Graph Representation Learning:

Embedding, GNNs, and Pre-Training

Yuxiao Dong
https://ericdongyx.github.io/

Microsoft Research, Redmond

Joint Work with

Jiezhong Qiu
Tsinghua
(Jie Tang)

Ziniu Hu
UCLA
(Yizhou Sun)

Hongxia Yang
Alibaba

Jing Zhang
Renmin U. of China

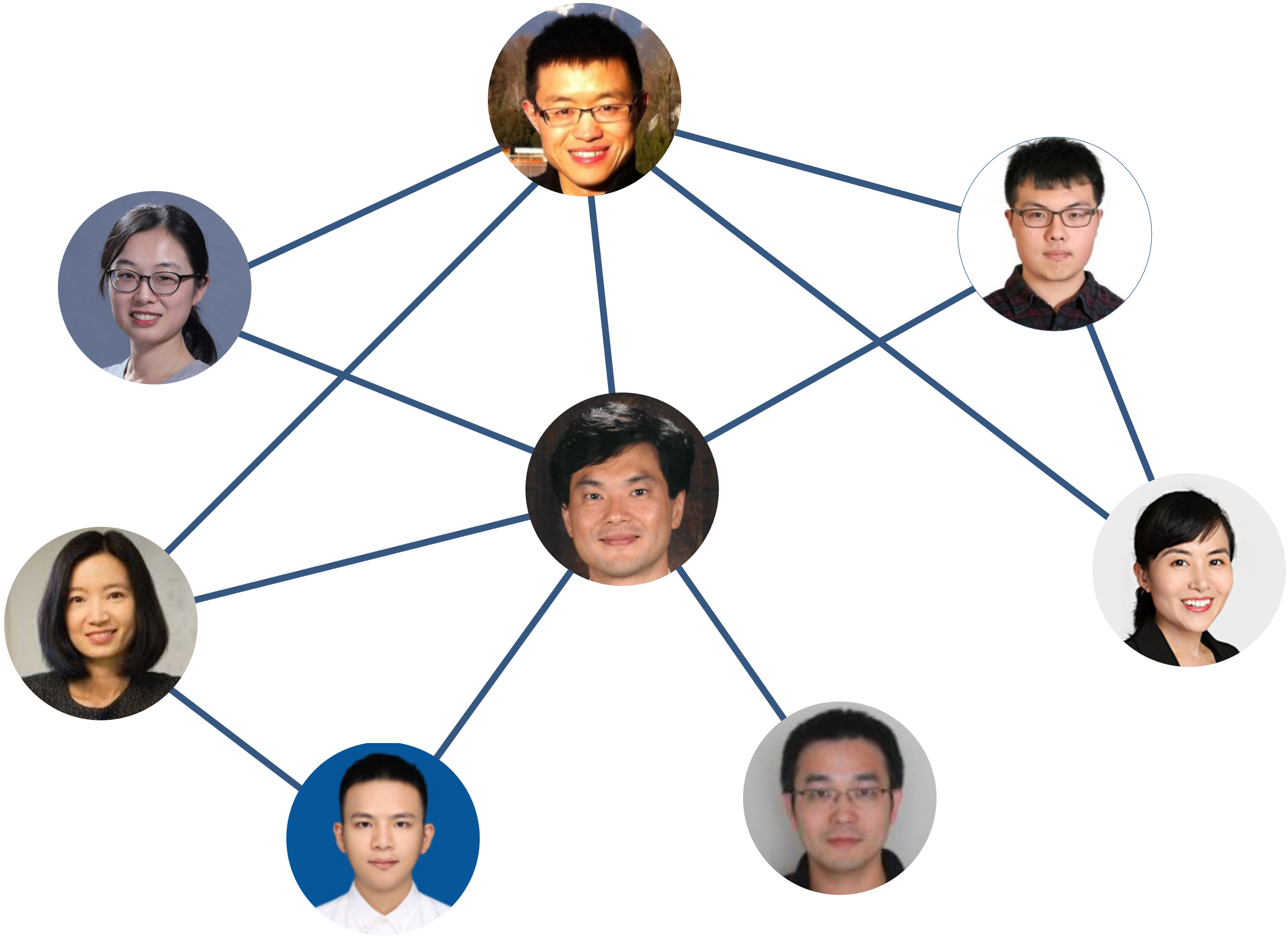
Jie Tang
Tsinghua

Yizhou Sun
UCLA

Hao Ma
Facebook AI

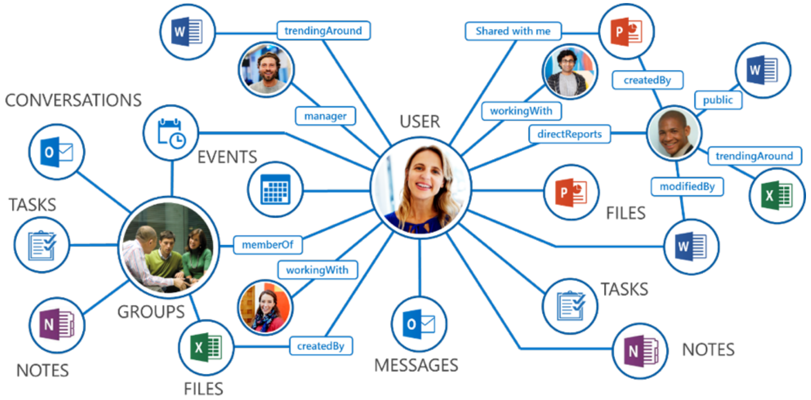
Kuansan Wang
Microsoft Research

Why Graphs?

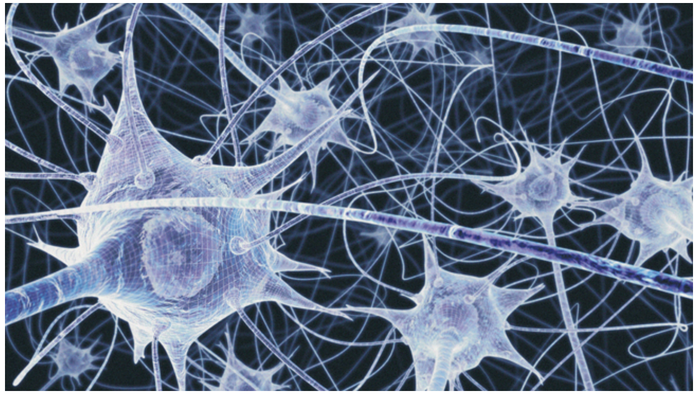


Graphs

Academic Graph

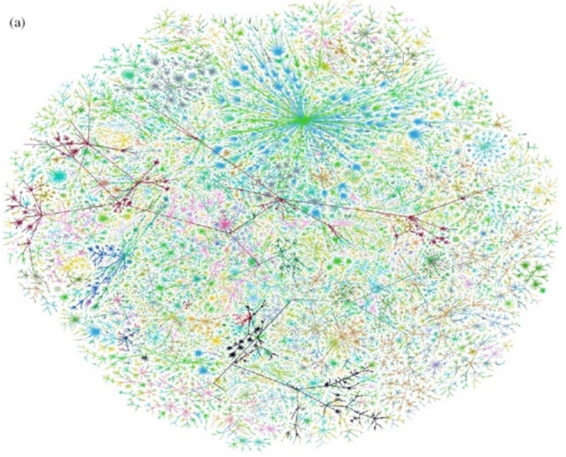


Office/Social Graph

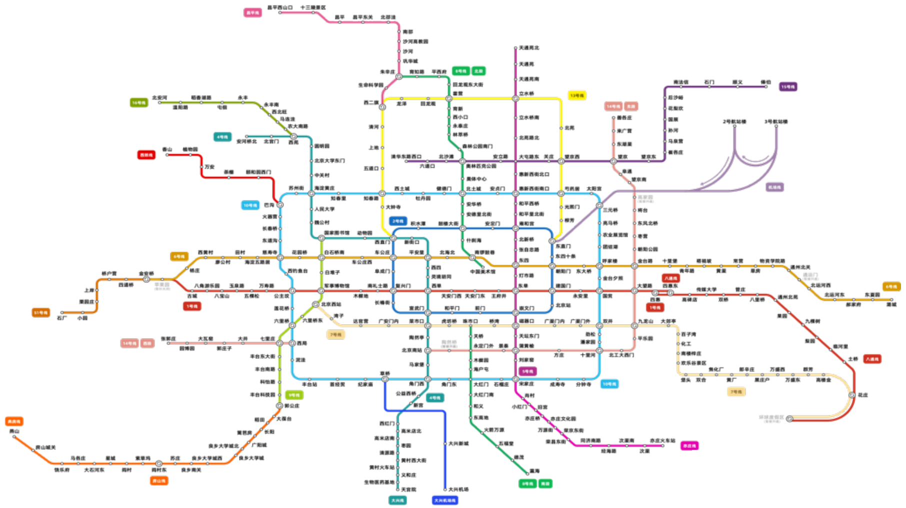


Biological Neural Networks

Knowledge Graph



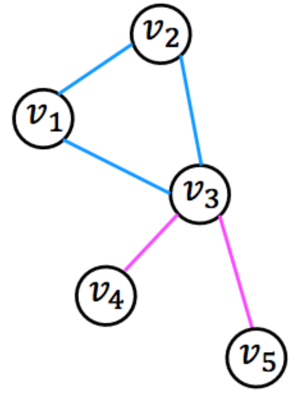
Internet

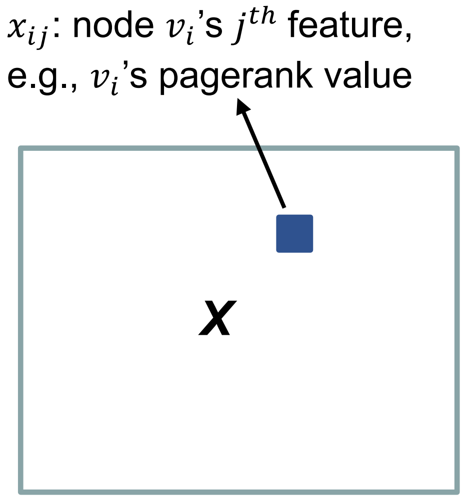


Transportation

figure credit: Web

The Graph Mining Paradigm





hand-crafted feature matrix

Graph & Network applications

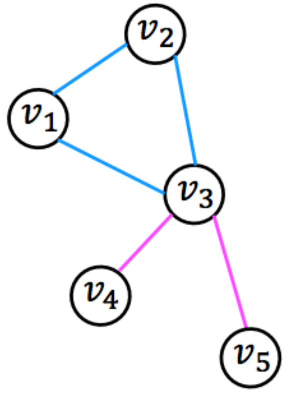
- Node classification
- Link prediction
- Community detection
- Anomaly detection
- Social influence
- Graph evolution
- … …

feature **engineering**

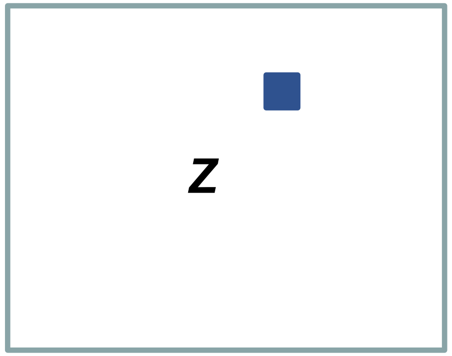
machine learning models

Structural Diversity and Homophily: A Study Across More Than One Hundred Big Networks. KDD 2017.

Graph Representation Learning



⇨



~~hand-crafted~~ **latent** feature matrix

⇨

Graph & Network applications
- Node classification
- Link prediction
- Community detection
- Anomaly detection
- Social influence
- Graph evolution
- … …

Feature ~~engineering~~ **learning**

machine learning models

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

Application: Embedding Heterogeneous Academic Graph

225,572,477
Papers

244,499,947
Authors

664,891
Topics

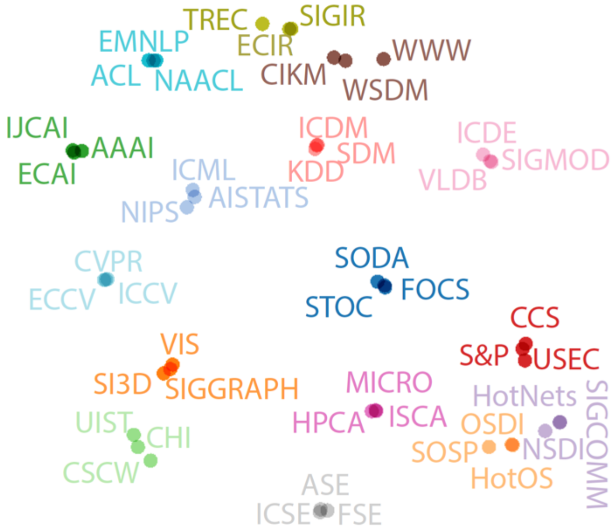
4,397
Conferences

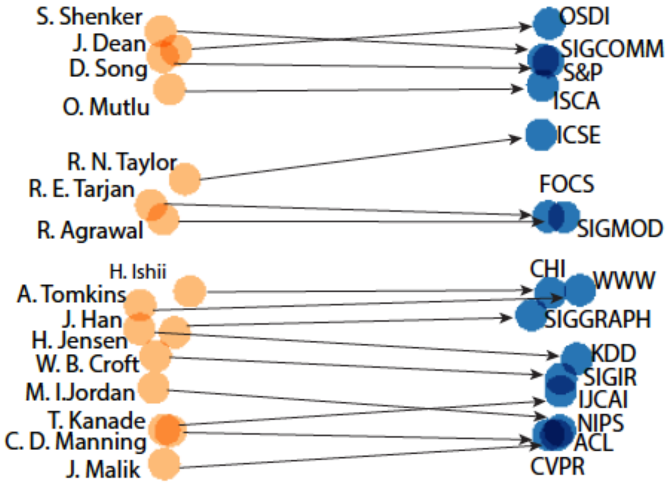
48,758
Journals

25,554
Institutions

Academic Graph

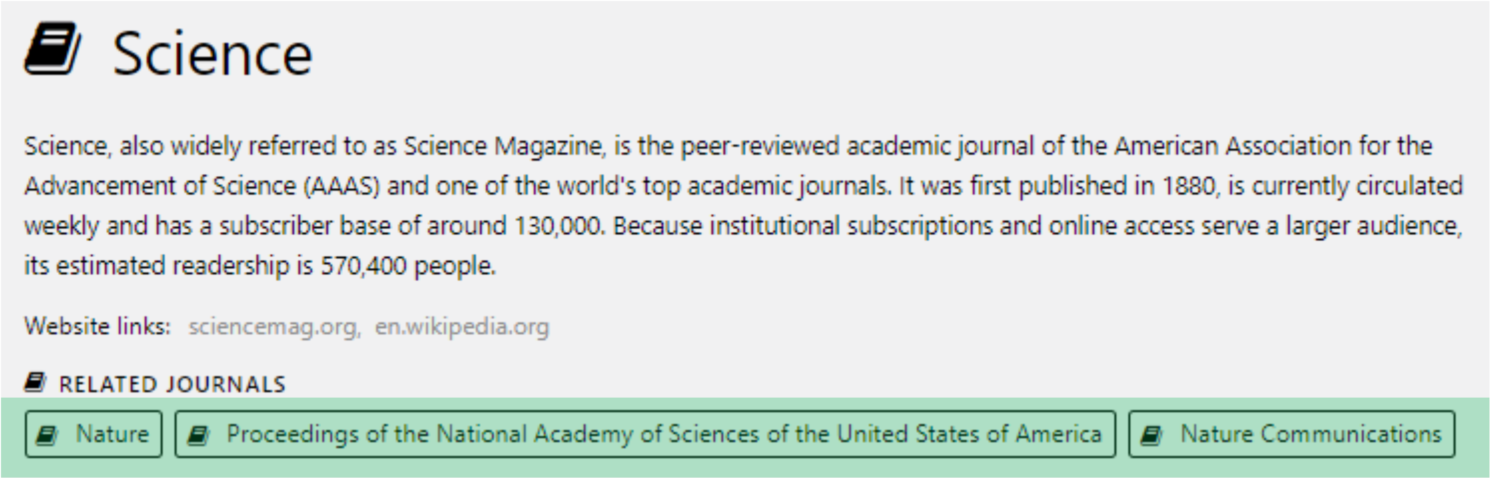
Graph Representation Learning

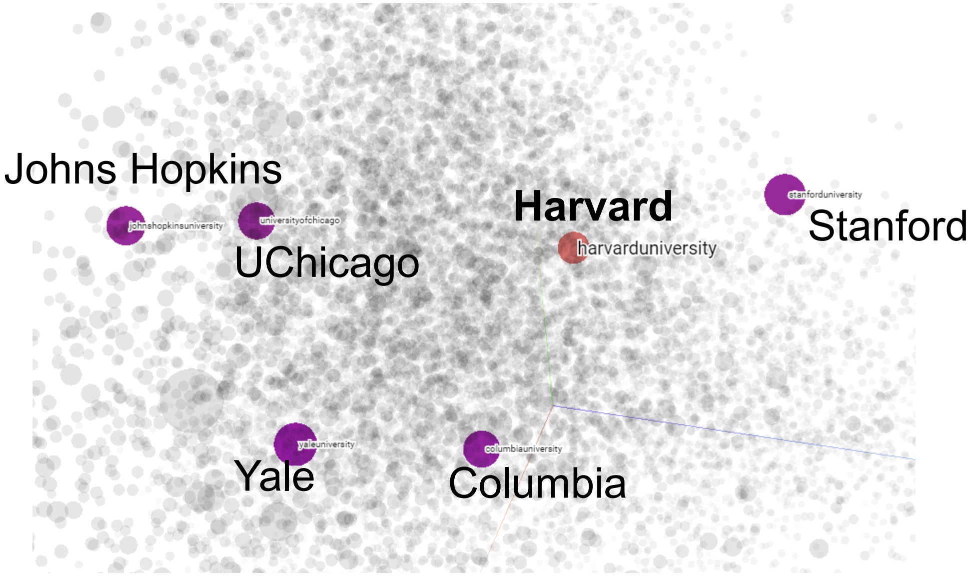




1. https://academic.microsoft.com/
2. Kuansan Wang et al. Microsoft Academic Graph: When experts are not enough. Quantitative Science Studies 1 (1), 396-413, 2020
3. Dong et al. metapath2vec: scalable representation learning for heterogeneous networks. In KDD 2017.
4. Code & data for metapath2vec: https://ericdongyx.github.io/metapath2vec/m2v.html

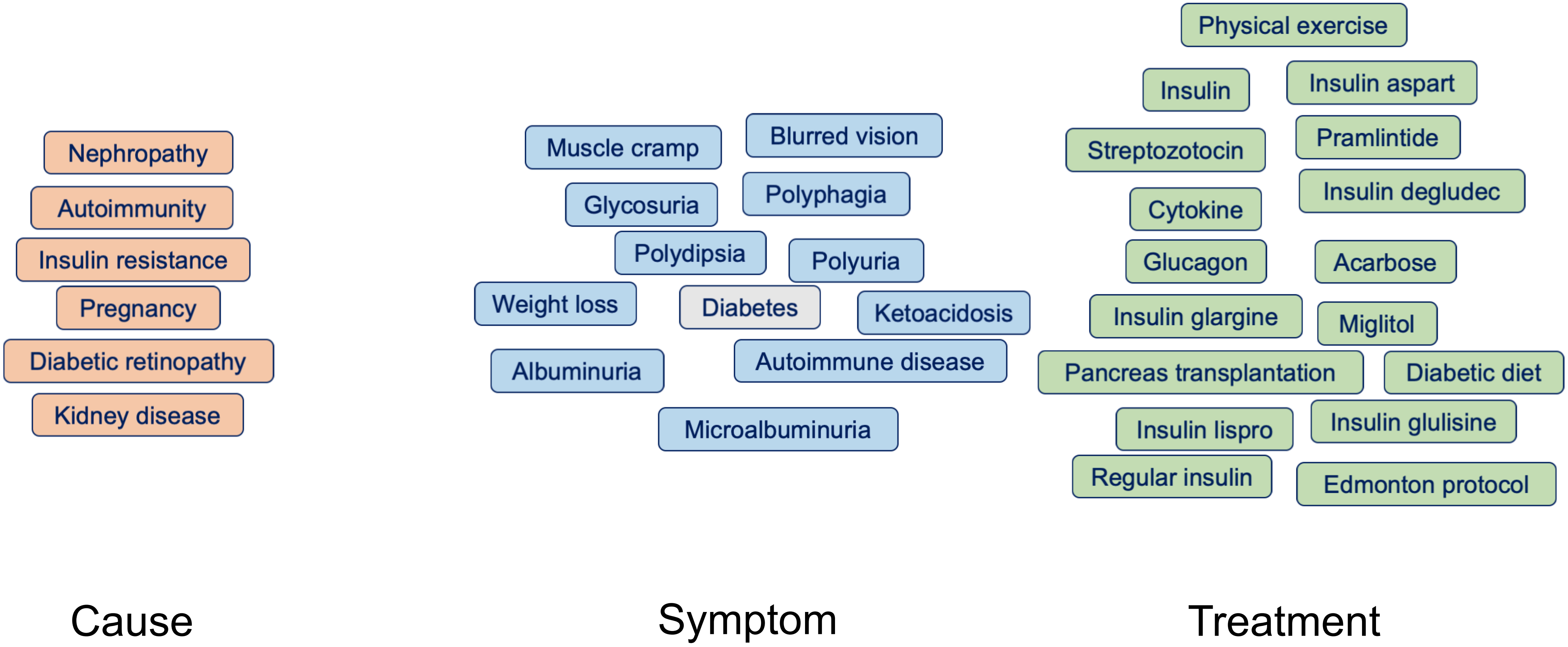
Application: Similarity Search & Recommendation



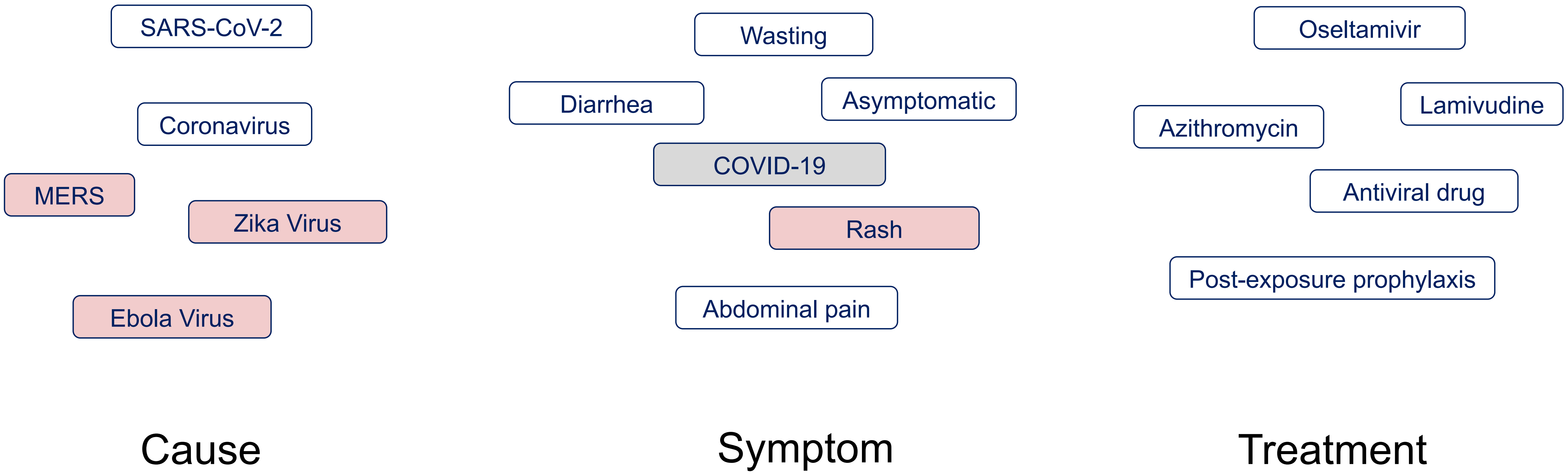


1. https://academic.microsoft.com/
2. Kuansan Wang et al. Microsoft Academic Graph: When experts are not enough. Quantitative Science Studies 1 (1), 396-413, 2020
3. Dong et al. metapath2vec: scalable representation learning for heterogeneous networks. In KDD 2017.
4. Code & data for metapath2vec: https://ericdongyx.github.io/metapath2vec/m2v.html

Application: Reasoning about **Diabetes** from MAG



Application: Reasoning about **COVID-19** from MAG

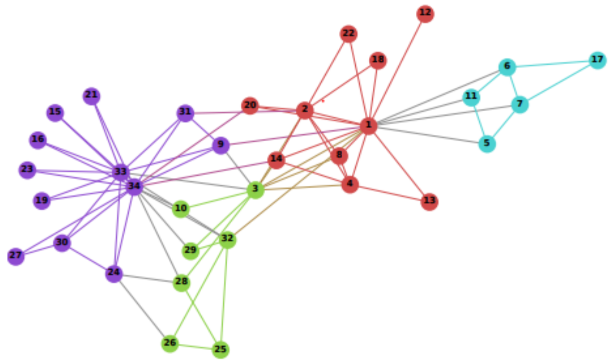


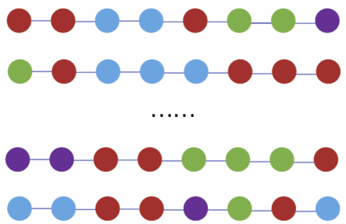
Graph Representation Learning

Network Embedding ⇔ **Matrix Factorization** ⇨ **GNNs** ⇨ **Pre-Training**

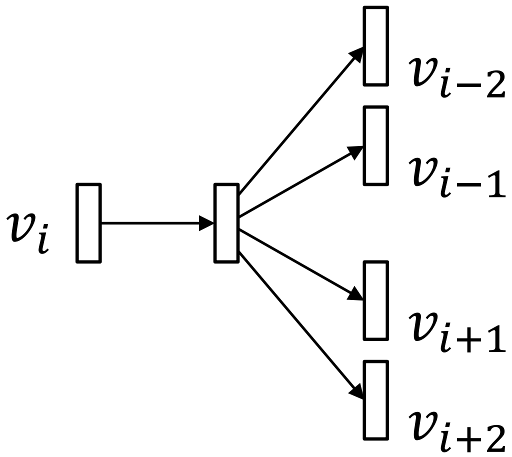
Network Embedding

Feature **learning**

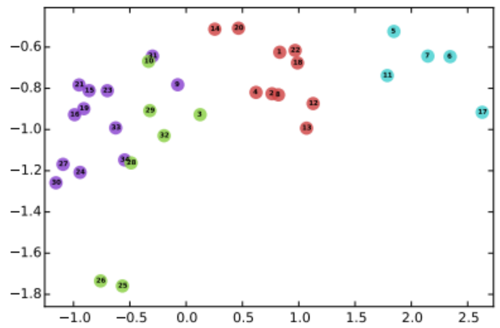




Sequences of objects
- Words in Text
- Nodes in graphs



Skip-Gram



1. Mikolov, et al. Efficient estimation of word representations in vector space. In *ICLR 2013*.
2. Perozzi et al. DeepWalk: Online learning of social representations. In *KDD' 14*, pp. 701–710.

Distributional Hypothesis of Harris

- **Word embedding**: words in similar contexts have similar meanings (e.g., skip-gram in word embedding)

- **Node embedding**: nodes in similar structural contexts are similar
  - DeepWalk: structural contexts are defined by co-occurrence over random walk paths

Harris, Z. (1954). Distributional structure. *Word*, 10(23): 146-162.

hide

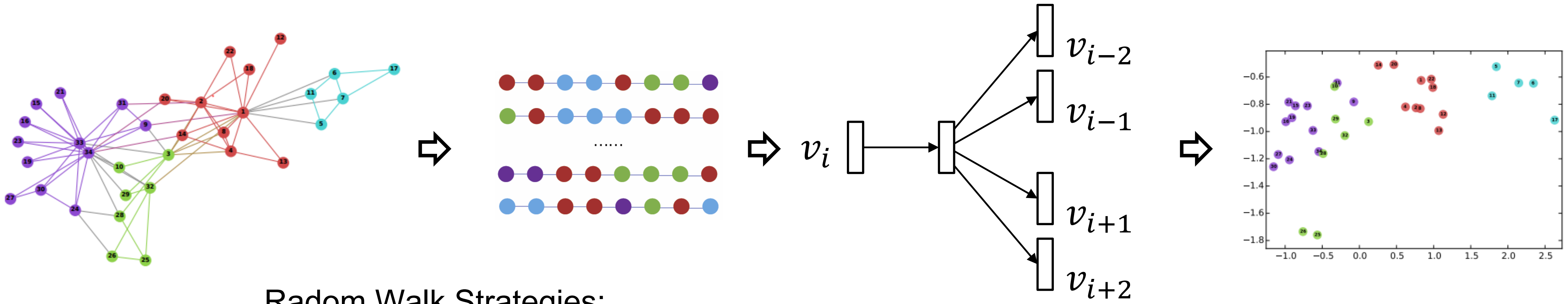
The Objective

$$\mathcal{L} = \sum_{v \in V} \sum_{c \in N_{rw}(v)} -\log(P(c|v)) \quad \Leftarrow \quad p(c|v) = \frac{\exp(\boldsymbol{z}_v^\top \boldsymbol{z}_c)}{\sum_{u \in V} \exp(\boldsymbol{z}_v^\top \boldsymbol{z}_u)}$$

$\mathcal{L}$ → to maximize the likelihood of node co-occurrence on a random walk path

$\boldsymbol{z}_v^\top \boldsymbol{z}_c$ → the possibility that node v and context c appear on a random walk path

Network Embedding: Random Walk + Skip-Gram



Radom Walk Strategies:

- DeepWalk (walk length > 1)
- LINE (walk length = 1)
- PTE (walk length = 1)
- node2vec (biased random walk)
- metapath2vec (heterogeneous rw)

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.**
4. Dong et al. **metapath2vec**: scalable representation learning for heterogeneous networks. In *KDD* 2017. **Most Cited Paper in KDD'17.**

Graph Representation Learning

Network Embedding ⇔ **Matrix Factorization** ⇒ **GNNs** ⇒ **Pre-Training**

- DeepWalk
- LINE
- Node2vec
- PTE
- …
- metapath2vec

hide

NetMF: Network Embedding as Matrix Factorization

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

$\boldsymbol{A}$ Adjacency matrix

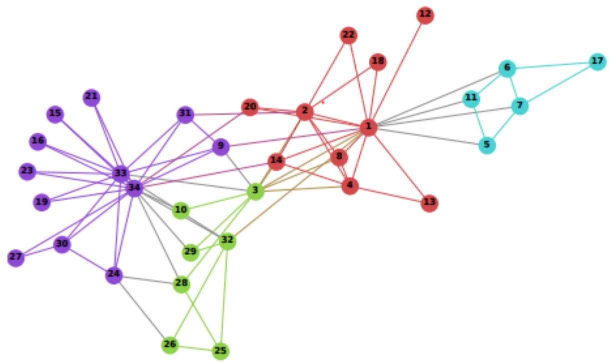
$\boldsymbol{D}$ Degree matrix

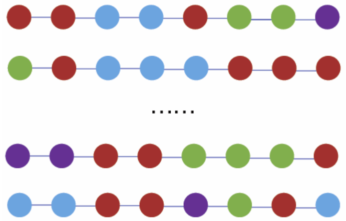
$$vol(G)=\sum_{i}\sum_{j}A_{ij}$$

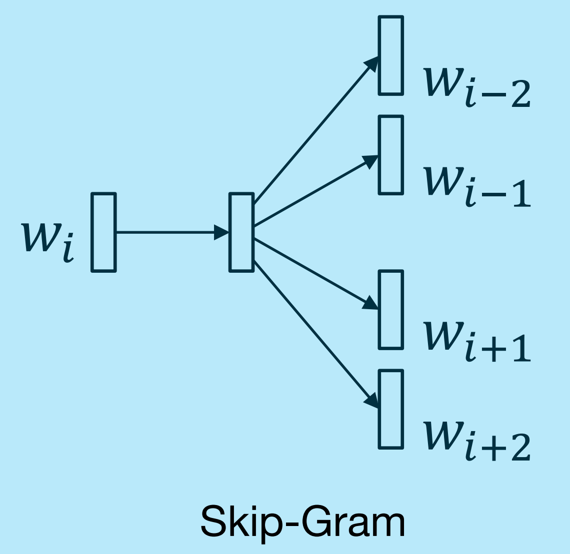
b: #negative samples
T: context window size

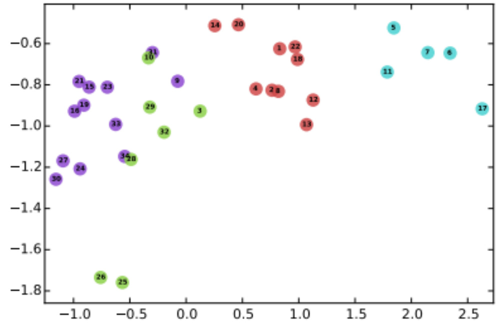
1. Qiu et al. Network embedding as matrix factorization: unifying deepwalk, line, pte, and node2vec. In *WSDM'18.*

Understanding Random Walk + Skip Gram









Graph Language

- G: graph
- $\boldsymbol{A}$**:** adjacency matrix
- $\boldsymbol{D}$:degree matrix
- $vol(G)$: volume of G

?

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

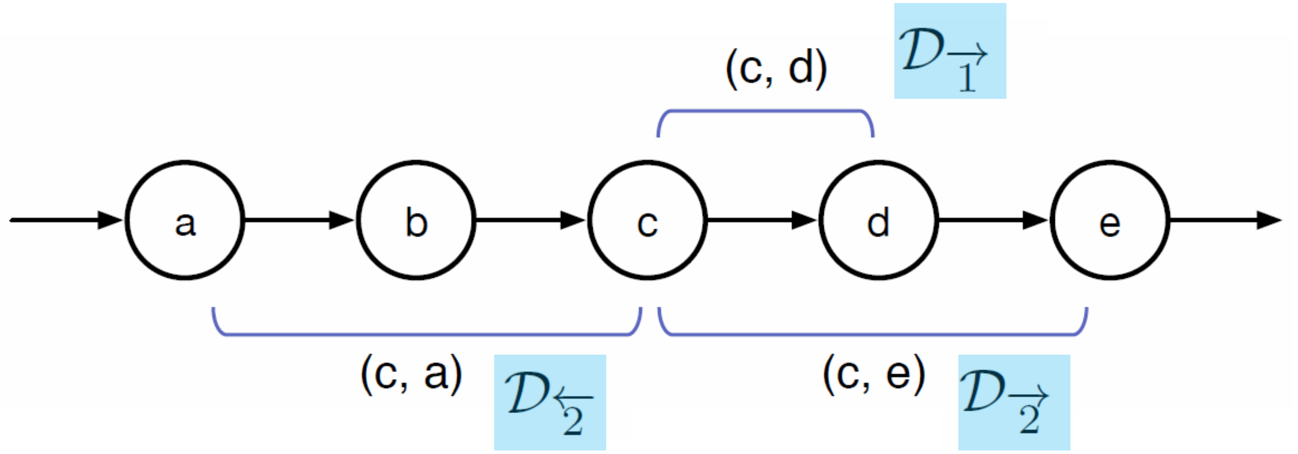
NLP Language

- #(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

Levy and Goldberg. Neural word embeddings as implicit matrix factorization. In *NIPS 2014*

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

Distinguish **direction** and **distance**

- Partition the multiset $\mathcal{D}$ into several sub-multisets according to the way in which each node and its context appear in a random walk node sequence.
- More formally, for $r = 1, 2, \cdots, T$, 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

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

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

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

$$\frac{\#(w,c)_{\overrightarrow{r}}}{|\mathcal{D}_{\overrightarrow{r}}|} \xrightarrow{p} \frac{d_w}{\mathrm{vol}(G)}(\boldsymbol{P}^r)_{w,c} \qquad \frac{\#(w,c)_{\overleftarrow{r}}}{|\mathcal{D}_{\overleftarrow{r}}|} \xrightarrow{p} \frac{d_c}{\mathrm{vol}(G)}(\boldsymbol{P}^r)_{c,w}$$

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

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

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

$$= \frac{\mathrm{vol}(G)}{2T}\left(\frac{1}{d_c}\sum_{r=1}^{T}(\boldsymbol{P}^r)_{w,c} + \frac{1}{d_w}\sum_{r=1}^{T}(\boldsymbol{P}^r)_{c,w}\right)$$

Understanding Random Walk + Skip Gram

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

$$\frac{\mathrm{vol}(G)}{2T}\left(\sum_{r=1}^{T}\boldsymbol{P}^r\boldsymbol{D}^{-1} + \sum_{r=1}^{T}\boldsymbol{D}^{-1}(\boldsymbol{P}^r)^\top\right)$$

$$=\frac{\mathrm{vol}(G)}{2T}\left(\sum_{r=1}^{T}\underbrace{\boldsymbol{D}^{-1}\boldsymbol{A}\times\cdots\times\boldsymbol{D}^{-1}\boldsymbol{A}}_{r \text{ terms}}\boldsymbol{D}^{-1} + \sum_{r=1}^{T}\boldsymbol{D}^{-1}\underbrace{\boldsymbol{A}\boldsymbol{D}^{-1}\times\cdots\times\boldsymbol{A}\boldsymbol{D}^{-1}}_{r \text{ terms}}\right)$$

$$=\frac{\mathrm{vol}(G)}{T}\sum_{r=1}^{T}\underbrace{\boldsymbol{D}^{-1}\boldsymbol{A}\times\cdots\times\boldsymbol{D}^{-1}\boldsymbol{A}}_{r \text{ terms}}\boldsymbol{D}^{-1} = \mathrm{vol}(G)\left(\frac{1}{T}\sum_{r=1}^{T}\boldsymbol{P}^r\right)\boldsymbol{D}^{-1}.$$

$$\log\left(\frac{\#(w,c)\,|\mathcal{D}|}{b\#(w)\cdot\#(c)}\right) = \log\left(\frac{\mathrm{vol}(G)}{b}\left(\frac{1}{T}\sum_{r=1}^{T}(\boldsymbol{D}^{-1}\boldsymbol{A})^r\right)\boldsymbol{D}^{-1}\right)$$

Graph Language

$\boldsymbol{A}$ Adjacency matrix

$\boldsymbol{D}$ Degree matrix

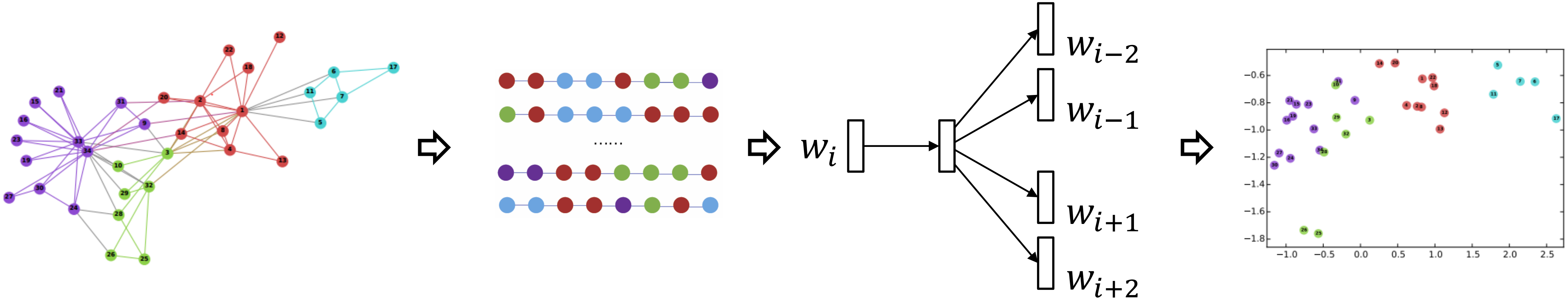
$\boldsymbol{P} = \boldsymbol{D}^{-1}\boldsymbol{A}$

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

b: #negative samples

T: context window size

Understanding Random Walk + Skip Gram



DeepWalk is asymptotically and implicitly factorizing

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

$\boldsymbol{A}$ Adjacency matrix

$\boldsymbol{D}$ Degree matrix

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

b: #negative samples

T: context window size

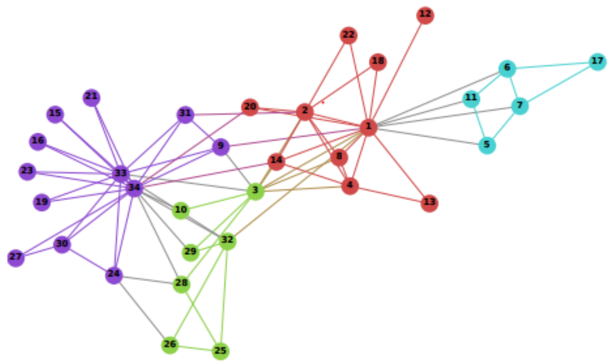
1. Qiu et al. Network embedding as matrix factorization: unifying deepwalk, line, pte, and node2vec. In *WSDM'18.*

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

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

Qiu et al. Network embedding as matrix factorization: unifying deepwalk, line, pte, and node2vec. In *WSDM'18.* **The most cited paper in WSDM'18 as of May 2019**

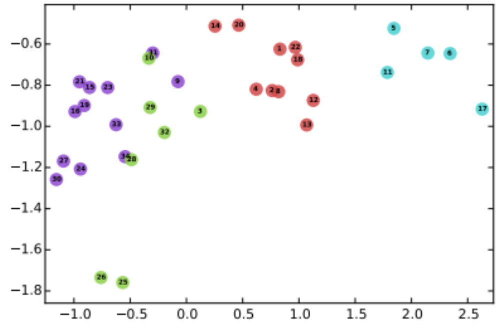
NetMF: Explicitly Factorizing the DeepWalk Matrix



⇨

Matrix Factorization

⇨



DeepWalk is asymptotically and implicitly factorizing

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

1. Qiu et al. Network embedding as matrix factorization: unifying deepwalk, line, pte, and node2vec. In *WSDM'18.*
2. Code &data for NetMF: https://github.com/xptree/NetMF

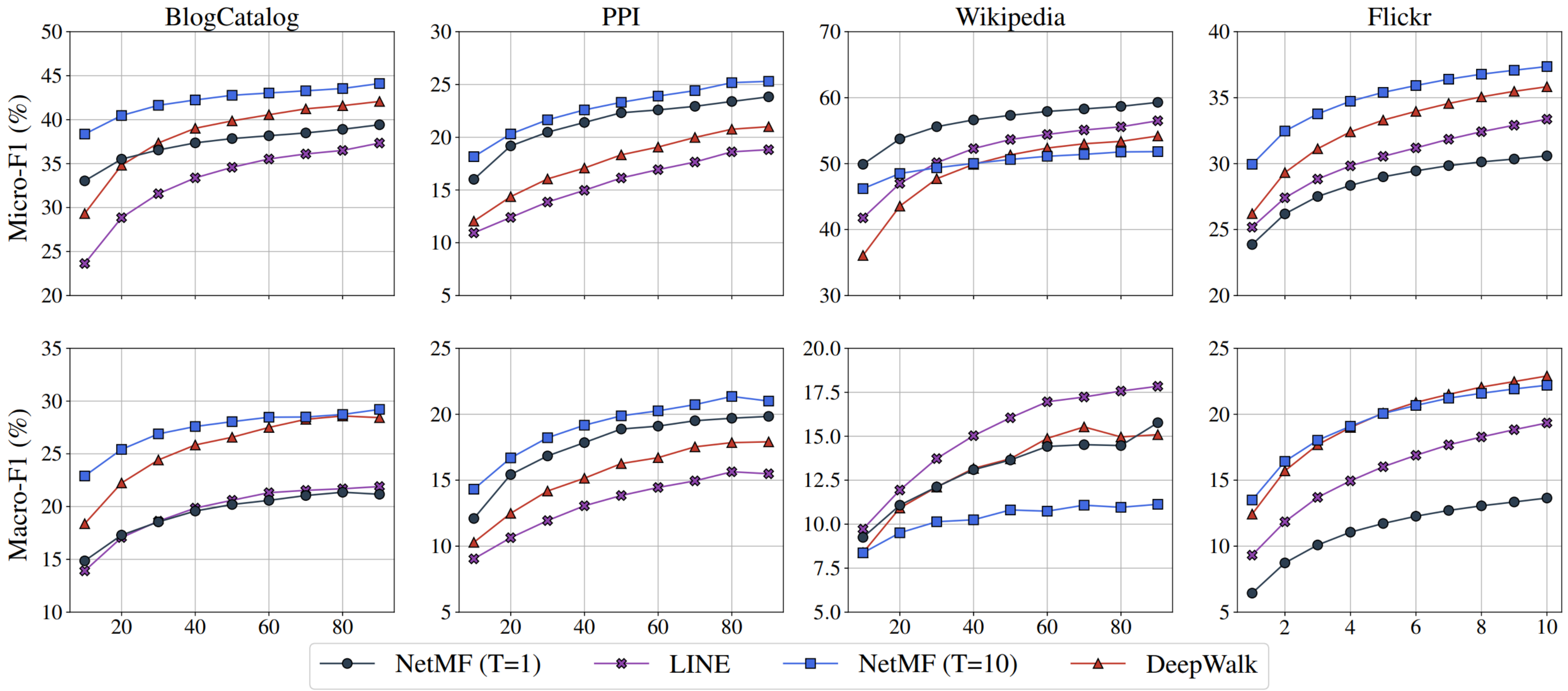
NetMF

1. Construction
2. Factorization

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

1. Qiu et al. Network embedding as matrix factorization: unifying deepwalk, line, pte, and node2vec. In *WSDM'18.*
2. Code &data for NetMF: https://github.com/xptree/NetMF

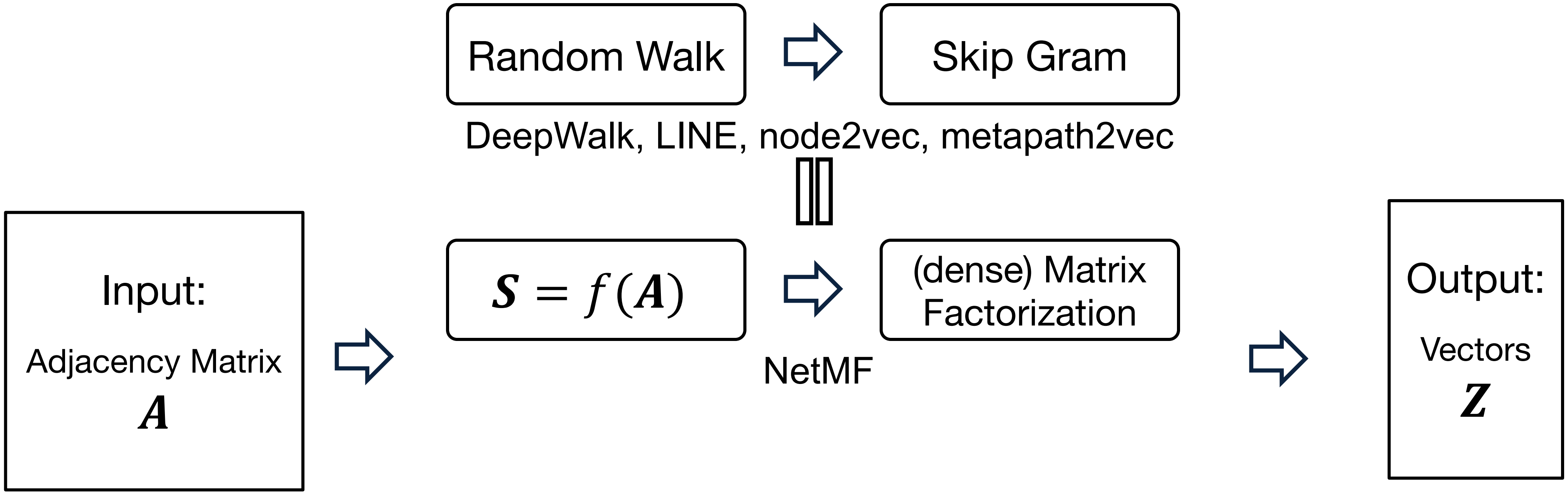
Results



Explicit matrix factorization (NetMF) offers performance gains over **implicit** matrix factorization (DeepWalk & LINE)

1. Qiu et al. Network embedding as matrix factorization: unifying deepwalk, line, pte, and node2vec. In *WSDM'18.*
2. Code &data for NetMF: https://github.com/xptree/NetMF

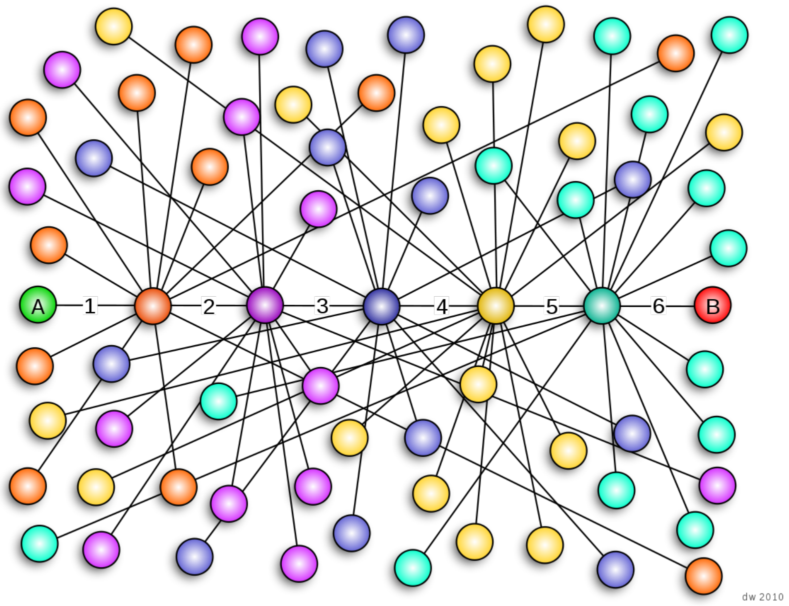
Network Embedding



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

1. Qiu et al. Network embedding as matrix factorization: unifying deepwalk, line, pte, and node2vec. In *WSDM'18.*
2. Code &data for NetMF: https://github.com/xptree/NetMF

Challenge?



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

n^2 non-zeros
Dense!!

Time complexity
$O(n^3)$

NetMF

How can we solve this issue?

1. Construction
2. Factorization

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

1. Qiu et al. NetSMF: Network embedding as sparse matrix factorization. In **WWW 2019.**
2. Code & data for NetSMF: https://github.com/xptree/NetSMF

NetSMF--Sparse

How can we solve this issue?

1. **Sparse** Construction
2. **Sparse** Factorization

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

1. Qiu et al. NetSMF: Network embedding as sparse matrix factorization. In **WWW 2019.**
2. Code & data for NetSMF: https://github.com/xptree/NetSMF

hide

Sparsify $\boldsymbol{S}$

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

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

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

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

Suppose $G = (V, E, \boldsymbol{A})$ and $\widetilde{G} = (V, \widetilde{E}, \widetilde{\boldsymbol{A}})$ are two weighted undirected networks. Let $\boldsymbol{L} = \boldsymbol{D}_G - \boldsymbol{A}$ and $\widetilde{\boldsymbol{L}} = \boldsymbol{D}_{\widetilde{G}} - \widetilde{\boldsymbol{A}}$ be their Laplacian matrices, respectively. We define G and $\widetilde{G}$ are $(1+\epsilon)$-spectrally similar if

$$\forall \boldsymbol{x} \in \mathbb{R}^n, (1-\epsilon) \cdot \boldsymbol{x}^\top \widetilde{\boldsymbol{L}} \boldsymbol{x} \leq \boldsymbol{x}^\top \boldsymbol{L} \boldsymbol{x} \leq (1+\epsilon) \cdot \boldsymbol{x}^\top \widetilde{\boldsymbol{L}} \boldsymbol{x}.$$

1. Dehua Cheng, Yu Cheng, Yan Liu, Richard Peng, and Shang-Hua Teng, Efficient Sampling for Gaussian Graphical Models via Spectral Sparsification, COLT 2015.
2. Dehua Cheng, Yu Cheng, Yan Liu, Richard Peng, and Shang-Hua Teng. Spectral sparsification of random-walk matrix polynomials. arXiv:1502.03496.

Sparsify $\boldsymbol{S}$

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

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

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

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

$$\alpha_1 = \cdots = \alpha_T = \frac{1}{T} \quad \Rightarrow$$

$$\begin{aligned} \boldsymbol{S} &= \log^\circ \left(\frac{\mathrm{vol}(G)}{b} \left(\frac{1}{T} \sum_{r=1}^{T} \left(\boldsymbol{D}^{-1}\boldsymbol{A}\right)^r \right) \boldsymbol{D}^{-1} \right) \\ &= \log^\circ \left(\frac{\mathrm{vol}(G)}{b} \boldsymbol{D}^{-1} (\boldsymbol{D} - \boldsymbol{L}) \boldsymbol{D}^{-1} \right) \\ &\approx \log^\circ \left(\frac{\mathrm{vol}(G)}{b} \boldsymbol{D}^{-1} (\boldsymbol{D} - \widetilde{\boldsymbol{L}}) \boldsymbol{D}^{-1} \right) \end{aligned}$$

1. Qiu et al. NetSMF: Network embedding as sparse matrix factorization. In **WWW 2019.**
2. Code & data for NetSMF: https://github.com/xptree/NetSMF

hide

NetSMF --- Sparse

- Construct a random walk matrix polynomial sparsifier, $\widetilde{\boldsymbol{L}}$
- Construct a NetMF matrix sparsifier.

$$\operatorname{trunc\_log}^{\circ}\left(\frac{\operatorname{vol}(G)}{b}\boldsymbol{D}^{-1}(\boldsymbol{D}-\widetilde{\boldsymbol{L}})\boldsymbol{D}^{-1}\right)$$

- Factorize the constructed matrix

1. Qiu et al. NetSMF: Network embedding as sparse matrix factorization. In **WWW 2019.**
2. Code & data for NetSMF: https://github.com/xptree/NetSMF

NetSMF---Bounded Approximation Error

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

$$=\log^\circ\left(\frac{\mathrm{vol}(G)}{b}\boldsymbol{D}^{-1}(\boldsymbol{D}-\boldsymbol{L})\boldsymbol{D}^{-1}\right) \longrightarrow \boldsymbol{M}$$

$$\approx\log^\circ\left(\frac{\mathrm{vol}(G)}{b}\boldsymbol{D}^{-1}(\boldsymbol{D}-\widetilde{\boldsymbol{L}})\boldsymbol{D}^{-1}\right) \longrightarrow \widetilde{\boldsymbol{M}}$$

Theorem

The singular value of $\widetilde{\boldsymbol{M}}-\boldsymbol{M}$ satisfies

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

Theorem

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

$$\left\|\mathrm{trunc\_log}^\circ\left(\frac{\mathrm{vol}(G)}{b}\widetilde{\boldsymbol{M}}\right)-\mathrm{trunc\_log}^\circ\left(\frac{\mathrm{vol}(G)}{b}\boldsymbol{M}\right)\right\|_F \leq \frac{4\epsilon\,\mathrm{vol}(G)}{b\sqrt{d_{\min}}}\sqrt{\sum_{i=1}^{n}\frac{1}{d_i}}.$$

1. Qiu et al. NetSMF: Network embedding as sparse matrix factorization. In **WWW 2019.**
2. Code & data for NetSMF: https://github.com/xptree/NetSMF

hide

Results

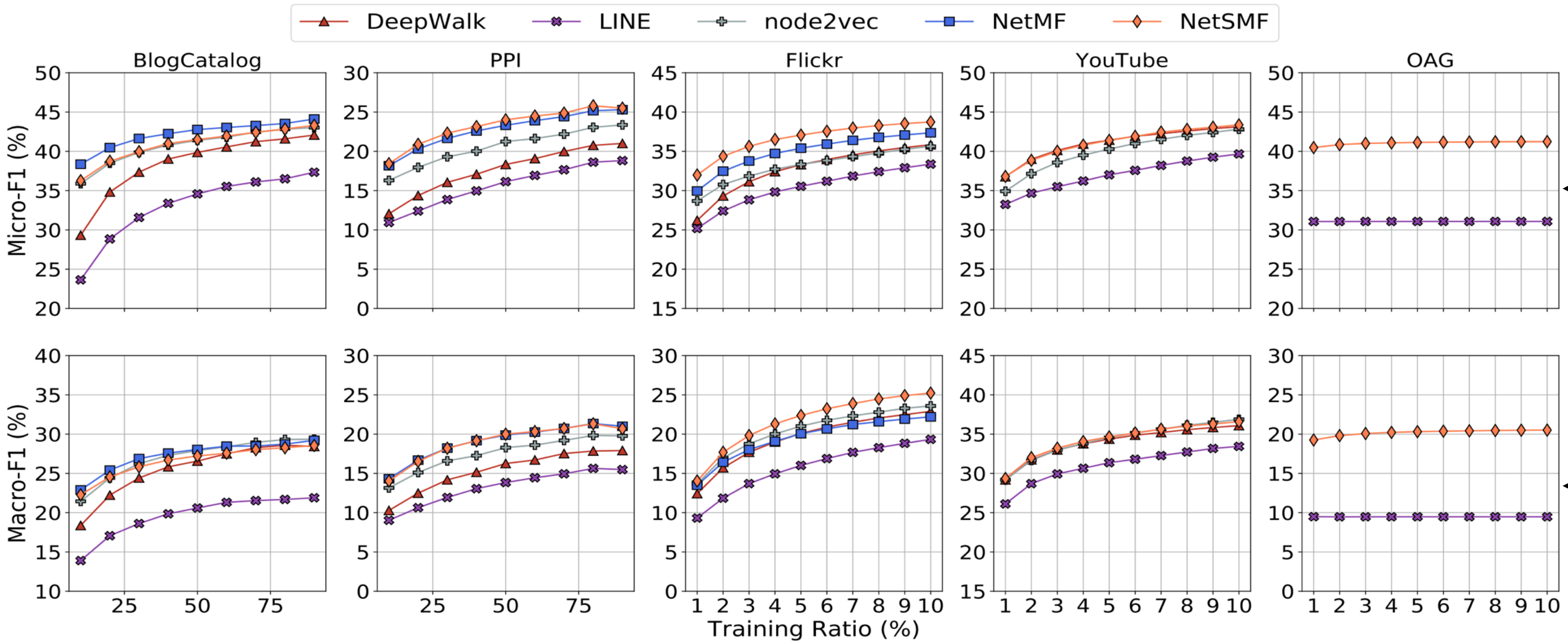
Dataset	BlogCatalog	PPI	Flickr	YouTube	OAG
$\|V\|$	10,312	3,890	80,513	1,138,499	67,768,244
$\|E\|$	333,983	76,584	5,899,882	2,990,443	895,368,962
#labels	39	50	195	47	19

- #non-zeros:
- ~4.5 Quadrillion → 45 Billion

	LINE	DeepWalk	node2vec	NetMF	NetSMF
BlogCatalog	40 mins	12 mins	56 mins	19 mins	13 mins
PPI	41 mins	4 mins	4 mins	1 min	10 secs
Flickr	42 mins	2.2 hours	21 hours	5 days	48 mins
YouTube	46 mins	4.3 hours	4 days	×	4.1 hours
OAG	2.6 hours	–	–	×	24 hours

1. Qiu et al. NetSMF: Network embedding as sparse matrix factorization. In **WWW 2019.**
2. Code & data for NetSMF: https://github.com/xptree/NetSMF

Results



30% improvements over LINE on billion-scale graphs

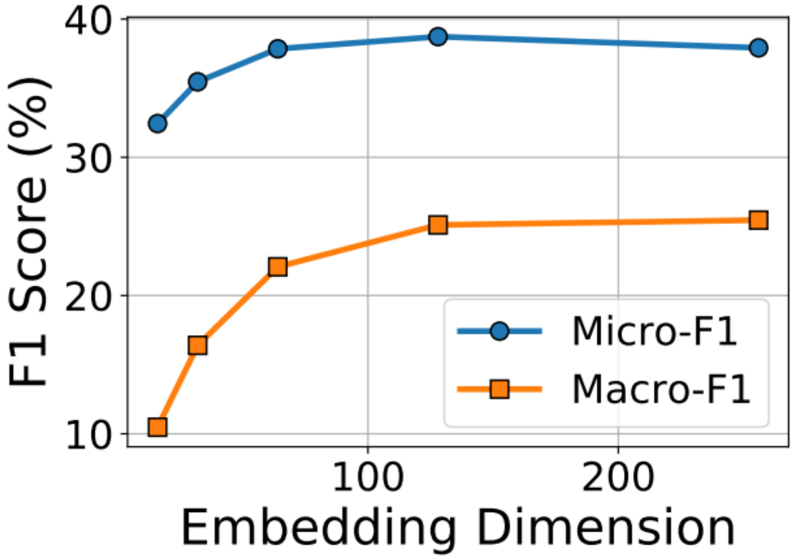
100% improvements over LINE on billion-scale graphs

Effectiveness: NetSMF (sparse MF) ≈ NetMF (explicit MF) **>** DeepWalk/LINE (implicit MF)

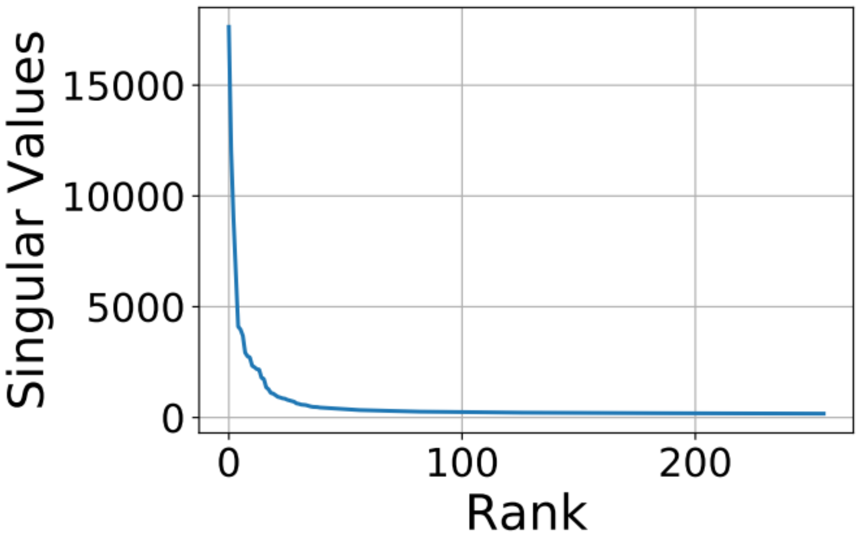
Efficiency: NetSMF (sparse MF) can handle billion-scale network embedding

1. Qiu et al. NetSMF: Network embedding as sparse matrix factorization. In **WWW 2019.**
2. Code & data for NetSMF: https://github.com/xptree/NetSMF

hide



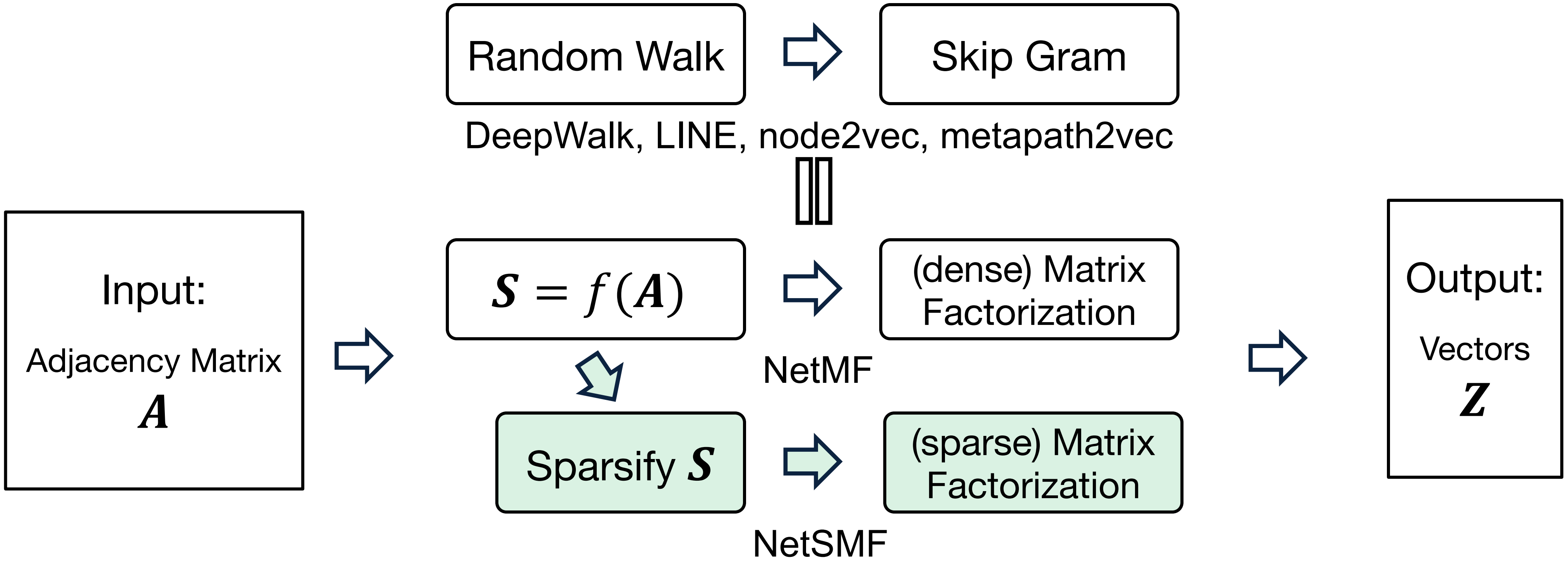
(a)



(b)

1. Qiu et al. NetSMF: Network embedding as sparse matrix factorization. In **WWW 2019.**
2. Code & data for NetSMF: https://github.com/xptree/NetSMF

Network Embedding

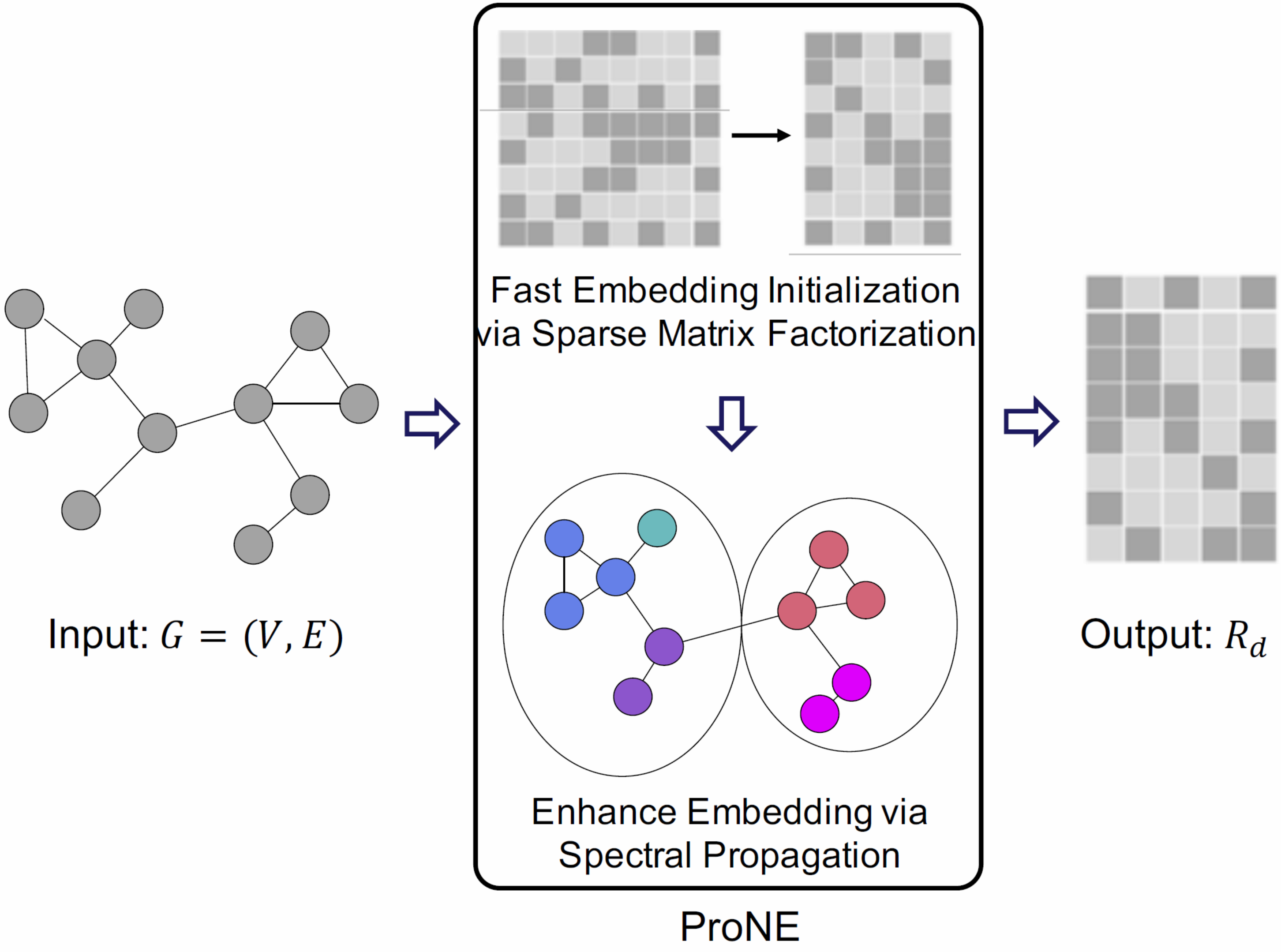


Incorporate network structures $\boldsymbol{A}$ into the similarity matrix $\boldsymbol{S}$, and then factorize $\boldsymbol{S}$

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

1. Qiu et al. NetSMF: Network embedding as sparse matrix factorization. In **WWW 2019.**
2. Code & data for NetSMF: https://github.com/xptree/NetSMF

ProNE: Propagation based Network Embedding



1. Zhang et al. ProNE: Fast and Scalable Network Representation Learning. In **IJCAI 2019**
2. Code & data for ProNE: https://github.com/THUDM/ProNE

Spectral Propagation

$R_d \leftarrow D^{-1}A(I_n - \tilde{L})\, R_d$

The idea of **Graph Neural Networks**

$D^{-1}A(I_n - \tilde{L})$ is $D^{-1}A$ modulated by the filter in the spectrum

$\widetilde{L} = Ug(\Lambda)U^T$ is the spectral filter of $L = I_n - D^{-1}A$

1. Zhang et al. ProNE: Fast and Scalable Network Representation Learning. In **IJCAI 2019**
2. Code & data for ProNE: https://github.com/THUDM/ProNE

hide

Chebyshev expansion for efficiency

- To avoid explicit eigendecomposition and Fourier transform
  - Chebyshev expansion $T_{i+1}(x) = 2xT_i(x) - T_{i-1}(x)$ with $T_0(x) = 1, T_1(x) = x$

$$\widetilde{L} = Udiag([g(\lambda_1), ..., g(\lambda_n)])U^T$$
$$\approx U \sum_{i=0}^{k-1} c_i(\theta) T_i(\bar{\Lambda}) U^T$$
$$= \sum_{i=0}^{k-1} c_i(\theta) T_i(\bar{L})$$

$\Rightarrow$

$$\widetilde{L} \approx B_0(\theta) T_0(\bar{L}) + 2 \sum_{i=1}^{k-1} (-)^i B_i(\theta) T_i(\bar{L})$$

1. Zhang et al. ProNE: Fast and Scalable Network Representation Learning. In **IJCAI 2019**
2. Code & data for ProNE: https://github.com/THUDM/ProNE

Efficiency

20 Threads — 1 Thread

Dataset	*DeepWalk*	*LINE*	*node2vec*	*ProNE*
PPI	272	70	828	**3**
Wiki	494	87	939	**6**
BlogCatalog	1,231	185	3,533	**21**
DBLP	3,825	1,204	4,749	**24**
Youtube	68,272	5,890	>5days	**627**

1.1M nodes — 19hours — 98mins — 10mins

Dataset	training ratio	0.01	0.03	0.05	0.07	0.09
DBLP	DeepWalk	49.3	55.0	57.1	57.9	58.4
	LINE	48.7	52.6	53.5	54.1	54.5
	node2vec	48.9	55.1	57.0	58.0	58.4
	GraRep	50.5	52.6	53.2	53.5	53.8
	HOPE	**52.2**	55.0	55.9	56.3	56.6
	ProNE (SMF)	50.8	54.9	56.1	56.7	57.0
	ProNE	48.8	**56.2**	**58.0**	**58.8**	**59.2**
	($\pm\sigma$)	($\pm$1.0)	($\pm$0.5)	($\pm$0.2)	($\pm$0.2)	($\pm$0.1)
Youtube	DeepWalk	38.0	40.1	41.3	42.1	42.8
	LINE	33.2	35.5	37.0	38.2	39.3
	ProNE (SMF)	36.5	40.2	41.2	41.7	42.1
	ProNE	**38.2**	**41.4**	**42.3**	**42.9**	**43.3**
	($\pm\sigma$)	($\pm$0.8)	($\pm$0.3)	($\pm$0.2)	($\pm$0.2)	($\pm$0.2)

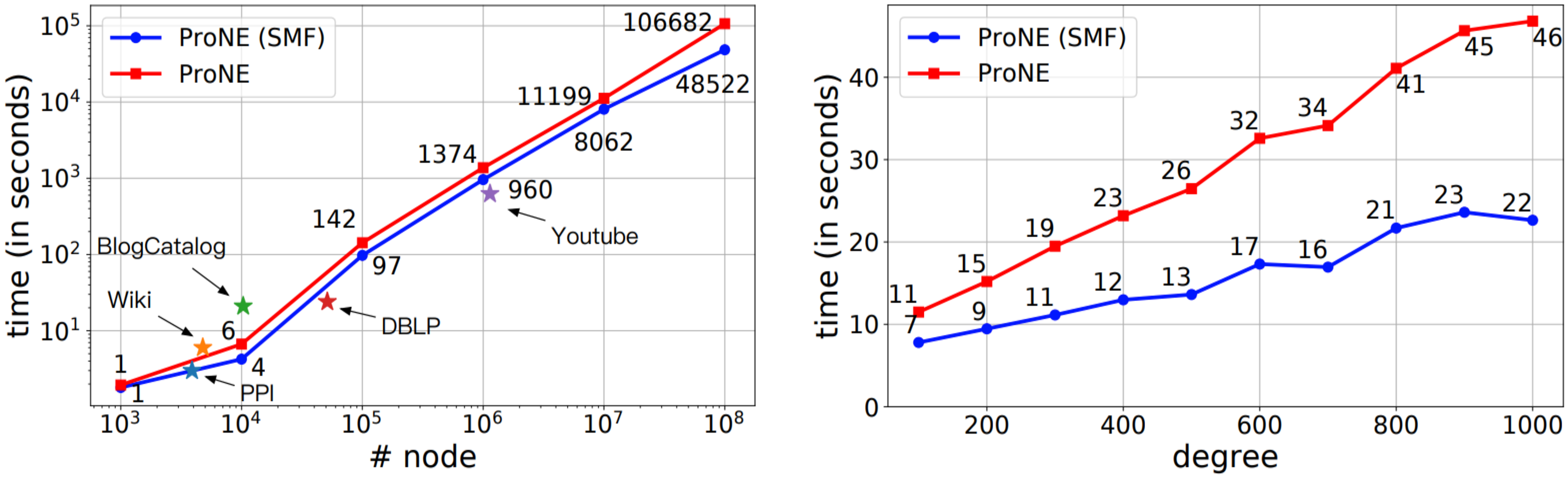
ProNE offers **10-400X** speedups (1 thread vs 20 threads)

Embed 100,000,000 nodes by 1 thread: 29 hours with performance superiority

1. Zhang et al. ProNE: Fast and Scalable Network Representation Learning. In **IJCAI 2019**
2. Code & data for ProNE: https://github.com/THUDM/ProNE

Scalability

hide

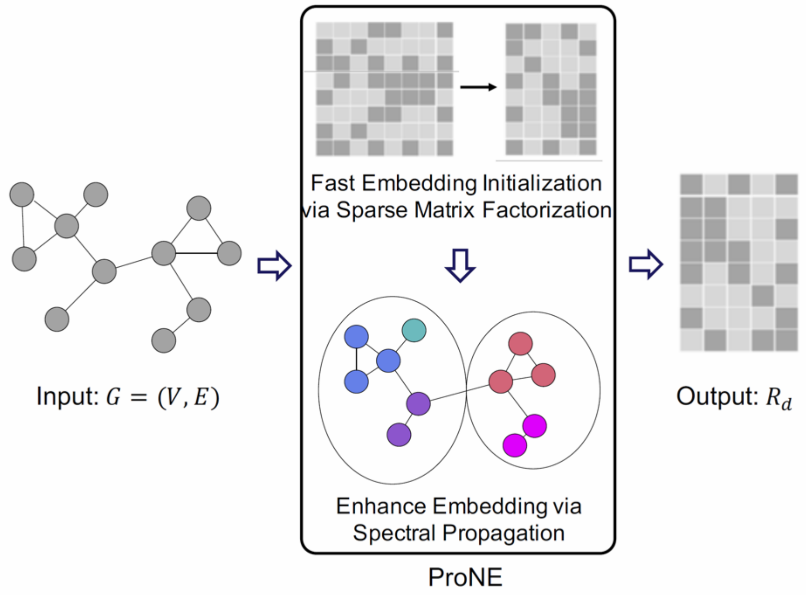


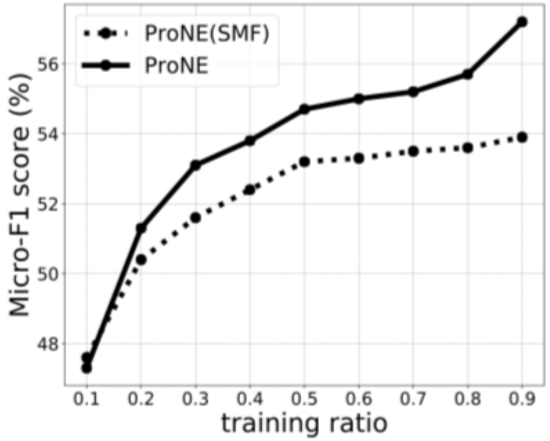
(a) The node degree is fixed to 10 and #nodes grows (b) #nodes is fixed to 10,000 and the node degree grows

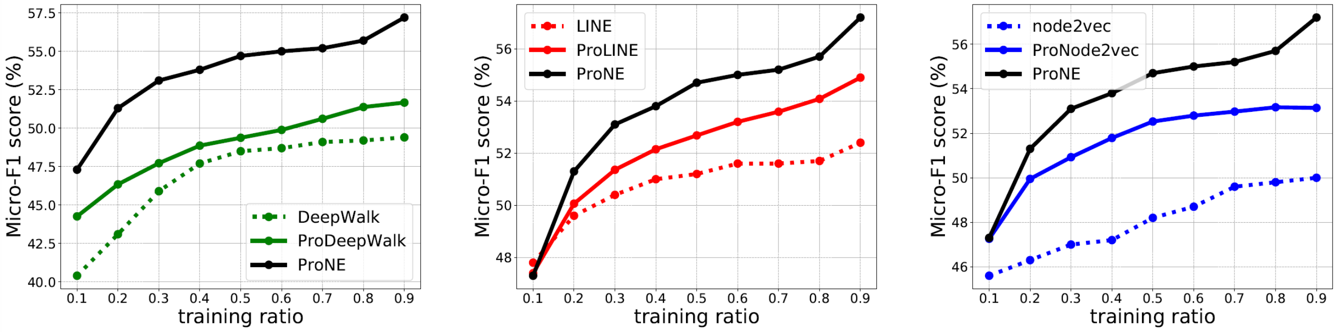
Embed 100,000,000 nodes by 1 thread: 29 hours with performance superiority

hide

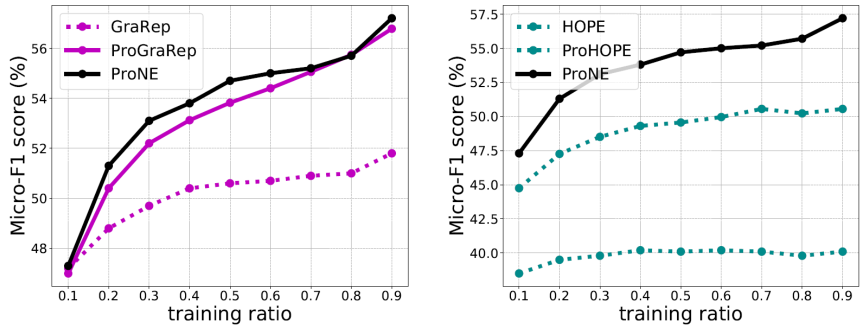
ProNE: A General Propagation Framework







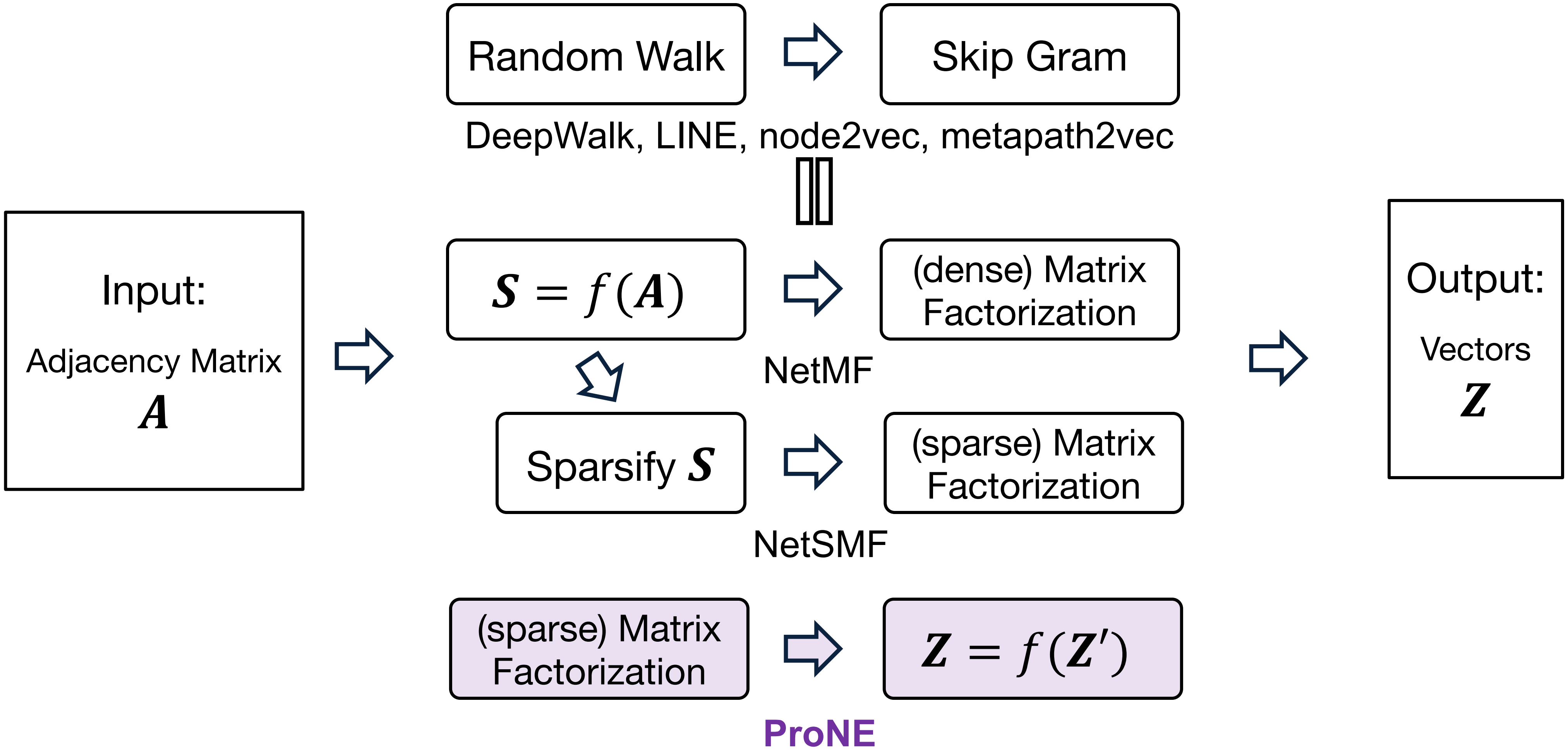
(a) ProDeepWalk (b) ProLINE (c) ProNode2vec



(d) ProGraRep (e) ProHOPE

1. Zhang et al. ProNE: Fast and Scalable Network Representation Learning. In **IJCAI 2019**
2. Code & data for ProNE: https://github.com/THUDM/ProNE

Network Embedding



Factorize A, and then incorporate network structures via spectral propagation

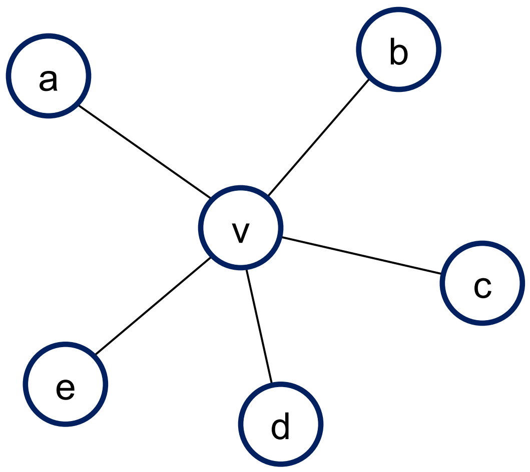
Graph Representation Learning

Network Embedding ⇔ **Matrix Factorization** ⇨ **GNNs** ⇨ **Pre-Training**

- DeepWalk
- LINE
- Node2vec
- PTE
- …
- metapath2vec

- NetMF
- NetSMF
- …
- ProNE (Propagation)

Connecting NE with Graph Neural Networks



- **ProNE**
  - Propagation based network embedding

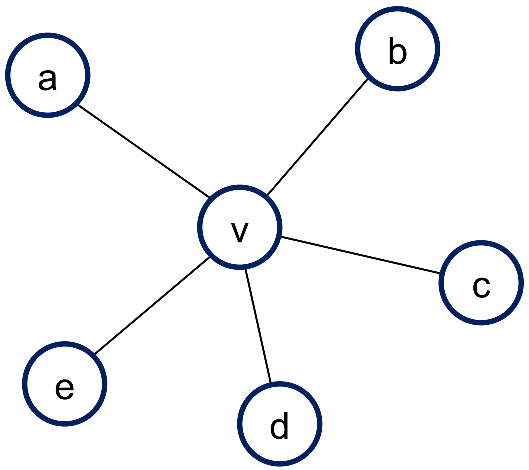
$$\boldsymbol{R}_d \leftarrow \boldsymbol{D}^{-1}\boldsymbol{A}(\boldsymbol{I}_n - \tilde{\boldsymbol{L}})\,\boldsymbol{R}_d$$

- **GNN**
  - Neighborhood aggregation: aggregate neighbor information and pass into a neural network

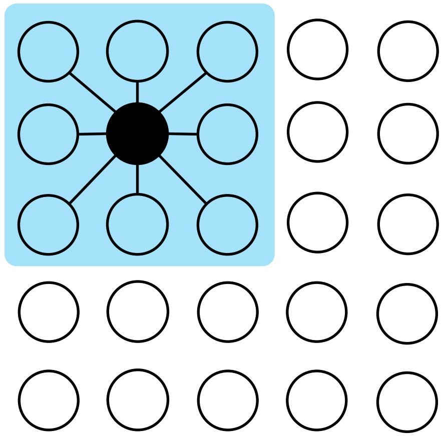
$$\boldsymbol{h}_v = f(\boldsymbol{h}_v, \boldsymbol{h}_a, \boldsymbol{h}_b, \boldsymbol{h}_c, \boldsymbol{h}_d, \boldsymbol{h}_e)$$

1. Justin Gilmer, et al. Neural message passing for quantum chemistry. In **ICML** 2017.
2. Zhang et al. ProNE: Fast and Scalable Network Representation Learning. In **IJCAI 2019**

Graph Neural Networks



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



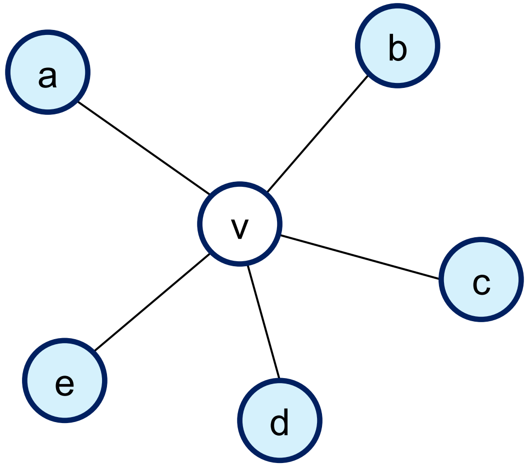
Graph Convolution **CNN**

Neighborhood Aggregation:

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

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

Graph Convolutional Networks



Non-linear activation function (e.g., ReLU)

parameters in layer k

$$\boldsymbol{h}_v^k = \sigma\Big(\boldsymbol{W}^k \sum_{u \in N(v) \cup v} \frac{\boldsymbol{h}_u^{k-1}}{\sqrt{|N(u)||N(v)|}}\Big)$$

node v's embedding at layer k

the neighbors of node v

$$\boldsymbol{H}^k = \sigma\left(\widehat{\boldsymbol{A}}\boldsymbol{H}^{(k-1)}\boldsymbol{W}^{(k)}\right)$$

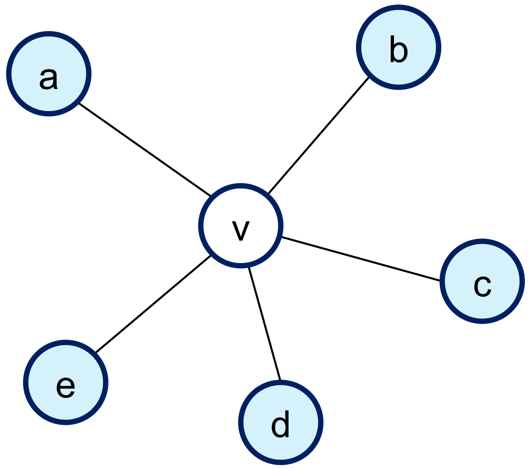
normalized Laplacian matrix

Aggregate info from neighborhood via the normalized Laplacian matrix

1. Kipf et al. Semisupervised Classification with Graph Convolutional Networks. ICLR 2017

Graph Convolutional Networks

hide



Aggregate from v's neighbors

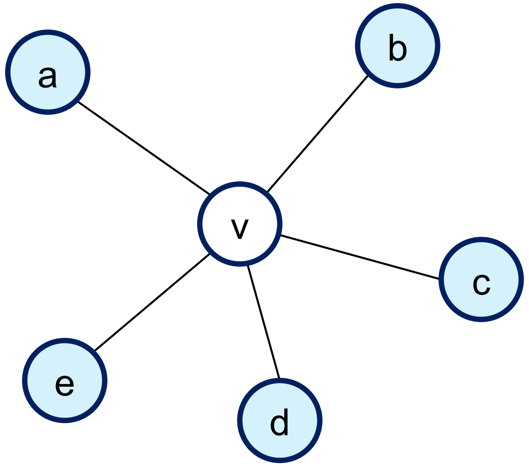
$$\boldsymbol{h}_v^k = \sigma(\boldsymbol{W}^k \sum_{u \in \mathrm{N}(v)} \frac{\boldsymbol{h}_u^{k-1}}{\sqrt{|N(u)||N(v)|}} + \boldsymbol{W}^k \sum_{v} \frac{\boldsymbol{h}_v^{k-1}}{\sqrt{|N(v)||N(v)|}})$$

Aggregate from itself

1. Kipf et al. Semisupervised Classification with Graph Convolutional Networks. ICLR 2017

hide

Graph Convolutional Networks



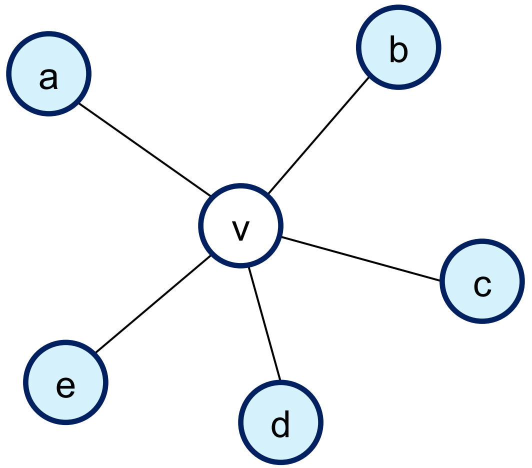
The same parameters for both its neighbors & itself

$$\boldsymbol{h}_v^k = \sigma\left(\boldsymbol{W}^k \sum_{u \in \mathrm{N}(v)} \frac{\boldsymbol{h}_u^{k-1}}{\sqrt{|N(u)||N(v)|}} + \boldsymbol{W}^k \sum_{v} \frac{\boldsymbol{h}_v^{k-1}}{\sqrt{|N(v)||N(v)|}}\right)$$

Kipf et al. Semisupervised Classification with Graph Convolutional Networks. ICLR 2017

hide

Graph Convolutional Networks



$$\boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{A}\boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{H}^{(k-1)}\boldsymbol{W}^{(k)}$$

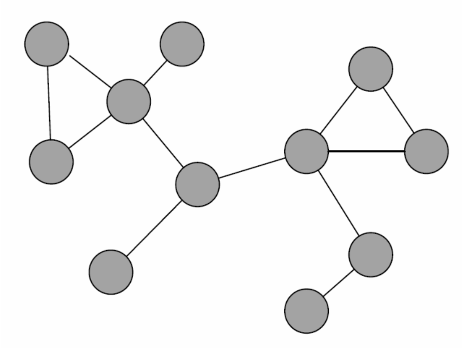
$$\boldsymbol{h}_v^k = \sigma(\boldsymbol{W}^k \sum_{u \in \mathrm{N}(v)} \frac{\boldsymbol{h}_u^{k-1}}{\sqrt{|N(u)||N(v)|}} + \boldsymbol{W}^k \sum_{v} \frac{\boldsymbol{h}_v^{k-1}}{\sqrt{|N(v)||N(v)|}})$$

$$\boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{I}\boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{H}^{(k-1)}\boldsymbol{W}^{(k)}$$

Kipf et al. Semisupervised Classification with Graph Convolutional Networks. ICLR 2017

hide

Graph Convolutional Networks



$G = (V, E, \boldsymbol{A})$

$\boldsymbol{H}^0 = \boldsymbol{X}$

$$\Rightarrow \boldsymbol{H}^k = \sigma\left(\boldsymbol{D}^{-\frac{1}{2}}(\boldsymbol{A} + \boldsymbol{I})\boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{H}^{(k-1)}\boldsymbol{W}^{(k)}\right) \Rightarrow \boldsymbol{Z} = \boldsymbol{H}^K$$

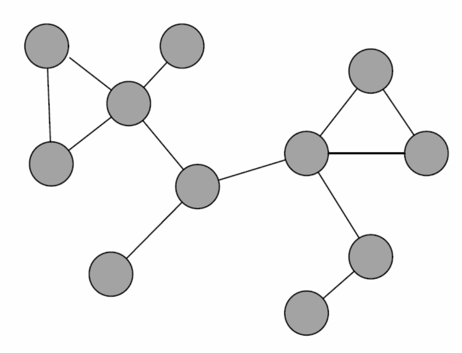
Input

Output

Kipf et al. Semisupervised Classification with Graph Convolutional Networks. ICLR 2017

hide

Graph Convolutional Networks



$G = (V, E, \boldsymbol{A})$

$\boldsymbol{H}^0 = \boldsymbol{X}$

$$\Rightarrow \boldsymbol{H}^k = \sigma\left(\boldsymbol{D}^{-\frac{1}{2}}(\boldsymbol{A}+\boldsymbol{I})\boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{H}^{(k-1)}\boldsymbol{W}^{(k)}\right) \Rightarrow \boldsymbol{Z} = \boldsymbol{H}^K$$

Input

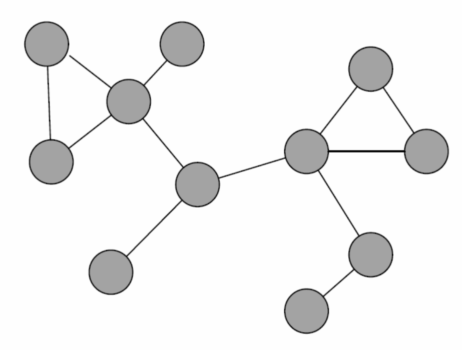
Output

- Model training
    - The common setting is to have an end to end training framework with a supervised task
    - That is, define a loss function over $\boldsymbol{Z}$

Kipf et al. Semisupervised Classification with Graph Convolutional Networks. ICLR 2017

hide

Graph Convolutional Networks



$G = (V, E, \boldsymbol{A})$

$\boldsymbol{H}^0 = \boldsymbol{X}$

$$\Rightarrow \boldsymbol{H}^k = \sigma\left(\boldsymbol{D}^{-\frac{1}{2}}(\boldsymbol{A} + \boldsymbol{I})\boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{H}^{(k-1)}\boldsymbol{W}^{(k)}\right) \Rightarrow \quad \boldsymbol{Z} = \boldsymbol{H}^K$$

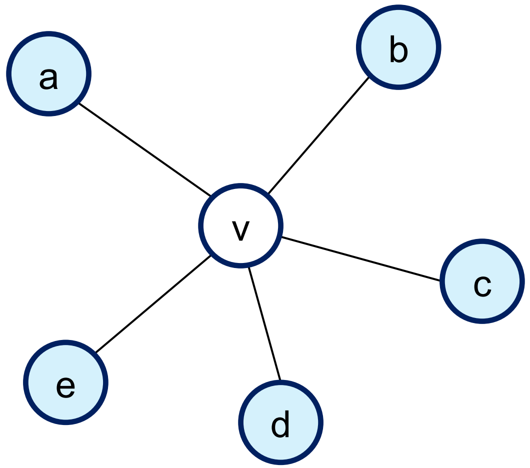
Input

Output

- Benefits: Parameter sharing for all nodes
  - #parameters is subline in |V|
  - Enable inductive learning for new nodes

hide

GraphSage



GCN

$$\boldsymbol{h}_v^k = \sigma(\boldsymbol{W}^k \sum_{u \in \mathrm{N}(v) \cup v} \frac{\boldsymbol{h}_u^{k-1}}{\sqrt{|N(u)||N(v)|}})$$

GraphSage

Instead of summation, it concatenates neighbor & self embeddings

$$\boldsymbol{h}_v^k = \sigma([\boldsymbol{A}^k \cdot \mathrm{AGG}\left(\{\boldsymbol{h}_u^{k-1}, \forall u \in N(v)\}\right), \boldsymbol{B}^k \boldsymbol{h}_v^{k-1}])$$

Generalized aggregation: any differentiable function that maps set of vectors to a single vector

Hamilton et al. Inductive Representation Learning on Large Graphs. NIPS 2017

hide

GraphSage

$$\boldsymbol{h}_v^k = \sigma([\boldsymbol{A}^k \cdot \boxed{\mathrm{AGG}}(\{\boldsymbol{h}_u^{k-1}, \forall u \in N(v)\}), \boldsymbol{B}^k \boldsymbol{h}_v^{k-1}])$$

- Mean:

$$\mathrm{AGG} = \sum_{u \in N(v)} \frac{\mathbf{h}_u^{k-1}}{|N(v)|}$$

- Pool
  - Transform neighbor vectors and apply symmetric vector function.

element-wise mean/max

$$\mathrm{AGG} = \boxed{\gamma}\left(\{\mathbf{Q}\mathbf{h}_u^{k-1}, \forall u \in N(v)\}\right)$$

- LSTM:
  - Apply LSTM to random permutation of neighbors.

$$\mathrm{AGG} = \mathrm{LSTM}\left([\mathbf{h}_u^{k-1}, \forall u \in \pi(N(v))]\right)$$

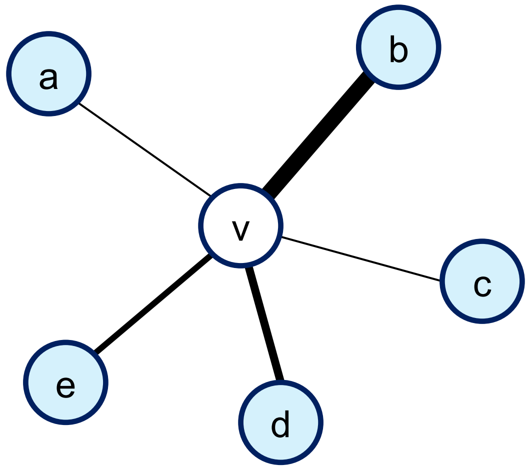
Hamilton et al. Inductive Representation Learning on Large Graphs. NIPS 2017
Slide snipping from “Hamiltion & Tang, AAAI 2019 Tutorial on Graph Representation Learning”

Graph Neural Network

hide

$$\boldsymbol{H}^{k} = \sigma\left(\boldsymbol{A}\boldsymbol{H}^{(k-1)}\boldsymbol{W}^{(k-1)}\right)$$

Graph Attention



GCN

$$\boldsymbol{h}_v^k = \sigma(\boldsymbol{W}^k \sum_{u \in \mathrm{N}(v) \cup v} \frac{\boldsymbol{h}_u^{k-1}}{\sqrt{|N(u)||N(v)|}})$$

Aggregate info from neighborhood via the normalized Laplacian matrix

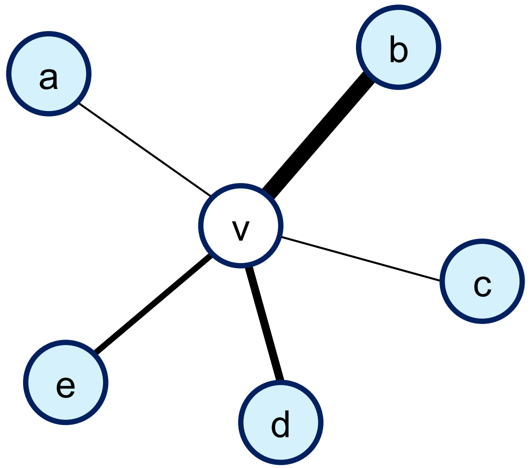
Graph Attention

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

Aggregate info from neighborhood via the learned attention

Velickovic et al. Graph Attention Networks. ICLR 2018

Graph Attention



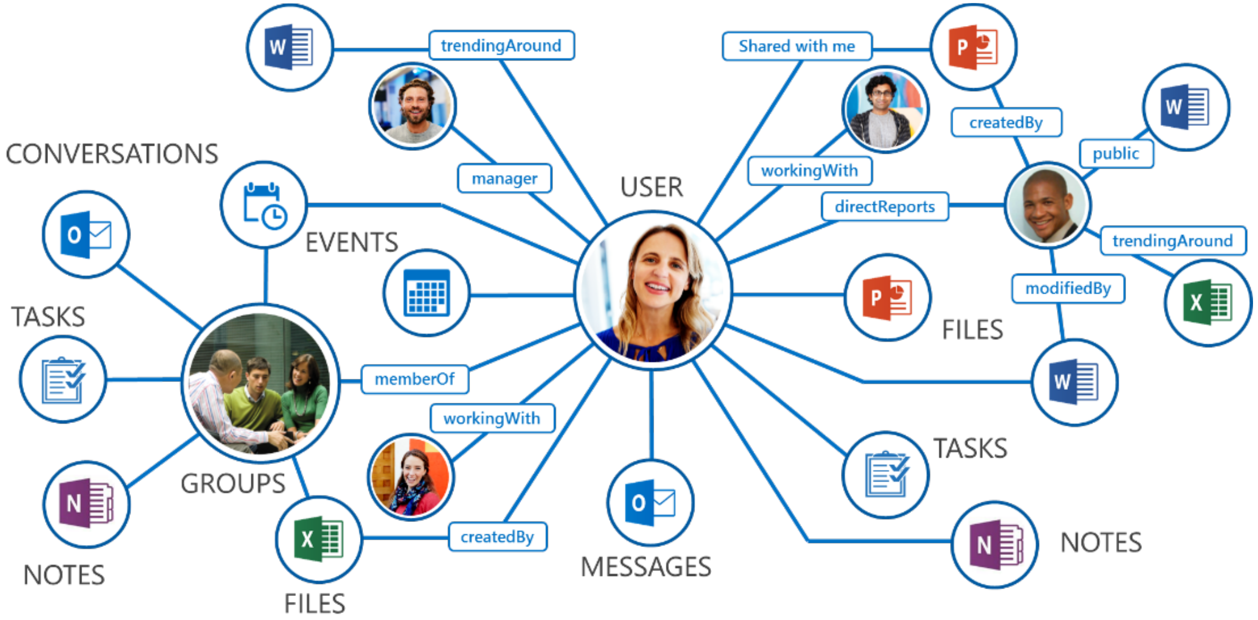
$$\alpha_{v,u} = \frac{\exp\left(\mathrm{LeakyReLU}\left(\mathbf{a}^\top[\mathbf{Q}\mathbf{h}_v, \mathbf{Q}\mathbf{h}_u]\right)\right)}{\sum_{u' \in N(v) \cup \{v\}} \exp\left(\mathrm{LeakyReLU}\left(\mathbf{a}^\top[\mathbf{Q}\mathbf{h}_v, \mathbf{Q}\mathbf{h}_{u'}]\right)\right)}$$

many ways to define attention!

Velickovic et al. Graph Attention Networks. ICLR 2018

Attention over Heterogeneous Graphs?

heterogeneous academic graph



heterogeneous office graph

Heterogeneous Graph Transformer (HGT)

- Current graph neural networks are not capable enough to capture graph **heterogeneity**
- **Heterogeneous Graph Transformer**
    - Unique parameters for each type of relationships

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

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

1.Ziniu Hu, et al. Heterogeneous Graph Transformer. **WWW 2020.**
2.Code & Data for HGT: https://github.com/acbull/pyHGT

Heterogeneous Graph Transformer (HGT)

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

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

- heterogeneous mutual attention

$$\textbf{Attention}_{HGT}(s,e,t) = \underset{\forall s \in N(t)}{\text{Softmax}}\left(\underset{i \in [1,h]}{\Big\|} ATT\text{-}head^i(s,e,t) \right) \qquad (3)$$

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

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

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

1.Ziniu Hu, et al. Heterogeneous Graph Transformer. **WWW 2020.**
2.Code & Data for HGT: https://github.com/acbull/pyHGT

hide

Heterogeneous Graph Transformer

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

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

- heterogeneous message passing

$$\textbf{Message}_{HGT}(s,e,t) = \underset{i \in [1,h]}{\Big\|} \textit{MSG-head}^i(s,e,t)$$

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

1.Ziniu Hu, et al. Heterogeneous Graph Transformer. **WWW 2020.**
2.Code & Data for HGT: https://github.com/acbull/pyHGT

hide

Heterogeneous Graph Transformer

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

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

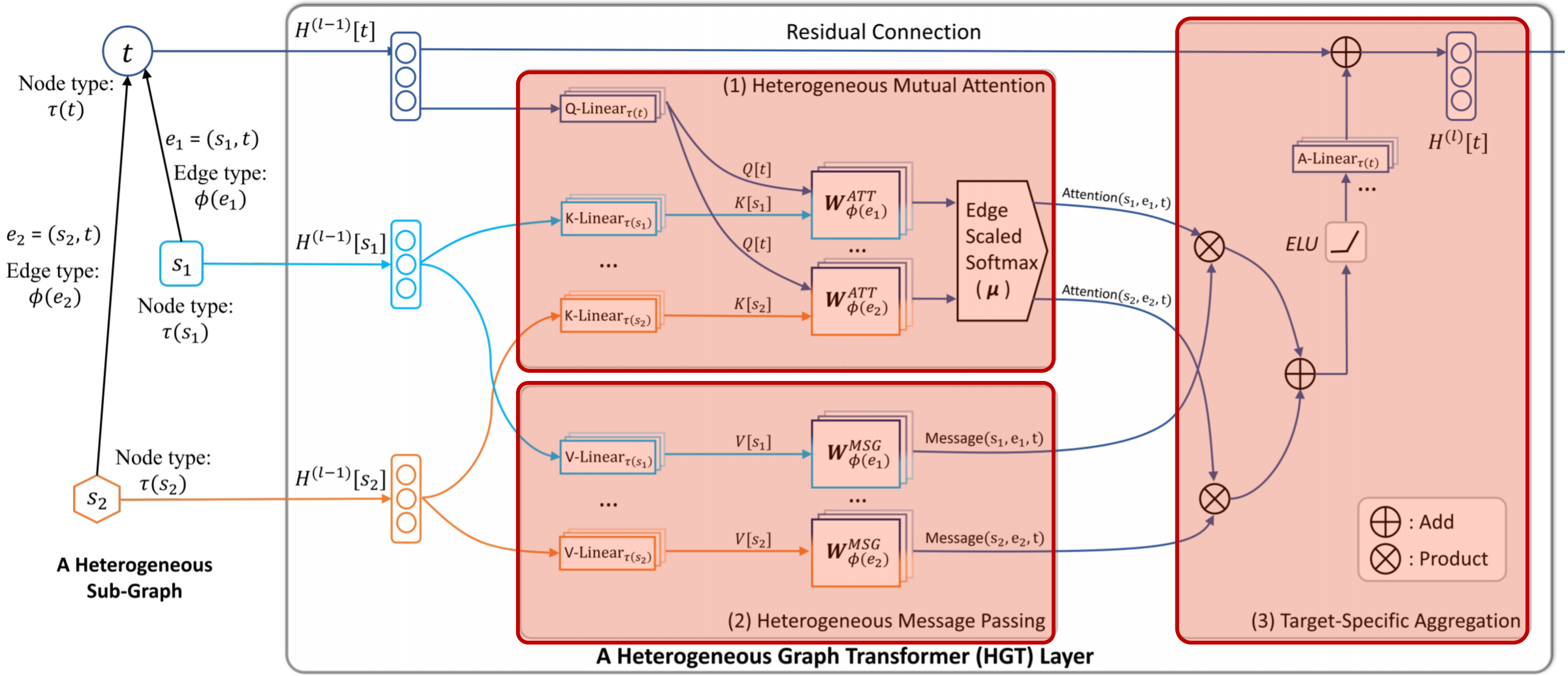
- target specific aggregation

$$\widetilde{H}^{(l)}[t] = \bigoplus_{\forall s \in N(t)} \Big(\textbf{Attention}_{HGT}(s, e, t) \cdot \textbf{Message}_{HGT}(s, e, t) \Big)$$

$$H^{(l)}[t] = \text{A-Linear}_{\tau(t)} \Big(\sigma \big(\widetilde{H}^{(l)}[t] \big) \Big) + H^{(l-1)}[t]$$

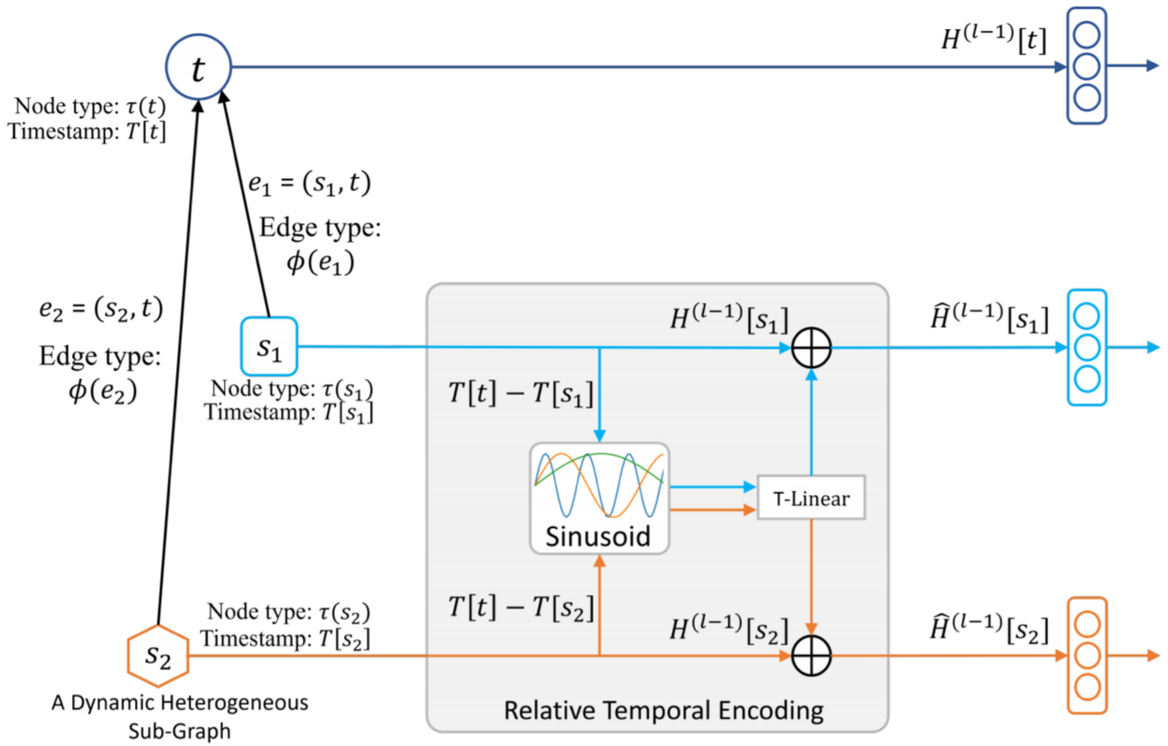
1.Ziniu Hu, et al. Heterogeneous Graph Transformer. **WWW 2020.**
2.Code & Data for HGT: https://github.com/acbull/pyHGT

Heterogeneous Graph Transformer (HGT)



1.Ziniu Hu, et al. Heterogeneous Graph Transformer. **WWW 2020.**
2.Code & Data for HGT: https://github.com/acbull/pyHGT

Heterogeneous Graph Transformer (HGT)



- Relative Temporal Encoding

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

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

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

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

1.Ziniu Hu, et al. Heterogeneous Graph Transformer. **WWW 2020.**
2.Code & Data for HGT: https://github.com/acbull/pyHGT

Experiments

236,963,398
Publications

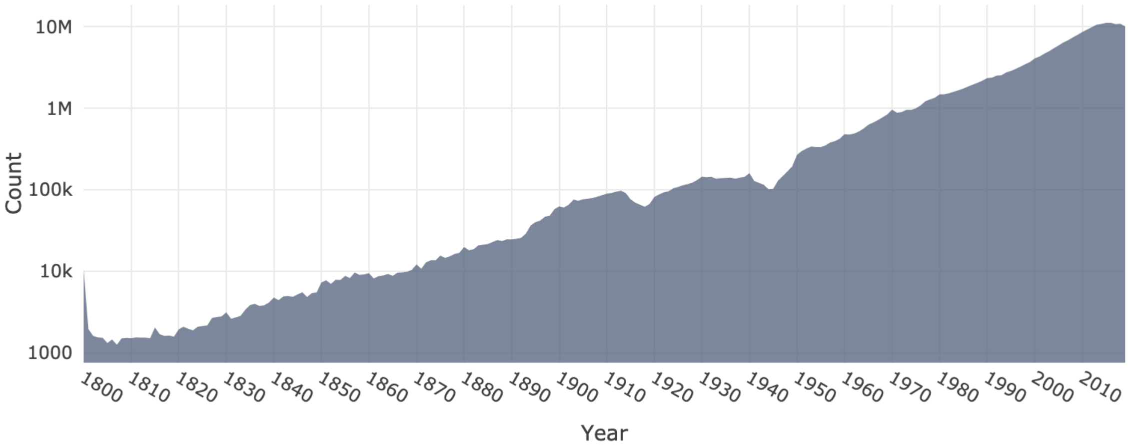
240,667,697
Authors

740,048
Topics

4,467
Conferences

48,876
Journals

25,759
Institutions



- Sampling subgraphs from large-scale graphs
  - From homogeneous graphs → LADIES algorithm
  - From heterogeneous graphs → **HGSampling** algo

1. Ziniu Hu, et al. Heterogeneous Graph Transformer. **WWW 2020.**
2. Difan Zou, et al. Layer-Dependent Importance Sampling for Training Deep and Large Graph Convolutional Networks. In NeurIPS'19.

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 Disambiguation	NDCG	.730±.064	.742±.057	.785±.063	.792±.052	.810±.049	.834±.058	.849±.066	**.857±.054**
	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

1. Ziniu Hu, et al. Heterogeneous Graph Transformer. **WWW 2020.**
2. Code & Data for HGT: https://github.com/acbull/pyHGT

Case Study

Experiments done in **2019**!

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

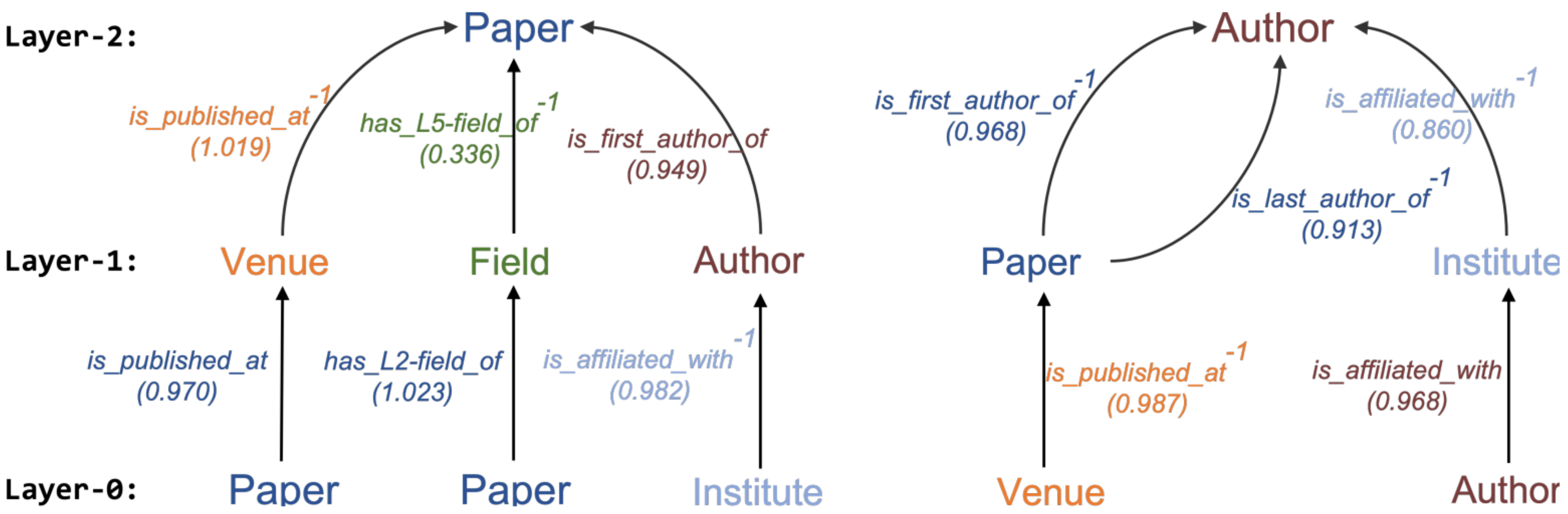
ML + DB + Web + AI + NLP!!!

CV + ML + AI

ML + CV + DL + NLP

1.Ziniu Hu, et al. Heterogeneous Graph Transformer. **WWW 2020.**
2.Code & Data for HGT: https://github.com/acbull/pyHGT

What is the Best Part of HGT?



Learn meta-paths & their weights implicitly!

1.Ziniu Hu, et al. Heterogeneous Graph Transformer. **WWW 2020.**
2.Code & Data for HGT: https://github.com/acbull/pyHGT

Graph Representation Learning

Network Embedding ⇔ **Matrix Factorization** ⇒ **GNNs** ⇒ **Pre-Training**

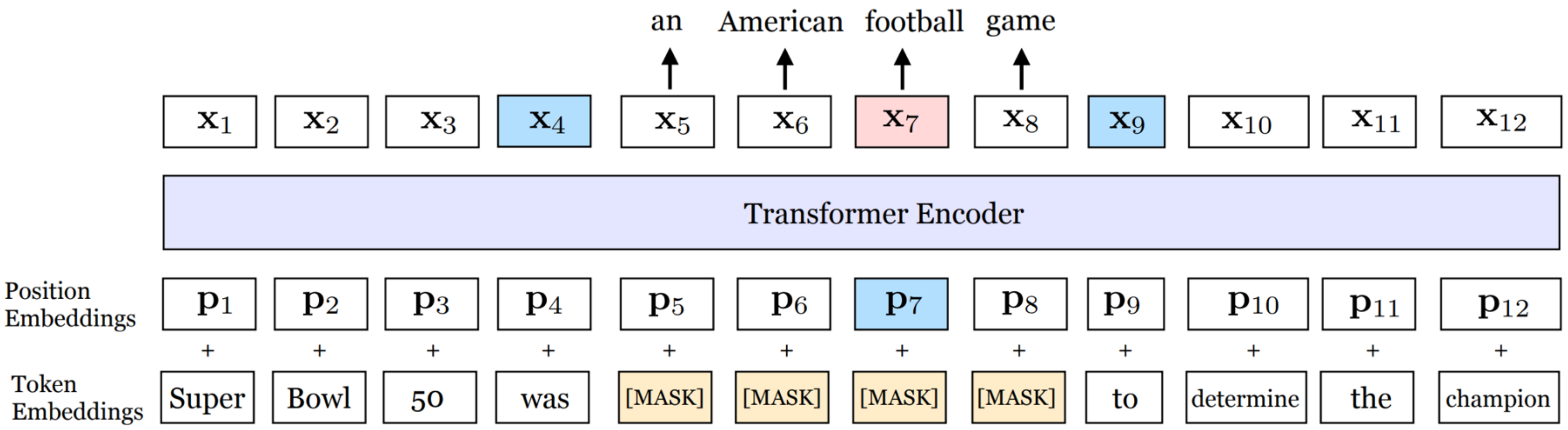
Network Embedding
- DeepWalk
- LINE
- Node2vec
- PTE
- …
- metapath2vec

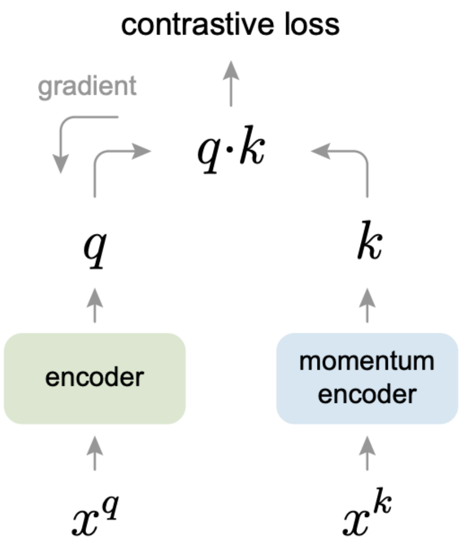
Matrix Factorization
- NetMF
- NetSMF
- …
- ProNE (Propagation)

GNNs
- GCN
- GAT
- GraphSage
- …
- GRAND
- HGT

Language and Image Pre-Training, Graphs?

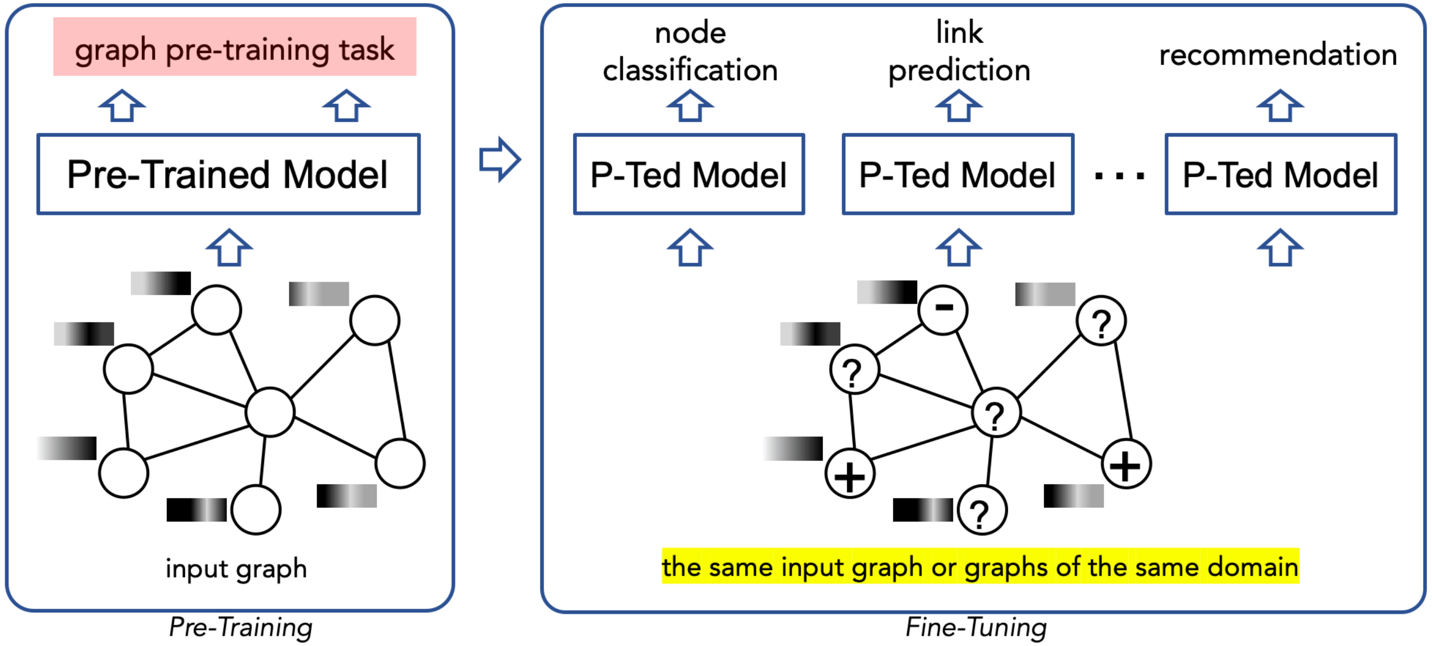
- Recent progress of pre-training models in NLP & CV
  - ELMO, BERT, XLNet, MoCo, etc.
    - Model level: Transformer
    - **Pre-training Task**: masked language modeling & next sentence prediction





GNN Pre-Training

- The FIRST graph pre-training setting:
  - To pre-train from one graph
  - To fine-tune for unseen tasks on the same graph or graphs of the same domain.

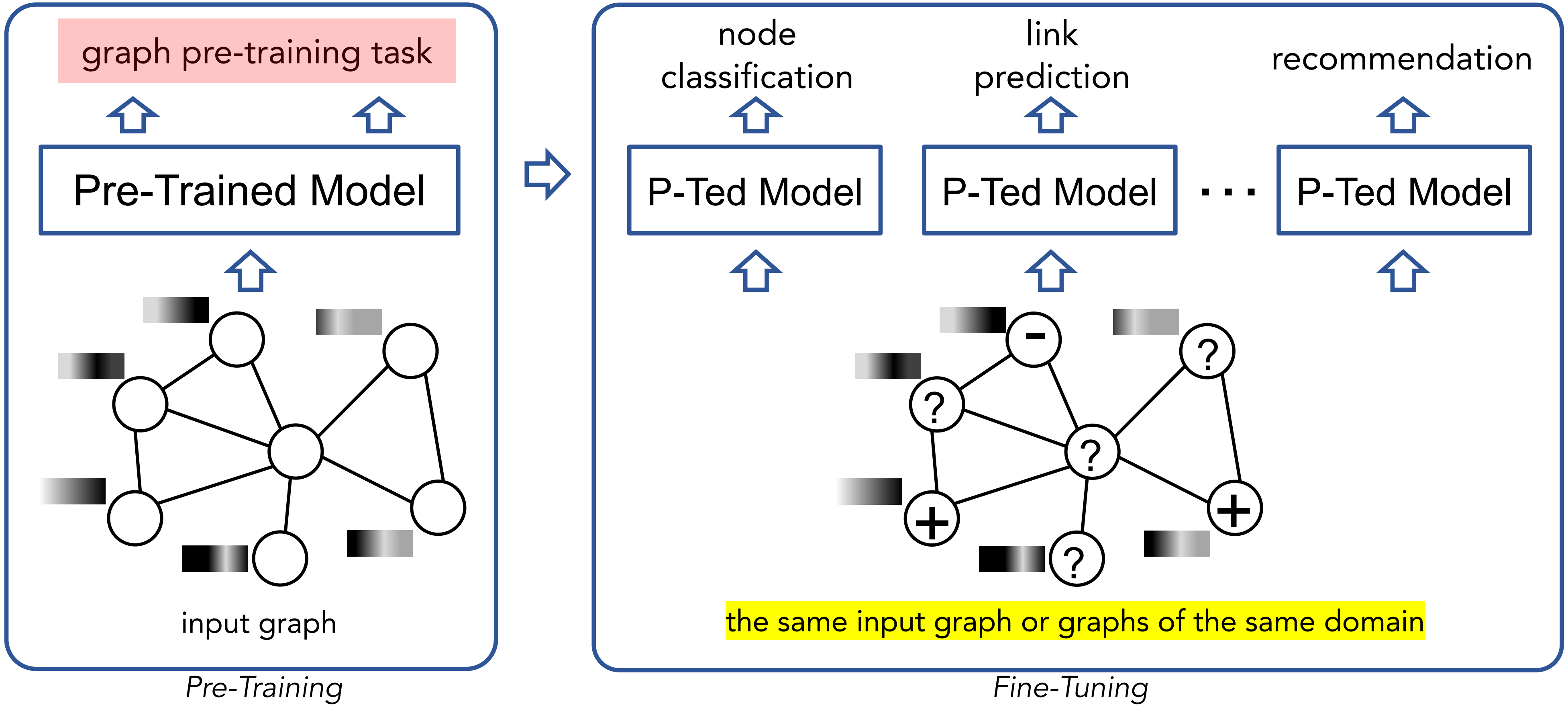


- How to do this?
  - Model level: GNNs?
  - Pre-training task: **self-supervised** tasks on graphs?

1.Ziniu Hu et al. GPT-GNN: Generative Pre-Training of Graph Neural Networks. **KDD** 2020.
2.Code & data for GPT-GNN: https://github.com/acbull/GPT-GNN

GNN Pre-Training

hide

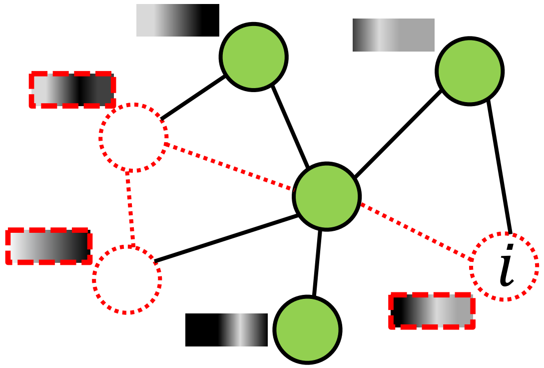


1.Ziniu Hu et al. GPT-GNN: Generative Pre-Training of Graph Neural Networks. **KDD** 2020.
2.Code & data for GPT-GNN: https://github.com/acbull/GPT-GNN

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

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



attribute and edge masked
input graph

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

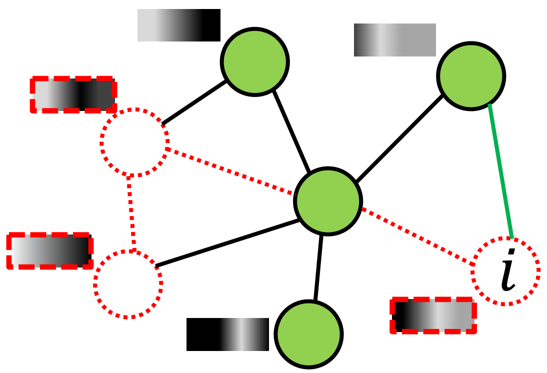
?

1.Ziniu Hu et al. GPT-GNN: Generative Pre-Training of Graph Neural Networks. **KDD** 2020.
2.Code & data for GPT-GNN: https://github.com/acbull/GPT-GNN

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

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

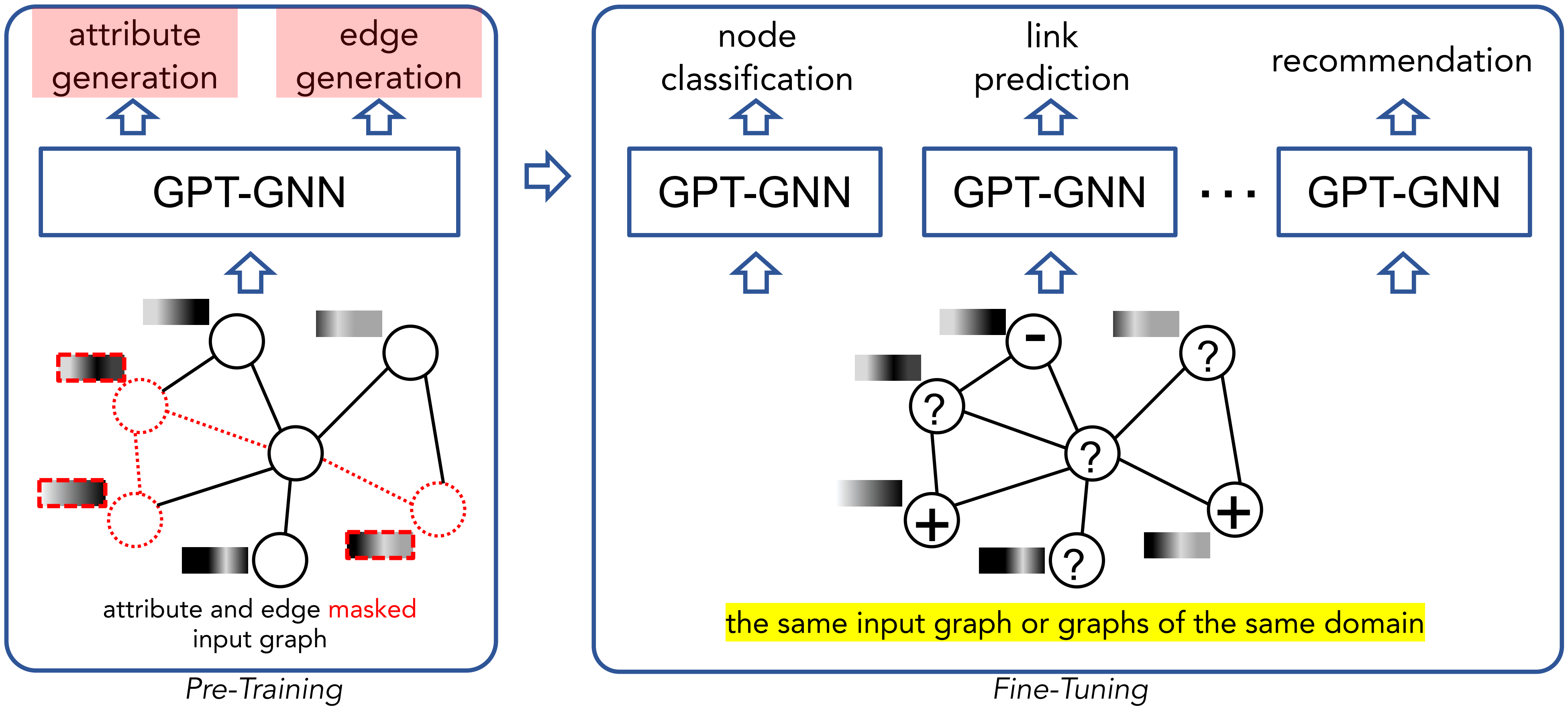


attribute and edge masked
input graph

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

1.Ziniu Hu et al. GPT-GNN: Generative Pre-Training of Graph Neural Networks. **KDD** 2020.
2.Code & data for GPT-GNN: https://github.com/acbull/GPT-GNN

GPT-GNN: Generative Pre-Training of GNNs



1.Ziniu Hu et al. GPT-GNN: Generative Pre-Training of Graph Neural Networks. **KDD** 2020.
2.Code & data for GPT-GNN: https://github.com/acbull/GPT-GNN

GPT-GNN: Generative Pre-Training of GNNs

- Data 1: Open Academic Graph

	Pre-Train	Fine-Tune
Tasks:	• Attribute Generation • Edge Generation	• Inferring the topic of each paper • Inferring the venue of each paper • Author name disambiguation

Base GNN model: Heterogeneous Graph Transformer (HGT)

1.Ziniu Hu et al. GPT-GNN: Generative Pre-Training of Graph Neural Networks. **KDD** 2020.
2.Code & data for GPT-GNN: https://github.com/acbull/GPT-GNN

GPT-GNN: Generative Pre-Training of GNNs

- Data 1: Open 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

1.Ziniu Hu et al. GPT-GNN: Generative Pre-Training of Graph Neural Networks. **KDD** 2020.
2.Code & data for GPT-GNN: https://github.com/acbull/GPT-GNN

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

1. Ziniu Hu et al. GPT-GNN: Generative Pre-Training of Graph Neural Networks. **KDD** 2020.
2. Code & data for GPT-GNN: https://github.com/acbull/GPT-GNN

GPT-GNN: Generative Pre-Training of GNNs

- Data 1: Open Academic Graph

	Pre-Train	Fine-Tune
Tasks:	• Attribute Generation • Edge Generation	• Inferring the topic of each paper • Inferring the venue of each paper • Author name disambiguation

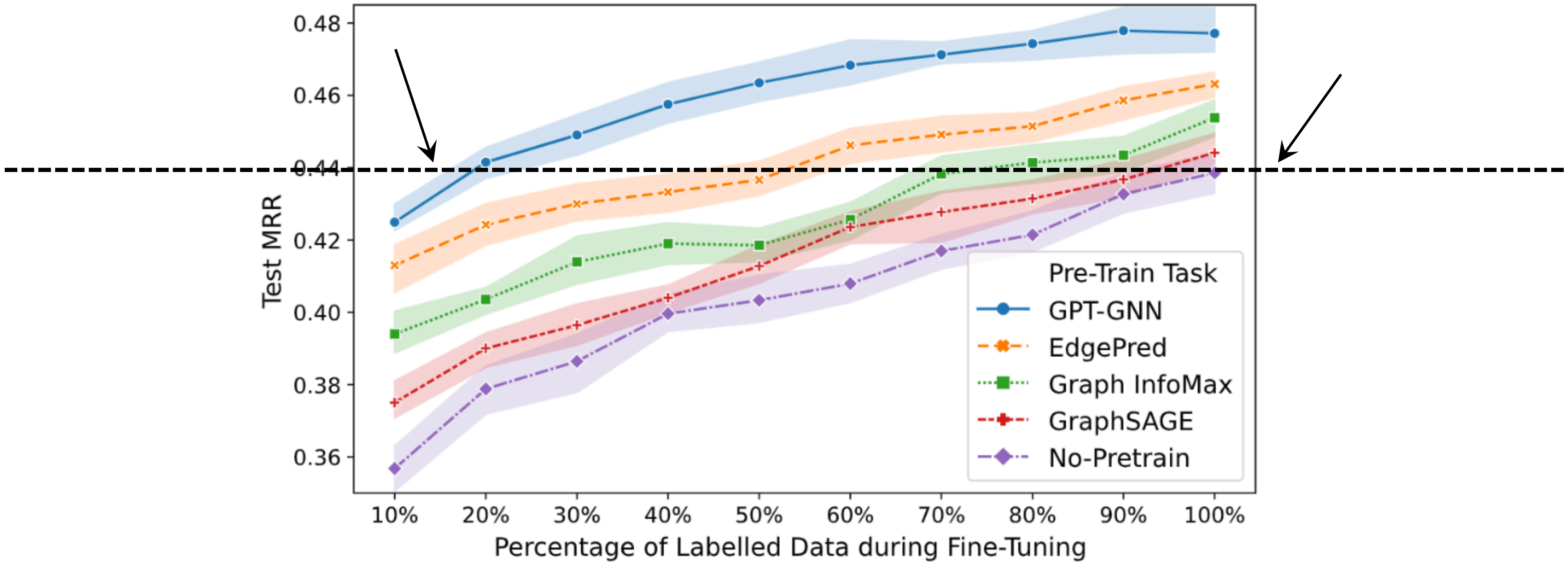
Base GNN model: Heterogeneous Graph Transformer (HGT)

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%

1. Ziniu Hu et al. GPT-GNN: Generative Pre-Training of Graph Neural Networks. **KDD** 2020.
2. Code & data for GPT-GNN: https://github.com/acbull/GPT-GNN

The Promise of Graph Pre-Training!

- During fine-turning



The GNN model **w/o** pre-training by using **100%** training dat**a**

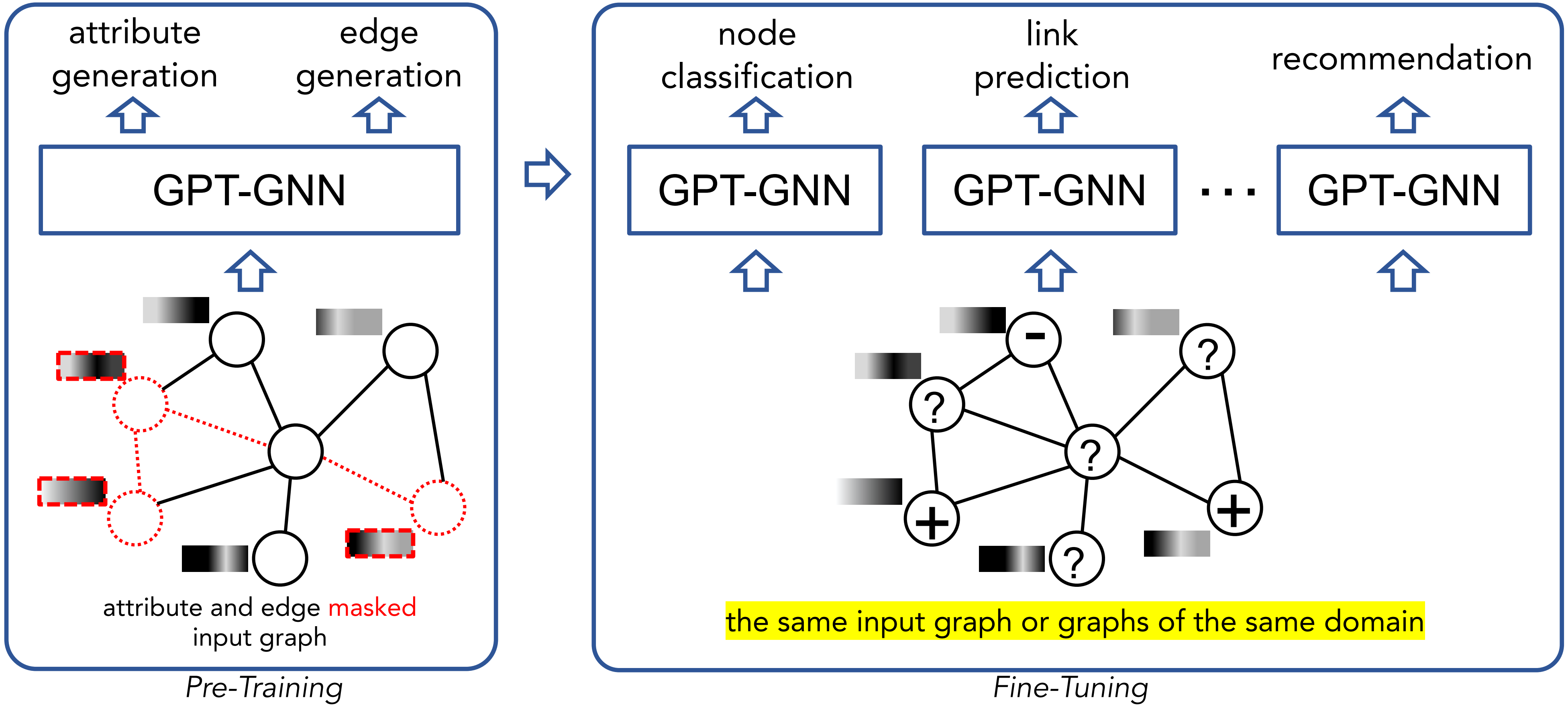
VS

The GNN model **with** pre-training by using **10-20%** training data

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

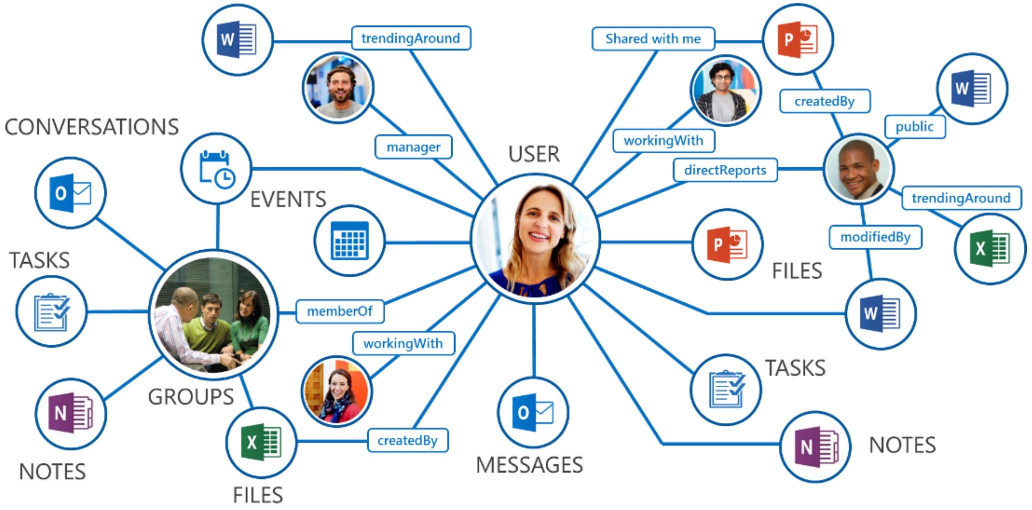
2.Code & data for GPT-GNN: https://github.com/acbull/GPT-GNN

GNN Pre-Training on the “Same” Networks

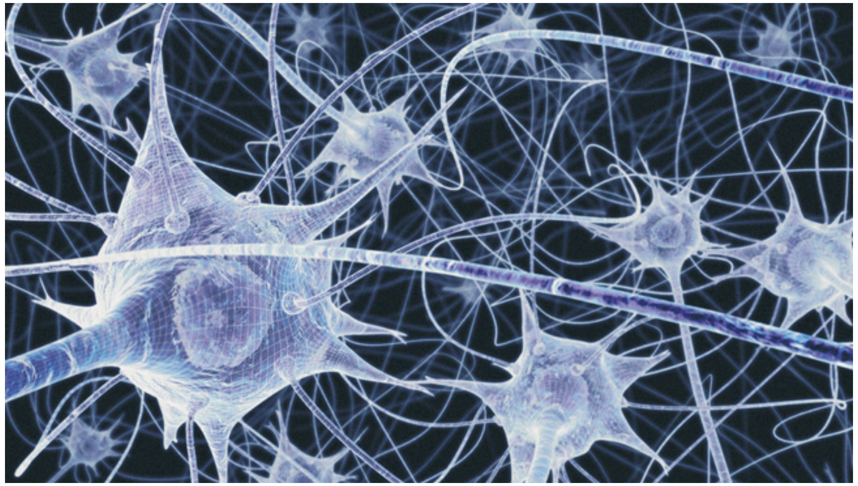


1.Ziniu Hu et al. GPT-GNN: Generative Pre-Training of Graph Neural Networks. **KDD** 2020.
2.Code & data for GPT-GNN: https://github.com/acbull/GPT-GNN

Graphs

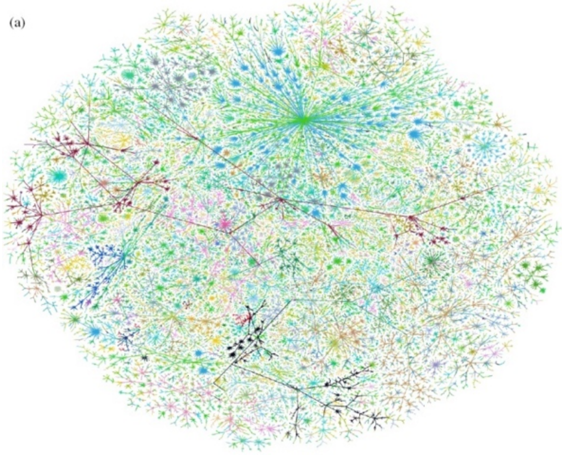


Office/Social Graph

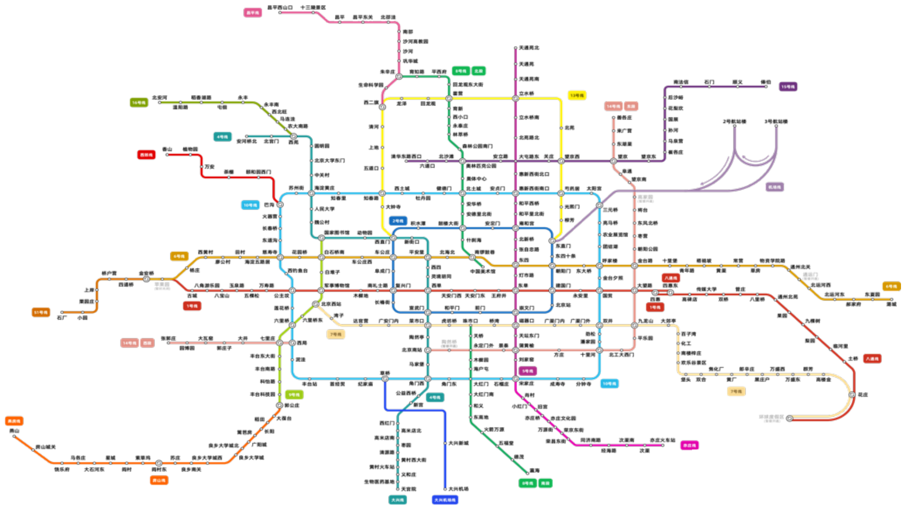


Biological Neural Networks

Knowledge Graph



Internet

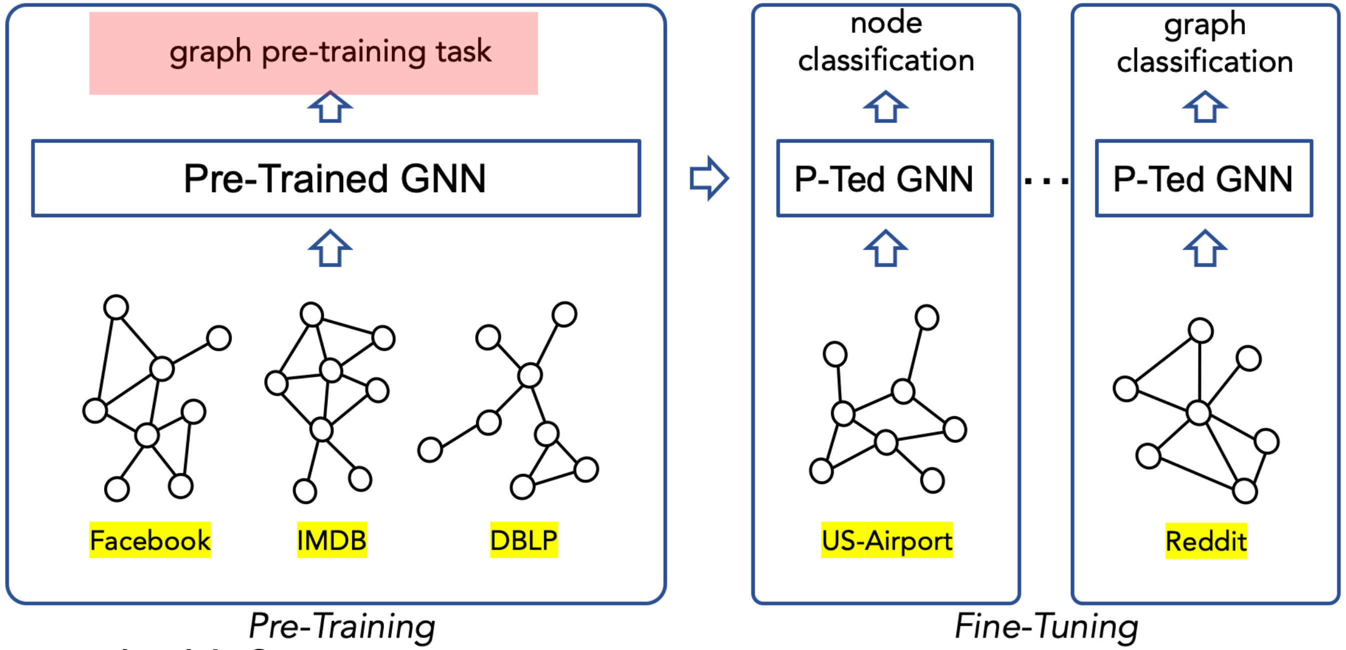


Transportation

figure credit: Web

GNN Pre-Training

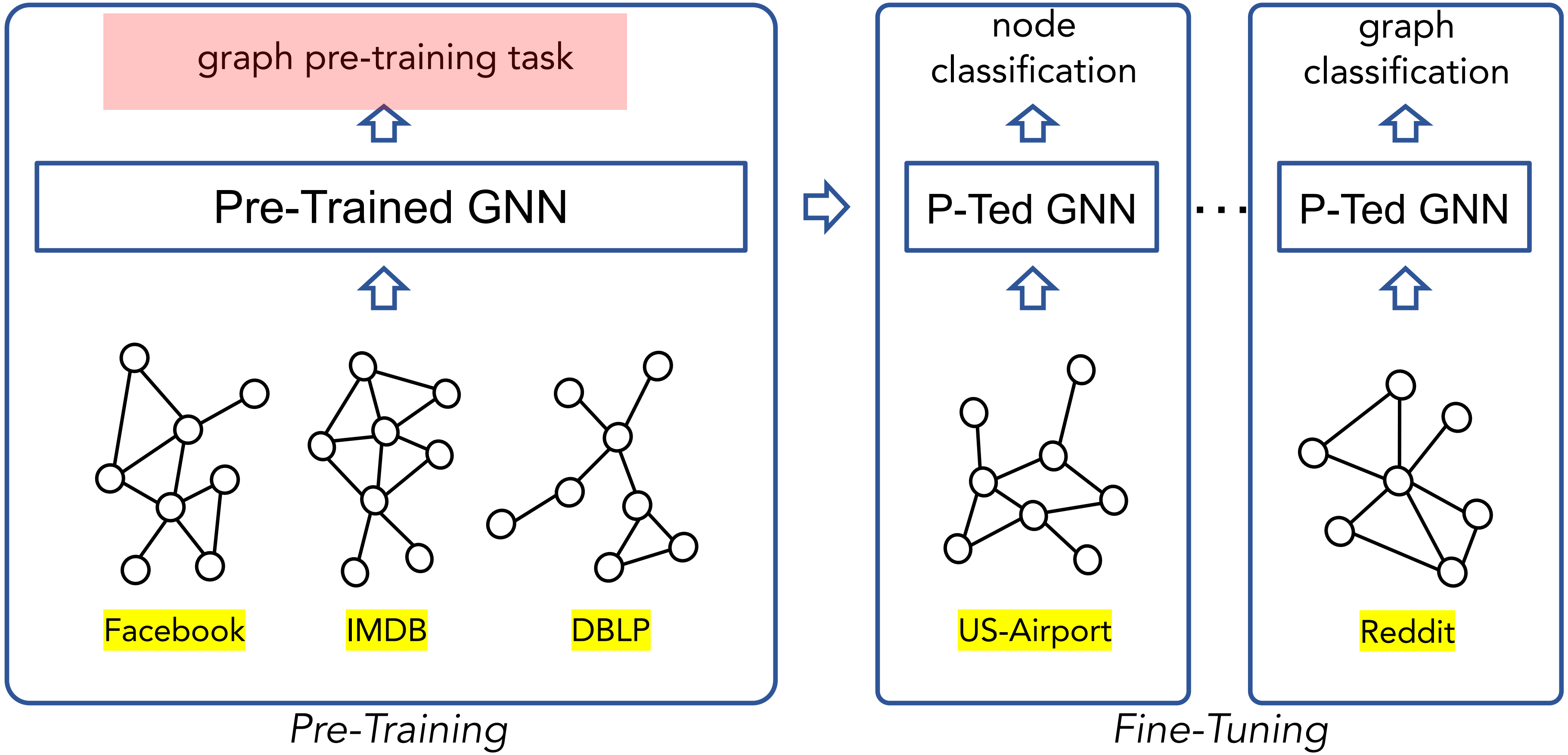
- The SECOND graph pre-training setting:
  - To pre-train from some graphs
  - To fine-tune for unseen tasks on unseen graphs



- How to do this?
  - Model level: GNNs?
  - Pre-training task: **self-supervised** tasks across graphs?

1.Jiezhong Qiu et al. GCC: Graph Contrastive Coding for Graph Neural Network Pre-Training. **KDD** 2020.
2.Code & Data for GCC: https://github.com/THUDM/GCC

GNN Pre-Training across Networks



1.Jiezhong Qiu et al. GCC: Graph Contrastive Coding for Graph Neural Network Pre-Training. **KDD** 2020.
2.Code & Data for GCC: https://github.com/THUDM/GCC

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

1. Jiezhong Qiu et al. GCC: Graph Contrastive Coding for Graph Neural Network Pre-Training. **KDD** 2020.
2. Code & Data for GCC: https://github.com/THUDM/GCC

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

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

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

- 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?

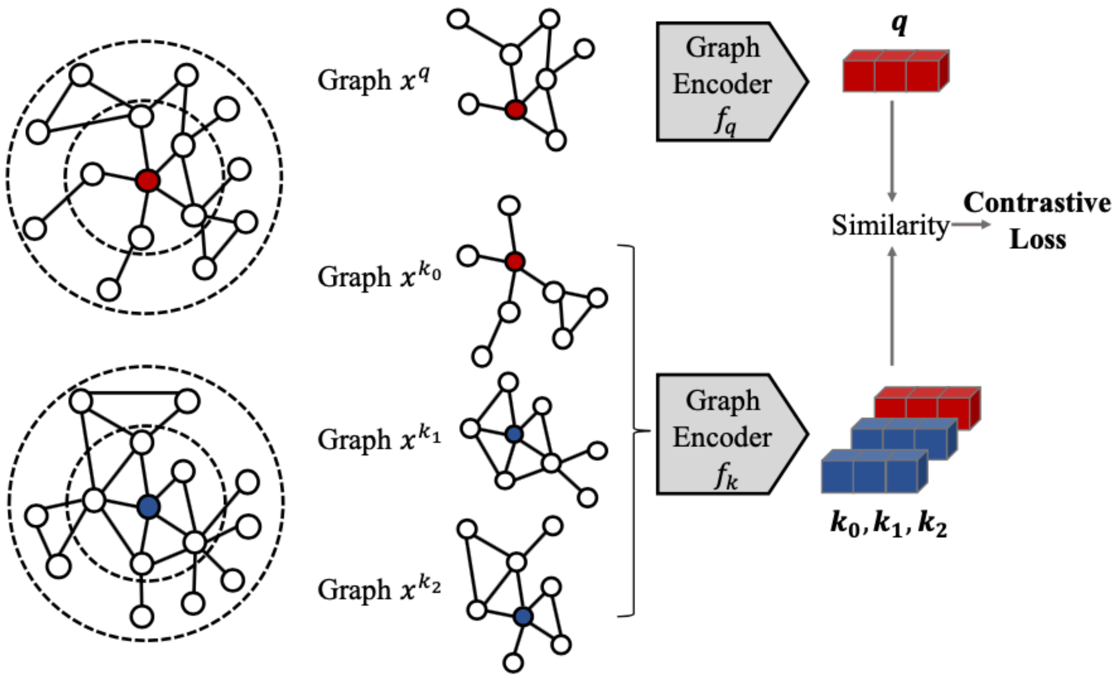
1. Zhirong Wu, Yuanjun Xiong, Stella X Yu, and Dahua Lin. Unsupervised feature learning via non-parametric instance discrimination. In CVPR ’18.
2. Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. 2020. Momentum contrast for unsupervised visual representation learning. In CVPR ’20.

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?

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

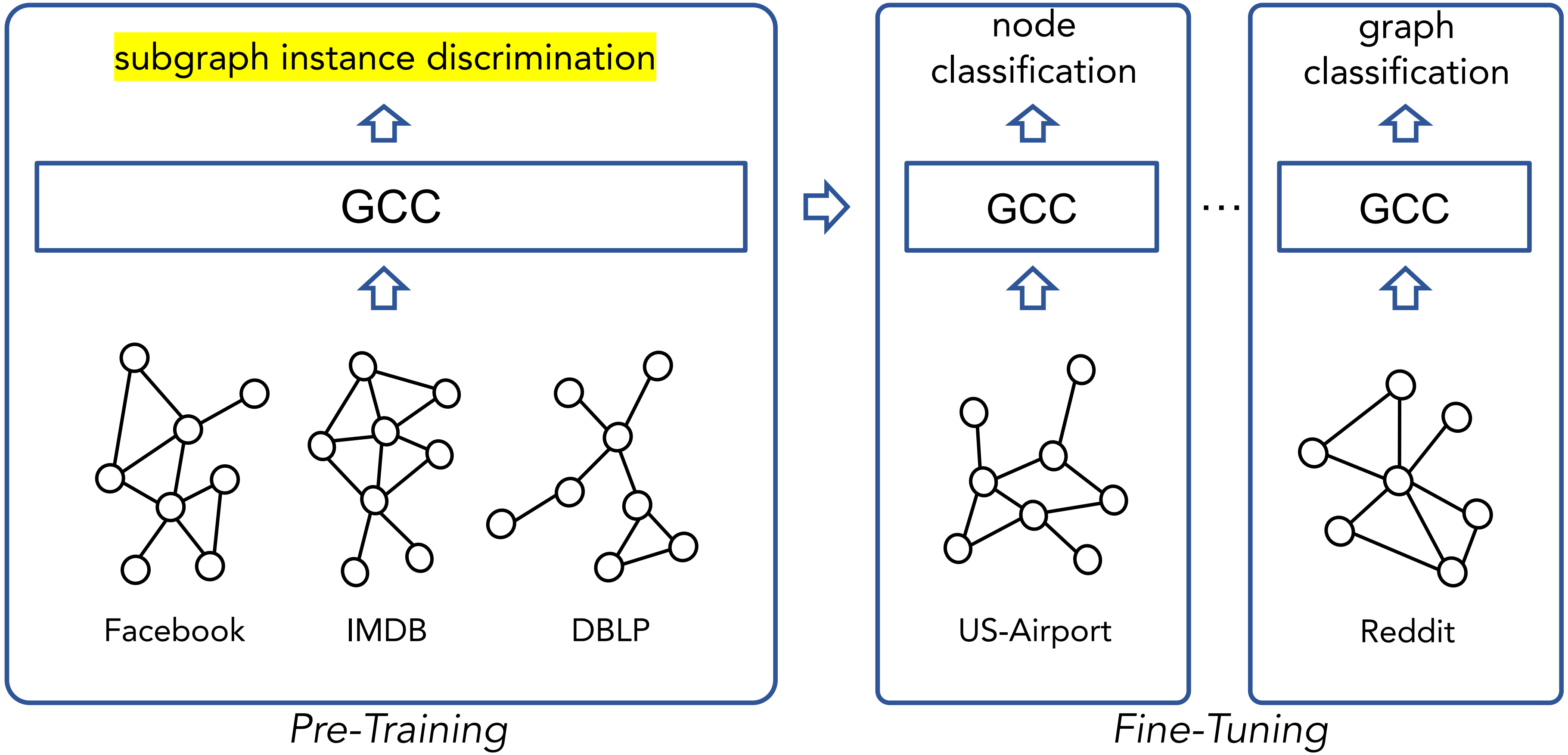
Subgraph instance discrimination



1.Jiezhong Qiu et al. GCC: Graph Contrastive Coding for Graph Neural Network Pre-Training. **KDD** 2020.
2.Code & Data for GCC: https://github.com/THUDM/GCC

hide

Graph Contrastive Coding (GCC)



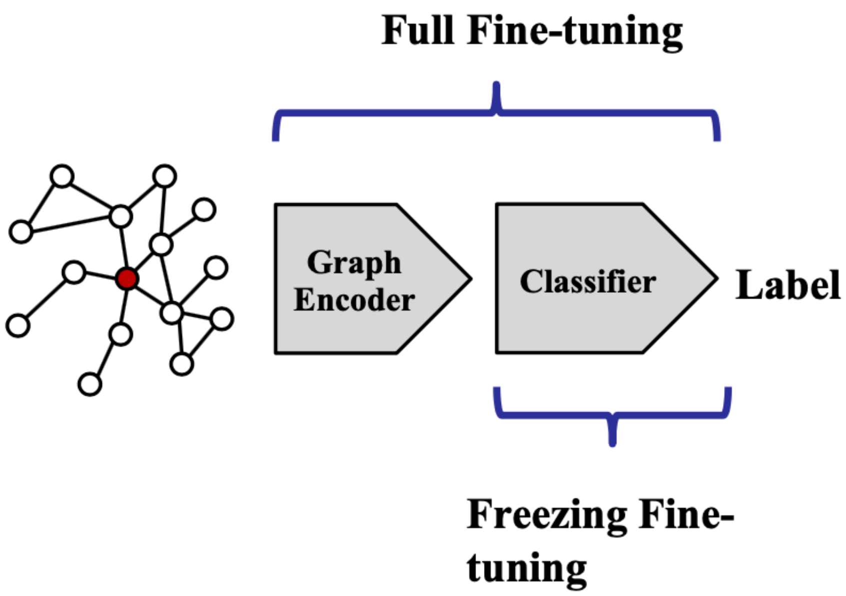
1. Jiezhong Qiu et al. GCC: Graph Contrastive Coding for Graph Neural Network Pre-Training. **KDD** 2020.
2. Code & Data for GCC: https://github.com/THUDM/GCC

GCC Pre-Training / Fine-Tuning

- pre-train on six graphs

Dataset	Academia	DBLP (SNAP)	DBLP (NetRep)	IMDB	Facebook	LiveJournal
$\lvert V \rvert$	137,969	317,080	540,486	896,305	3,097,165	4,843,953
$\lvert E \rvert$	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)



1.Jiezhong Qiu et al. GCC: Graph Contrastive Coding for Graph Neural Network Pre-Training. **KDD** 2020.
2.Code & Data for GCC: https://github.com/THUDM/GCC

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

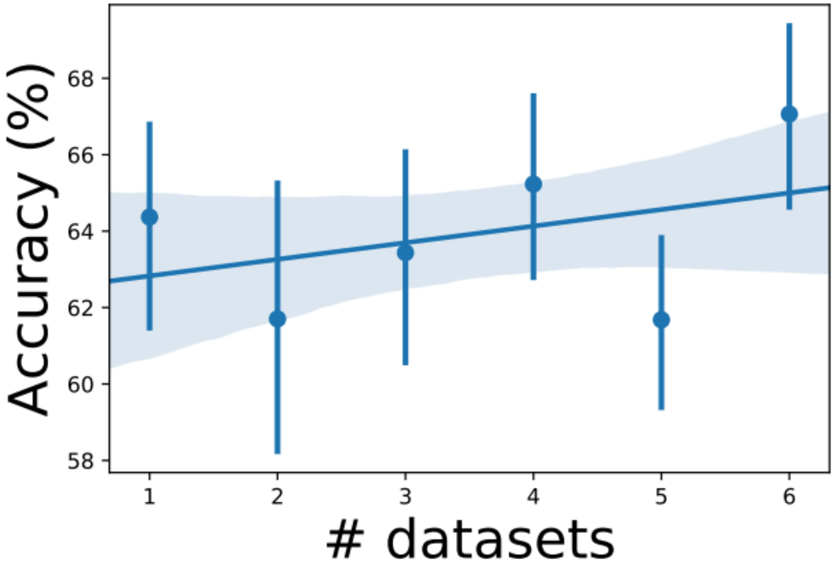
	KDD-ICDM		SIGIR-CIKM		SIGMOD-ICDE	
$\|V\|$	2,867	2,607	2,851	3,548	2,616	2,559
$\|E\|$	7,637	4,774	6,354	7,076	8,304	6,668
# groud truth		697		874		898
k	20	40	20	40	20	40
Random	0.0198	0.0566	0.0223	0.0447	0.0221	0.0521
RolX	0.0779	0.1288	0.0548	0.0984	0.0776	0.1309
Panther++	0.0892	0.1558	**0.0782**	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

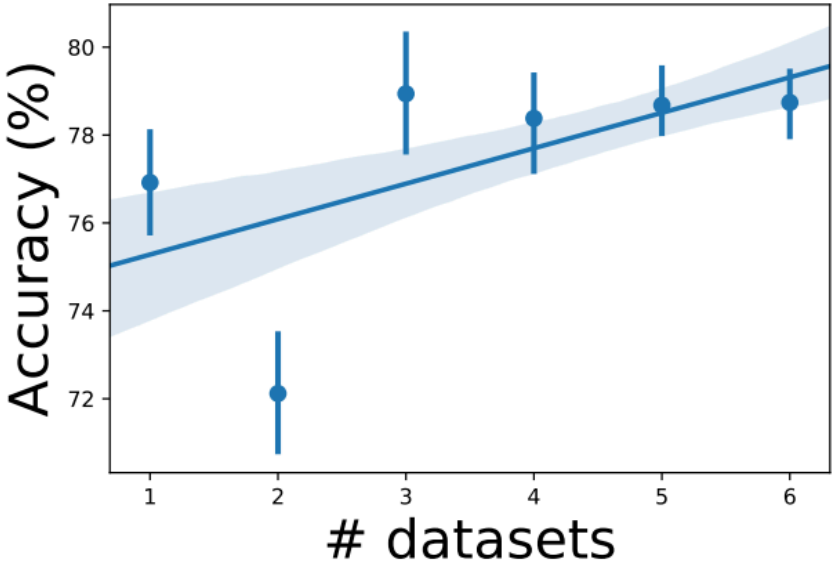
Datasets	IMDB-B	IMDB-M	COLLAB	RDT-B	RDT-M
# graphs	1,000	1,500	5,000	2,000	5,000
# classes	2	3	3	2	5
Avg. # nodes	19.8	13.0	74.5	429.6	508.5
DGK	67.0	44.6	73.1	78.0	41.3
graph2vec	71.1	50.4	–	75.8	47.9
InfoGraph	**73.0**	**49.7**	–	82.5	53.5
GCC (E2E, freeze)	71.7	49.3	74.7	87.5	52.6
GCC (MoCo, freeze)	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

1.Jiezhong Qiu et al. GCC: Graph Contrastive Coding for Graph Neural Network Pre-Training. **KDD** 2020.
2.Code & Data for GCC: https://github.com/THUDM/GCC

Results



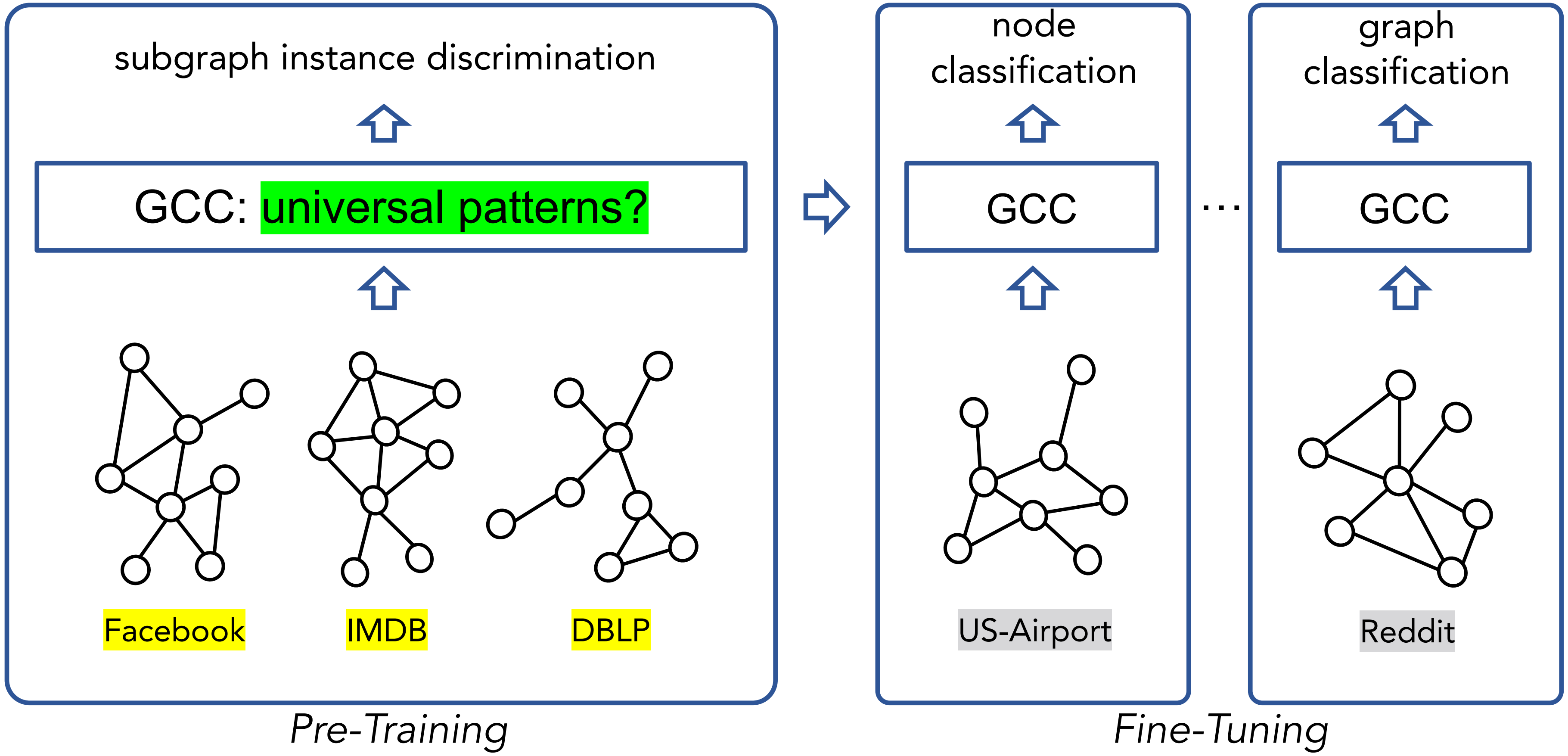
(a) US-Airport: $y = 0.4344x + 62.394$



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

1.Jiezhong Qiu et al. GCC: Graph Contrastive Coding for Graph Neural Network Pre-Training. **KDD** 2020.
2.Code & Data for GCC: https://github.com/THUDM/GCC

Results



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

1.Jiezhong Qiu et al. GCC: Graph Contrastive Coding for Graph Neural Network Pre-Training. **KDD** 2020.
2.Code & Data for GCC: https://github.com/THUDM/GCC

Graph Representation Learning

Network Embedding ⇔ **Matrix Factorization** ⇨ **GNNs** ⇨ **Pre-Training**

Network Embedding
- DeepWalk
- LINE
- Node2vec
- PTE
- …
- metapath2vec

Matrix Factorization
- NetMF
- NetSMF
- …
- ProNE (Propagation)

GNNs
- GCN
- GAT
- GraphSage
- …
- GRAND
- HGT

Pre-Training
- Generative
  - GPT-GNN
- Contrastive
  - GCC

What graph data to use?

OPEN GRAPH BENCHMARK

Realistic datasets

Flexible data loaders

Unified evaluation

https://ogb.stanford.edu/

Leaderboard for ogbn-arxiv

The classification accuracy on the test set. The higher, the better.

Rank	Method	Accuracy	Contact	References	Date
1	GCN	0.7174 ± 0.0029	Matthias Fey – OGB team	Paper, Code	May 1, 2020
2	GraphSAGE	0.7149 ± 0.0027	Matthias Fey – OGB team	Paper, Code	May 1, 2020
3	Node2vec	0.7007 ± 0.0013	Matthias Fey – OGB team	Paper, Code	May 1, 2020
4	MLP	0.5550 ± 0.0023	Matthias Fey – OGB team	Paper, Code	May 1, 2020

1. Weihua Hu, et al. Open Graph Benchmark: Datasets for Machine Learning on Graphs. arXiv 2020

Microsoft Academic Graph & AMiner & OAG

236,963,398
Publications

240,667,697
Authors

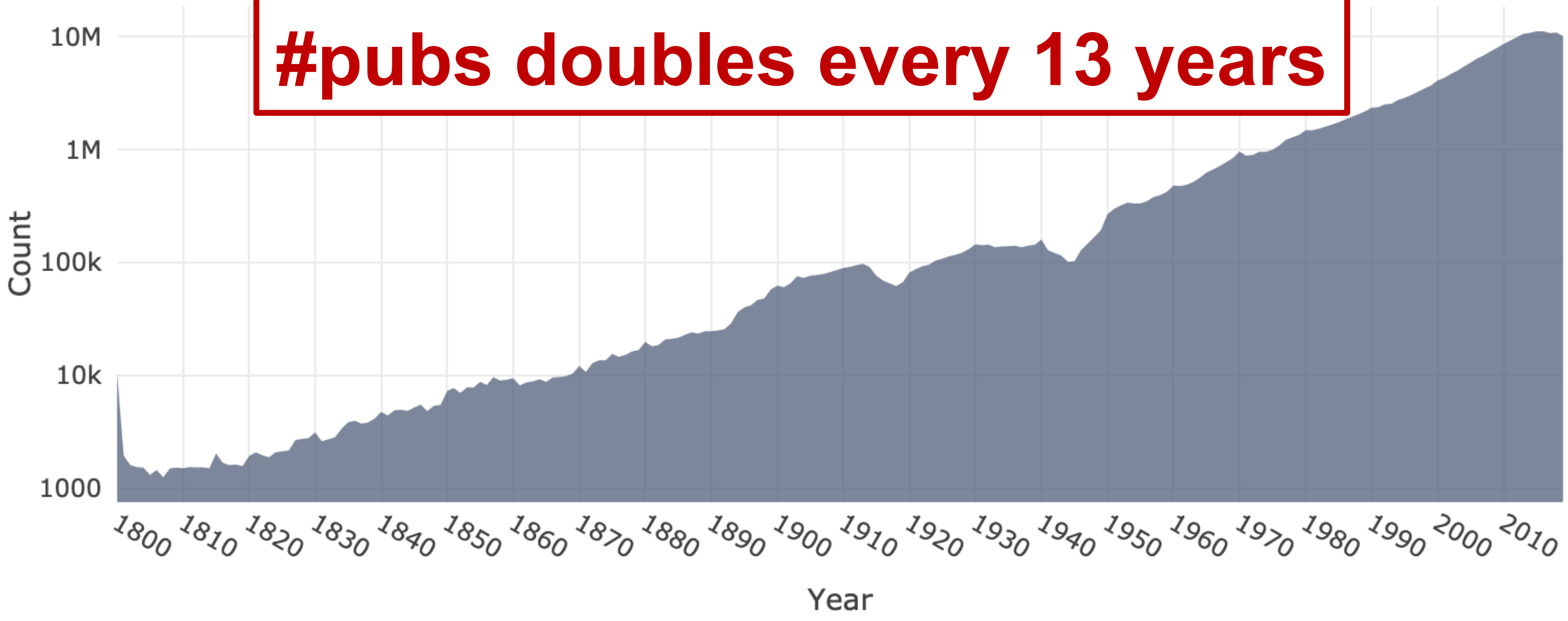
740,048
Topics

4,467
Conferences

48,876
Journals

25,759
Institutions

1800 --- 2019
#pubs doubles every 13 years



1. Kuansan Wang et al. Microsoft Academic Graph: When experts are not enough. Quantitative Science Studies 1 (1), 2020
2. Fanjin Zhang et al. OAG: Toward Linking Large-scale Heterogeneous Entity Graphs. KDD 2019.
3. Jie Tang et al. Arnetminer: extraction and mining of academic social networks. In KDD 2008.

References

1. Ziniu Hu et al. GPT-GNN: Generative Pre-Training of Graph Neural Networks. **KDD** 2020.
2. Jiezhong Qiu et al. GCC: Graph Contrastive Coding for Graph Neural Network Pre-Training. **KDD** 2020.
3. Kuansan Wang et al. Microsoft Academic Graph: When experts are not enough. Quantitative Science Studies 1 (1), 396-413, 2020.
4. Weihua Hu et al. Open Graph Benchmark: Datasets for Machine Learning on Graphs. arXiv 2020.
5. Feng et al. Graph Random Neural Networks. arXiv 2020.
6. Ziniu Hu et al. Heterogeneous Graph Transformer. **WWW** 2020.
7. Yuxiao Dong et al. Heterogeneous Network Representation Learning. **IJCAI** 2020.
8. Jiezhong Qiu et al. *NetSMF: Large-Scale Network Embedding as Sparse Matrix Factorization*. **WWW** 2019.
9. Jie Zhang et al. *ProNE: Fast and Scalable Network Representation Learning*. **IJCAI** 2019.
10. Fanjing Zhang et al. OAG: Toward Linking Large-scale Heterogeneous Entity Graphs. ACM ***KDD*** 2019.
11. Xian Wu et al. Neural Tensor Decomposition. **WSDM** 2019.
12. Jiezhong Qiu et al. *Network Embedding as Matrix Factorization: Unifying DeepWalk, LINE, PTE, and node2vec.* **WSDM** 2018.
13. Yuxiao Dong et al. *metapath2vec: Scalable Representation Learning for Heterogeneous Networks*. **KDD** 2017.
14. Perozzi et al. DeepWalk: Online learning of social representations. In **KDD**’ 14.
15. Tang et al. LINE: Large scale information network embedding. In **WWW’15**.
16. Grover and Leskovec. node2vec: Scalable feature learning for networks. In **KDD’16**.
17. Harris, Z. (1954). Distributional structure. ***Word***, 10(23): 146-162.
18. Kipf et al. Semisupervised Classification with Graph Convolutional Networks. **ICLR** 2017
19. Velickovic et al. Graph Attention Networks. **ICLR** 2018
20. Hamilton et al. Inductive Representation Learning on Large Graphs. **NeurIPS** 2017
21. Defferrard et al. Convolutional Neural Networks on Graphs with Fast Locailzied Spectral Filtering. In **NeurIPS** 2016
22. Jacob Devlin et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. **NAACL-HLT** 2019.
23. Justin Gilmer, et al. Neural message passing for quantum chemistry. arXiv: 2017.
24. Kaiming He, et al. Momentum contrast for unsupervised visual representation learning. arXiv: 2019
25. Tomas Mikolov et al. Distributed representations of words and phrases and their compositionality. **NeurIPS** 2013.
26. Petar Velickovic et al. Deep Graph Infomax. In **ICLR** 19.
27. Zhen Yang et al. Understanding Negative Sampling in Graph Representation Learning. **KDD** 2020.

Thank you!

Jiezhong Qiu
Tsinghua
(Jie Tang)

Ziniu Hu
UCLA
(Yizhou Sun)

Hongxia Yang
Alibaba

Jing Zhang
Renmin U. of China

Jie Tang
Tsinghua

Yizhou Sun
UCLA

Hao Ma
Facebook AI

Kuansan Wang
Microsoft Research

Papers & data & code available at https://ericdongyx.github.io/
ericdongyx@gmail.com