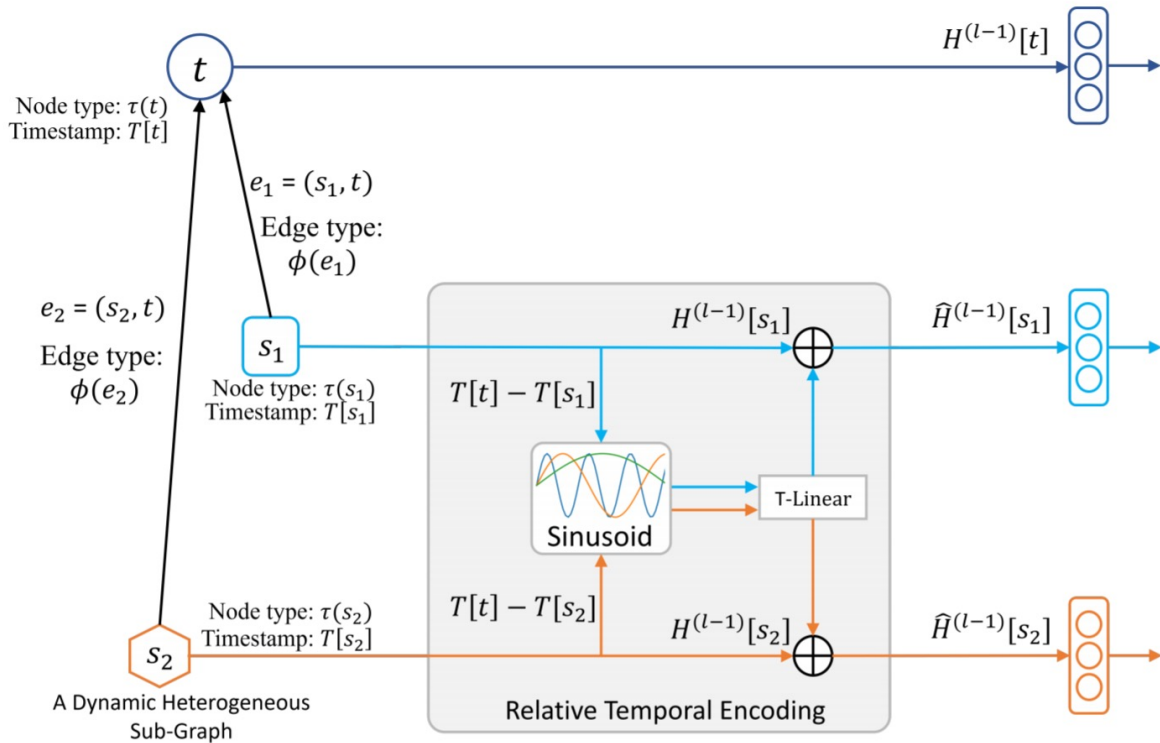


Figure Credit: [Michael Bronstein](#)

The common strategy: Slice the dynamic graph into multiple timestamps

Relative Temporal Encoding (RTE) in HGT



$$\hat{H}^{(l-1)}[s] = H^{(l-1)}[s] + RTE(\Delta T(t, s))$$

$$RTE(\Delta T(t, s)) = \text{T-Linear}\left(\text{Base}(\Delta T_{t,s})\right)$$

$$\text{Base}(\Delta T(t, s), 2i) = \sin\left(\Delta T_{t,s} / 10000^{\frac{2i}{d}}\right)$$

$$\text{Base}(\Delta T(t, s), 2i + 1) = \cos\left(\Delta T_{t,s} / 10000^{\frac{2i+1}{d}}\right)$$

- Maintain all edges in different timestamps

Billion-Scale Dynamic Heterogeneous Graphs?



Academic Graph



Microsoft Office Graph



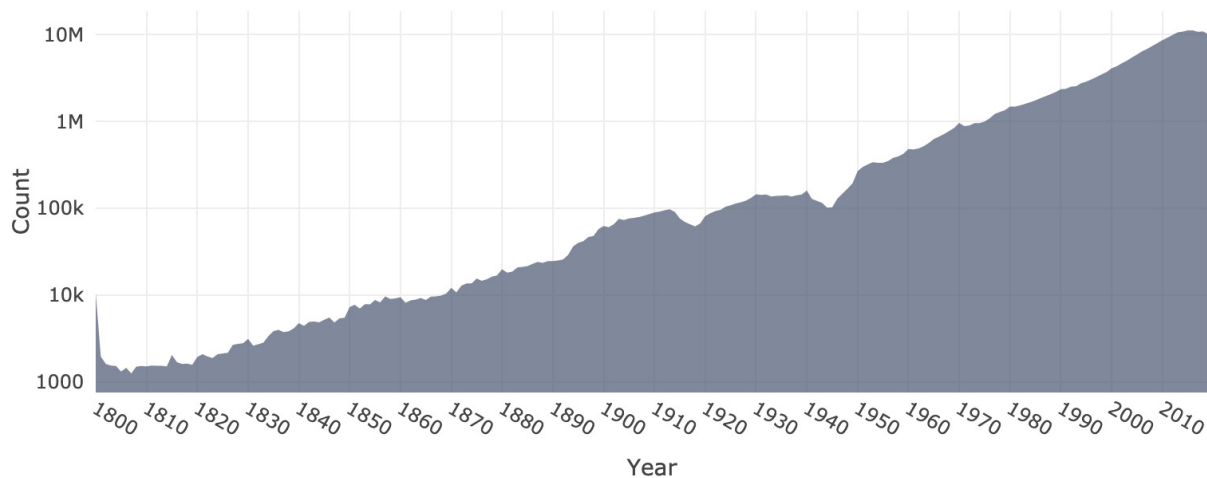
LinkedIn Economic Graph



Facebook Entity Graph

Experiments

AMiner & Microsoft Academic Graph



HGT



- Infer paper field
- Infer paper venue
- Name disambiguation
- Fake paper detection
- ...

Results

GNN Models		GCN [7]	RGCN [12]	GAT [21]	HetGNN [25]	HAN [22]	HGT _{noHeter}	HGT _{noTime}	HGT
# of Parameters		1.69M	8.80M	1.69M	8.41M	9.45M	3.12M	7.44M	8.20M
Paper-Field (L1)	NDCG	.558±.141	.563±.128	.601±.103	.615±.084	.617±.096	.674±.086	.702±.089	.735±.084
	MRR	.513±.136	.526±.105	.587±.096	.595±.076	.604±.092	.652±.078	.676±.082	.713±.081
Paper-Field (L2)	NDCG	.241±.074	.258±.046	.276±.049	.271±.062	.281±.051	.301±.046	.307±.052	.332±.048
	MRR	.192±.067	.206±.052	.228±.045	.231±.053	.242±.049	.257±.058	.260±.064	.276±.071
Paper-Venue	NDCG	.303±.066	.354±.051	.461±.057	.447±.071	.478±.062	.515±.059	.538±.064	.551±.062
	MRR	.114±.070	.198±.047	.244±.052	.226±.059	.269±.067	.295±.060	.322±.048	.334±.061
Author	NDCG	.730±.064	.742±.057	.785±.063	.792±.052	.810±.049	.834±.058	.849±.066	.857±.054
Disambiguation	MRR	.762±.042	.786±.048	.843±.044	.852±.058	.876±.056	.903±.041	.911±.043	.918±.048

HGT offers ~9–21% improvements over existing (heterogeneous) GNNs

Case Study

Experiments done w/o 2020 data!

Venue	Time	Top-5 Most Similar Venues
WWW	2000	SIGMOD, VLDB, NSDI, GLOBECOM, SIGIR
	2010	GLOBECOM, KDD, CIKM, SIGIR, SIGMOD
	2020	KDD, GLOBECOM, SIGIR, WSDM, SIGMOD
KDD	2000	SIGMOD, ICDE, ICDM, CIKM, VLDB
	2010	ICDE, WWW, NeurIPS, SIGMOD, ICML
	2020	NeurIPS, SIGMOD, WWW, AAAI, EMNLP
NeurIPS	2000	ICCV, ICML, ECCV, AAAI, CVPR
	2010	ICML, CVPR, ACL, KDD, AAAI
	2020	ICML, CVPR, ICLR, ICCV, ACL

DB + Networking + IR



DM + Networking + IR + DB

DB + DM



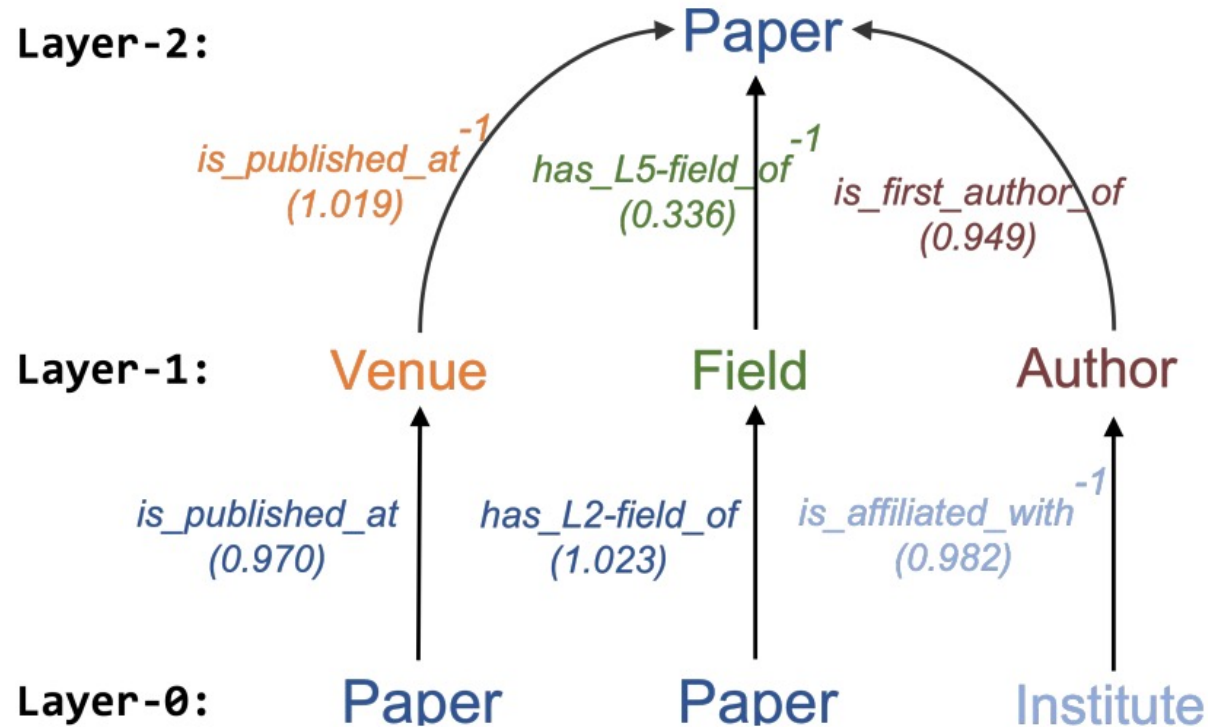
ML + DB + Web + AI + NLP!!!

CV + ML + AI



ML + CV + DL + NLP

What is the Best Part of HGT?



Learn meta-paths & their weights implicitly and automatically!

Powering the Microsoft Office Graph



One enterprise graph (monthly)

- 1.6 billion entities
 - 7 types of entities
- 7.8 trillion edges

Anomaly detection on Microsoft Office Graph

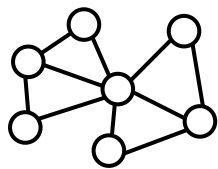
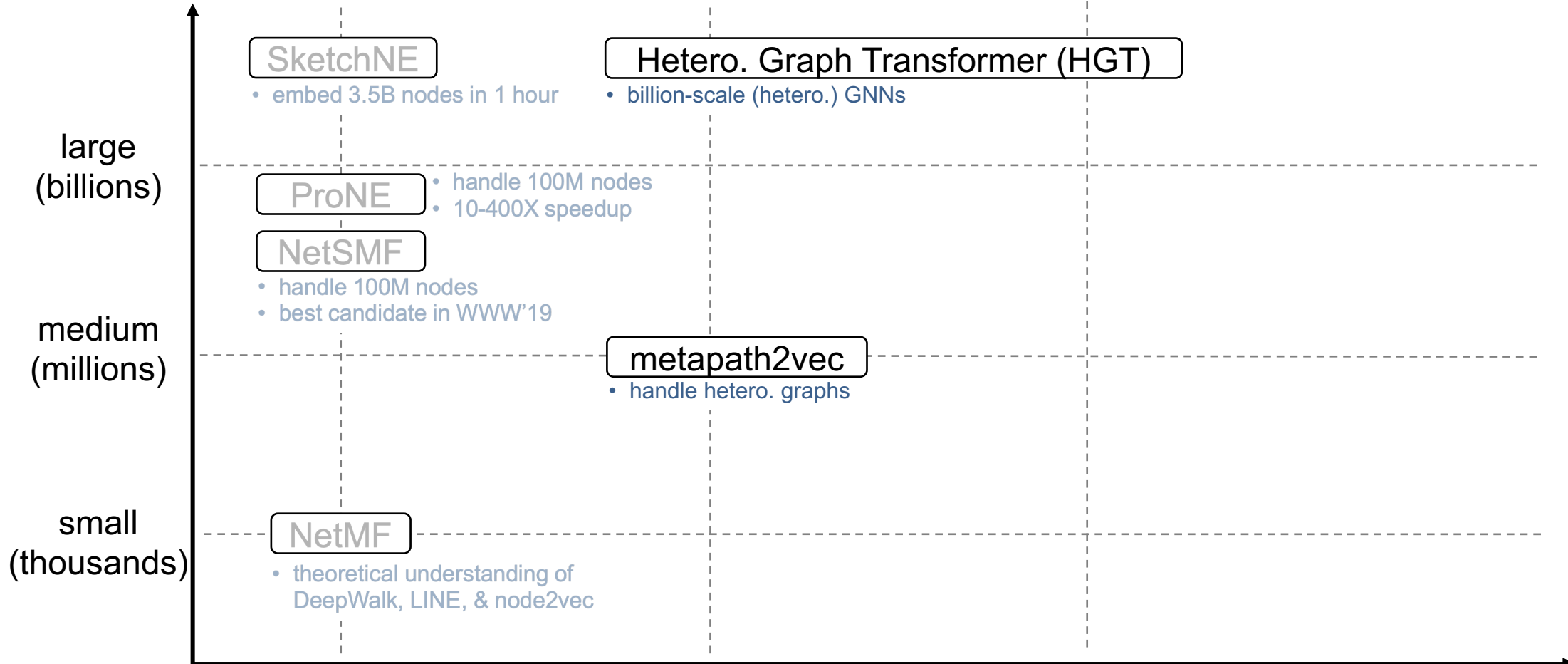
	Prec.	Recall	F1	Accu.
GraphSage	+0.00	+0.09	+0.06	+0.03
Graph Attention	+0.01	+0.11	+0.08	+0.03
HGT	+0.01	+0.30	+0.19	+0.07

Pre-trained
HGT on
one
enterprise

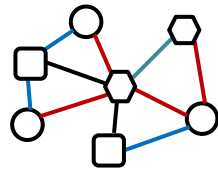


Other
enterprise
customers w/o
data access

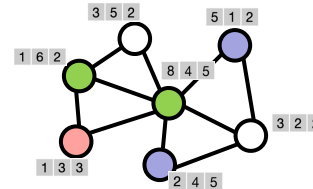
Heterogeneous Graphs



structure

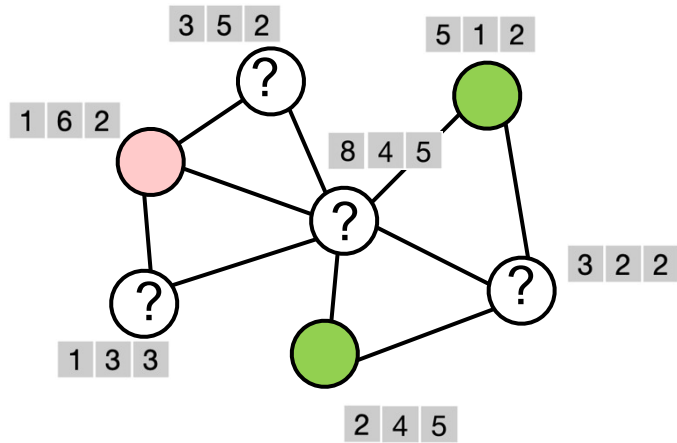


heterogeneous structure / knowledge graph

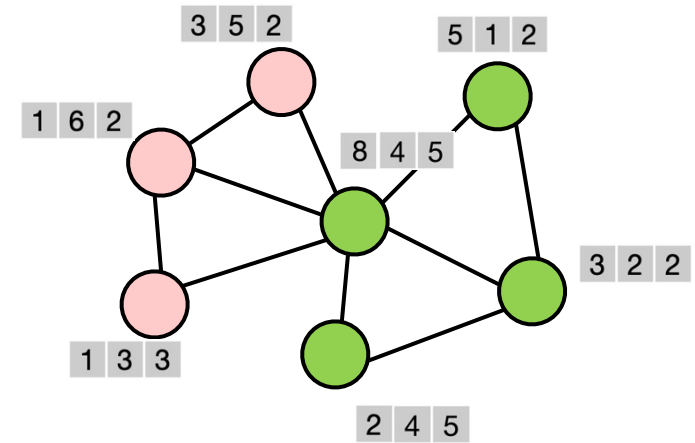
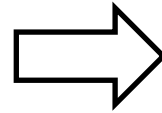


structure with features

Semi-Supervised Learning on Graphs

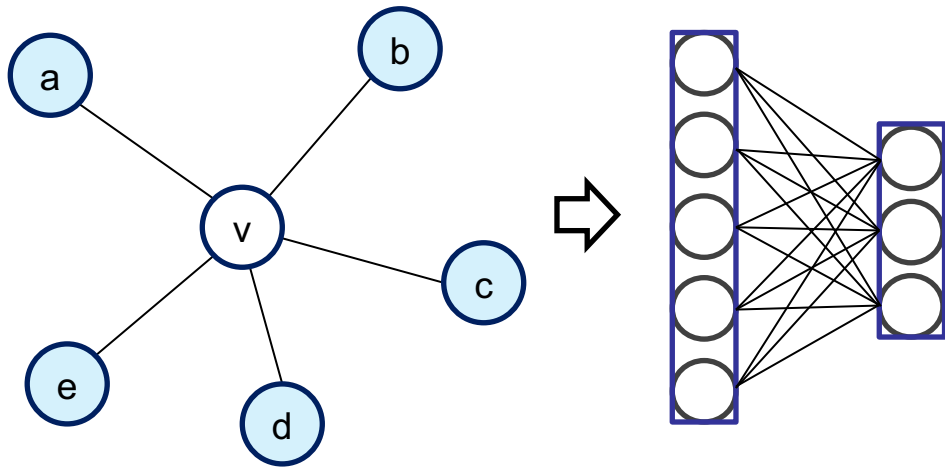


Input: a partially labeled & attributed graph



Output: infer the labels of unlabeled nodes

Graph Neural Networks (GNNs)



node v 's embedding at $k + 1$

non-linear activation function (e.g. ReLU)

$$\mathbf{H}^{k+1} = \sigma(\hat{\mathbf{A}}\mathbf{H}^{(k)}\mathbf{W}^{(k)})$$

normalized Laplacian matrix

$$\mathbf{H}^{k+1} = \sigma\left(\mathbf{W}^k \sum_{u \in N(v) \cup v} \frac{\mathbf{H}_u^k}{\sqrt{|N(u)||N(v)|}}\right)$$

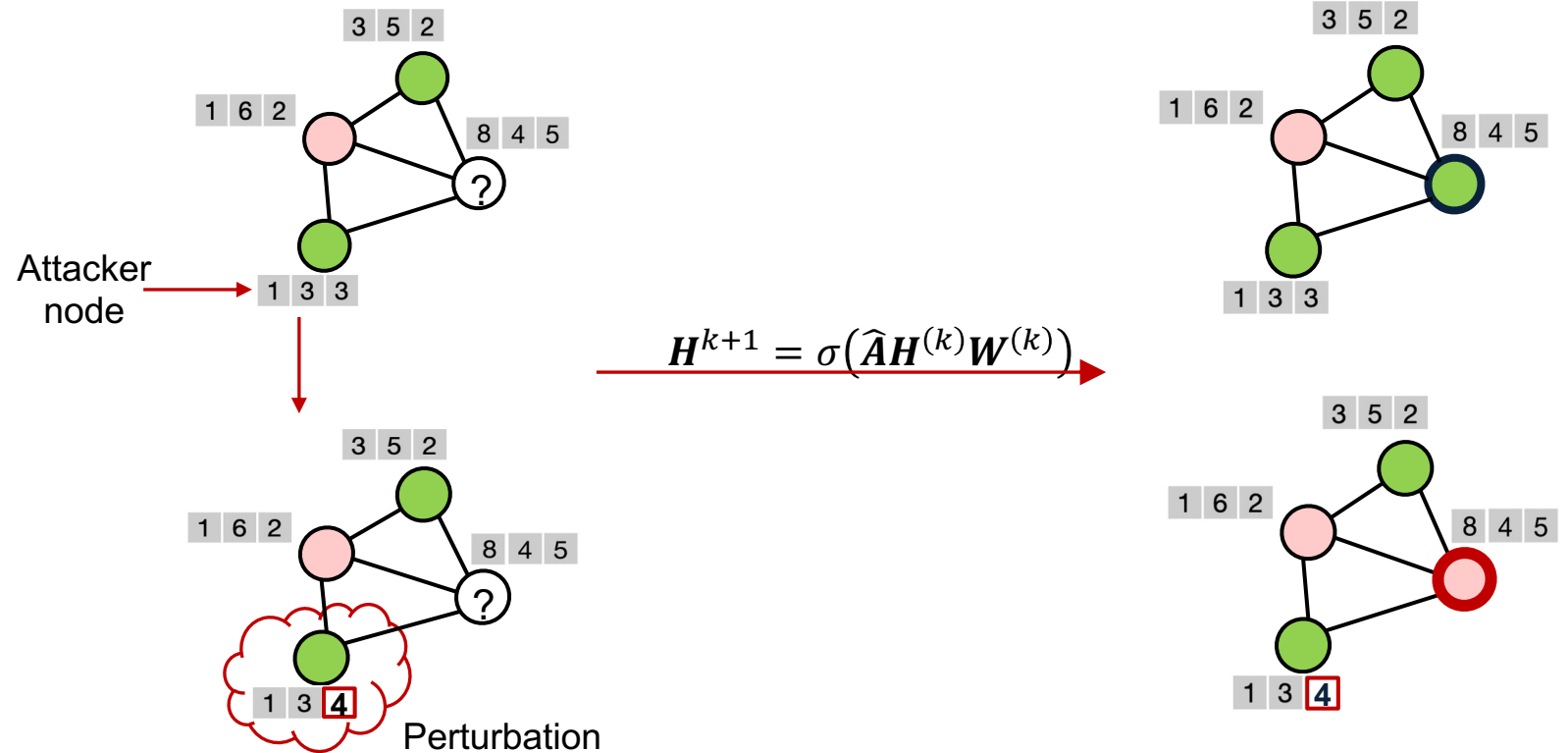
the neighbors of node v

Graph Neural Networks

- **Non-Robust:** each node is highly dependent with its neighbors, making GNNs **non-robust** to noises

$$H^{k+1} = \sigma(\widehat{A}H^{(k)}W^{(k)})$$

a deterministic propagation



Graph Neural Networks

- **Non-Robust:** each node is highly dependent with its neighbors, making GNNs **non-robust** to noises
- **Over-Smoothing:** stacking many GNNs layers may cause **over-smoothing**

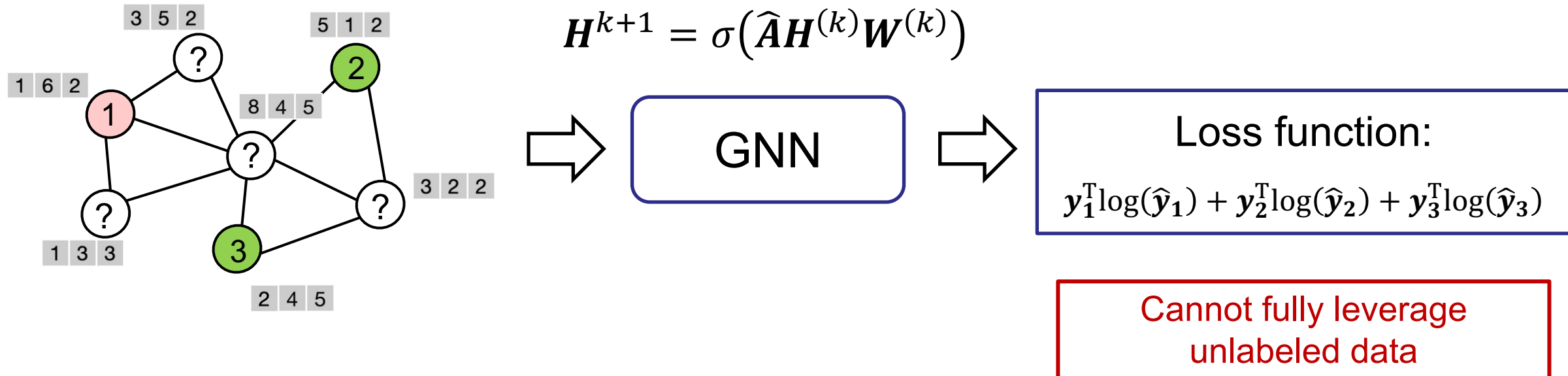
$$H^{k+1} = \sigma(\widehat{A}H^{(k)}W^{(k)})$$

feature propagation is
Laplacian smoothing,
coupled with
non-linear transformation

Graph Neural Networks

- **Non-Robust:** each node is highly dependent with its neighbors, making GNNs **non-robust** to noises
- **Over-Smoothing:** stacking many GNNs layers may cause **over-smoothing**
- **Over-Fitting:** under semi-supervised settings, standard training is easy to **over-fit** the scarce labels

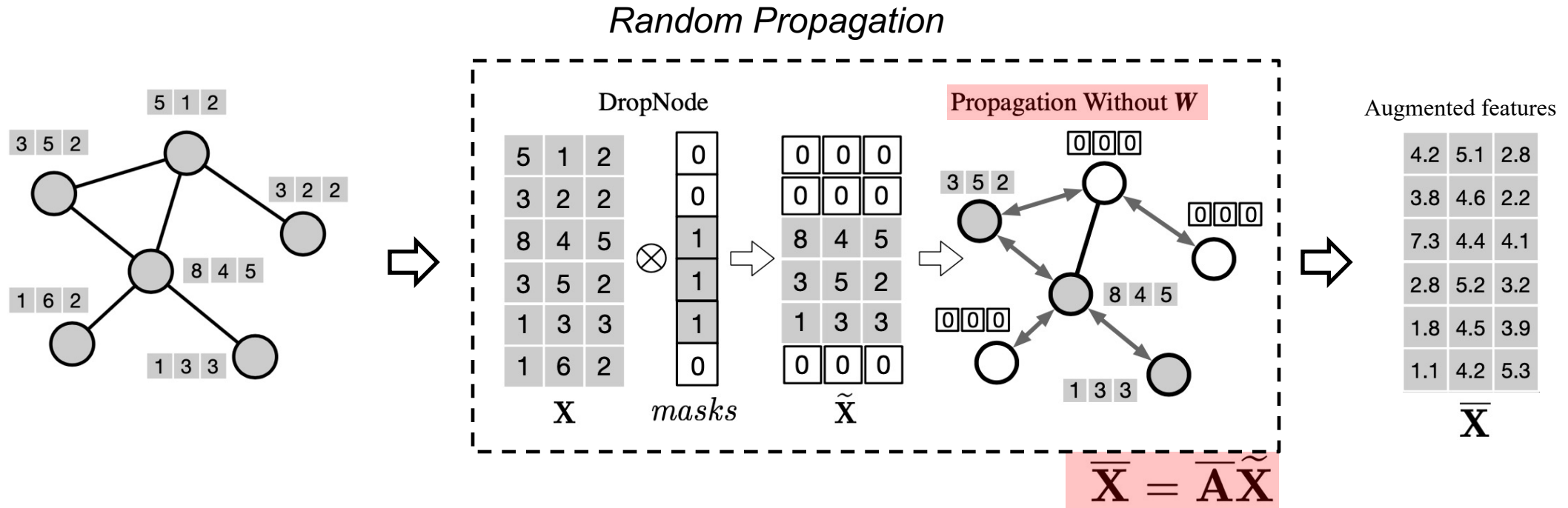
The standard training flow for GNNs:



Graph Random Neural Network (GRAND)

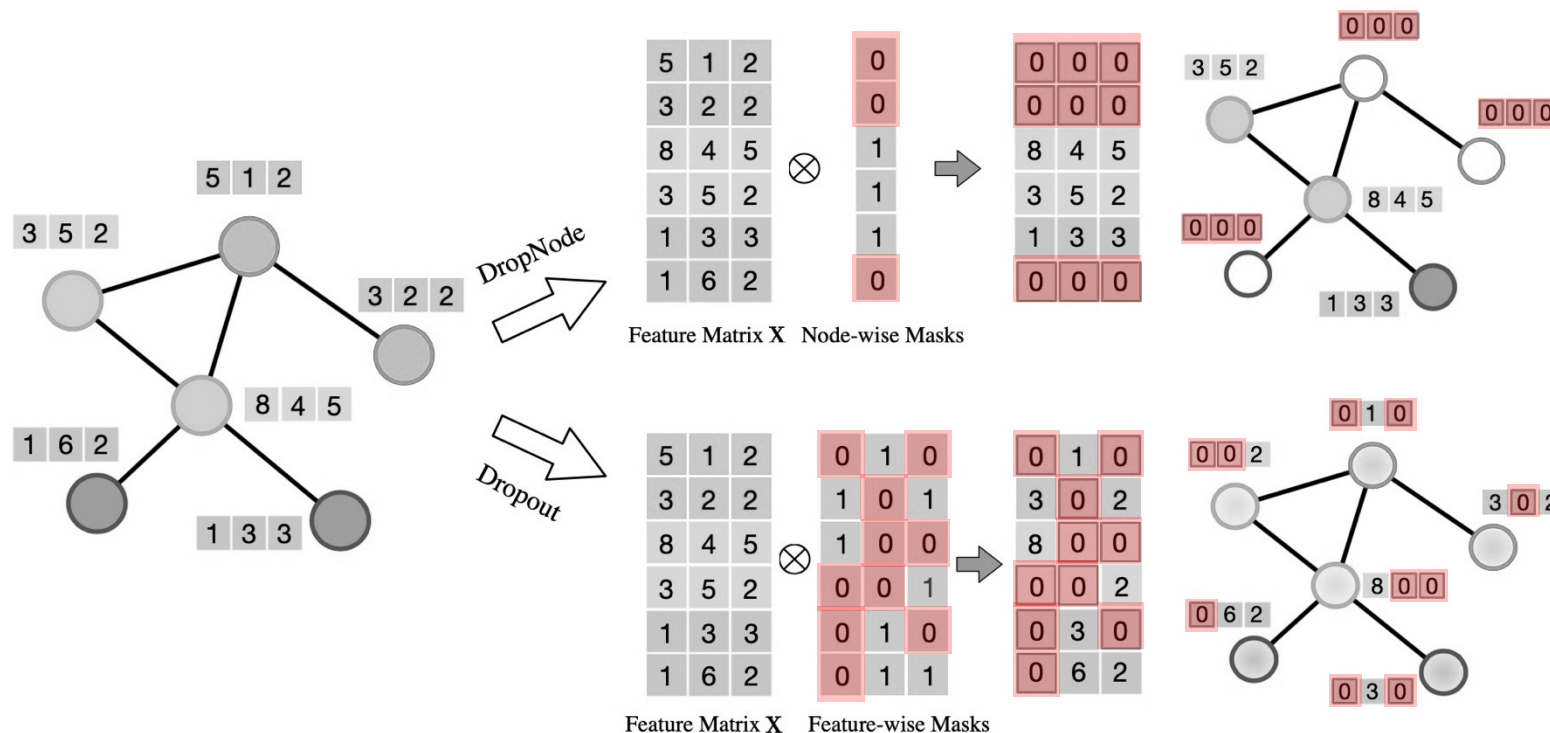
- **Random Propagation (DropNode + Propagation):**

- **Enhancing robustness:** Each node is enabled to be not sensitive to specific neighborhoods.
- **Mitigating over-smoothing and overfitting:** Decouple feature propagation from feature transformation.



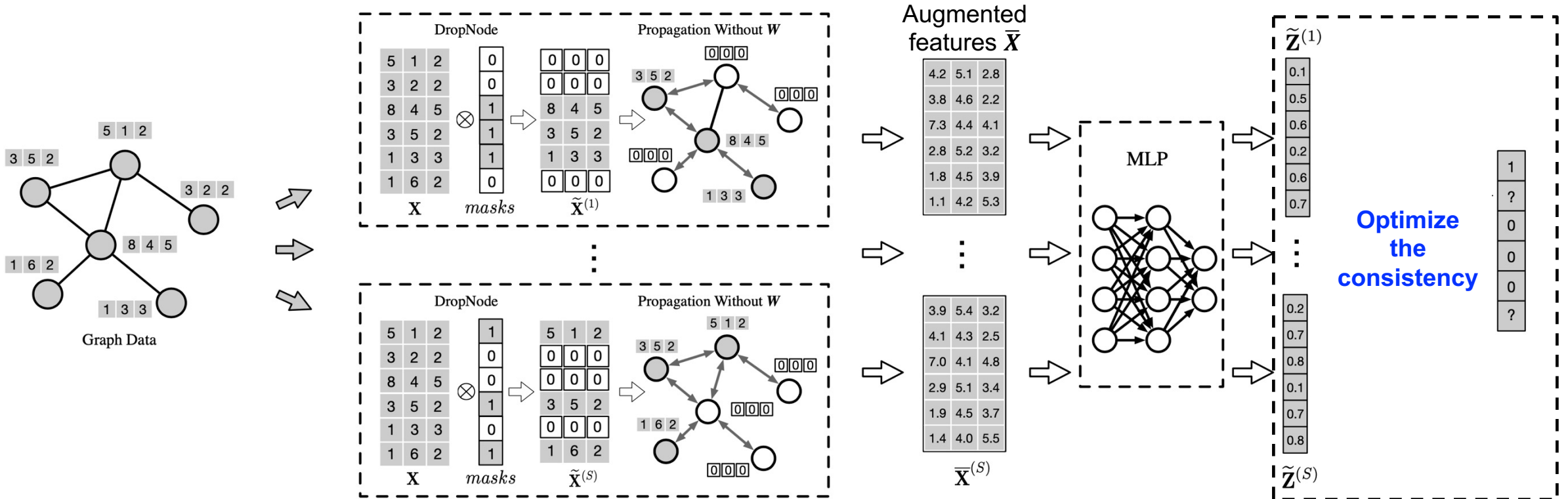
Random Propagation: DropNode vs Dropout

- Dropout drops each element in X independently
- DropNode drops the entire features of selected nodes, i.e., the row vectors of X , randomly



Graph Random Neural Network (GRAND)

- Consistency Regularized Training:
 - Generates S data augmentations of the graph
 - Optimizing the consistency among S augmentations of the graph.



Random Propagation as data augmentation

S Augmentations

Consistency Regularization ?

GRAND: Consistency Regularization

Distributions of a node
after augmentations

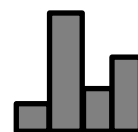


Average

$$\bar{\mathbf{Z}}_i = \frac{1}{S} \sum_{s=1}^S \tilde{\mathbf{Z}}_i^{(s)}$$



Sharpening



$$\bar{\mathbf{Z}}'_{ik} = \bar{\mathbf{Z}}_{ik}^{\frac{1}{T}} \Bigg/ \sum_{j=0}^{C-1} \bar{\mathbf{Z}}_{ij}^{\frac{1}{T}}$$

$$\mathcal{L}_{sup} = -\frac{1}{S} \sum_{s=1}^S \sum_{i=0}^{m-1} \mathbf{Y}_i^{\top} \log \tilde{\mathbf{Z}}_i^{(s)}$$

+

⇒

$$\mathcal{L} = \mathcal{L}_{sup} + \lambda \mathcal{L}_{con}$$

$$\mathcal{L}_{con} = \frac{1}{S} \sum_{s=1}^S \sum_{i=0}^{n-1} \mathcal{D}(\bar{\mathbf{Z}}'_i, \tilde{\mathbf{Z}}_i^{(s)})$$

↕

Graph Random Neural Network (GRAND)

Input:

Adjacency matrix $\hat{\mathbf{A}}$, feature matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$, times of augmentations in each epoch S , DropNode probability δ .

Output:

Prediction \mathbf{Z} .

1: **while** not convergence **do**

2: **for** $s = 1 : S$ **do**

3: Apply DropNode via Algorithm 1: $\tilde{\mathbf{X}}^{(s)} \sim \text{DropNode}(\mathbf{X}, \delta)$.

4: Perform propagation: $\bar{\mathbf{X}}^{(s)} = \frac{1}{K+1} \sum_{k=0}^K \hat{\mathbf{A}}^k \tilde{\mathbf{X}}^{(s)}$.

5: Predict class distribution using MLP: $\tilde{\mathbf{Z}}^{(s)} = P(\mathbf{Y} | \bar{\mathbf{X}}^{(s)}; \Theta)$.

6: **end for**

7: Compute supervised classification loss \mathcal{L}_{sup} via Eq. 4 and consistency regularization loss via Eq. 6.

8: Update the parameters Θ by gradients descending:

$$\nabla_{\Theta} \mathcal{L}_{sup} + \lambda \mathcal{L}_{con}$$

9: **end while**

10: Output prediction \mathbf{Z} via Eq. 8.

**Generate
 S Augmentations**

**Consistency
Regularization**

Consistency Regularized Training Algorithm

Graph Random Neural Network (GRAND)

- With Consistency Regularization Loss:
 - Random propagation can enforce the consistency of the classification confidence between each node and its all multi-hop neighborhoods.

$$\mathbb{E}_\epsilon (\mathcal{L}_{con}) \approx \mathcal{R}^c(\mathbf{W}) = \sum_{i=0}^{n-1} z_i^2 (1 - z_i)^2 \text{Var}_\epsilon (\overline{\mathbf{A}}_i \tilde{\mathbf{X}} \cdot \mathbf{W})$$

$$\mathcal{R}_{DN}^c(\mathbf{W}) = \frac{\delta}{1 - \delta} \sum_{j=0}^{n-1} \left[(\mathbf{X}_j \cdot \mathbf{W})^2 \sum_{i=0}^{n-1} (\overline{\mathbf{A}}_{ij})^2 z_i^2 (1 - z_i)^2 \right]$$

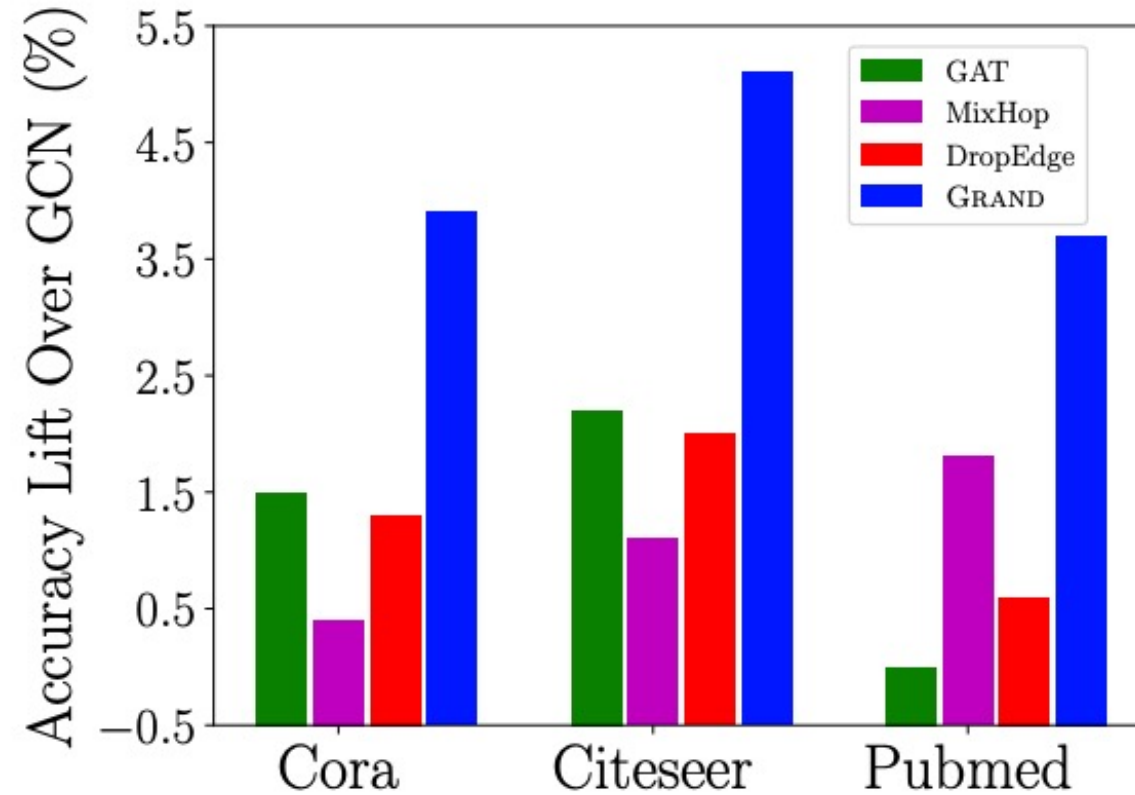
$$\mathcal{R}_{Do}^c(\mathbf{W}) = \frac{\delta}{1 - \delta} \sum_{h=0}^{d-1} \mathbf{W}_h^2 \sum_{j=0}^{n-1} \left[\mathbf{X}_{jh}^2 \sum_{i=0}^{n-1} z_i^2 (1 - z_i)^2 (\overline{\mathbf{A}}_{ij})^2 \right]$$

- With Supervised Cross-Entropy Loss:
 - Random propagation can enforce the consistency of the classification confidence between each node and its labeled multi-hop neighborhoods.

Results

	Method	Cora	Citeseer	Pubmed
GCNs	GCN [19]	81.5	70.3	79.0
	GAT [32]	83.0±0.7	72.5±0.7	79.0±0.3
	APPNP [20]	83.8±0.3	71.6±0.5	79.7±0.3
	Graph U-Net [11]	84.4±0.6	73.2±0.5	79.6±0.2
	SGC [36]	81.0±0.0	71.9±0.1	78.9±0.0
	MixHop [1]	81.9±0.4	71.4±0.8	80.8±0.6
	GMNN [28]	83.7	72.9	81.8
	GraphNAS [12]	84.2±1.0	73.1±0.9	79.6±0.4
Sampling GCNs	GraphSAGE [16]	78.9±0.8	67.4±0.7	77.8±0.6
	FastGCN [7]	81.4±0.5	68.8±0.9	77.6±0.5
Regularization GCNs	VBAT [10]	83.6±0.5	74.0±0.6	79.9±0.4
	G ³ NN [24]	82.5±0.2	74.4±0.3	77.9±0.4
	GraphMix [33]	83.9±0.6	74.5±0.6	81.0±0.6
	DropEdge [29]	82.8	72.3	79.6
	GRAND	85.4±0.4	75.4±0.4	82.7±0.6

Results



*GRAND achieves **much more significant** performance lifts in all three datasets!*

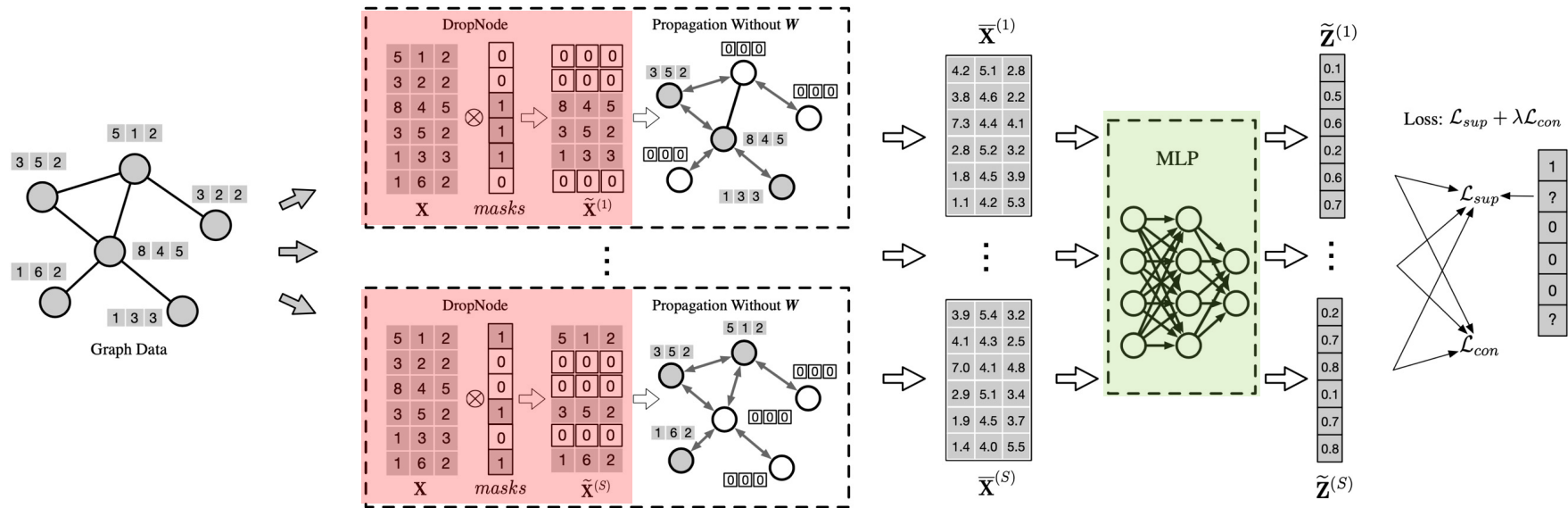
Larger Graphs

Table 5: Results on large datasets.

Method	Cora Full	Coauthor CS	Coauthor Physics	Amazon Computer	Amazon Photo	Citation CS
GCN	62.2 ± 0.6	91.1 ± 0.5	92.8 ± 1.0	82.6 ± 2.4	91.2 ± 1.2	49.9 ± 2.0
GAT	51.9 ± 1.5	90.5 ± 0.6	92.5 ± 0.9	78.0 ± 19.0	85.7 ± 20.3	49.6 ± 1.7
GRAND	63.5 ± 0.6	92.9 ± 0.5	94.6 ± 0.5	85.7 ± 1.8	92.5 ± 1.7	52.8 ± 1.2

- Feng et al. Graph Random Neural Networks for Semi-Supervised Learning on Graphs. <https://arxiv.org/abs/2005.11079>, 2020
- Code & data for Grand: <https://github.com/Grand20/grand>

The Design Choices in GRAND



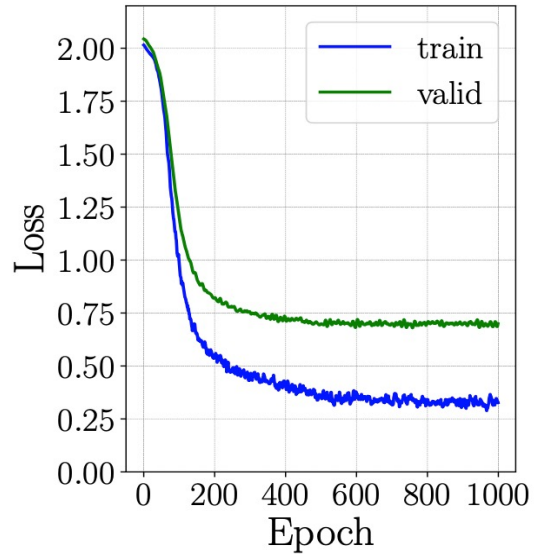
GRAND_dropout	84.9±0.4	75.0±0.3	81.7±1.0
GRAND_GCN	84.5±0.3	74.2±0.3	80.0±0.3
GRAND_GAT	84.3±0.4	73.2±0.4	79.2±0.6
GRAND	85.4±0.4	75.4±0.4	82.7±0.6

Ablation Study

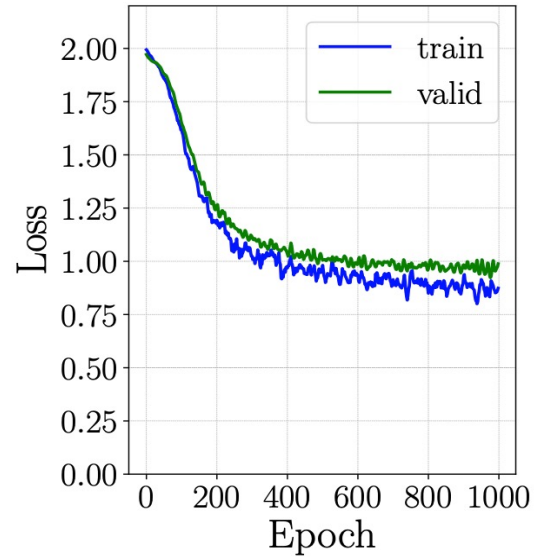
Method	Cora	Citeseer	Pubmed
GCN [19]	81.5	70.3	79.0
GAT [32]	83.0±0.7	72.5±0.7	79.0±0.3
APPNP [20]	83.8±0.3	71.6±0.5	79.7±0.3
Graph U-Net [11]	84.4±0.6	73.2±0.5	79.6±0.2
SGC [36]	81.0±0.0	71.9±0.1	78.9±0.0
MixHop [1]	81.9±0.4	71.4±0.8	80.8±0.6
GMNN [28]	83.7	72.9	81.8
GraphNAS [12]	84.2±1.0	73.1±0.9	79.6±0.4
DropEdge [29]	82.8	72.3	79.6
w/o CR	84.4±0.5	73.1±0.6	80.9±0.8
w/o mDN	84.7±0.4	74.8±0.4	81.0±1.1
w/o sharpening	84.6±0.4	72.2±0.6	81.6±0.8
w/o CR & DN	83.2±0.5	70.3±0.6	78.5±1.4

1. Each of the designed components contributes to the success of GRAND.
2. GRAND w/o consistency regularization outperforms almost *all 8 non-regularization based GCNs & DropEdge*

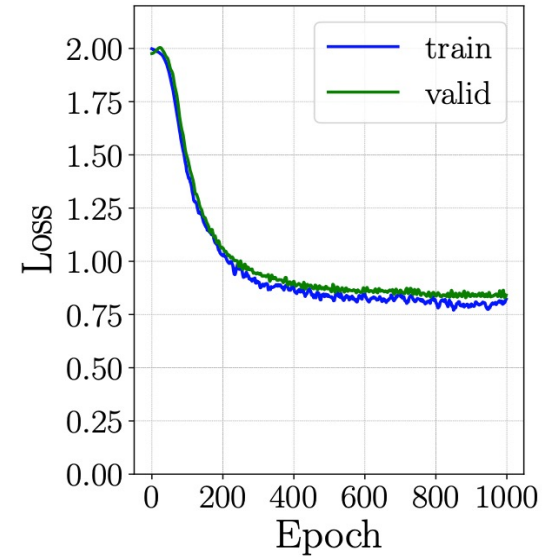
Generalization



(a) Without CR and RP



(b) Without CR

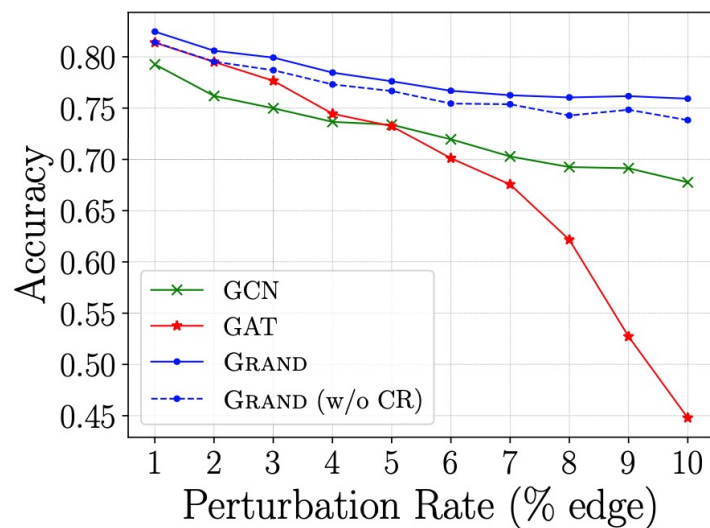


(c) GRAND

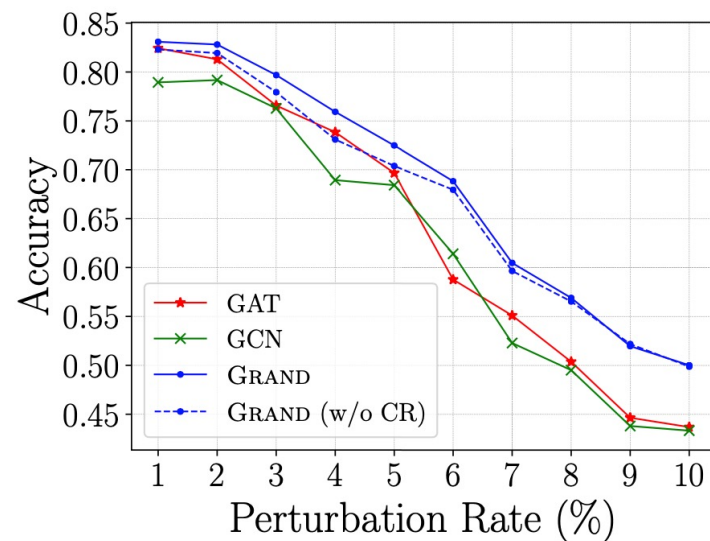
Both the random propagation and consistency regularization improve GRAND's generalization capability

- Feng et al. Graph Random Neural Networks for Semi-Supervised Learning on Graphs. <https://arxiv.org/abs/2005.11079>, 2020
- Code & data for Grand: <https://github.com/Grand20/grand>

Robustness



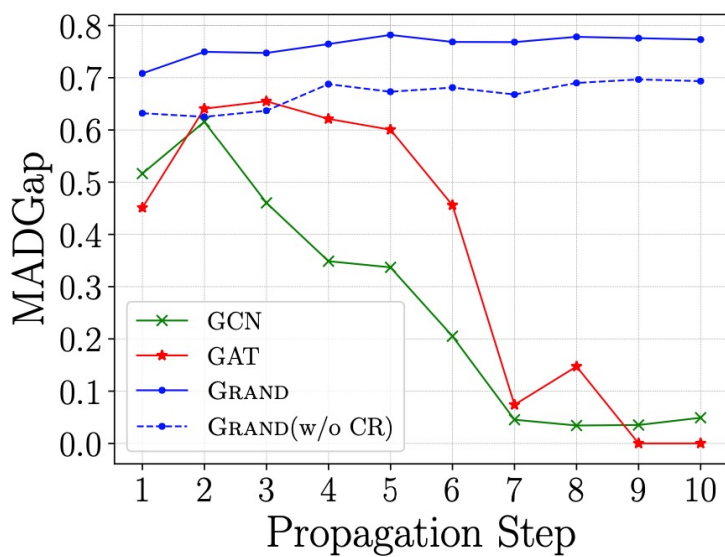
(a) Random Attack



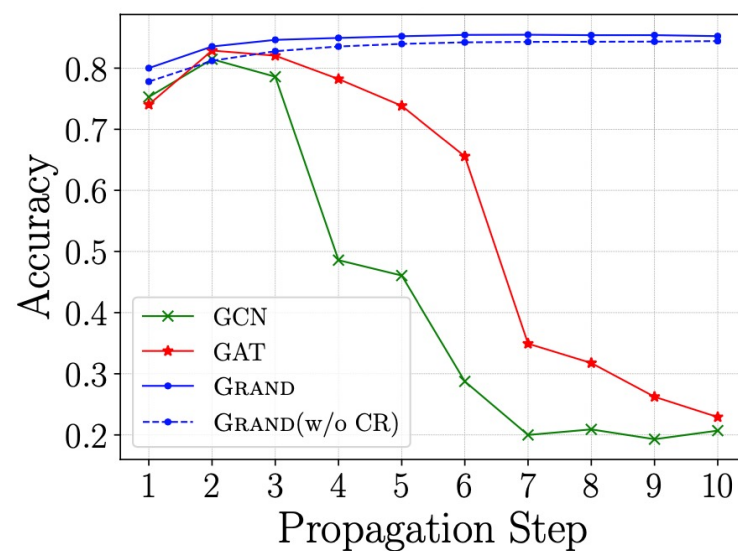
(b) Metattack

GRAND (with or w/o) consistency regularization is more robust than GCN and GAT.

Over-Smoothing



(a) MADGap



(b) Classification Results

GRAND is very powerful to relieve over-smoothing, when GCN & GAT are very vulnerable to it

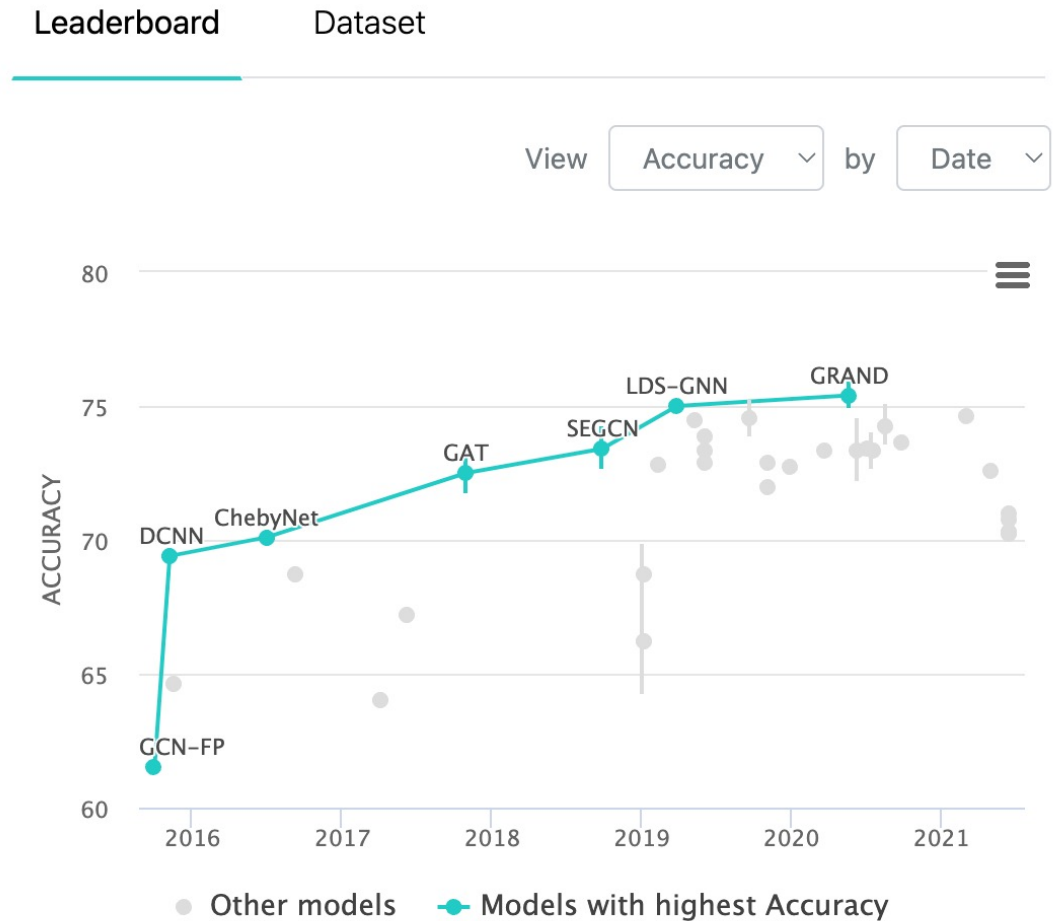
GRAND

- Paperwithcode

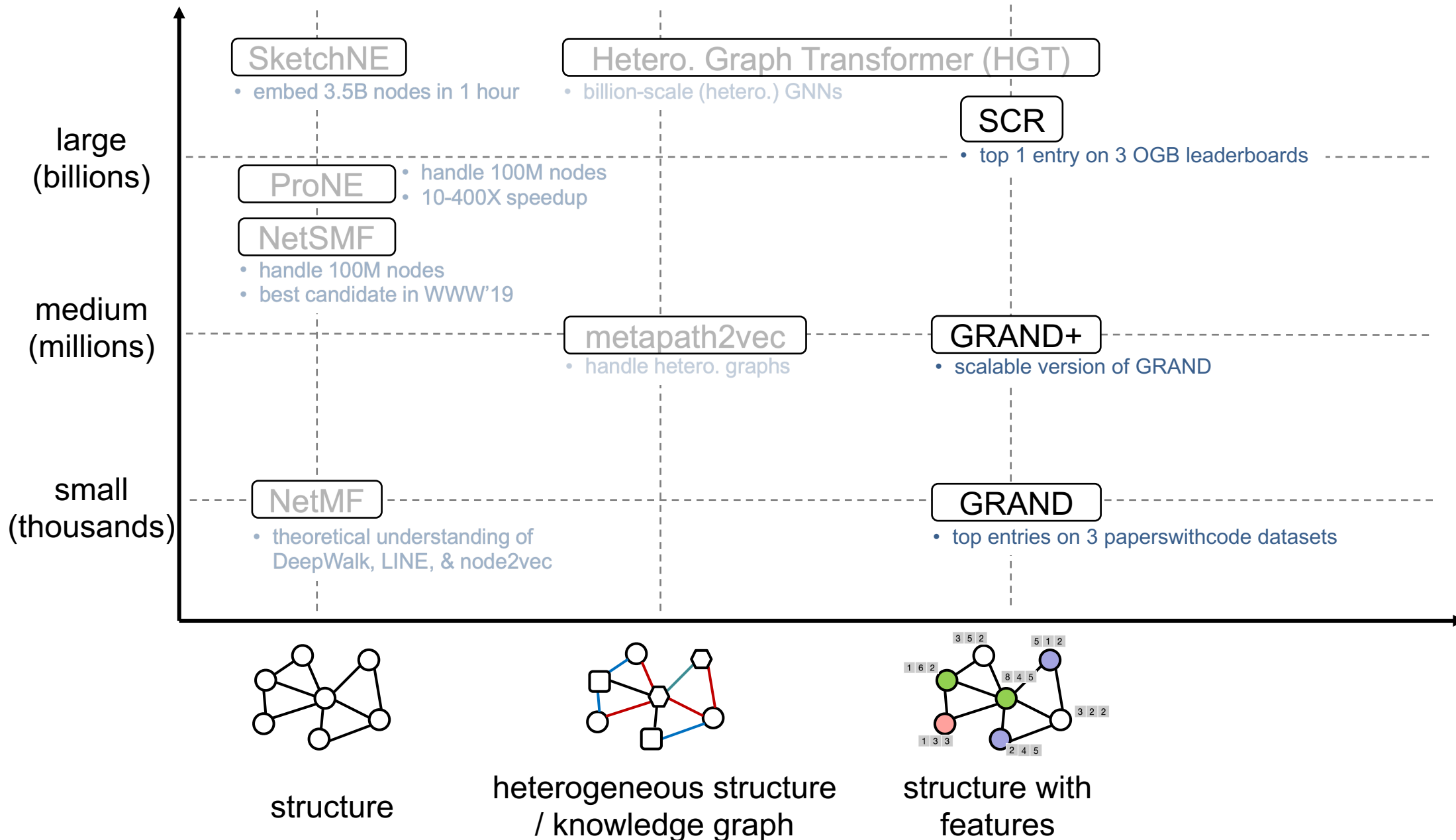
📄 State of the Art Node Classification on CiteSeer with Public Split: fixed 20 nodes per class

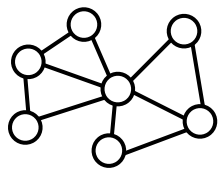
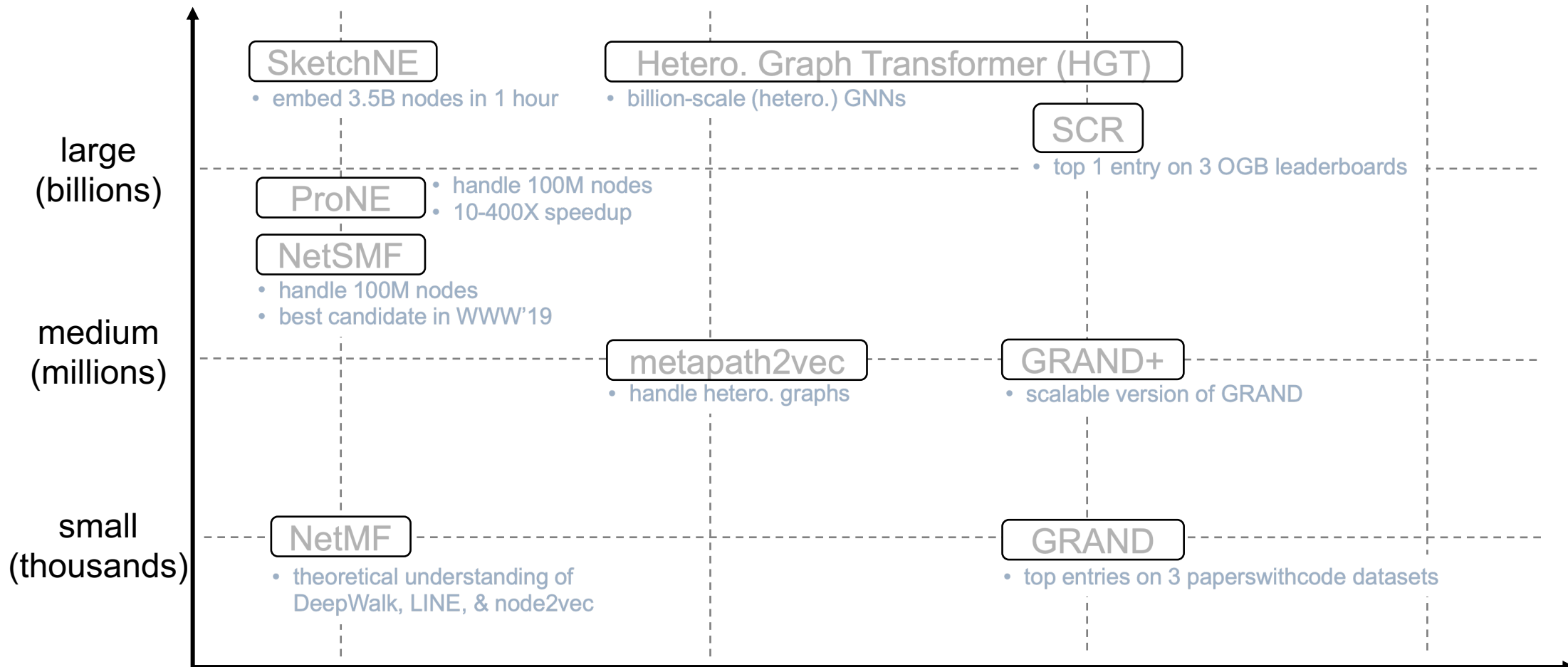
📄 Ranked #3 Node Classification on PubMed with Public Split: fixed 20 nodes per class

📄 Ranked #2 Node Classification on Cora with Public Split: fixed 20 nodes per class

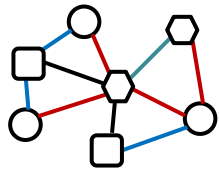


Semi-Supervised GNNs

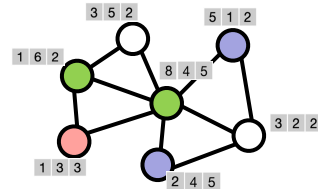




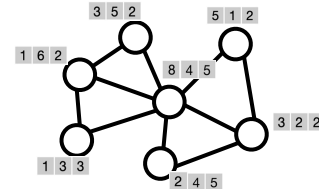
structure



heterogeneous structure / knowledge graph



structure with features



no label (pre-training)

Remaining Challenges in Graph Learning

- Common issues in graph research
 - Usually expensive and even infeasible to access sufficient labeled data
 - Sometimes need to handle out-of-distribution predictions
- Solution: graph pre-training?
 - Great success in language & image pre-training, e.g., ELMO, BERT, MoCo, GPT-n...

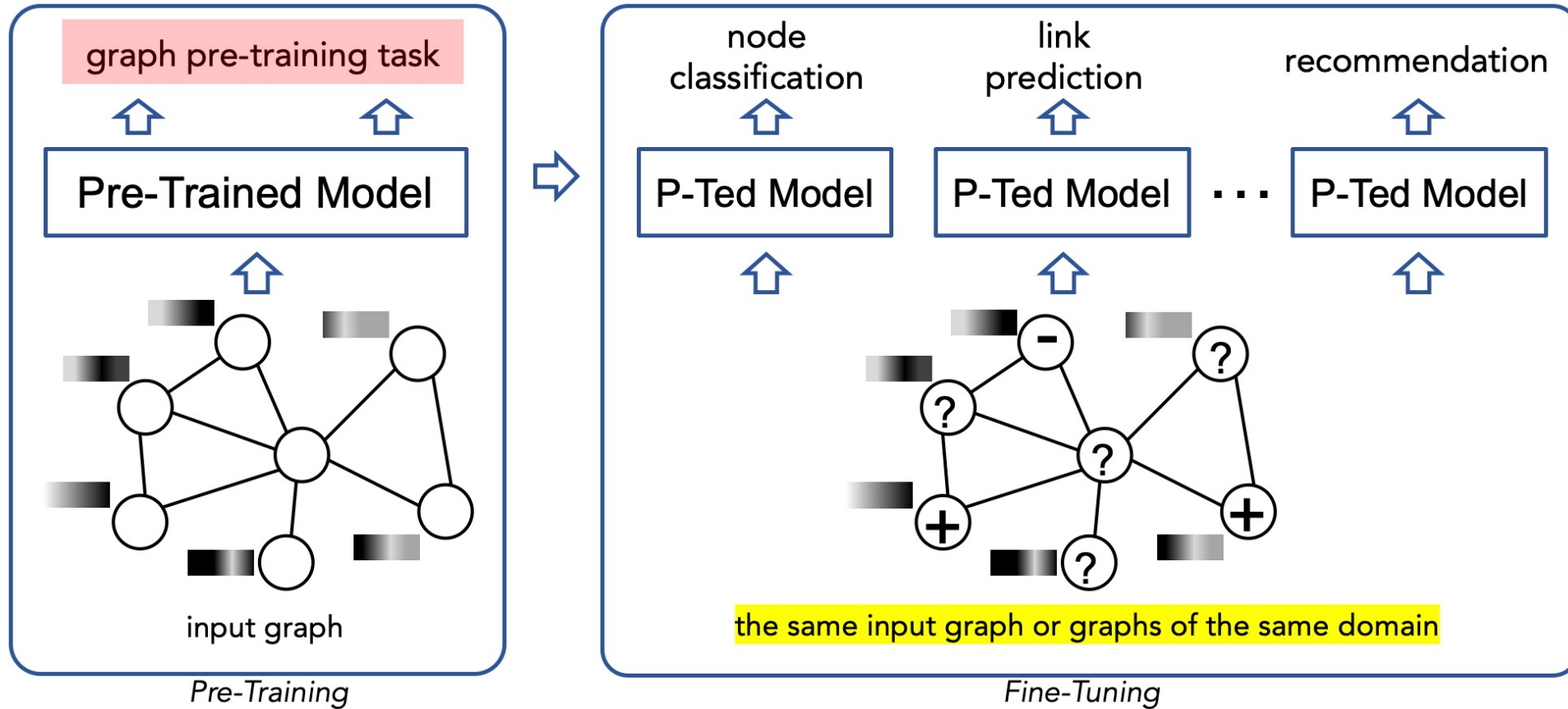
BERT for NLP

- Model level: Transformer
- Pre-training Task: MLM & NSP

Pre-training for Graphs

- **Model level: graph neural nets?**
- **Pre-training Task: ?**

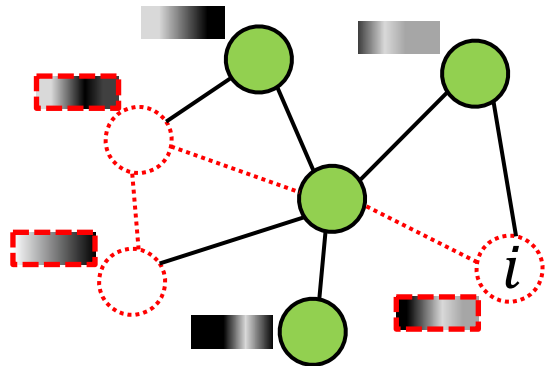
GNN Pre-Training



GPT-GNN: Generative Pre-Training of GNNs

- Model the graph distribution $p(G; \theta)$ by learning to reconstruct the input graph.
 - Factorize the graph likelihood into two terms:
 - Attribute Generation
 - Edge Generation

$$\log p_{\theta}(X, E) = \sum_{i=1}^{|\mathcal{V}|} \log p_{\theta}(X_i, E_i | X_{<i}, E_{<i}).$$



attribute and edge **masked**
input graph

$$p_{\theta}(X_i, E_i | X_{<i}, E_{<i}) \\ = p_{\theta}(X_i | X_{<i}, E_{<i}) \cdot p_{\theta}(E_i | X_{<i}, E_{<i})$$

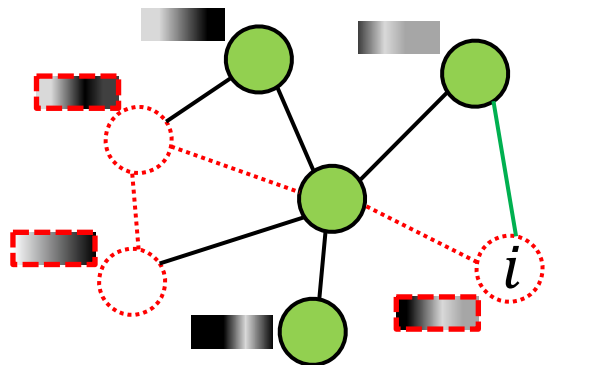
?

Lose the dependency between X_i and E_i

GPT-GNN: Generative Pre-Training of GNNs

- Model the graph distribution $p(G; \theta)$ by learning to reconstruct the input graph.
 - Factorize the graph likelihood into two terms:
 - Attribute Generation: given observed edges, generate node attributes
 - Edge Generation: given observed edges and generated attributes, generate masked edges

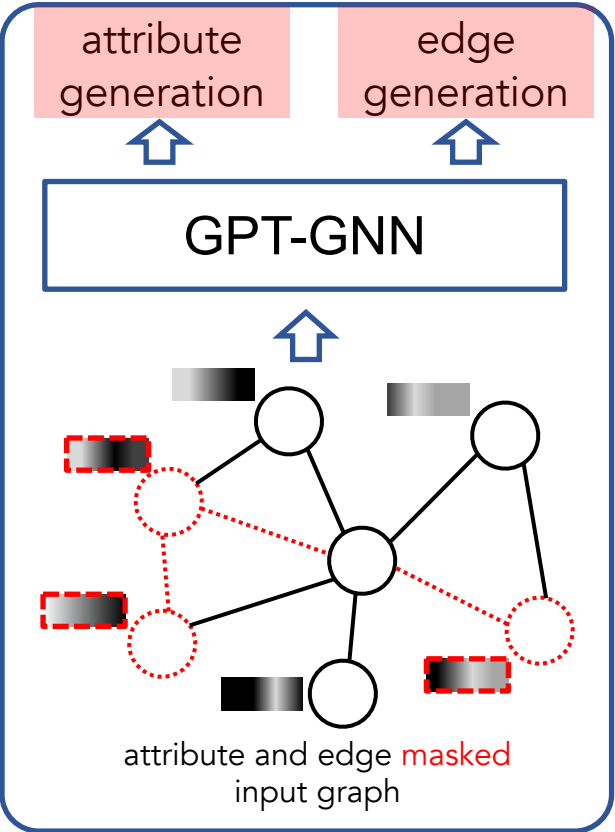
$$\log p_{\theta}(X, E) = \sum_{i=1}^{|\mathcal{V}|} \log p_{\theta}(X_i, E_i | X_{<i}, E_{<i}).$$



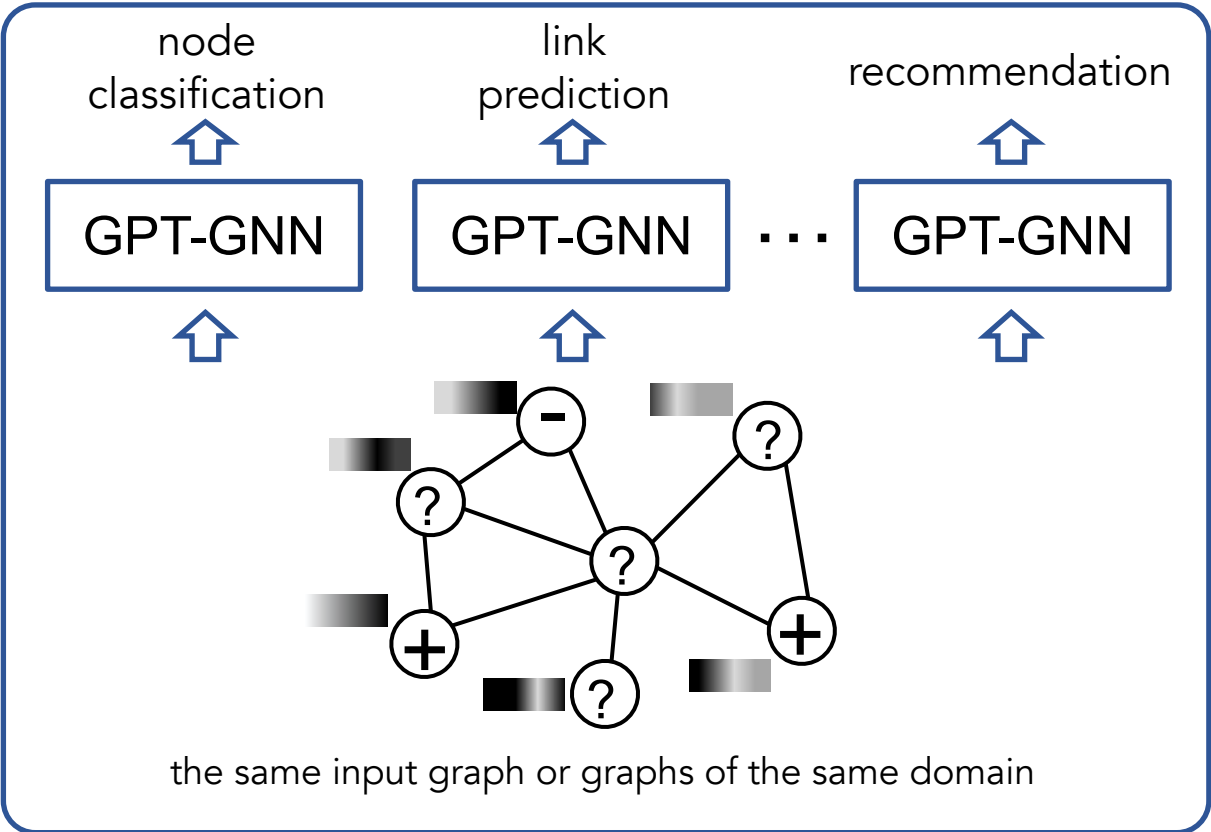
attribute and edge masked
input graph

$$\begin{aligned} & p_{\theta}(X_i, E_i | X_{<i}, E_{<i}) \\ &= \sum_o p_{\theta}(X_i, E_{i, \neg o} | E_{i, o}, X_{<i}, E_{<i}) \cdot p_{\theta}(E_{i, o} | X_{<i}, E_{<i}) \\ &= \mathbb{E}_o \left[p_{\theta}(X_i, E_{i, \neg o} | E_{i, o}, X_{<i}, E_{<i}) \right] \\ &= \mathbb{E}_o \left[\underbrace{p_{\theta}(X_i | E_{i, o}, X_{<i}, E_{<i})}_{1) \text{ generate attributes}} \cdot \underbrace{p_{\theta}(E_{i, \neg o} | E_{i, o}, X_{\leq i}, E_{<i})}_{2) \text{ generate edges}} \right]. \end{aligned}$$

GPT-GNN: Generative Pre-Training of GNNs



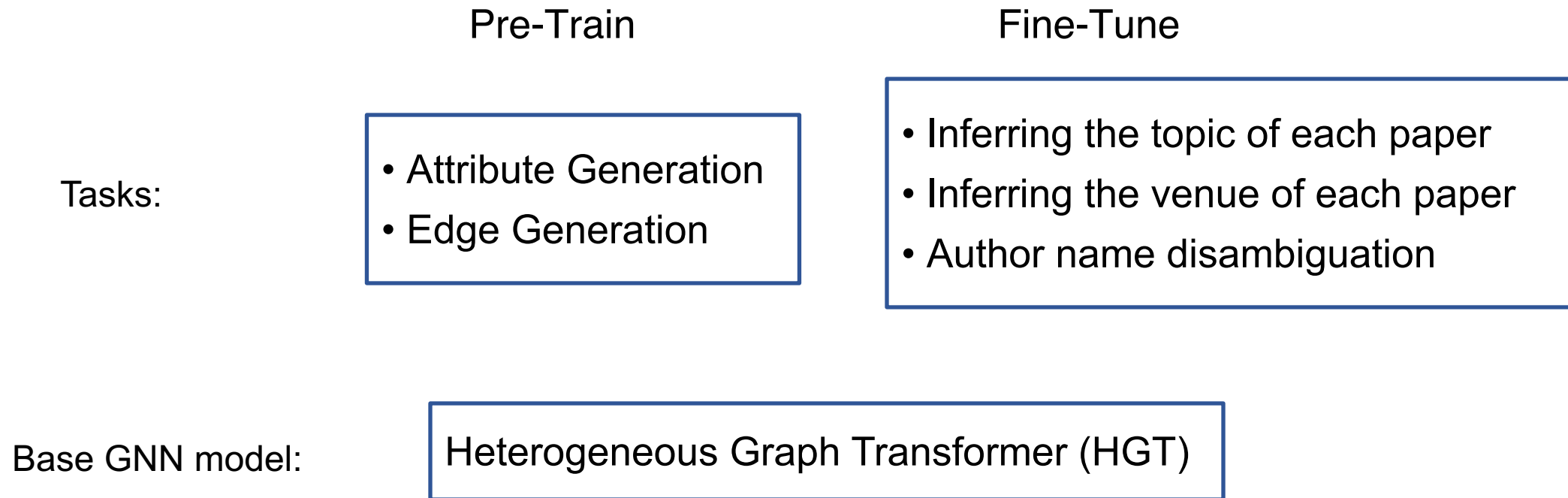
Pre-Training



Fine-Tuning

GPT-GNN: Generative Pre-Training of GNNs

- Data: Microsoft Academic Graph



GPT-GNN: Generative Pre-Training of GNNs

- Data: Microsoft Academic Graph

	Pre-Train	Fine-Tune
No Transfer:	CS Academic Graph	CS Academic Graph
Field Transfer:	Med, Bio, Physics...	CS Academic Graph
Time Transfer:	CS before 2014	CS after 2014
Time + Field Transfer:	Med, Bio, Physics... before 2014	CS after 2014

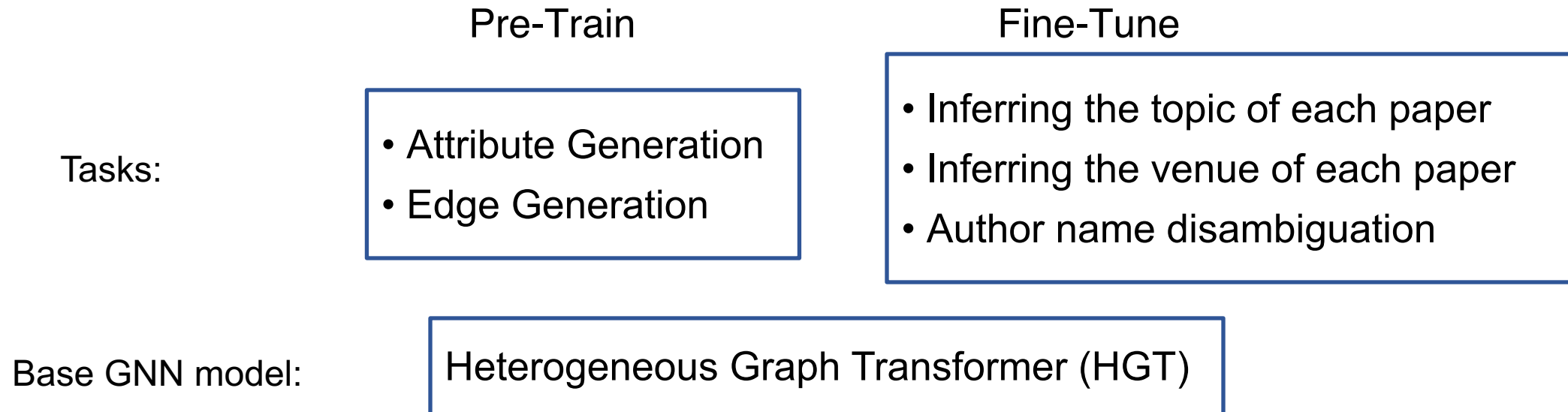
GPT-GNN: Generative Pre-Training of GNNs

Downstream Dataset		OAG		
Evaluation Task		Paper-Field	Paper-Venue	Author ND
No Pre-train		.346±.149	.598±.122	.813±.105
Field Transfer	GAE	.403±.114	.626±.093	.836±.084
	GraphSAGE (unsp.)	.368±.125	.609±.096	.818±.092
	Graph Infomax	.387±.112	.612±.097	.827±.084
	GPT-GNN (Attr)	.396±.118	.623±.105	.834±.086
	GPT-GNN (Edge)	.413±.109	.635±.096	.842±.093
	GPT-GNN	.420±.107	.641±.098	.848±.102
Time Transfer	GAE	.384±.117	.619±.101	.828±.095
	GraphSAGE (unsp.)	.352±.121	.601±.105	.815±.093
	Graph Infomax	.369±.116	.606±.102	.821±.089
	GPT-GNN (Attr)	.374±.114	.614±.098	.826±.089
	GPT-GNN (Edge)	.397±.105	.629±.102	.836±.088
	GPT-GNN	.405±.108	.635±.101	.840±.093
Time + Field Transfer	GAE	.371±.124	.611±.108	.821±.102
	GraphSAGE (unsp.)	.349±.130	.602±.118	.812±.097
	Graph Infomax	.360±.121	.600±.102	.815±.093
	GPT-GNN (Attr)	.364±.115	.609±.103	.824±.094
	– (w/o node separation)	.347±.128	.601±.102	.813±.108
	GPT-GNN (Edge)	.390±.116	.622±.104	.830±.105
	– (w/o adaptive queue)	.376±.121	.617±.115	.828±.104
GPT-GNN	.397±.112	.628±.108	.833±.102	

- **All pre-training frameworks** help the performance of GNNs
 - GAE, GraphSage, Graph Infomax
 - GPT-GNN
- **GPT-GNN helps the most** by achieving a relative performance gain of 9.1% over the base model without pre-training
- **Both self-supervised tasks in GPT-GNN** help the pre-training framework
 - Attribute generation
 - Edge generation

GPT-GNN: Generative Pre-Training of GNNs

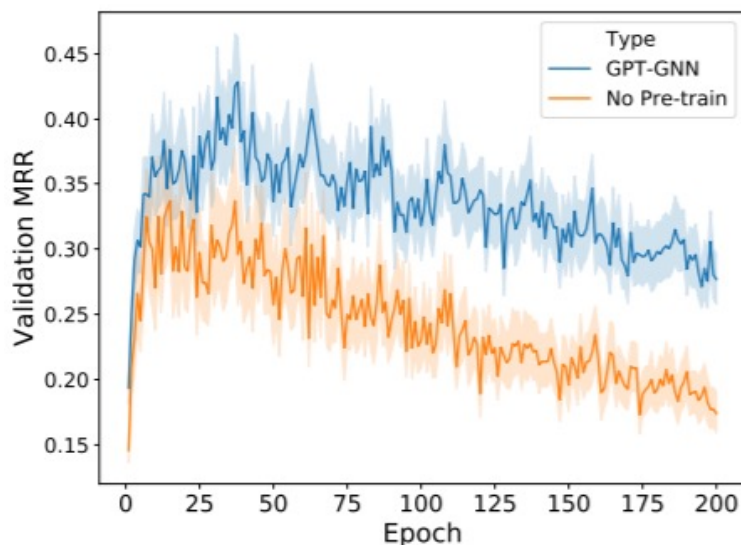
- Data: Microsoft Academic Graph



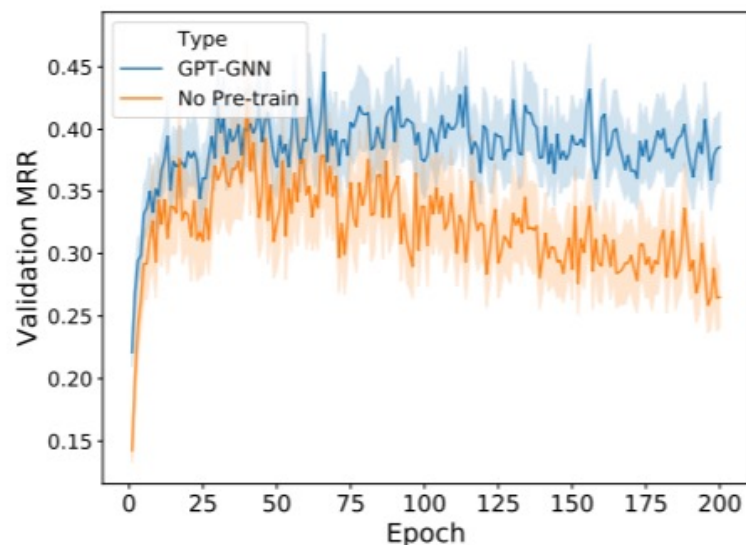
Model	HGT	GCN	GAT	RGCN	HAN
No Pre-train	.346	.327	.318	.296	.332
GPT-GNN	.420	.359	.382	.351	.406
Relative Gain	21.4%	9.8%	20.1%	18.9%	22.3%

Downstream Dataset	OAG (citation)	Reddit
No Pre-train	.281±.087	.873±.036
GAE	.296±.095	.885±.039
GraphSAGE (unsp.)	.287±.093	.880±.042
Graph Infomax	.291±.086	.877±.034
GPT-GNN	.309±.081	.896±.028

The Promise of Graph Pre-Training!



(a) Data Percentage: 10%



(b) Data Percentage: 20%

Predict Paper Title

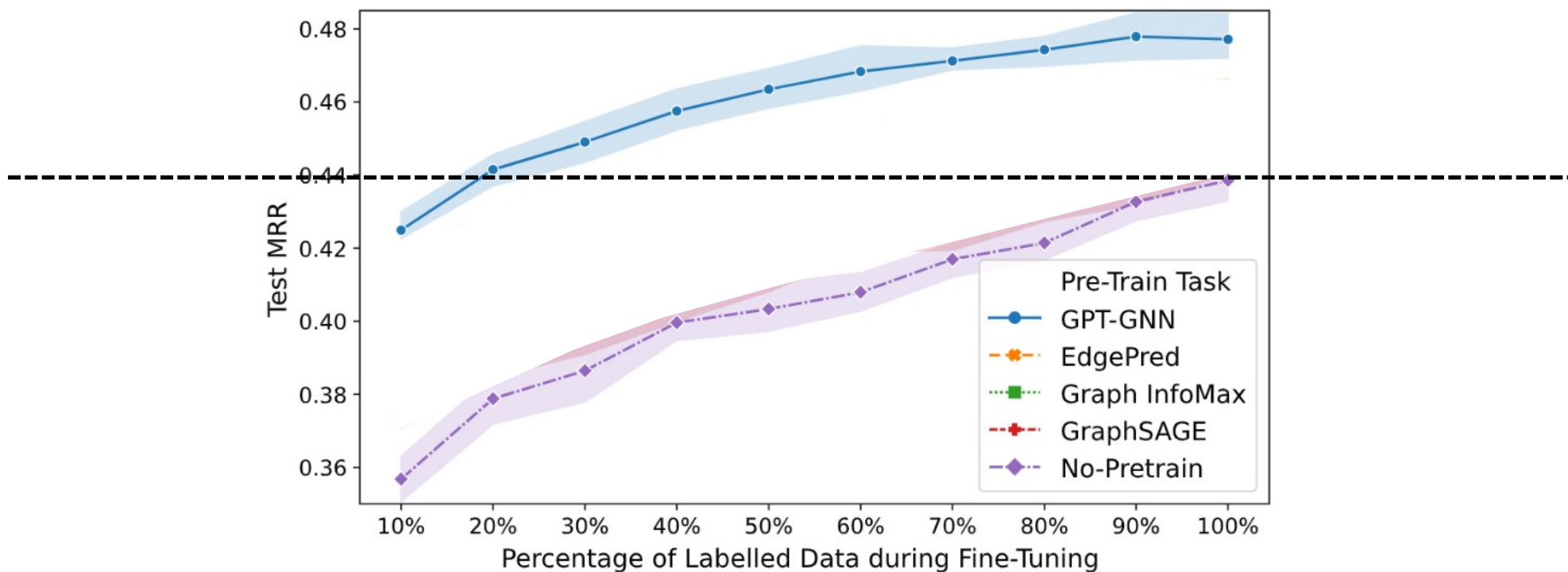
person recognition system using automatic probabilistic classification
a novel framework using spectrum sensing in wireless systems
a efficient evaluation of a distributed data storage service storage
parameter control in wireless sensor networks networks networks
a experimental system for for to the analysis of graphics

GroundTruth Paper Title

person re-identification by probabilistic relative distance comparison
a secure collaborative spectrum sensing strategy in cyber physical systems
an empirical analysis of a large scale mobile cloud storage service
optimal parameter estimation under controlled communication over sensor networks
an interactive computer graphics approach to surface representation

The Promise of Graph Pre-Training!

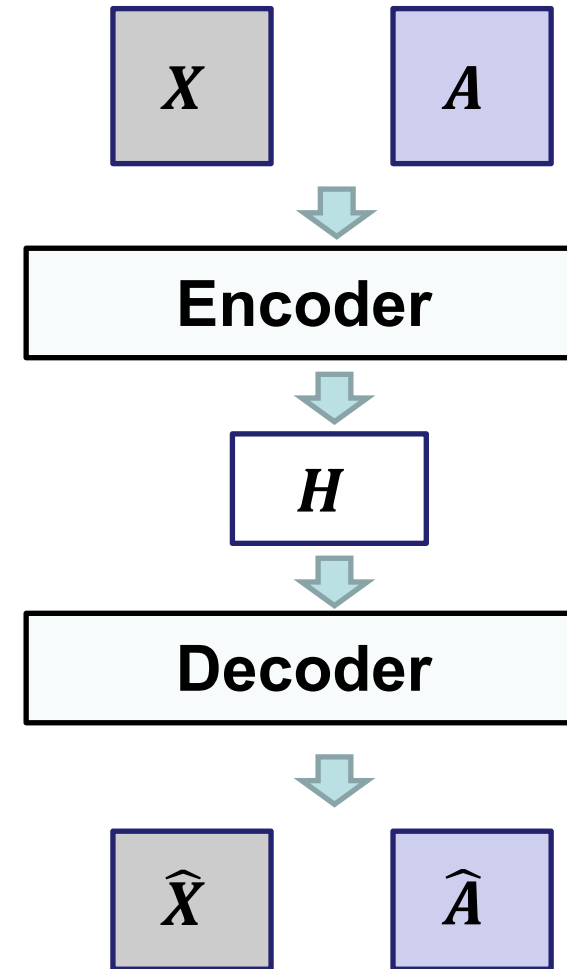
During fine-tuning



The GNN model **w/o** pre-training with **100%** training data
VS
The pre-trained GNN model with **10-20%** training data

Graph AutoEncoder

- $G = (V, A, X)$
 - $A \in \{0, 1\}^{N \times N}$: adjacency matrix,
 - $X \in \mathbb{R}^{N \times d}$: node features
- Encoding
 - $H = f_E(A, X)$,
- Decoding
 - $G' = f_D(A, H)$
- Reconstruction objectives:
 - graph structure (link)
 - node features

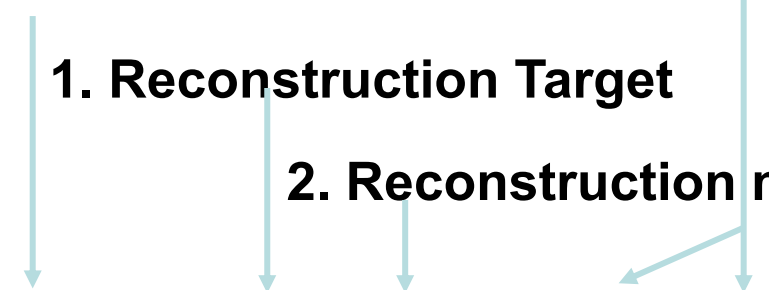


4. Error function

3. Decoding strategy

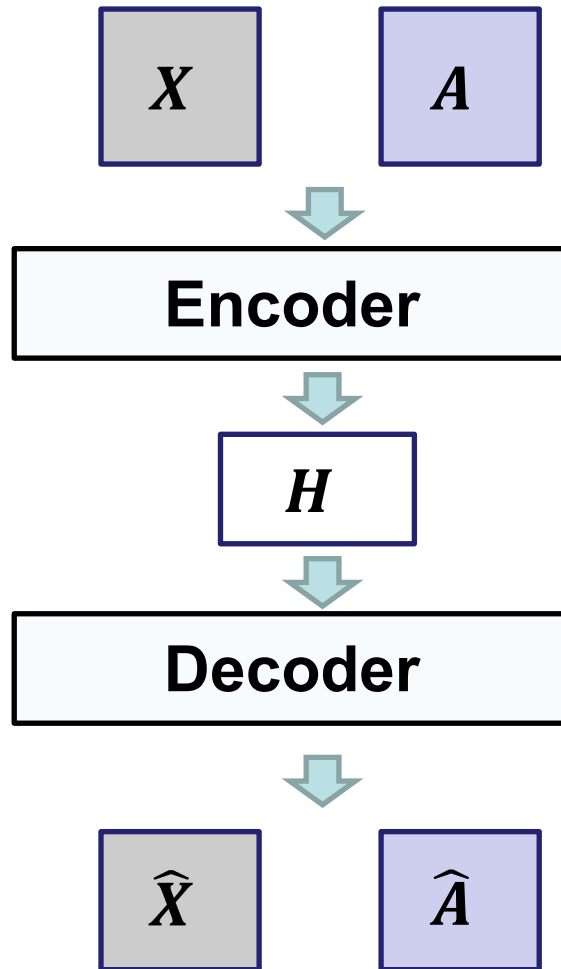
1. Reconstruction Target

2. Reconstruction method



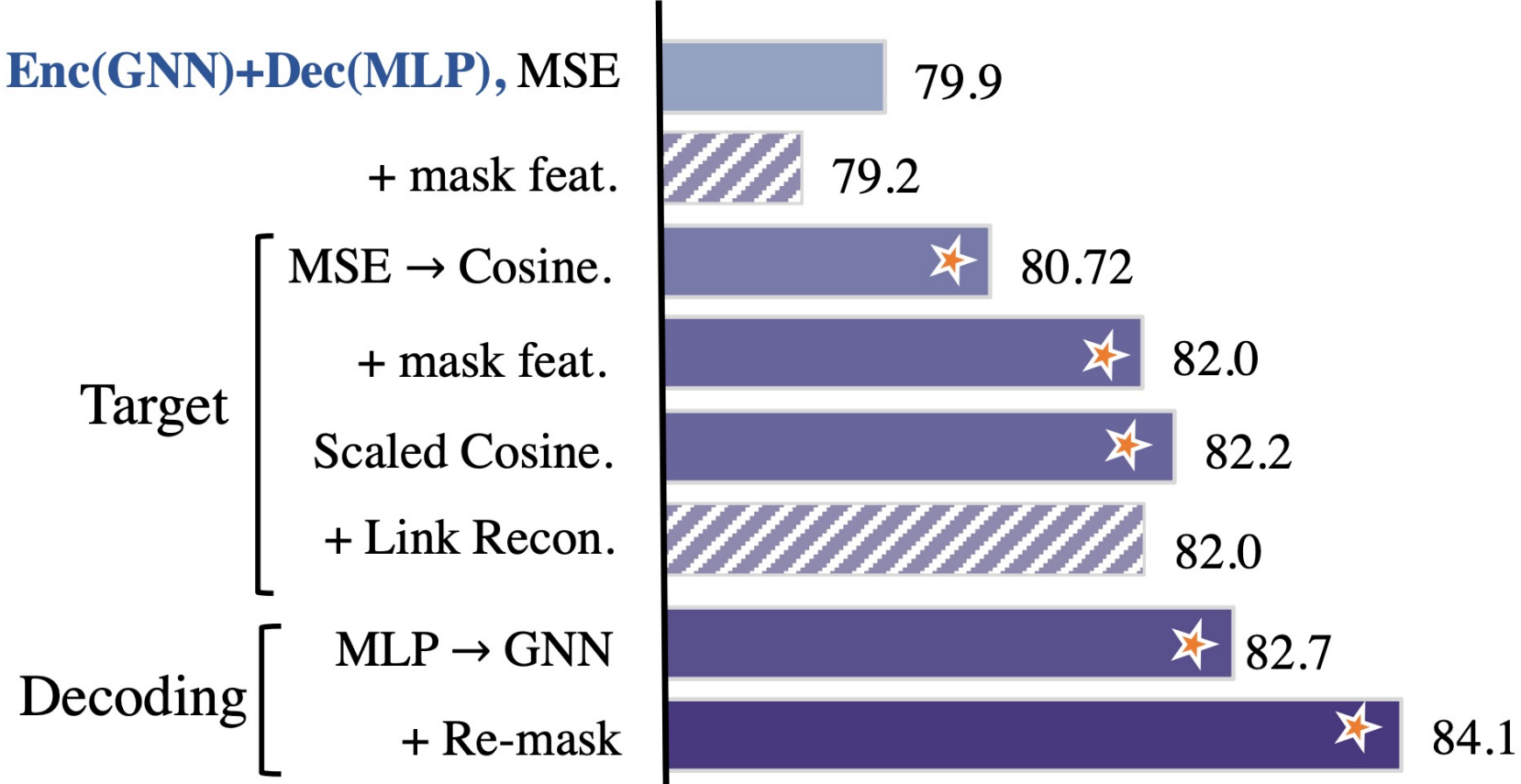
Methods	Feat. Loss	AE	No Struc.	Mask Feat.	GNN Decoder	Re-mask Dec.	Space
VGAE [20]	n/a	✓	-	-	-	-	$O(N^2)$
ARVGA [26]	n/a	✓	-	-	-	-	$O(N^2)$
MGAE [42]	MSE	✓	-	✓	-	-	$O(N)$
GALA [27]	MSE	✓	✓	-	✓	-	$O(N)$
GATE [31]	MSE	✓	-	-	✓	-	$O(N)$
AttrMask [16]	CE	✓	✓	✓	-	-	$O(N)$
GPT-GNN [17]	MSE	-	-	✓	-	-	$O(N)$
AGE [3]	n/a	✓	-	-	-	-	$O(N^2)$
NodeProp [18]	MSE	✓	✓	✓	-	-	$O(N)$

Generative SSL for Graphs: Graph AutoEncoder?

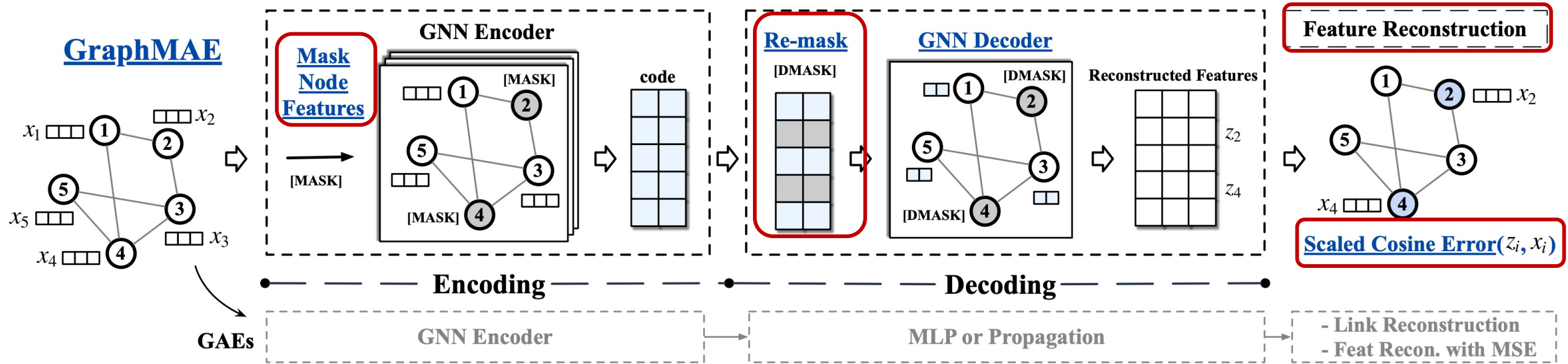


1. What to reconstruct ?
2. How to avoid trivial solutions ?
3. How to design the decoding ?
4. What error function to use ?

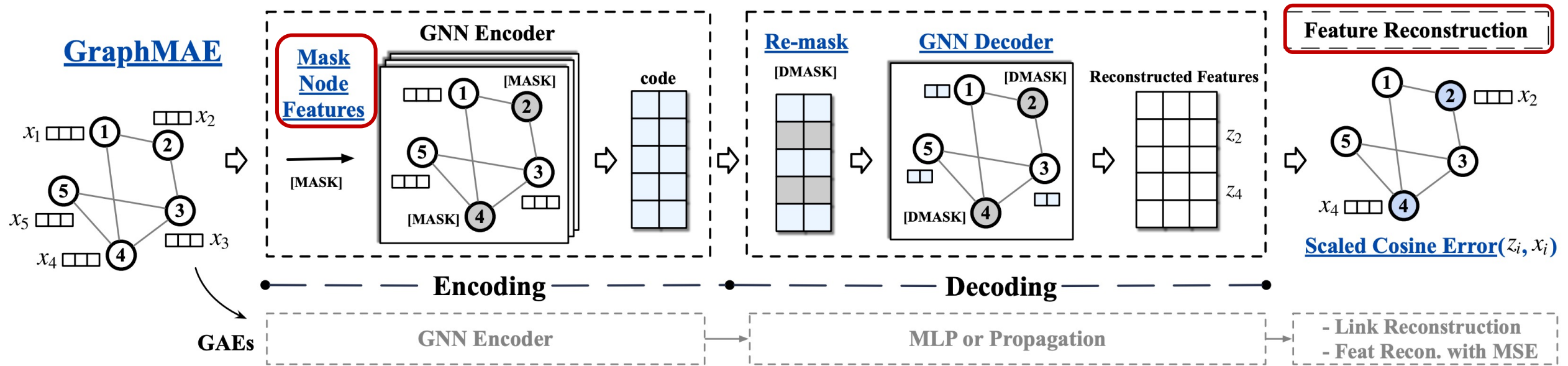
Graph AutoEncoder



GraphMAE



Masked Feature Reconstruction

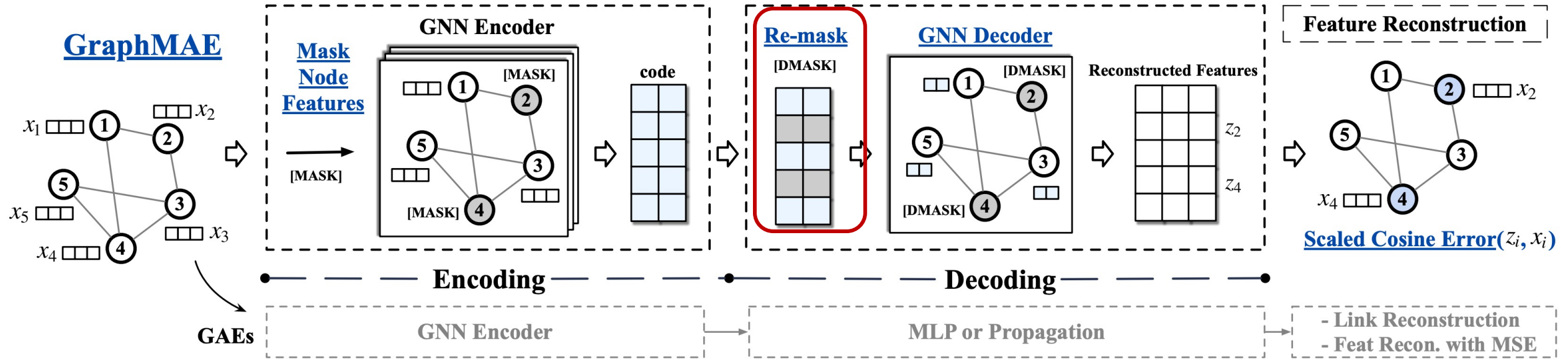


- Feature construction as the learning objective
- Masked feature reconstruction
 1. Sample a subset of nodes $\tilde{V} \subset V$
 2. Replace node feature with [MASK]

$$\tilde{x}_i = \begin{cases} \mathbf{x}_{[M]} & v_i \in \tilde{V} \\ \mathbf{x}_i & v_i \notin \tilde{V} \end{cases}$$

- $H = f_E(A, \tilde{X})$

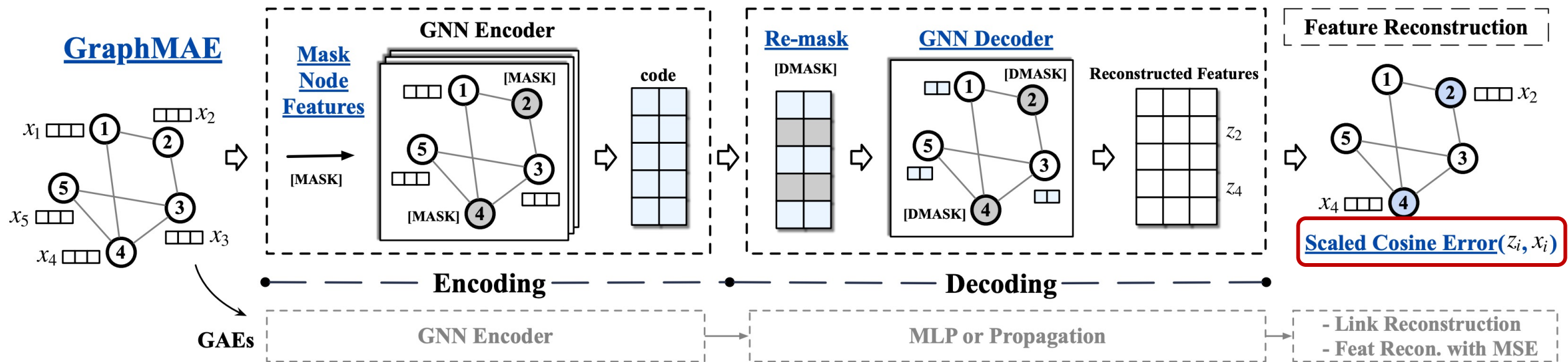
GNNs as Decoder with Re-Mask Decoding



- Use a GNN as the decoder
 - A more expressive decoder helps reconstruct low informative features
- Re-mask node features before decoder
 - Re-mask the “masked” nodes

$$\tilde{H} = \text{Remask}(H), \quad Z = f_D(A, \tilde{H}) \quad \tilde{h}_i = \begin{cases} h_{[M]} & v_i \in \tilde{\mathcal{V}} \\ h_i & v_i \notin \tilde{\mathcal{V}} \end{cases}$$

Scaled Cosine Error as the Criterion



- MSE fails, especially for continuous features
 - Sensitivity & low selectivity
- Scaled cosine error as the criterion
 - Cosine error & Scaled coefficient

$$L_{MSE} = \frac{1}{|\tilde{\mathcal{V}}|} \sum_{v_i \in \tilde{\mathcal{V}}} (x_i - z_i)^2$$

$$\mathcal{L}_{SCE} = \frac{1}{|\tilde{\mathcal{V}}|} \sum_{v_i \in \tilde{\mathcal{V}}} \left(1 - \frac{x_i^T z_i}{\|x_i\| \cdot \|z_i\|}\right)^\gamma, \gamma \geq 1,$$

GraphMAE vs GAEs

4. Error function

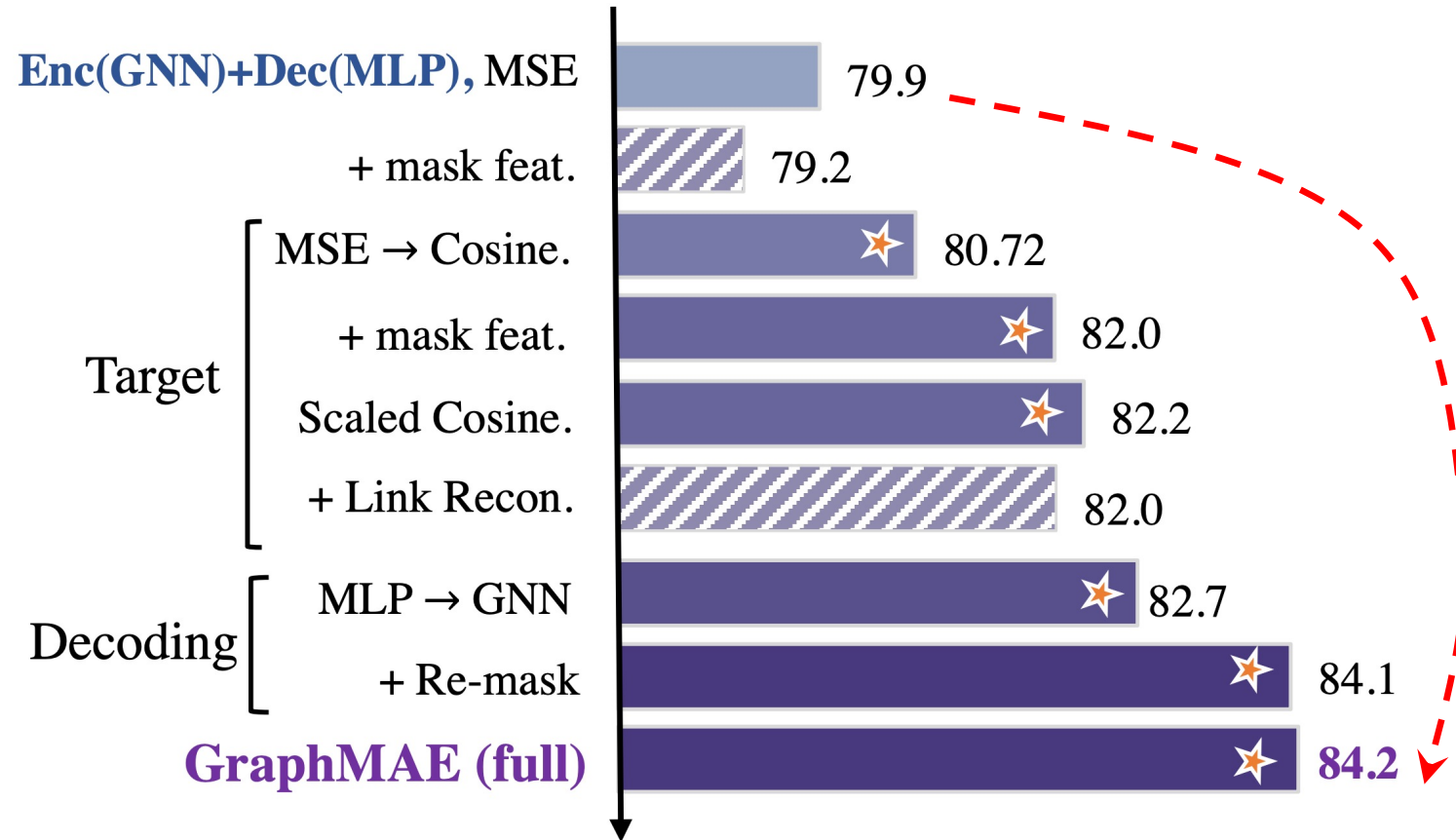
3. Decoding strategy

1. Reconstruction Target

2. Reconstruction method

Methods	Feat. Loss	AE	No Struc.	Mask Feat.	GNN Decoder	Re-mask Dec.	Space
VGAE [20]	n/a	✓	-	-	-	-	$O(N^2)$
ARVGA [26]	n/a	✓	-	-	-	-	$O(N^2)$
MGAE [42]	MSE	✓	-	✓	-	-	$O(N)$
GALA [27]	MSE	✓	✓	-	✓	-	$O(N)$
GATE [31]	MSE	✓	-	-	✓	-	$O(N)$
AttrMask [16]	CE	✓	✓	✓	-	-	$O(N)$
GPT-GNN [17]	MSE	-	-	✓	-	-	$O(N)$
AGE [3]	n/a	✓	-	-	-	-	$O(N^2)$
NodeProp [18]	MSE	✓	✓	✓	-	-	$O(N)$
GraphMAE	SCE	✓	✓	✓	✓	✓	$O(N)$

GraphMAE



(b) The effect of GraphMAE designs on the performance on Cora dataset.

Node classification

Node Classification

Table 1: Experiment results in unsupervised representation learning for node classification. We report Micro-F1(%) score for PPI and accuracy(%) for the other datasets.

	Dataset	Cora	CiteSeer	PubMed	Ogbn-arxiv	PPI	Reddit
Supervised	GCN	81.5	70.3	79.0	71.74±0.29	75.7±0.1	95.3±0.1
	GAT	83.0±0.7	72.5±0.7	79.0±0.3	72.10±0.13	97.30±0.20	96.0±0.1
Self-supervised	GAE	71.5±0.4	65.8±0.4	72.1±0.5	-	-	-
	GPT-GNN	80.1±1.0	68.4±1.6	76.3±0.8	-	-	-
	GATE	83.2±0.6	71.8±0.8	<u>80.9±0.3</u>	-	-	-
	DGI	82.3±0.6	71.8±0.7	76.8±0.6	70.34±0.16	63.80±0.20	94.0±0.10
	MVGRL	83.5±0.4	73.3±0.5	80.1±0.7	-	-	-
	GRACE ¹	81.9±0.4	71.2±0.5	80.6±0.4	71.51±0.11	69.71±0.17	94.72±0.04
	BGRL ¹	82.7±0.6	71.1±0.8	79.6±0.5	<u>71.64±0.12</u>	<u>73.63±0.16</u>	94.22±0.03
	InfoGCL	83.5±0.3	73.5±0.4	79.1±0.2	-	-	-
	CCA-SSG ¹	<u>84.0±0.4</u>	73.1±0.3	<u>81.0±0.4</u>	71.24±0.20	73.34±0.17	<u>95.07±0.02</u>
	GraphMAE	84.2±0.4	<u>73.4±0.4</u>	81.1±0.4	71.75±0.17	74.50±0.29	96.01±0.08

Graph Classification

Table 2: Experiment results in unsupervised representation learning for graph classification. We report accuracy(%) for all datasets.

	Dataset	IMDB-B	IMDB-M	PROTEINS	COLLAB	MUTAG	REDDIT-B	NCI1
Supervised	GIN	75.1±5.1	52.3±2.8	76.2±2.8	80.2±1.9	89.4±5.6	92.4±2.5	82.7±1.7
	DiffPool	72.6±3.9	-	75.1±3.5	78.9±2.3	85.0±10.3	92.1±2.6	-
Graph Kernels	WL	72.30±3.44	46.95±0.46	72.92±0.56	-	80.72±3.00	68.82±0.41	80.31±0.46
	DGK	66.96±0.56	44.55±0.52	73.30±0.82	-	87.44±2.72	78.04±0.39	80.31±0.46
Self-supervised	graph2vec	71.10±0.54	50.44±0.87	73.30±2.05	-	83.15±9.25	75.78±1.03	73.22±1.81
	Infograph	73.03±0.87	49.69±0.53	74.44±0.31	70.65±1.13	89.01±1.13	82.50±1.42	76.20±1.06
	GraphCL	71.14±0.44	48.58±0.67	74.39±0.45	71.36±1.15	86.80±1.34	<u>89.53±0.84</u>	77.87±0.41
	JOAO	70.21±3.08	49.20±0.77	<u>74.55±0.41</u>	69.50±0.36	87.35±1.02	85.29±1.35	78.07±0.47
	GCC	72.0	49.4	-	78.9	-	89.8	-
	MVGRL	74.20±0.70	51.20±0.50	-	-	<u>89.70±1.10</u>	84.50±0.60	-
	InfoGCL	<u>75.10±0.90</u>	<u>51.40±0.80</u>	-	<u>80.00±1.30</u>	91.20±1.30	-	<u>80.20±0.60</u>
	GraphMAE	75.52±0.66	51.63±0.52	75.30±0.39	80.32±0.46	88.19±1.26	88.01±0.19	80.40±0.30

Transfer Learning

Table 3: Experiment results in transfer learning on molecular property prediction benchmarks. The model is first pre-trained on ZINC15 and then finetuned on the following datasets. We report ROC-AUC(%) scores.

	BBBP	Tox21	ToxCast	SIDER	ClinTox	MUV	HIV	BACE	Avg.
No-pretrain	65.5±1.8	74.3±0.5	63.3±1.5	57.2±0.7	58.2±2.8	71.7±2.3	75.4±1.5	70.0±2.5	67.0
ContextPred	64.3±2.8	<u>75.7±0.7</u>	63.9±0.6	60.9±0.6	65.9±3.8	75.8±1.7	77.3±1.0	79.6±1.2	70.4
AttrMasking	64.3±2.8	76.7±0.4	64.2±0.5	<u>61.0±0.7</u>	71.8±4.1	74.7±1.4	77.2±1.1	79.3±1.6	71.1
Infomax	68.8 ±0.8	75.3 ±0.5	62.7 ±0.4	58.4 ±0.8	69.9±3.0	75.3 ±2.5	76.0 ±0.7	75.9 ±1.6	70.3
GraphCL	69.7±0.7	73.9±0.7	62.4±0.6	60.5±0.9	76.0±2.7	69.8±2.7	78.5±1.2	75.4±1.4	70.8
JOAO	70.2±1.0	75.0±0.3	62.9±0.5	60.0±0.8	<u>81.3±2.5</u>	71.7±1.4	76.7±1.2	77.3±0.5	71.9
GraphLoG	72.5±0.8	<u>75.7±0.5</u>	63.5±0.7	61.2±1.1	76.7±3.3	<u>76.0±1.1</u>	<u>77.8±0.8</u>	83.5±1.2	<u>73.4</u>
GraphMAE	<u>72.0±0.6</u>	75.5±0.6	<u>64.1±0.3</u>	60.3±1.1	82.3±1.2	76.3±2.4	77.2±1.0	<u>83.1±0.9</u>	73.8

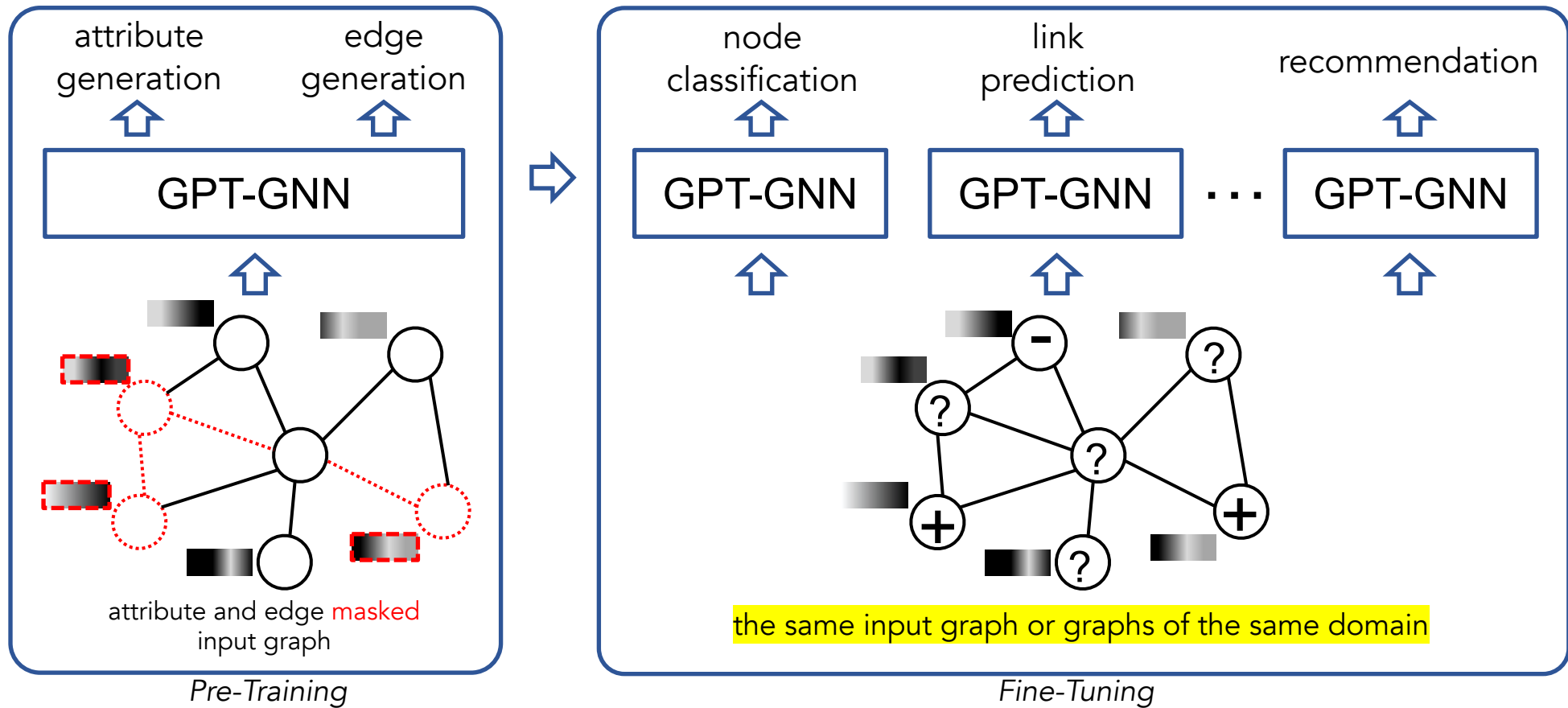
Ablation Study

Table 4: Ablation studies of decoder type, re-mask and reconstruction criterion on node- and graph-level benchmarks.

	Dataset	Node-Level			Graph-Level	
		Cora	PubMed	Arxiv	MUTAG	IMDB-B
COMP.	GraphMAE	84.2	81.1	71.75	88.19	75.52
	w/o mask	79.7	77.9	70.97	82.58	74.42
	w/o re-mask	82.7	80.0	71.61	86.29	74.42
	w/ MSE	79.1	73.1	67.44	86.30	74.04
Decoder	MLP	82.2	80.4	71.54	87.16	73.94
	GCN	81.3	79.1	71.59	87.78	74.54
	GIN	81.8	80.2	71.41	88.19	75.52
	GAT	84.2	81.1	71.75	86.27	74.04

Effects of the decoder type, objective function and mask strategy

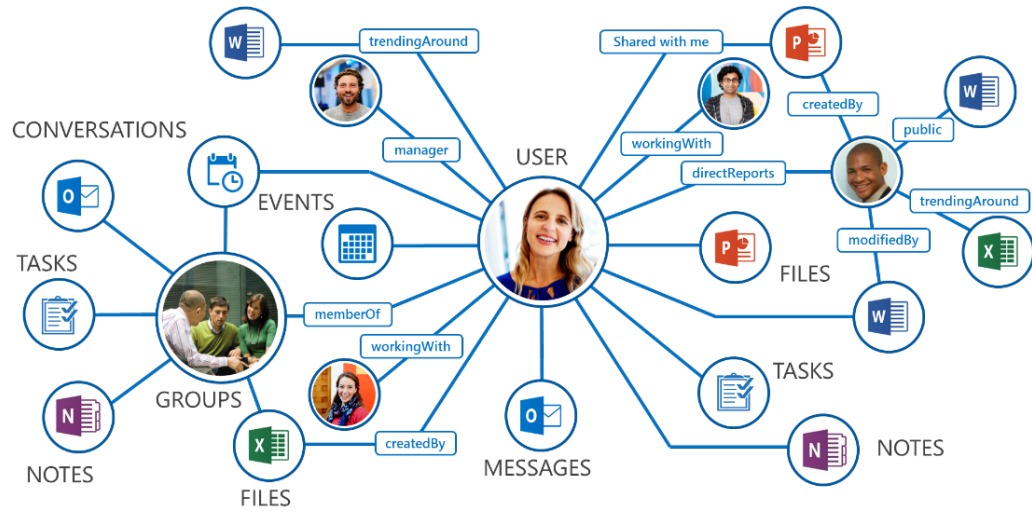
GNN Pre-Training on the “Same” Networks



1. Ziniu Hu et al. GPT-GNN: Generative Pre-Training of Graph Neural Networks. **KDD 2020**.

2. Zhenyu Hou et al. GraphMAE: Self-supervised graph autoencoders. **KDD 2022**.

Many Graphs



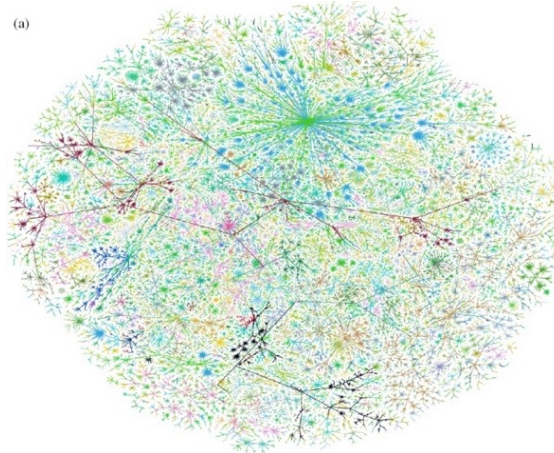
Office/Social Graph



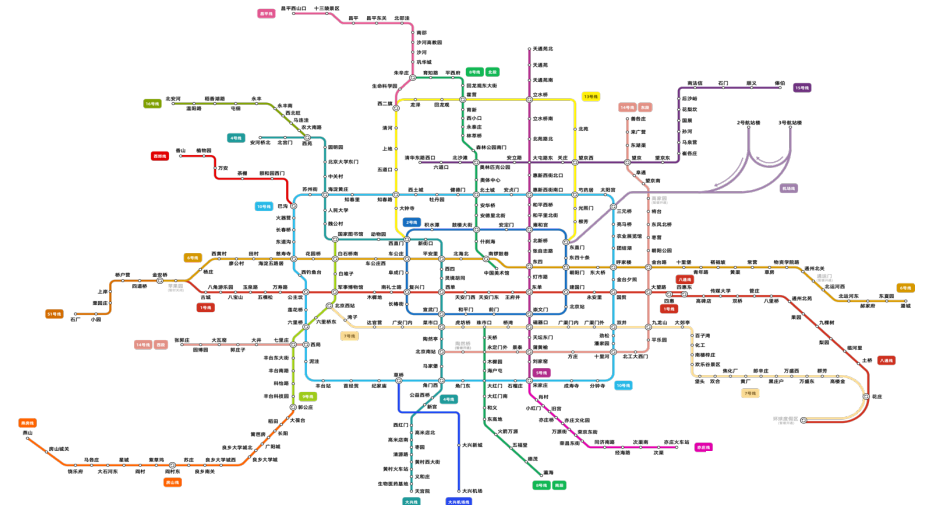
Biological Neural Networks



Knowledge Graph

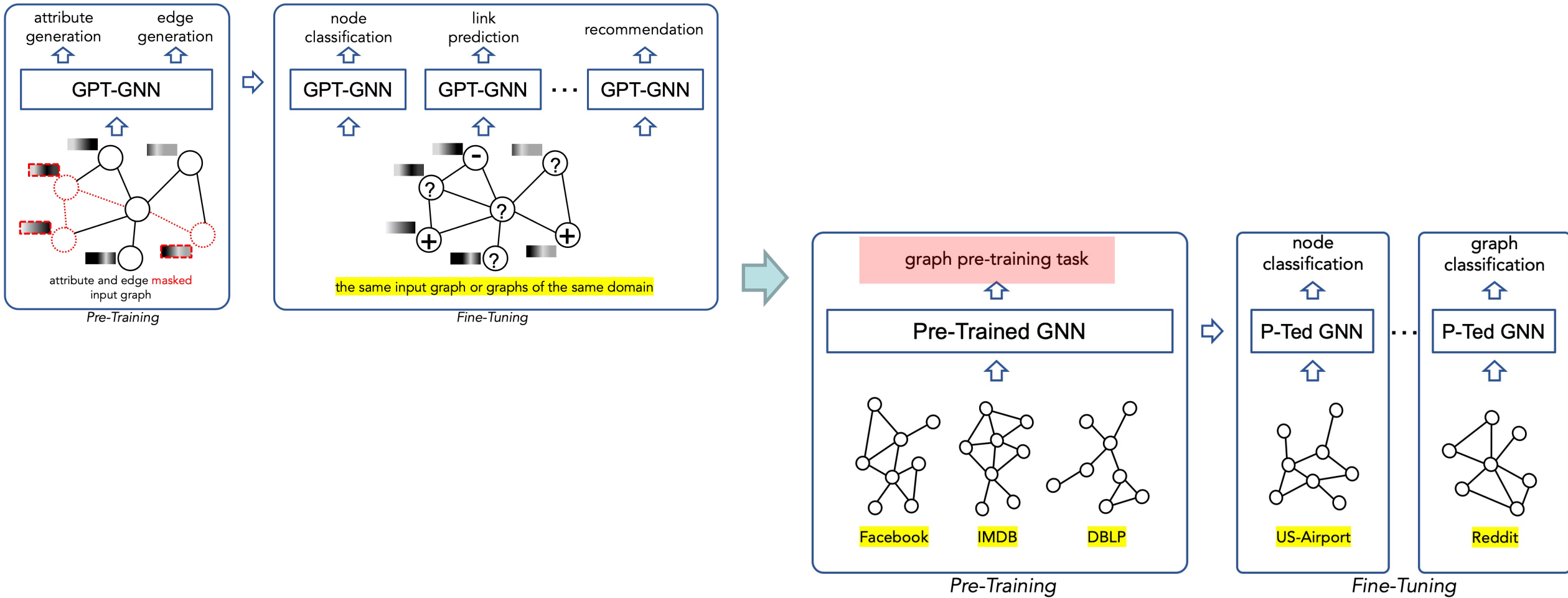


Internet



Transportation

GNN Pre-Training



GNN Pre-Training across Networks

- What are the requirements?
 - **structural similarity**, it maps vertices with similar local network topologies close to each other in the vector space
 - **transferability**, it is compatible with vertices and graphs unseen by the pre-training algorithm

GNN Pre-Training across Networks

- The Idea: Contrastive learning
 - **pre-training task:** instance discrimination
 - **InfoNCE objective:** output instance representations that are capable of capturing the similarities between instances

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

- query instance x^q
- query \mathbf{q} (embedding of x^q), i.e., $\mathbf{q} = f(x^q)$
- dictionary of keys $\{\mathbf{k}_0, \mathbf{k}_1, \dots, \mathbf{k}_K\}$
- key $\mathbf{k} = f(x^k)$

- Contrastive learning for graphs?
 - Q1: How to define instances in graphs?
 - Q2: How to define (dis) similar instance pairs in and across graphs?
 - Q3: What are the proper graph encoders?

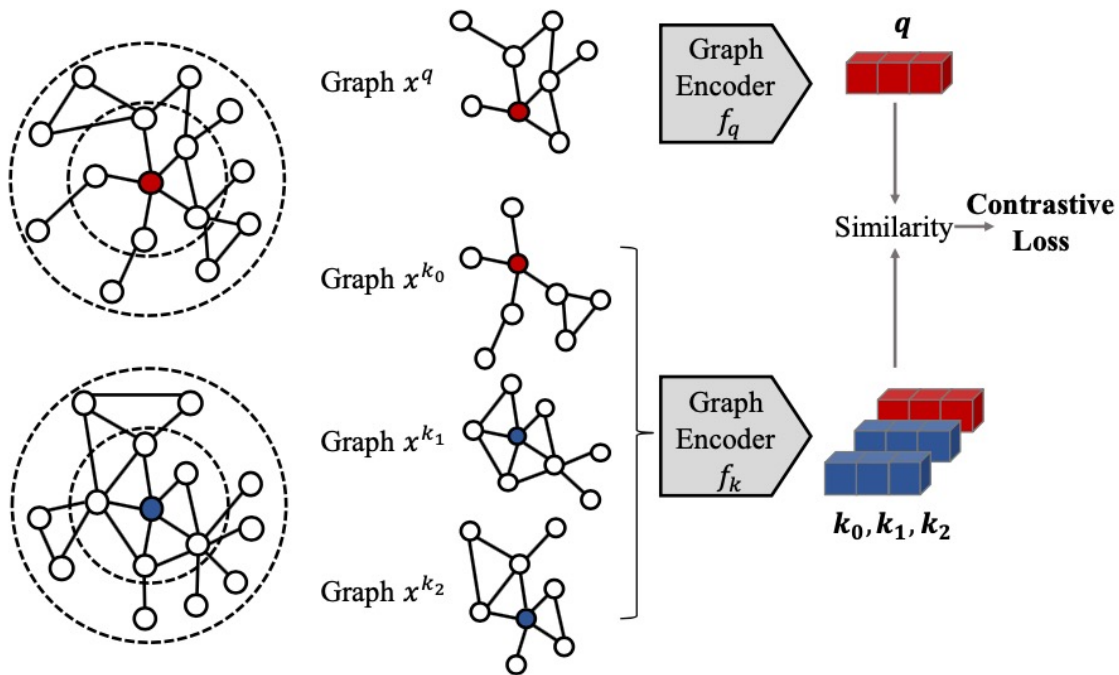
Graph Contrastive Coding (GCC)

Contrastive learning for graphs

- Q1: How to define instances in graphs?
- Q2: How to define (dis) similar instance pairs in and across graphs?
- Q3: What are the proper graph encoders?

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

Subgraph instance
discrimination



GCC Pre-Training / Fine-Tuning

- pre-train on six graphs

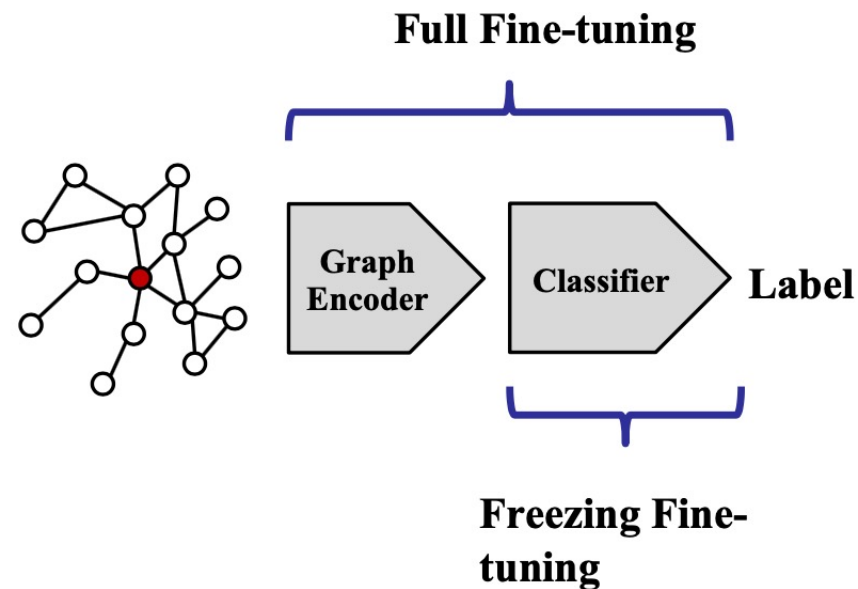
Dataset	Academia	DBLP (SNAP)	DBLP (NetRep)	IMDB	Facebook	LiveJournal
$ V $	137,969	317,080	540,486	896,305	3,097,165	4,843,953
$ E $	739,384	2,099,732	30,491,458	7,564,894	47,334,788	85,691,368

- fine-tune on **different** graphs

- US-Airport & AMiner academic graph
 - Node classification
- COLLAB, RDT-B, RDT-M, & IMDB-B, IMDB-M
 - Graph classification
- AMiner academic graph
 - Similarity search

- The base GNN

- Graph Isomorphism Network (GIN)



Results

Node Classification

Datasets	US-Airport	H-index
$ V $	1,190	5,000
$ E $	13,599	44,020
ProNE	62.3	69.1
GraphWave	60.2	70.3
Struc2vec	66.2	> 1 Day
GCC (E2E, freeze)	64.8	78.3
GCC (MoCo, freeze)	65.6	75.2
GCC (rand, full)	64.2	76.9
GCC (E2E, full)	68.3	80.5
GCC (MoCo, full)	67.2	80.6

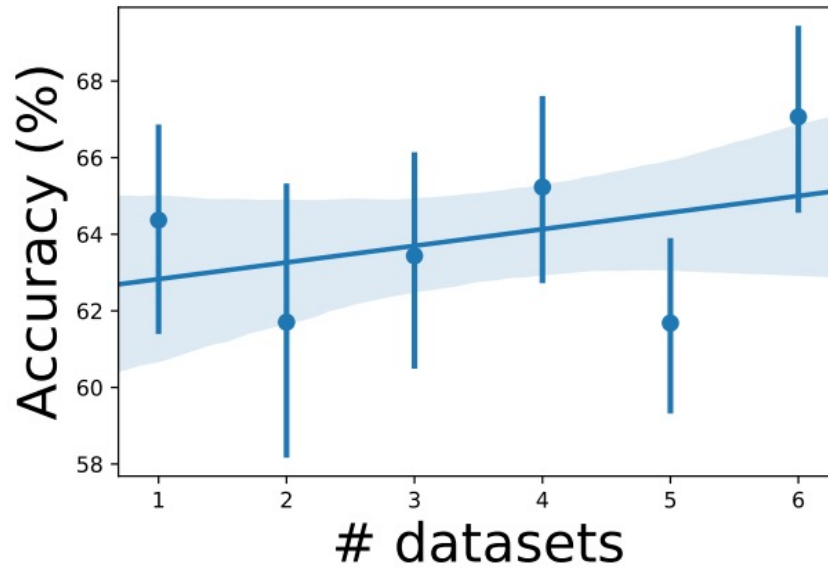
Similarity Search

	KDD-ICDM		SIGIR-CIKM		SIGMOD-ICDE	
$ V $	2,867	2,607	2,851	3,548	2,616	2,559
$ E $	7,637	4,774	6,354	7,076	8,304	6,668
# ground truth	697		874		898	
k	20	40	20	40	20	40
Random	0.0198	0.0566	0.0223	0.0447	0.0221	0.0521
RoIX	0.0779	0.1288	0.0548	0.0984	0.0776	0.1309
Panther++	0.0892	0.1558	0.0782	0.1185	0.0921	0.1320
GraphWave	0.0846	0.1693	0.0549	0.0995	0.0947	0.1470
GCC (E2E)	0.1047	0.1564	0.0549	0.1247	0.0835	0.1336
GCC (MoCo)	0.0904	0.1521	0.0652	0.1178	0.0846	0.1425

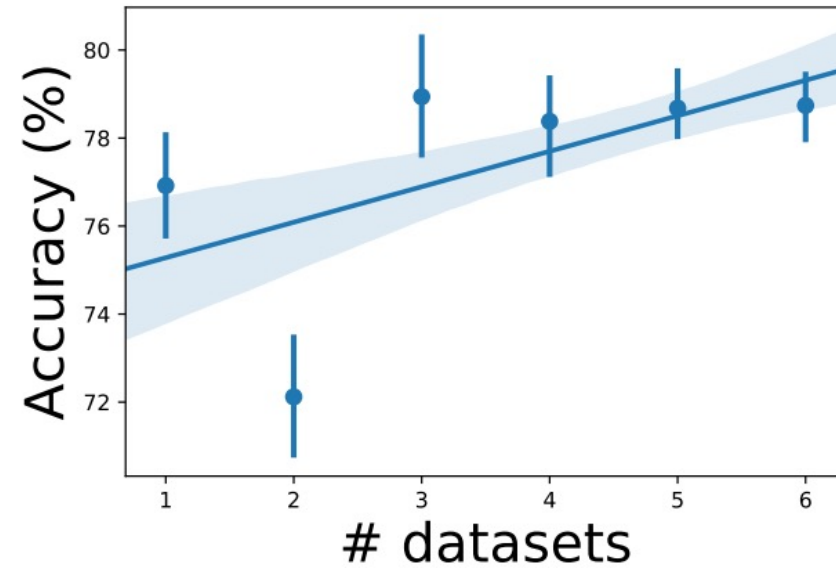
Graph Classification

Datasets	IMDB-B	IMDB-M	COLLAB	RDT-B	RDT-M
# graphs	1,000	1,500	5,000	2,000	5,000
# classes	2	3	3	2	5
Avg. # nodes	19.8	13.0	74.5	429.6	508.5
DGK	67.0	44.6	73.1	78.0	41.3
graph2vec	71.1	50.4	–	75.8	47.9
InfoGraph	73.0	49.7	–	82.5	53.5
GCC (E2E, freeze)	71.7	49.3	74.7	87.5	52.6
GCC (MoCo, freeze)	72.0	49.4	78.9	89.8	53.7
DGCNN	70.0	47.8	73.7	–	–
GIN	75.6	51.5	80.2	89.4	54.5
GCC (rand, full)	75.6	50.9	79.4	87.8	52.1
GCC (E2E, full)	70.8	48.5	79.0	86.4	47.4
GCC (MoCo, full)	73.8	50.3	81.1	87.6	53.0

Results

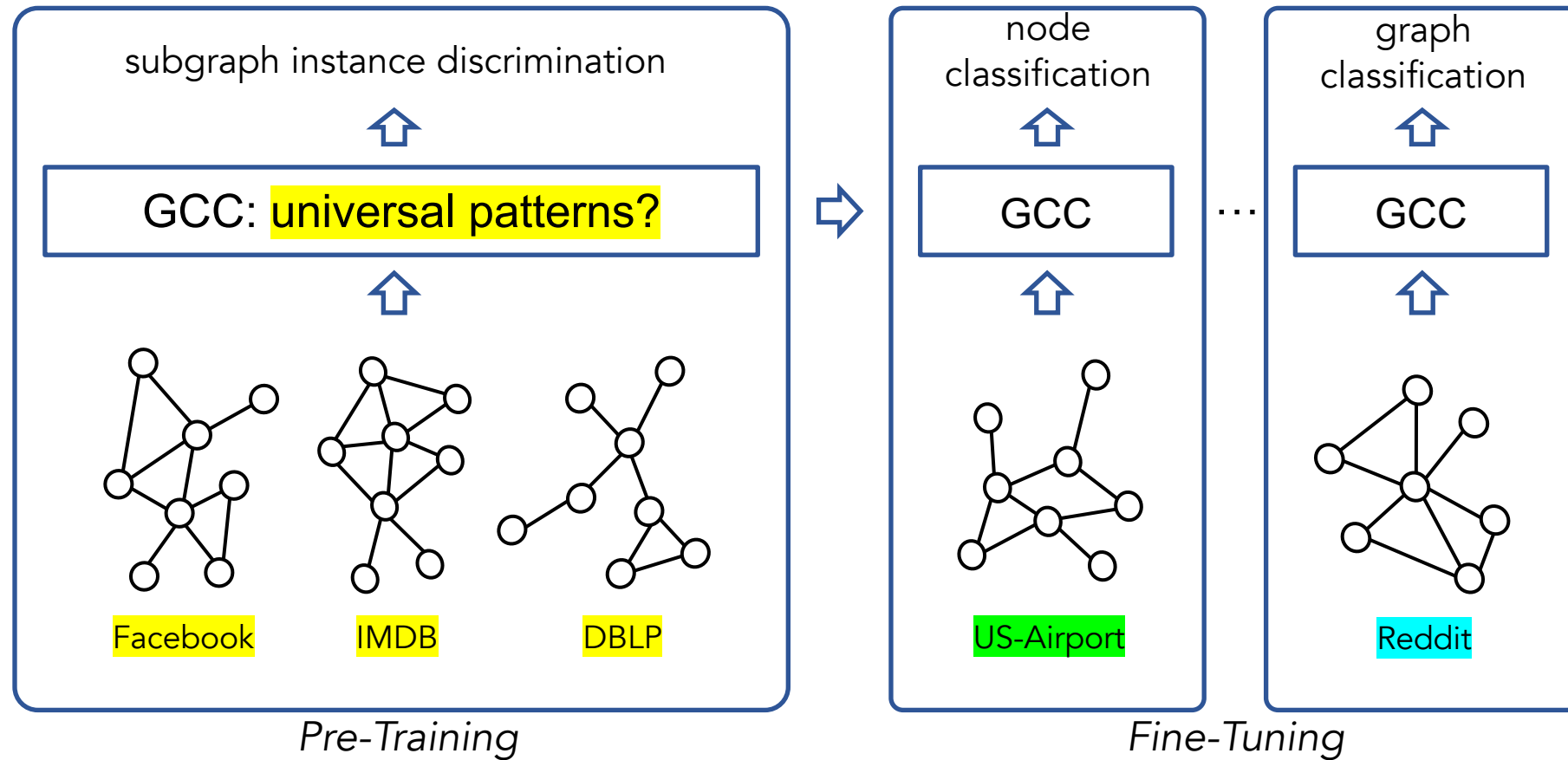


(a) **US-Airport:** $y = 0.4344x + 62.394$

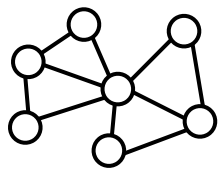
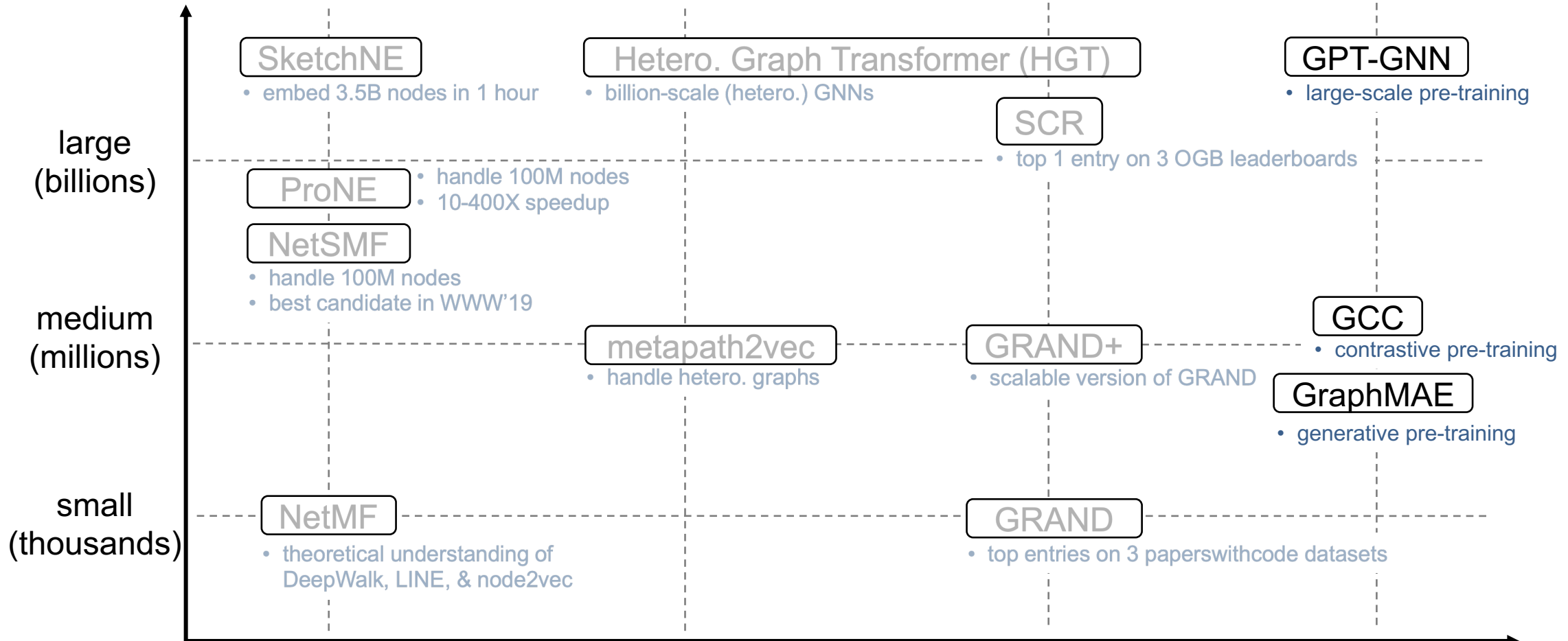


(b) **COLLAB:** $y = 0.8065x + 74.4737$

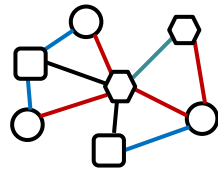
Does the pre-training of GNNs learn the **universal structural patterns** across networks?



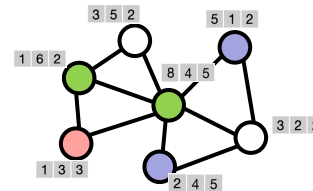
Graph Pre-Training and Self-Supervised Learning



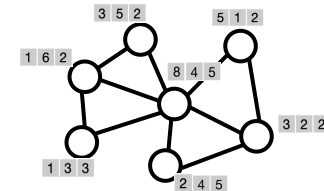
structure



heterogeneous structure / knowledge graph

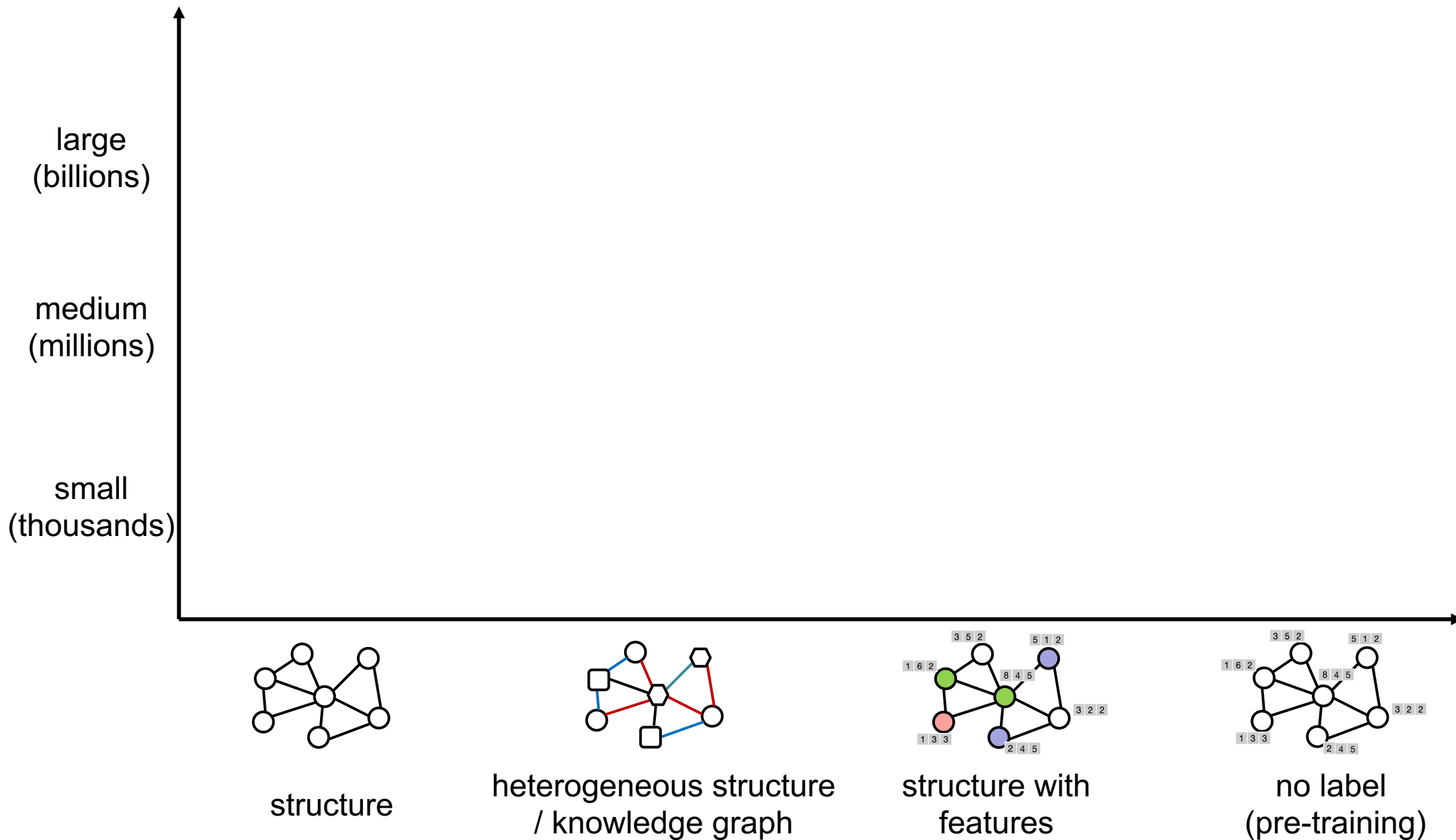


structure with features



no label (pre-training)

Graph Pre-Training and Self-Supervised Learning

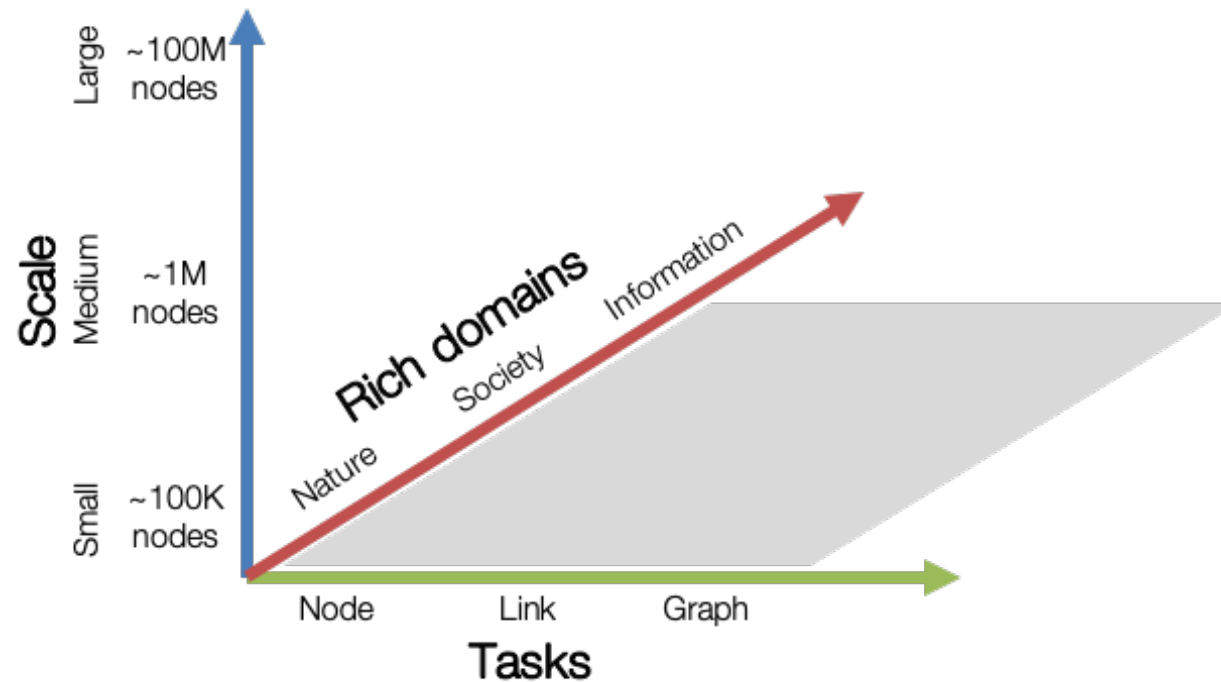


Why a Graph ML Benchmark?

- 1) Current datasets are **small**:
 - Too small to be realistic
 - Hard to reliably and rigorously evaluate algorithms
- 2) Evaluation protocol is **not unified**:
 - Every paper uses its own train/test split and metrics
 - Performance across papers is not comparable
- 3) Dataset splits follow **conventional random splits**:
 - Unrealistic for real-world applications
 - Accuracies are over-optimistic under conventional splits

Open Graph Benchmark (OGB)

Large-scale, realistic, and diverse benchmark datasets for graph ML.

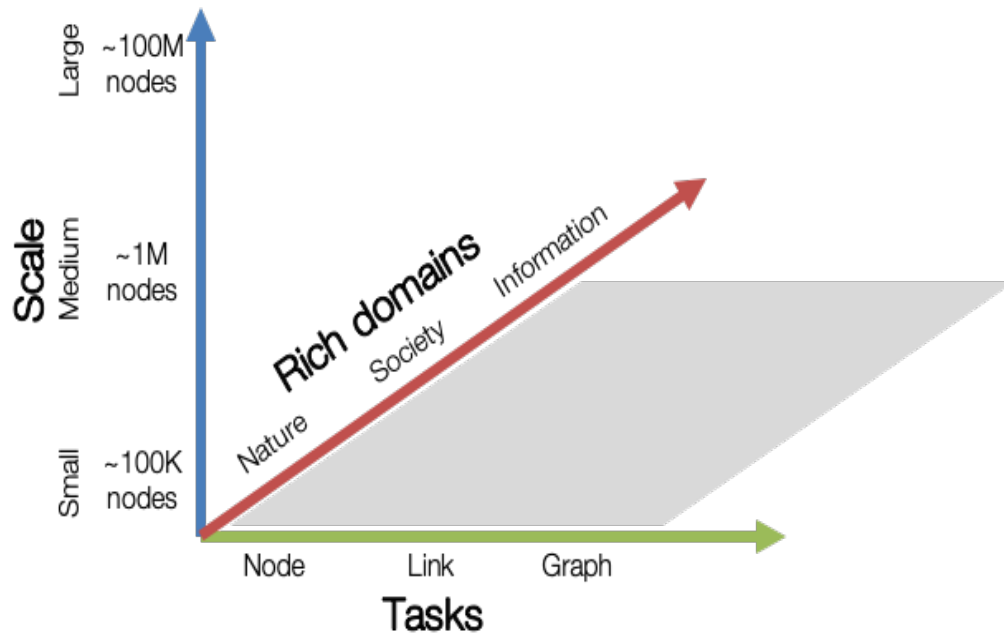


Paper: NeurIPS 2020

Leaderboards: <https://ogb.stanford.edu/>

Open Graph Benchmark (OGB)

- Covers diverse ML tasks, domains, and scales.
- Open to suggestion from the community.



Task		Node property prediction ogbn-		
Domain		Nature	Society	Information
Small			arxiv	
Medium		proteins	products	mag
Large			papers100M	

Task		Link property prediction ogbl-		
Domain		Nature	Society	Information
Small		ddi	collab	biokg
Medium		ppa	citation	wikikg
Large				

Task		Graph property prediction ogbg-		
Domain		Nature	Society	Information
Small		molhiv		
Medium		molpcba / ppa		code
Large				



Paper: NeurIPS 2020

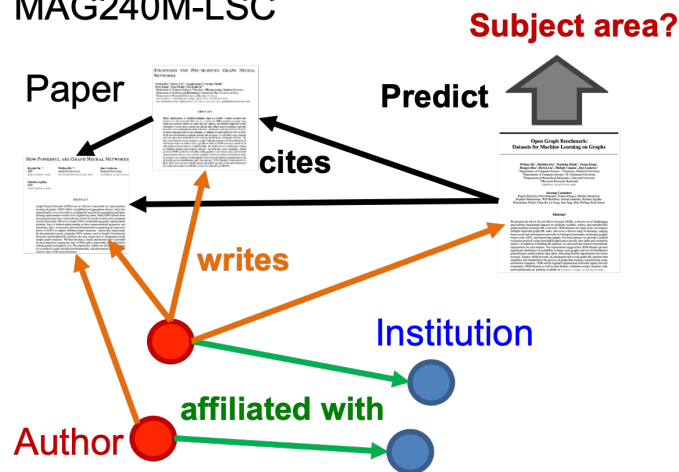
Leaderboards: <https://ogb.stanford.edu/>

Open Graph Benchmark (OGB)

- An end-to-end pipeline for graph ML research
 1. Large-scale datasets for **key task categories**
 - Node/link/graph property prediction
 2. **Data loader** for automatically downloading, processing, & splitting the datasets.
 - Compatible to Pytorch Geometric, DGL, & CogDL
 3. **Evaluator** for unified automatic evaluation.
- We envision OGB to be common, community-driven platform for graph ML research & teaching resource

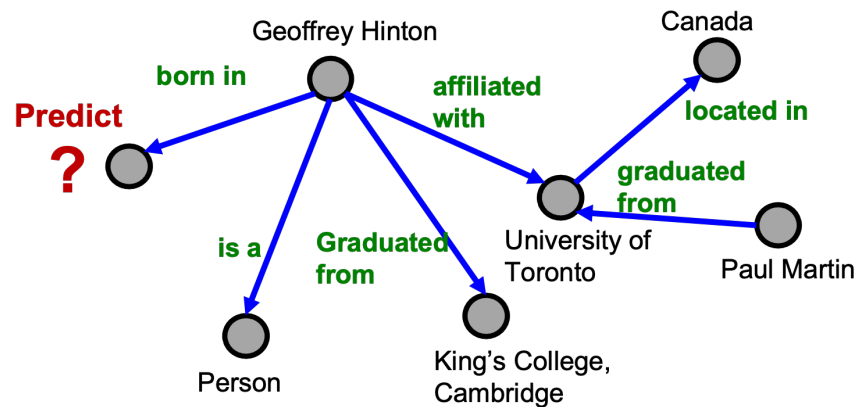


Node-level
MAG240M-LSC



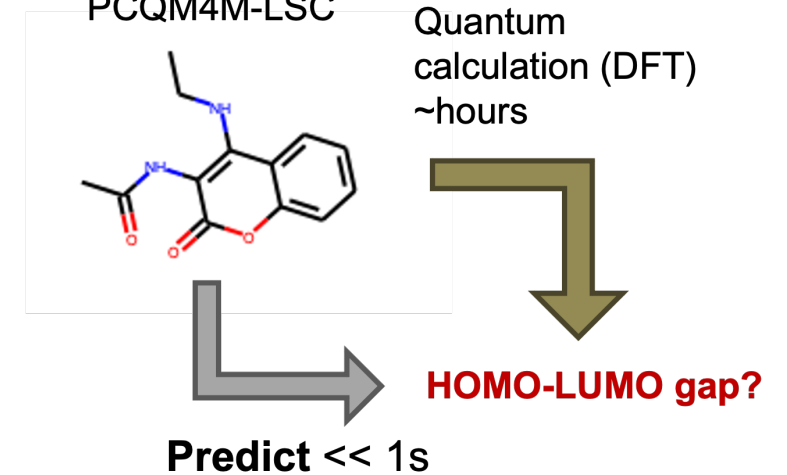
- Baidu
- DeepMind
- Synerise AI

Link-level
WikiKG90M-LSC



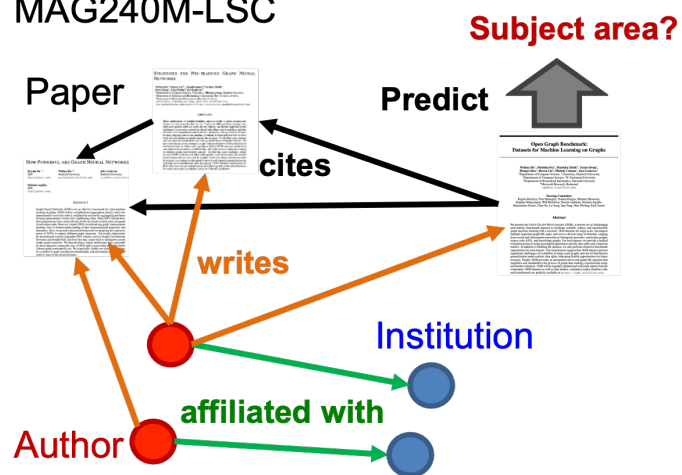
- Baidu
- Harbin Inst. of Tech.
- USTC

Graph-level
PCQM4M-LSC

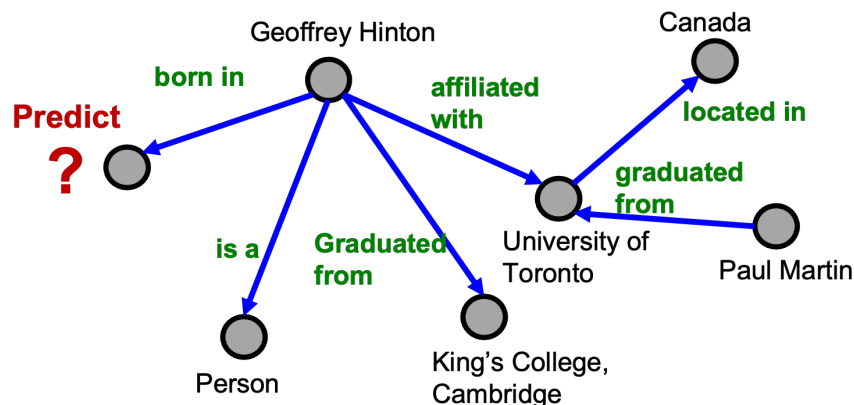


- MSR
- Baidu
- DeepMind

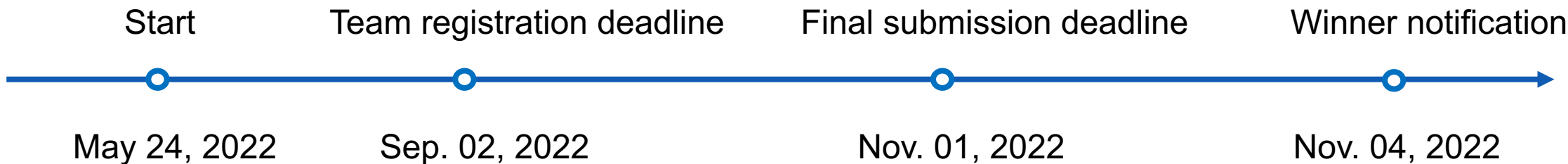
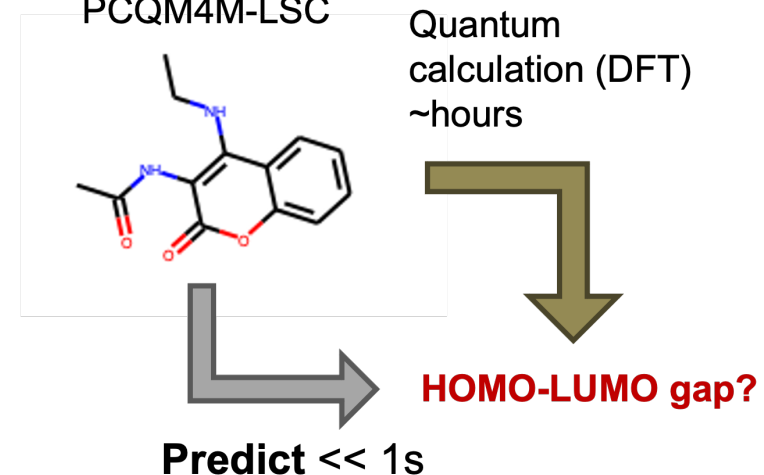
Node-level
MAG240M-LSC



Link-level
WikiKG90M-LSC



Graph-level
PCQM4M-LSC



CogDL.ai

You prepare the data, and
CogDL does **everything else.**



A Unified Trainer

CogDL integrates a unified trainer with decoupled modules for the GNN training. Based on this unique design, CogDL can provide extensive features such as hyperparameter optimization, distributed training, training techniques, and experiment management.



Efficient Sparse Operators

Efficiency is one of the most significant characteristics of CogDL. CogDL develops well-optimized sparse kernel operators to speed up the training of GNN models, enabling it become the most competitive graph libraries for efficiency.



Ease of Use

We provide simple APIs in CogDL such that users only need to write one line of code to train and evaluate any graph representation learning methods. In addition, CogDL also collects and maintains the state-of-the-art configurations, facilitating open, robust, and reproducible deep learning research on graphs.

Total Downloads

25,435

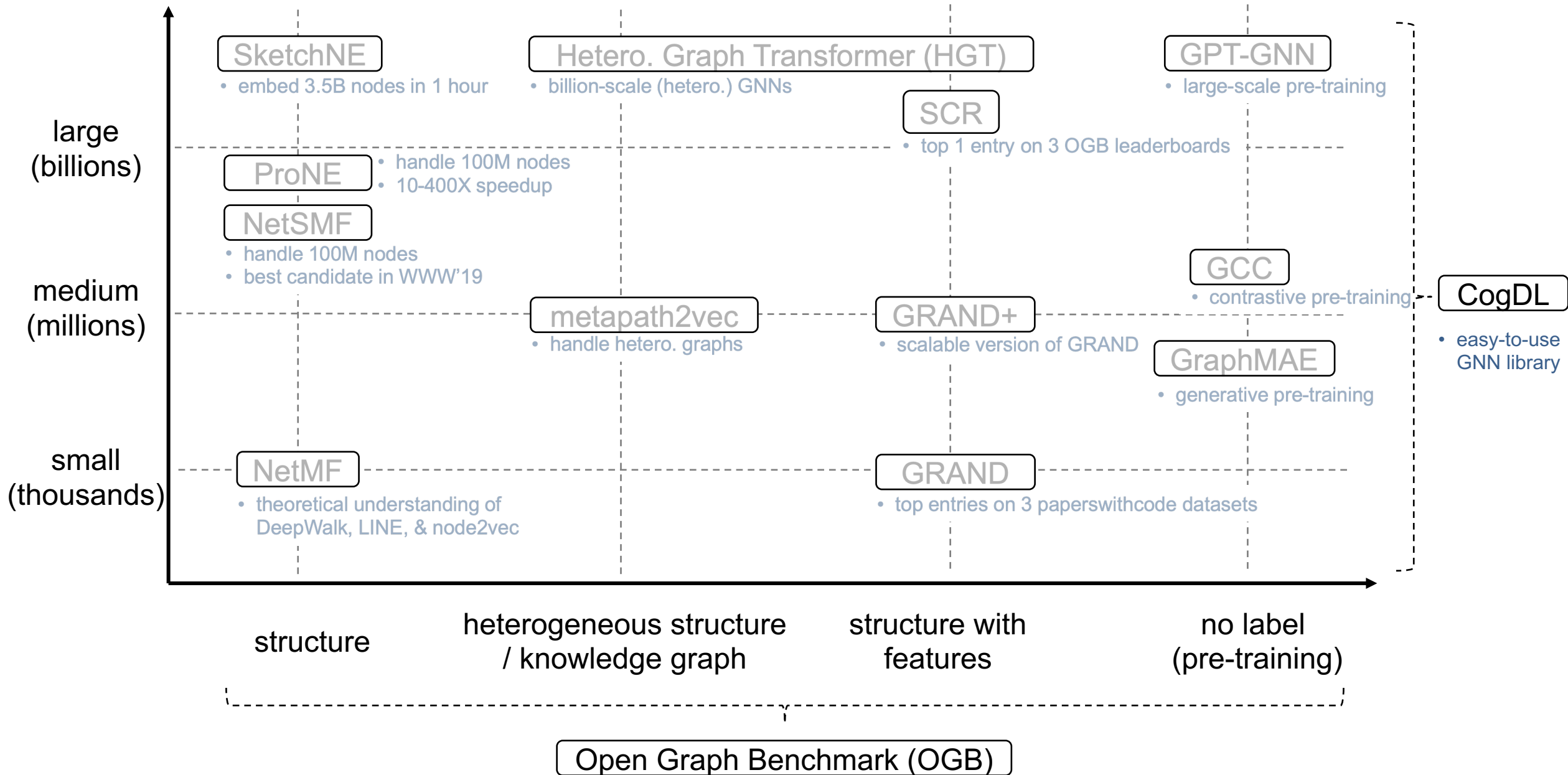
Total Stars

1,191

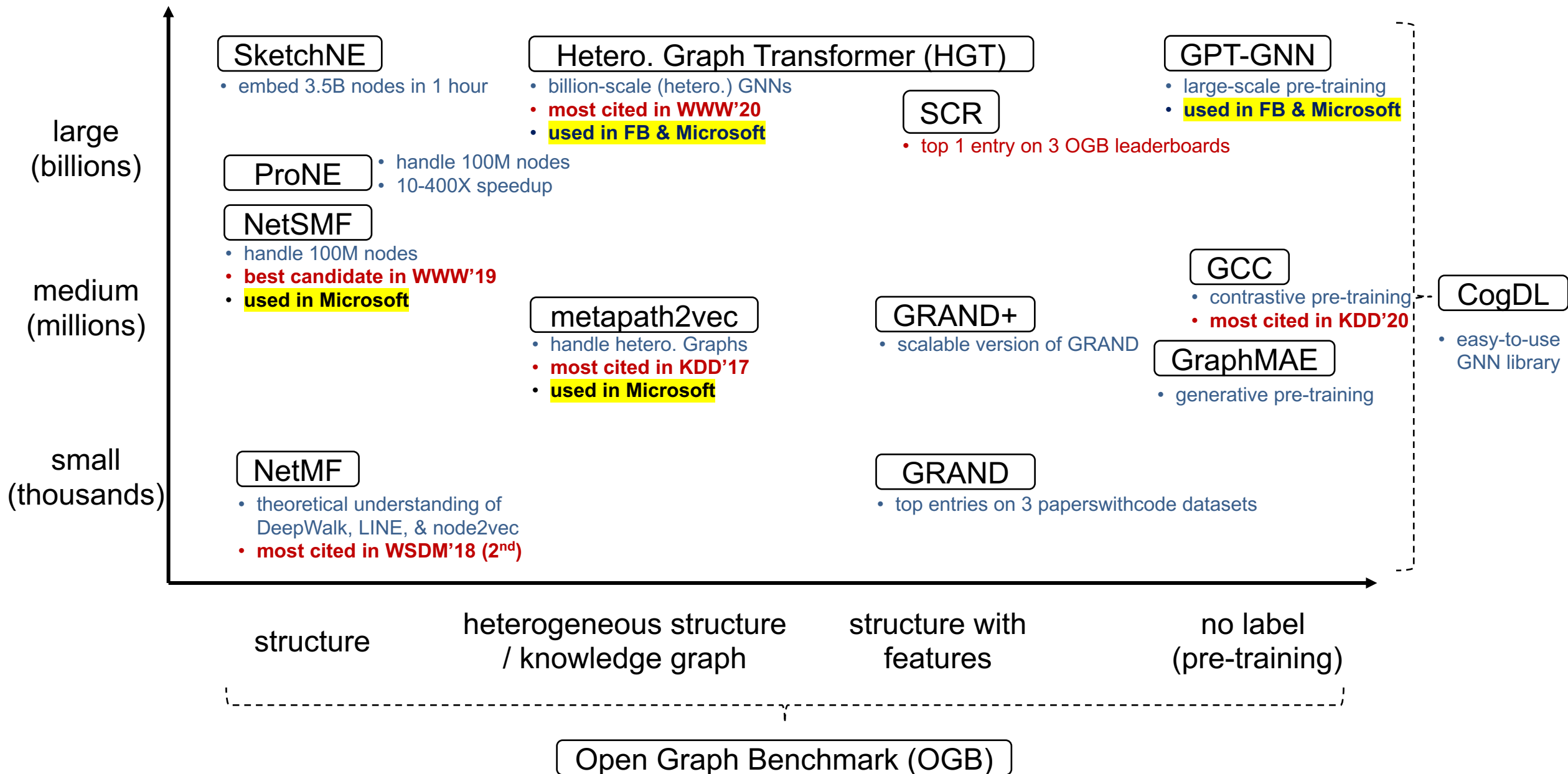
Total Forks

273

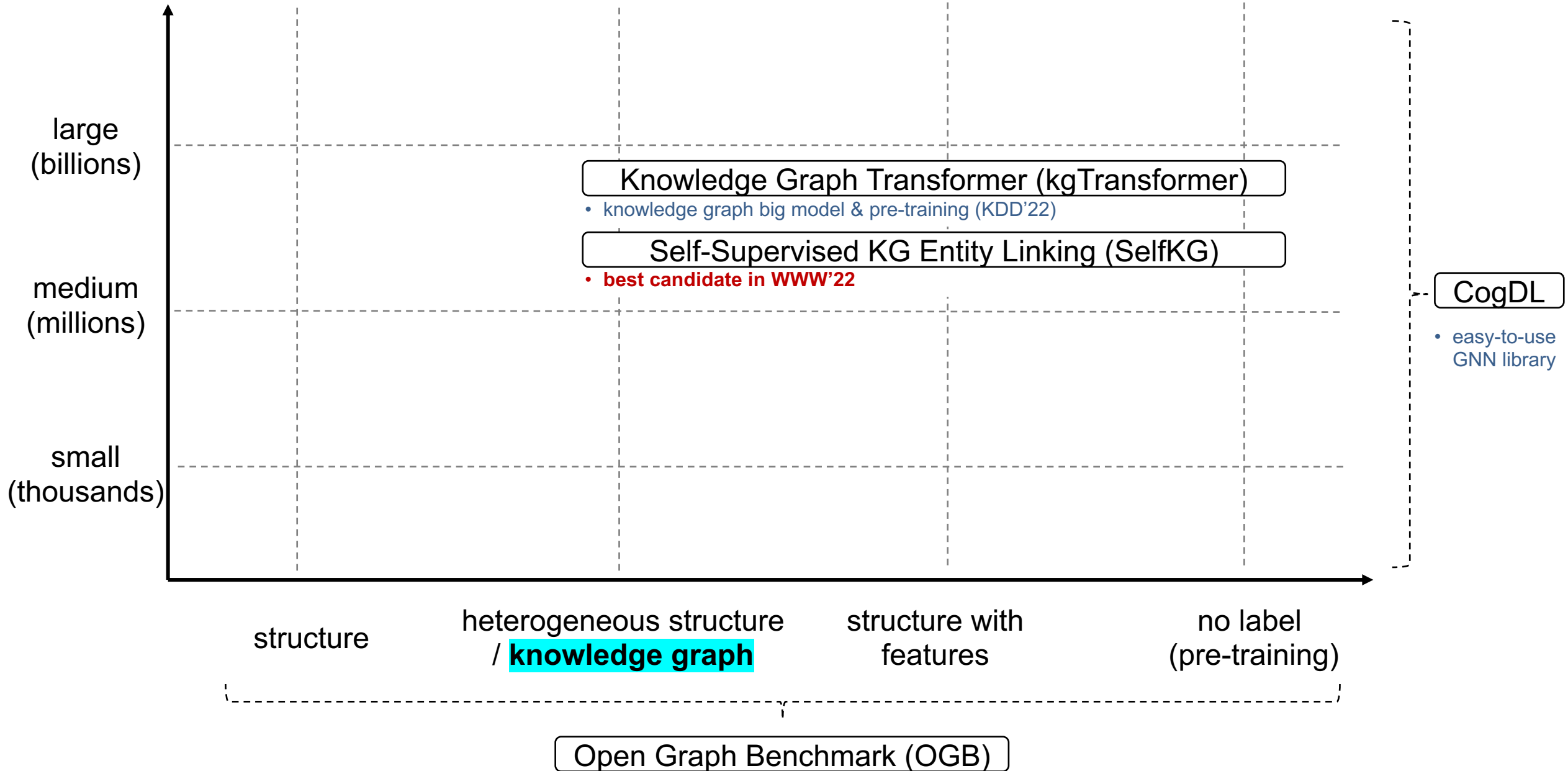
Open Data & Toolkit



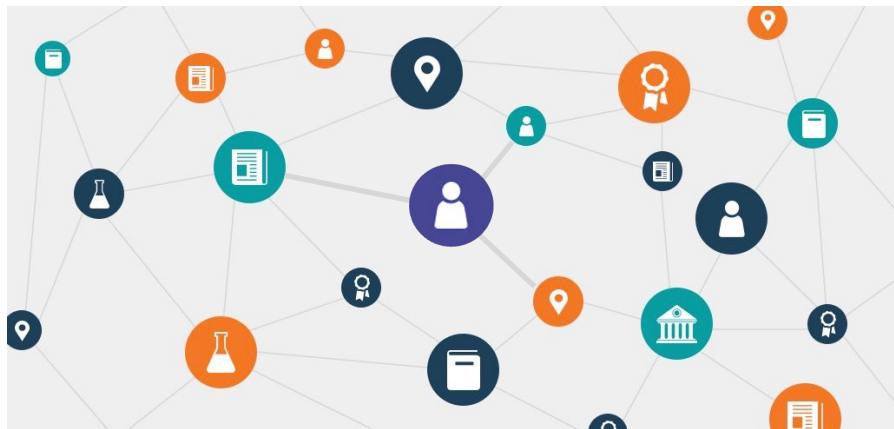
Graph Representation Learning and Pre-Training



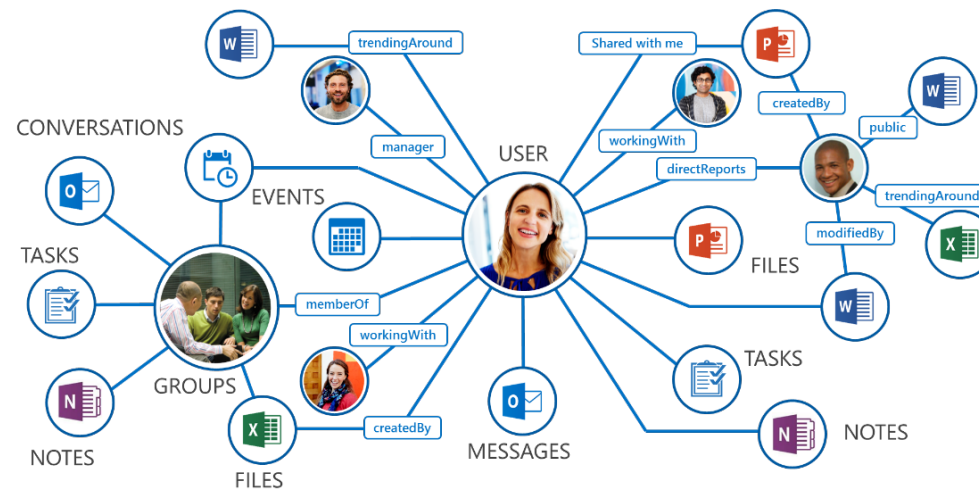
Graph Representation Learning and Pre-Training



Pre-Training with *Knowledge*



Academic Graph



Microsoft Office Graph

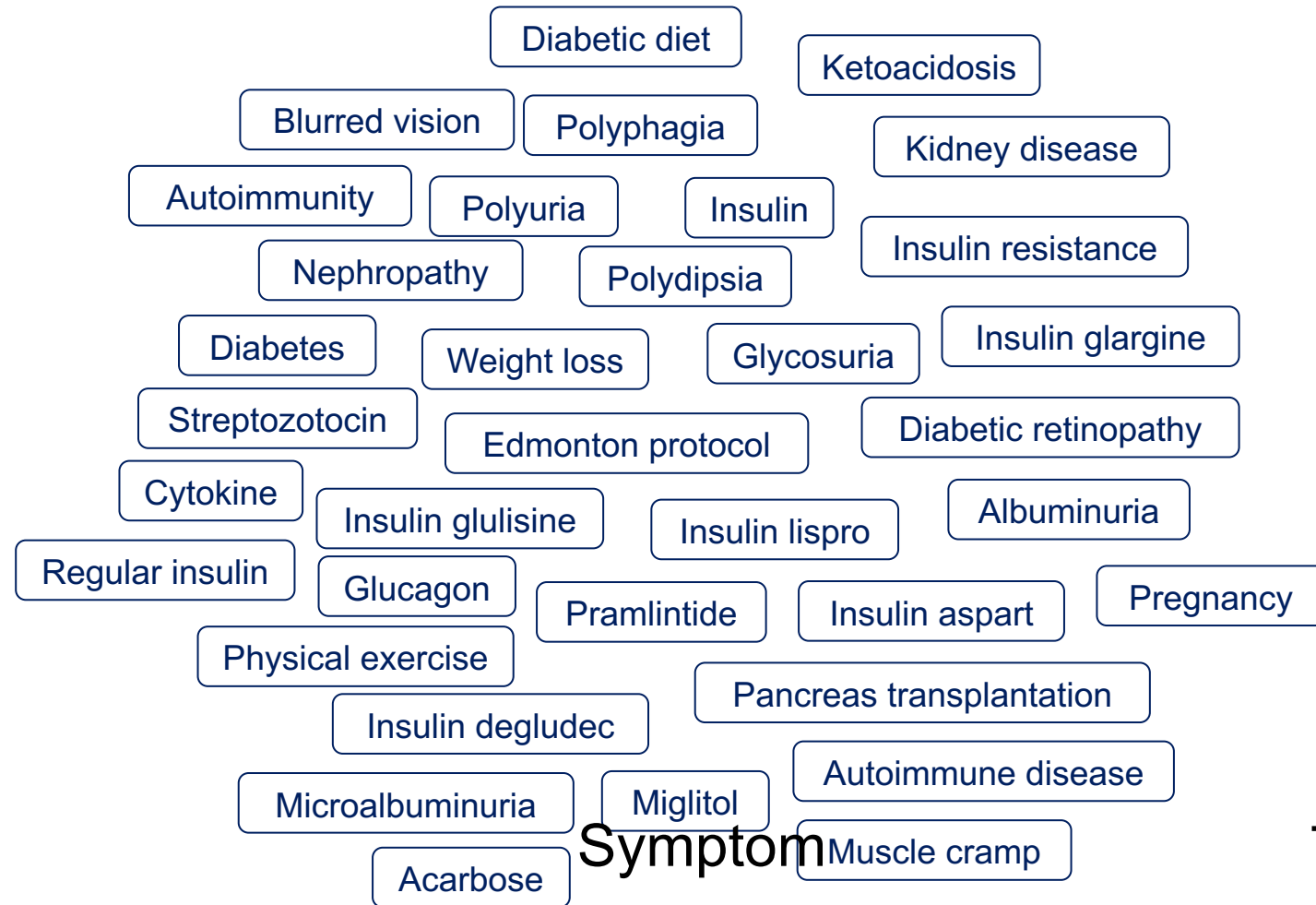


LinkedIn Economic Graph



Facebook Entity Graph

Neural Symbolic Reasoning

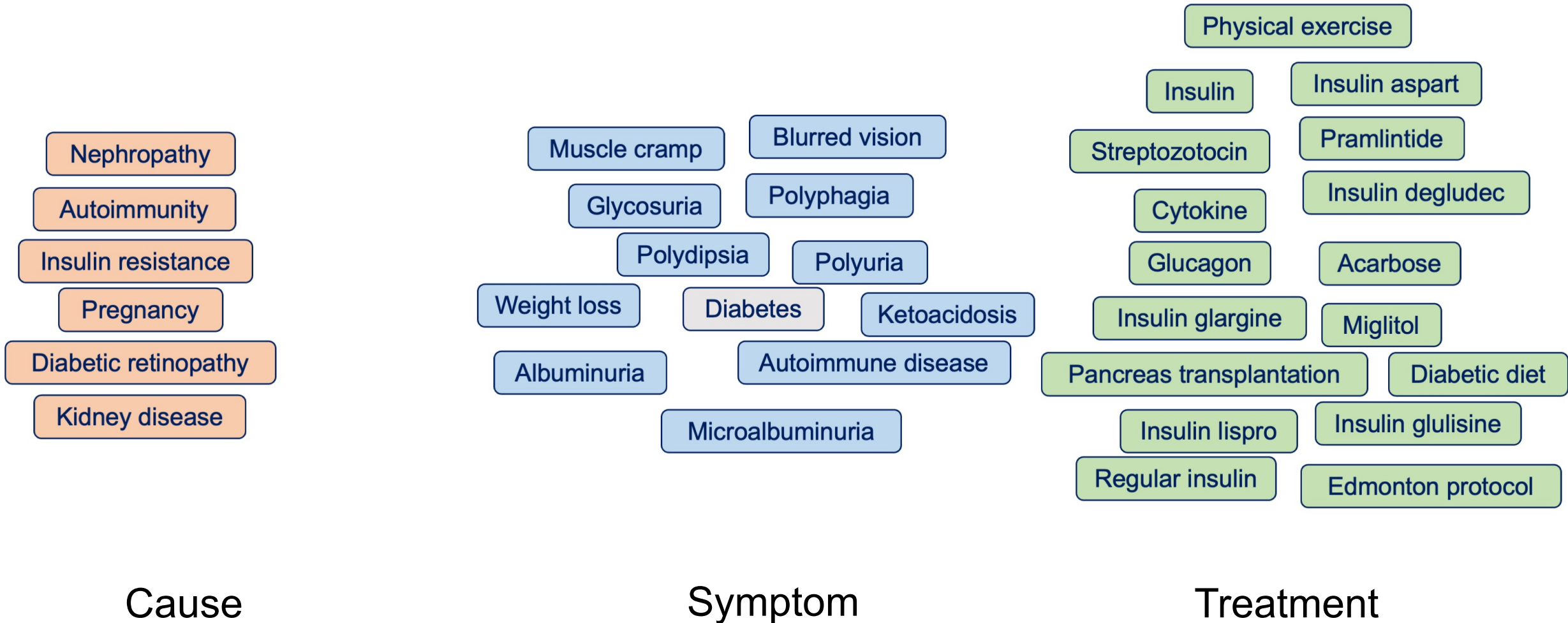


Cause

Symptom

Treatment

Neural Symbolic Reasoning



References

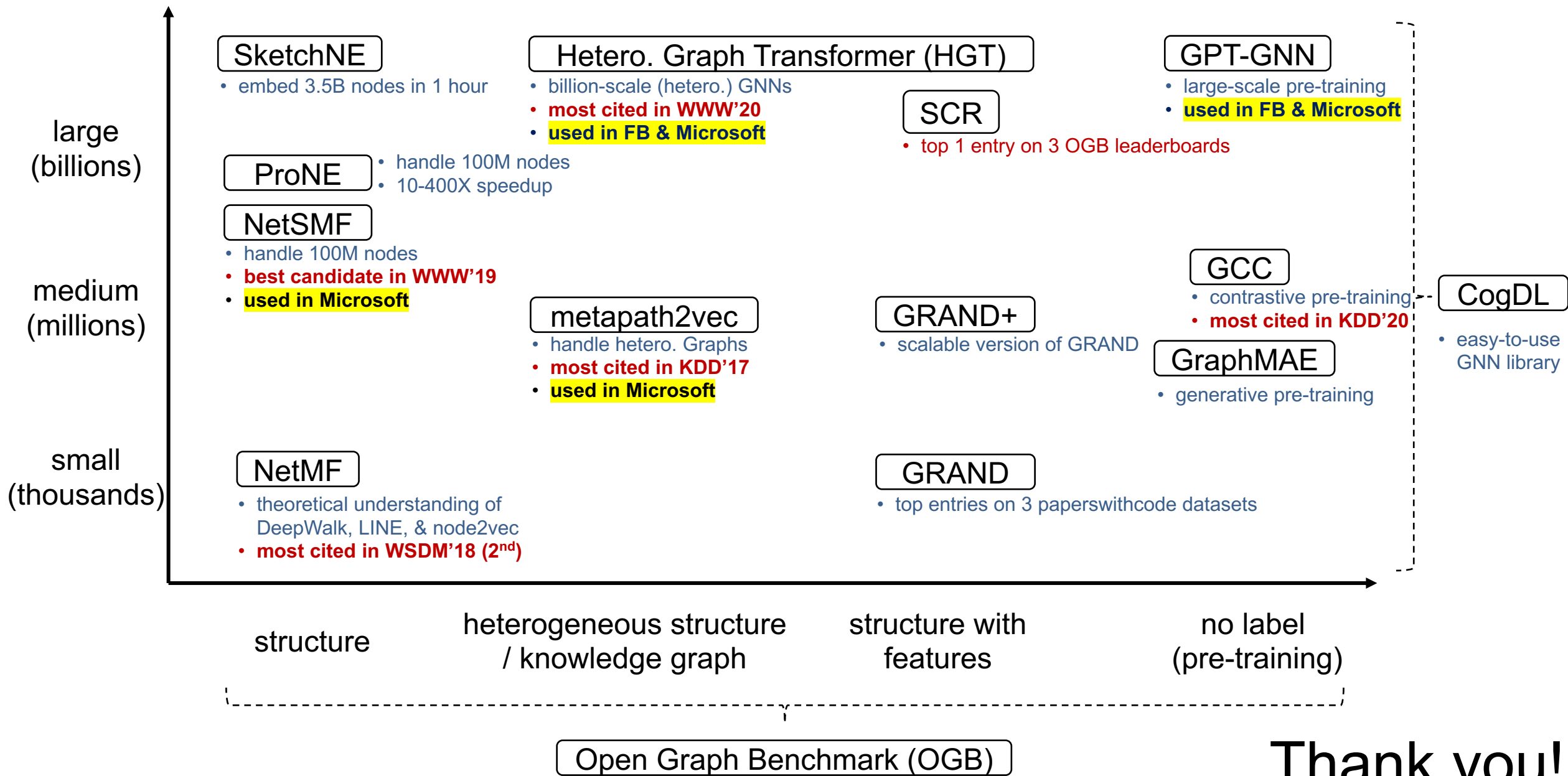
1. Zhenyu Hou, et al. *GraphMAE: Self-Supervised Masked Graph Autoencoders*. KDD 2022.
2. Xiao Liu, et al. *Mask and Reason: Pre-Training Knowledge Graph Transformers for Complex Logical Queries*. KDD 2022.
3. Xiao Liu, et al. *SelfKG: Self-Supervised Entity Alignment in Knowledge Graphs*. WWW 2022. **Best Paper Candidate**.
4. Wenzheng Feng, et al. *GRAND+: Scalable Graph Random Neural Networks*. WWW 2022.
5. Yukuo Cen, et al. *CogDL: A Unified Library for Graph Neural Networks*. <https://cogdl.ai/>.
6. Chenhui Zhang, et al. *SCR: Training Graph Neural Networks with Consistency Regularization*. arXiv.
7. Tinglin Huang, et al. *MixGCF: An Improved Training Method for Graph Neural Network-based Recommender Systems*. KDD 2021.
8. Xu Zou, et al. *TDGIA: Effective Injection Attacks on Graph Neural Networks*. KDD 2021.
9. Wenzheng Feng, et al. *Graph Random Neural Networks for Semi-Supervised Learning on Graphs*. NeurIPS 2020.
10. Weihua Hu et al. *Open Graph Benchmark: Datasets for Machine Learning on Graphs*. NeurIPS 2020.
11. Ziniu Hu et al. *GPT-GNN: Generative Pre-Training of Graph Neural Networks*. KDD 2020. *Top cited in KDD'20*
12. Jiezhong Qiu et al. *GCC: Graph Contrastive Coding for Graph Neural Network Pre-Training*. KDD 2020. *Most cited in KDD'20*
13. Ziniu Hu et al. *Heterogeneous Graph Transformer*. WWW 2020. *Most cited in WWW'20*
14. Yuxiao Dong et al. *Heterogeneous Network Representation Learning*. IJCAI 2020.
15. Jie Zhang et al. *ProNE: Fast and Scalable Network Representation Learning*. IJCAI 2019.
16. Jiezhong Qiu et al. *NetSMF: Large-Scale Network Embedding as Sparse Matrix Factorization*. WWW 2019. **Best Paper Candidate**
17. Jiezhong Qiu et al. *Network Embedding as Matrix Factorization: Unifying DeepWalk, LINE, PTE, and node2vec*. WSDM 2018. *2nd Most cited in WSDM'18*
18. Yuxiao Dong et al. *metapath2vec: Scalable Representation Learning for Heterogeneous Networks*. KDD 2017. *Most cited in KDD'17*

Papers & Data & Code available at <https://keg.cs.tsinghua.edu.cn/yuxiao/>

Thank you!

*Xiao Liu, Jiezhong Qiu, Ziniu Hu, Wenzheng Feng, Zhenyu Hou
Yukuo Cen, Weihua Hu, Jie Zhang, Chenhui Zhang, Yuyang Xie
Hao Ma, Kuansan Wang, Yizhou Sun, Jure Leskovec, Jie Tang*

Graph Representation Learning and Pre-Training



Thank you!