Advanced Network Embedding

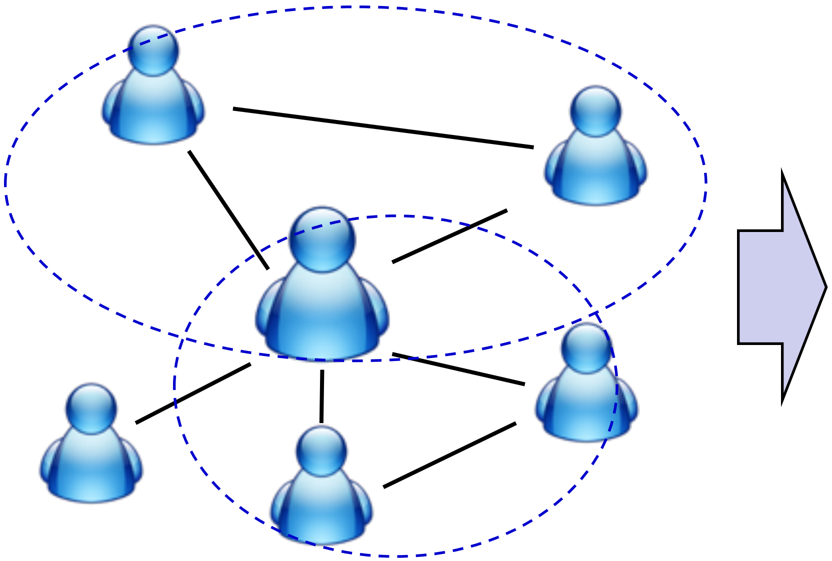
Yukuo Cen
GNN Center, Zhipu AI
KEG, Tsinghua University
Advisors: Yuxiao Dong, Jie Tang

Course Link: https://cogdl.ai/gnn2022/

CogDL is publicly available at https://github.com/THUDM/cogdl

Review Representation Learning for Graphs

Representation Learning/
Graph Embedding

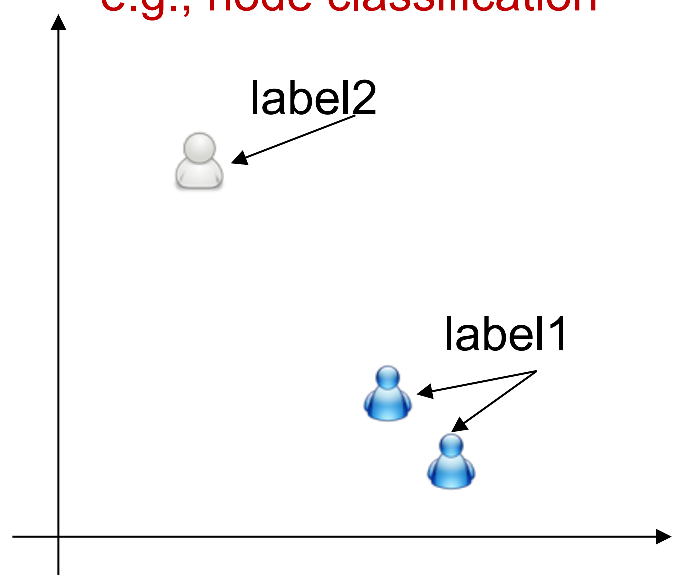


d-dimensional vector, *d*<<|*V*|

0.8	0.2	0.3	…	0.0	0.0

Users with the same label are located in the *d*-dimensional space closer than those with different labels

e.g., node classification



Questions

- What are the fundamentals underlying the different methods?

or

- Can we unify the different network embedding approaches?

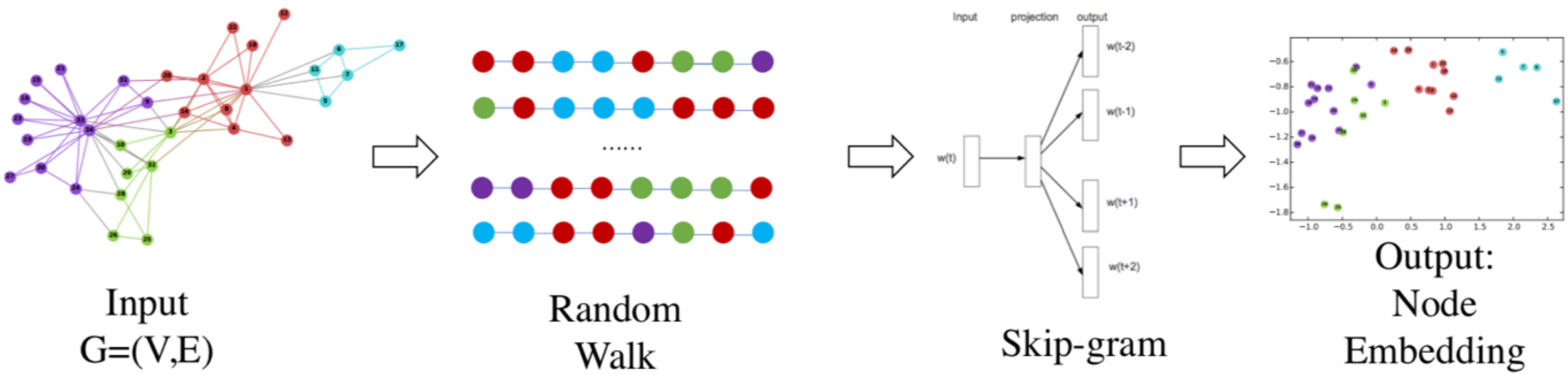
Unifying DeepWalk, LINE, PTE, and node2vec into Matrix Forms

Algorithm	Closed Matrix Form
DeepWalk	$\log\left(\mathrm{vol}(G)\left(\frac{1}{T}\sum_{r=1}^{T}(\boldsymbol{D}^{-1}\boldsymbol{A})^r\right)\boldsymbol{D}^{-1}\right)-\log b$
LINE	$\log\left(\mathrm{vol}(G)\boldsymbol{D}^{-1}\boldsymbol{A}\boldsymbol{D}^{-1}\right)-\log b$
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	$\log\left(\frac{\frac{1}{2T}\sum_{r=1}^{T}\left(\sum_u \boldsymbol{X}_{w,u}\underline{\boldsymbol{P}}_{c,w,u}^{r}+\sum_u \boldsymbol{X}_{c,u}\underline{\boldsymbol{P}}_{w,c,u}^{r}\right)}{(\sum_u \boldsymbol{X}_{w,u})(\sum_u \boldsymbol{X}_{c,u})}\right)-\log b$

$\boldsymbol{A}$: $\boldsymbol{A}\in\mathbb{R}_+^{|V|\times|V|}$ is G's adjacency matrix with $\boldsymbol{A}_{i,j}$ as the edge weight between vertices i and j;
$\boldsymbol{D}_{\mathrm{col}}$: $\boldsymbol{D}_{\mathrm{col}}=\mathrm{diag}(\boldsymbol{A}^\top\boldsymbol{e})$ is the diagonal matrix with column sum of $\boldsymbol{A}$;
$\boldsymbol{D}_{\mathrm{row}}$: $\boldsymbol{D}_{\mathrm{row}}=\mathrm{diag}(\boldsymbol{A}\boldsymbol{e})$ is the diagonal matrix with row sum of $\boldsymbol{A}$;
$\boldsymbol{D}$: For undirected graphs ($\boldsymbol{A}^\top=\boldsymbol{A}$), $\boldsymbol{D}_{\mathrm{col}}=\boldsymbol{D}_{\mathrm{row}}$. For brevity, $\boldsymbol{D}$ represents both $\boldsymbol{D}_{\mathrm{col}}$ & $\boldsymbol{D}_{\mathrm{row}}$.
$\boldsymbol{D}=\mathrm{diag}(d_1,\cdots,d_{|V|})$, where d_i represents generalized degree of vertex i;
$\mathrm{vol}(G)$: $\mathrm{vol}(G)=\sum_i\sum_j \boldsymbol{A}_{i,j}=\sum_i d_i$ is the volume of an weighted graph G;
T & b: The context window size and the number of negative sampling in skip-gram, respectively.

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

Starting with DeepWalk



DeepWalk Algorithm

Algorithm 1: DeepWalk

for $n = 1, 2, \ldots, N$ **do**
- Pick w_1^n according to a probability distribution $P(w_1)$;
- Generate a vertex sequence $(w_1^n, \cdots, w_L^n)$ of length L by a random walk on network G;
- **for** $j = 1, 2, \ldots, L - T$ **do**
  - **for** $r = 1, \ldots, T$ **do**
    - Add vertex-context pair (w_j^n, w_{j+r}^n) to multiset $\mathcal{D}$;
    - Add vertex-context pair (w_{j+r}^n, w_j^n) to multiset $\mathcal{D}$;

Run SGNS on $\mathcal{D}$ with b negative samples.

Skip-gram with Negative Sampling

- SGNS maintains a multiset $\mathcal{D}$ that counts the occurrence of each word-context pair (w, c)
- Objective

$$\mathcal{L} = \sum_{w} \sum_{c} \left(\#(w, c) \log g(x_w^T x_c) + \frac{b\#(w)\#(c)}{|\mathcal{D}|} \log g(-x_w^T x_c) \right)$$

where x_w and x_c are d-dimenational vector

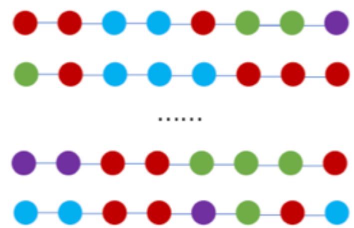
- For sufficiently large dimension d, the objective above is equivalent to factorizing the PMI matrix[1]

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

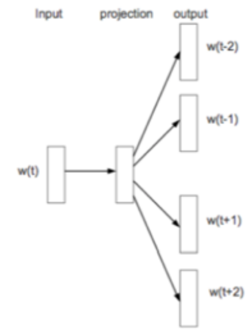
1. Levy and Goldberg. Neural word embeddings as implicit matrix factorization. In *NIPS 2014*

PMI Matrix of Random Walks on a Graph

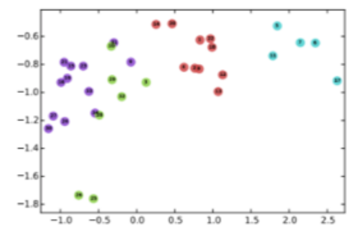
Input
G=(V,E)



Random
Walk



Skip-gram



Output:
Node
Embedding

Levy&Goldberg (NIPS'14)

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

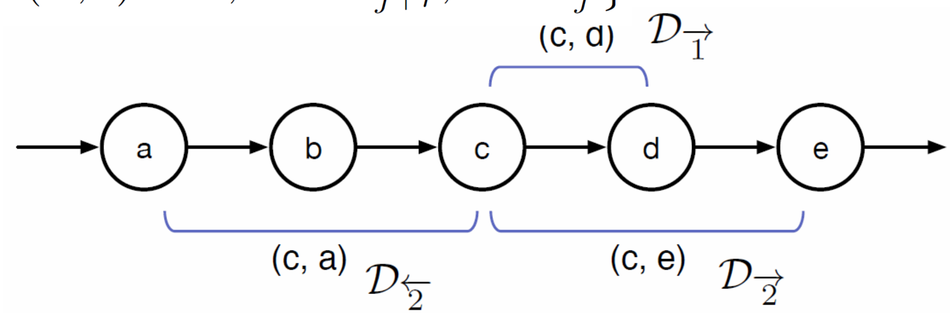
?

Understanding random walk + skip gram

- 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

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

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



THEOREM 2.1. *Denote $\boldsymbol{P} = \boldsymbol{D}^{-1}\boldsymbol{A}$, when $L \to \infty$, we have*

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

THEOREM 2.2. *When $L \to \infty$, we have*

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

Understanding random walk + skip gram

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

Understanding random walk + skip gram

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

the length of random walk $L \rightarrow \infty$

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

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

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

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

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

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

Understanding random walk + skip gram

- Write it in matrix form:

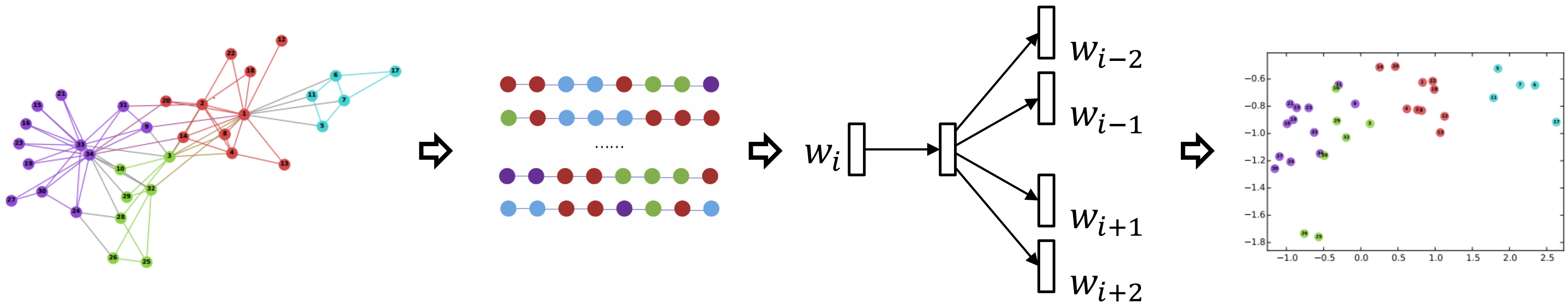
$$\frac{\#(w,c)\,|\mathcal{D}|}{\#(w)\cdot\#(c)} \xrightarrow{p} \frac{\mathrm{vol}(G)}{2T}\left(\frac{1}{d_c}\sum_{r=1}^{T}(\boldsymbol{P}^r)_{w,c} + \frac{1}{d_w}\sum_{r=1}^{T}(\boldsymbol{P}^r)_{c,w}\right)$$

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

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

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

DeepWalk is factorizing a matrix



DeepWalk is asymptotically and implicitly factorizing

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

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

$\boldsymbol{A}$ Adjacency matrix

$\boldsymbol{D}$ Degree matrix

b: #negative samples

T: context window size

LINE

- Objective of LINE:

$$\mathcal{L} = \sum_{i=1}^{|V|} \sum_{j=1}^{|V|} \left(\boldsymbol{A}_{i,j} \log g\left(\boldsymbol{x}_i^\top \boldsymbol{y}_j\right) + \frac{b d_i d_j}{\mathrm{vol}(G)} \log g\left(-\boldsymbol{x}_i^\top \boldsymbol{y}_j\right) \right).$$

- Align it with the Objective of SGNS:

$$\mathcal{L} = \sum_{w} \sum_{c} \left(\#(w,c) \log g\left(\boldsymbol{x}_w^\top \boldsymbol{y}_c\right) + \frac{b\#(w)\#(c)}{|\mathcal{D}|} \log g\left(-\boldsymbol{x}_w^\top \boldsymbol{y}_c\right) \right).$$

- LINE is actually factorizing

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

- Recall DeepWalk's matrix form:

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

Observation LINE is a special case of DeepWalk ($T = 1$).

PTE

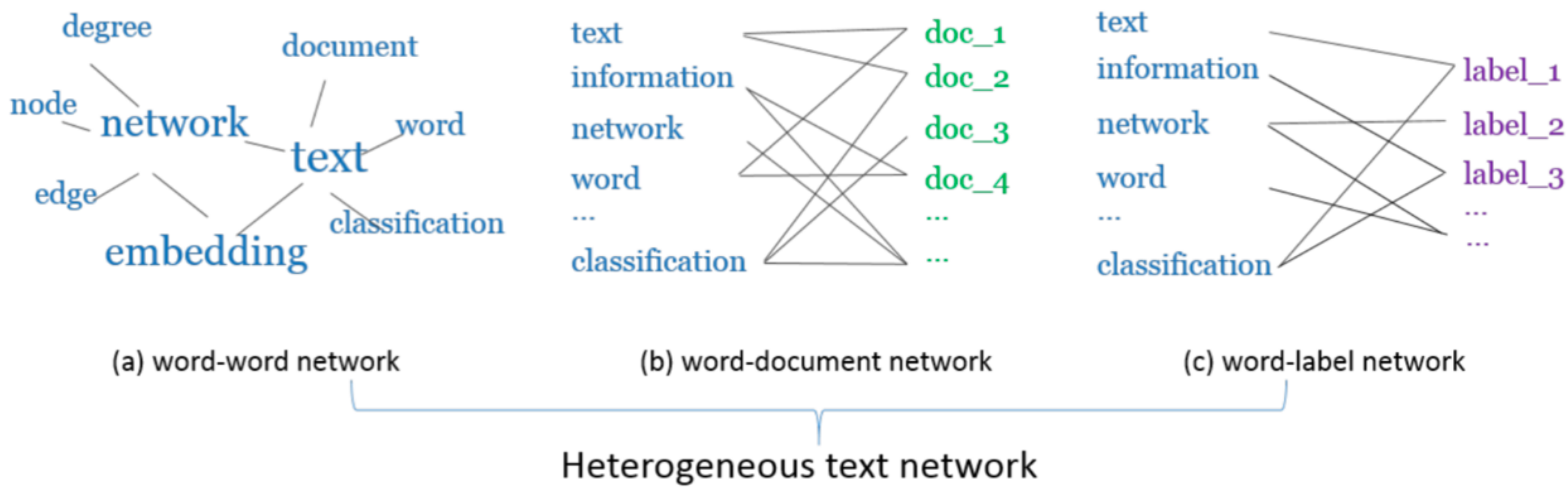


Figure 2: Heterogeneous Text Network.

$$\log\left(\begin{bmatrix} \alpha\,\mathrm{vol}(G_{\mathsf{ww}})(\boldsymbol{D}^{\mathsf{ww}}_{\mathsf{row}})^{-1}\boldsymbol{A}_{\mathsf{ww}}(\boldsymbol{D}^{\mathsf{ww}}_{\mathsf{col}})^{-1} \\ \beta\,\mathrm{vol}(G_{\mathsf{dw}})(\boldsymbol{D}^{\mathsf{dw}}_{\mathsf{row}})^{-1}\boldsymbol{A}_{\mathsf{dw}}(\boldsymbol{D}^{\mathsf{dw}}_{\mathsf{col}})^{-1} \\ \gamma\,\mathrm{vol}(G_{\mathsf{lw}})(\boldsymbol{D}^{\mathsf{lw}}_{\mathsf{row}})^{-1}\boldsymbol{A}_{\mathsf{lw}}(\boldsymbol{D}^{\mathsf{lw}}_{\mathsf{col}})^{-1} \end{bmatrix}\right) - \log b,$$

node2vec — 2nd Order Random Walk

$$\underline{\boldsymbol{T}}_{u,v,w} = \begin{cases} \frac{1}{p} & (u,v) \in E, (v,w) \in E, u = w; \\ 1 & (u,v) \in E, (v,w) \in E, u \neq w, (w,u) \in E; \\ \frac{1}{q} & (u,v) \in E, (v,w) \in E, u \neq w, (w,u) \notin E; \\ 0 & \text{otherwise.} \end{cases}$$

$$\underline{\boldsymbol{P}}_{u,v,w} = \text{Prob}\left(w_{j+1} = u | w_j = v, w_{j-1} = w\right) = \frac{\underline{\boldsymbol{T}}_{u,v,w}}{\sum_u \underline{\boldsymbol{T}}_{u,v,w}}.$$

Stationary Distribution

$$\sum_w \underline{\boldsymbol{P}}_{u,v,w} \boldsymbol{X}_{v,w} = \boldsymbol{X}_{u,v}$$

Existence guaranteed by Perron-Frobenius theorem, but may not be unique.

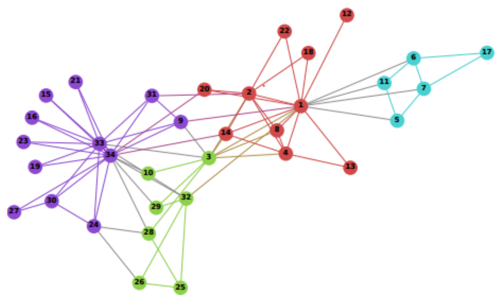
$$\frac{\#(\boldsymbol{w}, \boldsymbol{c}) \, |\mathcal{D}|}{\#(\boldsymbol{w}) \cdot \#(\boldsymbol{c})} \xrightarrow{p} \frac{\frac{1}{2T} \sum_{r=1}^{T} \left(\sum_u \boldsymbol{X}_{\boldsymbol{w},u} \underline{\boldsymbol{P}}^{r}_{\boldsymbol{c},\boldsymbol{w},u} + \sum_u \boldsymbol{X}_{\boldsymbol{c},u} \underline{\boldsymbol{P}}^{r}_{\boldsymbol{w},\boldsymbol{c},u} \right)}{\left(\sum_u \boldsymbol{X}_{\boldsymbol{w},u}\right) \left(\sum_u \boldsymbol{X}_{\boldsymbol{c},u}\right)}$$

Unifying DeepWalk, LINE, PTE, and node2vec into Matrix Forms

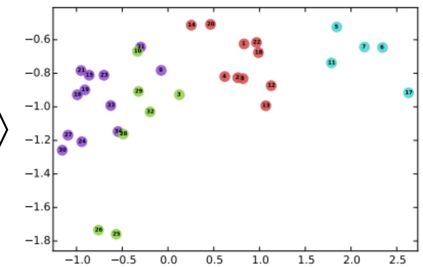
Algorithm	Closed Matrix Form
DeepWalk	$\log\left(\mathrm{vol}(G)\left(\frac{1}{T}\sum_{r=1}^{T}(\boldsymbol{D}^{-1}\boldsymbol{A})^{r}\right)\boldsymbol{D}^{-1}\right)-\log b$
LINE	$\log\left(\mathrm{vol}(G)\boldsymbol{D}^{-1}\boldsymbol{A}\boldsymbol{D}^{-1}\right)-\log b$
PTE	$\log\left(\begin{bmatrix}\alpha\,\mathrm{vol}(G_{\mathrm{ww}})(\boldsymbol{D}_{\math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

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

NetMF: explicitly factorizing the DW matrix



Matrix Factorization



A unified algorithm NetMF to explicitly factorizes the derived matrix

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

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

Explicitly factorize the matrix

Algorithm 3: NetMF for a Small Window Size T

1 Compute $\boldsymbol{P}^1, \cdots, \boldsymbol{P}^T$;
2 Compute $\boldsymbol{M} = \frac{\mathrm{vol}(G)}{bT} \left(\sum_{r=1}^{T} \boldsymbol{P}^r \right) \boldsymbol{D}^{-1}$;
3 Compute $\boldsymbol{M}' = \max(\boldsymbol{M}, 1)$;
4 Rank-d approximation by SVD: $\log \boldsymbol{M}' = \boldsymbol{U}_d \boldsymbol{\Sigma}_d \boldsymbol{V}_d^\top$;
5 **return** $\boldsymbol{U}_d \sqrt{\boldsymbol{\Sigma}_d}$ as network embedding.

Algorithm 4: NetMF for a Large Window Size T

1 Eigen-decomposition $\boldsymbol{D}^{-1/2} \boldsymbol{A} \boldsymbol{D}^{-1/2} \approx \boldsymbol{U}_h \boldsymbol{\Lambda}_h \boldsymbol{U}_h^\top$;
2 Approximate $\boldsymbol{M}$ with
$\hat{\boldsymbol{M}} = \frac{\mathrm{vol}(G)}{b} \boldsymbol{D}^{-1/2} \boldsymbol{U}_h \left(\frac{1}{T} \sum_{r=1}^{T} \boldsymbol{\Lambda}_h^r \right) \boldsymbol{U}_h^\top \boldsymbol{D}^{-1/2}$;
3 Compute $\hat{\boldsymbol{M}}' = \max(\hat{\boldsymbol{M}}, 1)$;
4 Rank-d approximation by SVD: $\log \hat{\boldsymbol{M}}' = \boldsymbol{U}_d \boldsymbol{\Sigma}_d \boldsymbol{V}_d^\top$;
5 **return** $\boldsymbol{U}_d \sqrt{\boldsymbol{\Sigma}_d}$ as network embedding.

- approximate $\boldsymbol{D}^{-1/2}\boldsymbol{A}\boldsymbol{D}^{-1/2}$ with its top-h eigenpairs $\boldsymbol{U}_h \boldsymbol{\Lambda}_h \boldsymbol{U}_h^\top$
- decompose using Arnoldi algorithm[1]

1. R. Lehoucq, D. Sorensen, and C.Yang. 1998. ARPACK users' guide: solution of large-scale eigenvalue problems with implicitly restarted Arnoldi methods. SIAM.

Experimental Setup

Label Classification:

- BlogCatelog, PPI, Wikipedia, Flickr
- Logistic Regression
- NetMF $(T = 1)$ v.s. LINE
- NetMF $(T = 10)$ v.s. DeepWalk

Table 1: Statistics of Datasets.

Dataset	BlogCatalog	PPI	Wikipedia	Flickr
$\|V\|$	10,312	3,890	4,777	80,513
$\|E\|$	333,983	76,584	184,812	5,899,882
#Labels	39	50	40	195

** Code available at https://github.com/xptree/NetMF

Results

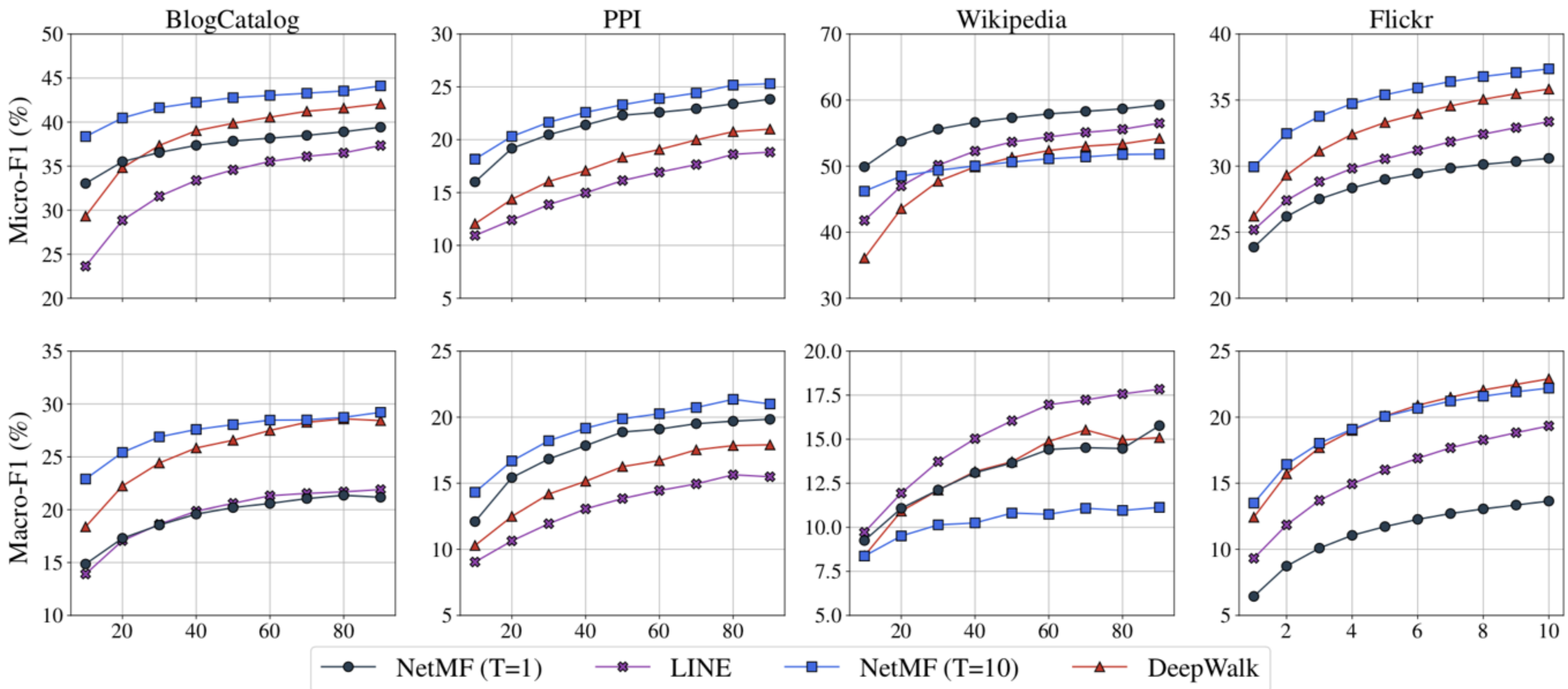
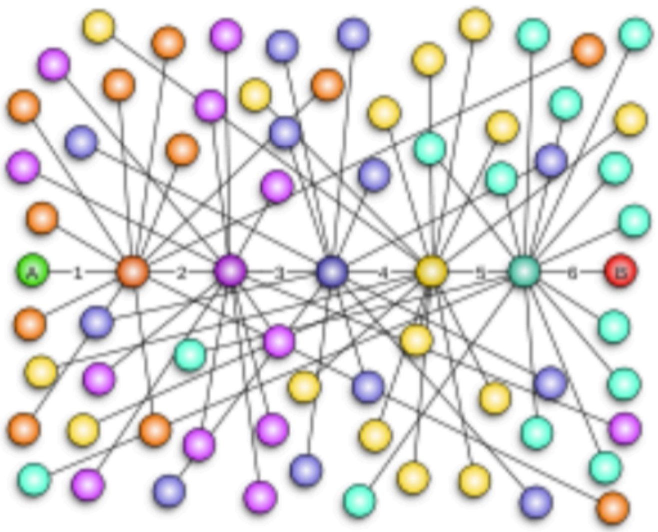
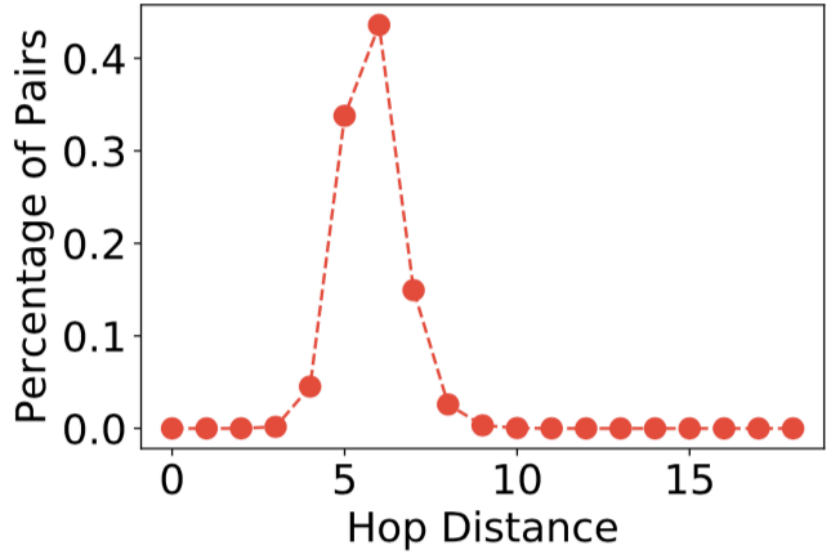


Figure 5: Predictive performance on varying the ratio of training data. The x-axis represents the ratio of labeled data (%), and the y-axis in the top and bottom rows denote the Micro-F1 and Macro-F1 scores respectively.

Challenge in NetMF



Small world



Academic graph

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

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 $O(T^2 m \epsilon^{-2} \log^2 n)$

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

1. D. Cheng, Y. Cheng, Y. Liu, R. Peng, and S.H. Teng, Efficient Sampling for Gaussian Graphical Models via Spectral Sparsification, COLT 2015.
2. D. Cheng, Y. Cheng, Y. Liu, R. Peng, and S.H. 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 $O(T^2 m \epsilon^{-2} \log^2 n)$

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

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

NetSMF --- Sparse

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

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

- Factorize the constructed matrix

1. J. Qiu, Y. Dong, H. Ma, J. Li, C. Wang, K. Wang, and J. Tang. NetSMF: Large-Scale Network Embedding as Sparse Matrix Factorization. WWW'19.

NetSMF — Approximation Error

Denote $\boldsymbol{M} = \boldsymbol{D}^{-1}\left(\boldsymbol{D} - \boldsymbol{L}\right)\boldsymbol{D}^{-1}$ in

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

and $\widetilde{\boldsymbol{M}}$ to be its sparsifier the we constructed.

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

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

Datasets

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

** Code available at https://github.com/xptree/NetSMF

Results

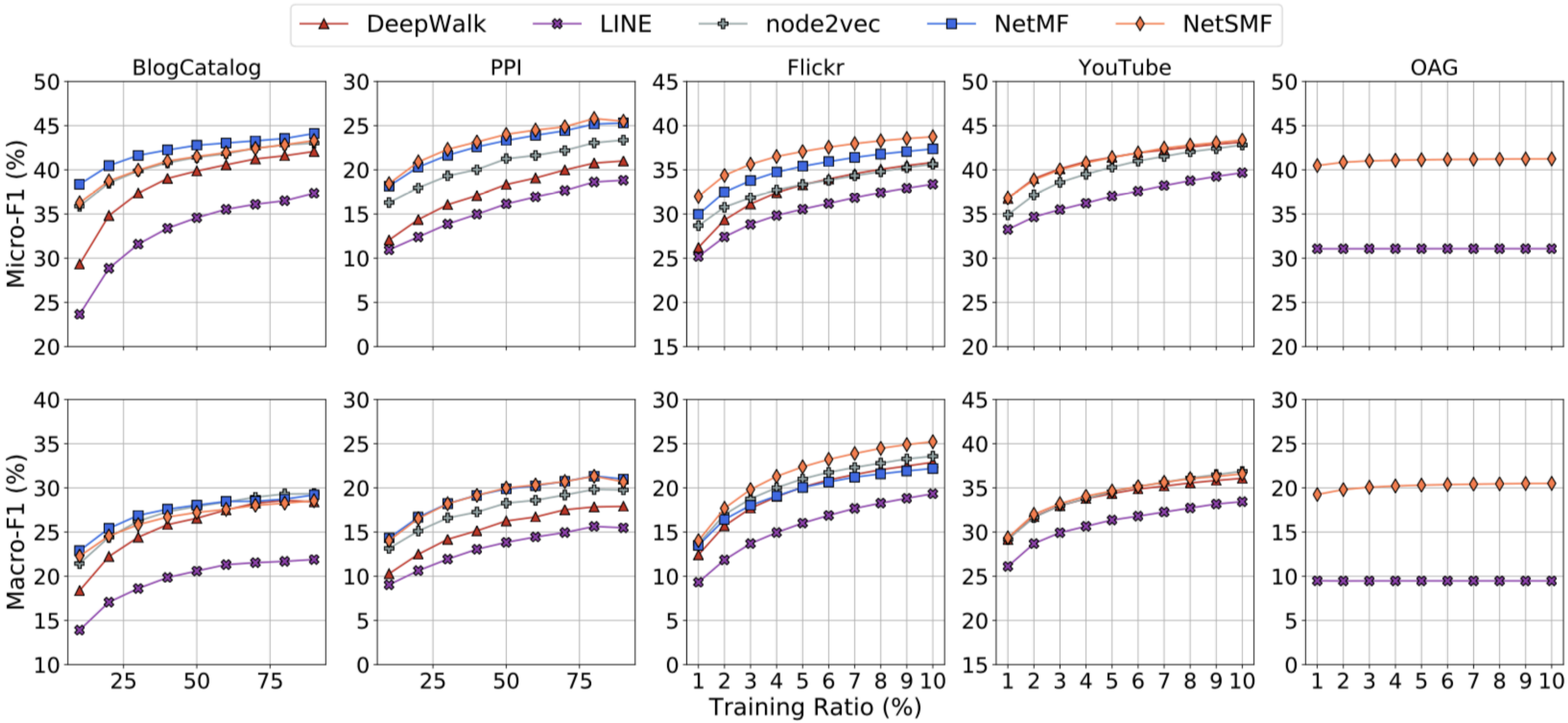
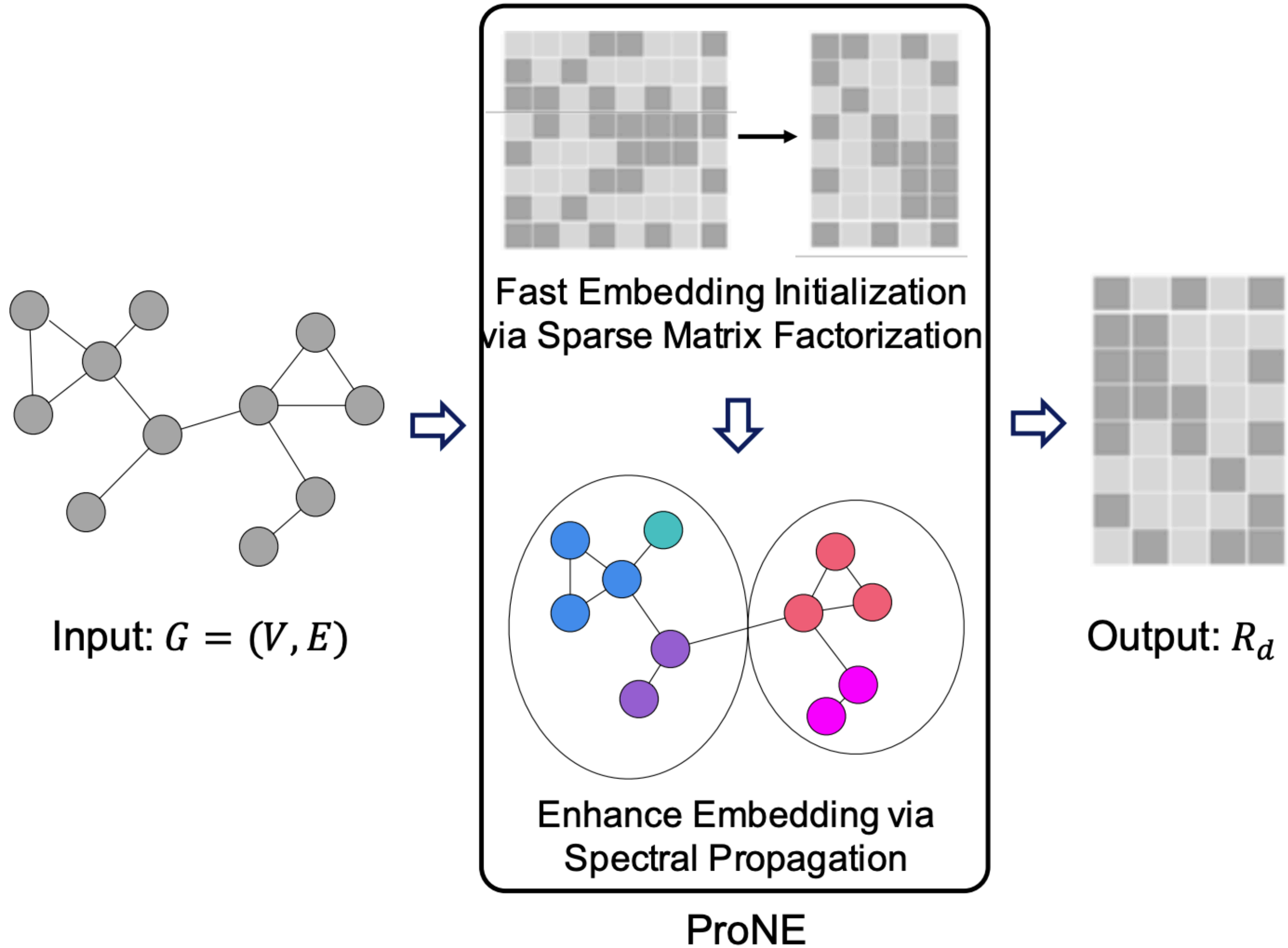


Figure 7: Predictive performance on varying the ratio of training data. The x-axis represents the ratio of labeled data (%), and the y-axis in the top and bottom rows denote the Micro-F1 and Macro-F1 scores

** Code available at https://github.com/xptree/NetSMF

ProNE: Fast and Scalable Network Embedding



1. J. Zhang, Y. Dong, Y. Wang, J. Tang, and M. Ding. ProNE: Fast and Scalable Network Representation Learning. IJCAI'19.

NE as Sparse Matrix Factorization

- node-context set $\mathcal{D} = E$ (sparsity)
- Probability of context v_j given node v_i $\hat{p}_{i,j} = \sigma(r_i^T c_j)$
- Objective: $l = -\sum_{(i,j)\in\mathcal{D}} p_{i,j} \ln \hat{p}_{i,j}$, $p_{ij} = A_{ij}/D_{ii}$
- Modify the loss (sum over the edge-->sparse)

$$l = -\sum_{(i,j)\in\mathcal{D}} [p_{i,j} \ln \sigma(r_i^T c_j) + \lambda P_{\mathcal{D},j} \ln \sigma(-r_i^T c_j)]$$

- Local negative samples drawn from $P_{\mathcal{D},j} \propto \sum_{i:(i,j)\in\mathcal{D}} p_{i,j}$

NE as Sparse Matrix Factorization

- Let the partial derivative w.r.t. $r_i^T c_j$ be zero

$$r_i^T c_j = \ln p_{i,j} - \ln(\lambda P_{D,j}), \quad (v_i, v_j) \in \mathcal{D}$$

- Matrix to be factorized (sparse)

$$M_{i,j} = \begin{cases} \ln p_{i,j} - \ln(\lambda P_{D,j}) & , (v_i, v_j) \in \mathcal{D} \\ 0 & , (v_i, v_j) \notin \mathcal{D} \end{cases}$$

NE as Sparse Matrix Factorization

- Compared with matrix factorization method (e.g., NetMF)

$$\log\left(\frac{\mathrm{vol}(G)}{b}\left(\frac{1}{T}\sum_{r=1}^{T}\left(\boldsymbol{D}^{-1}\boldsymbol{A}\right)^{r}\right)\boldsymbol{D}^{-1}\right) \quad \textbf{V.S.} \quad M_{i,j}=\begin{cases}\ln p_{i,j}-\ln(\lambda P_{D,j}) & ,(v_i,v_j)\in\mathcal{D}\\ 0 & ,(v_i,v_j)\notin\mathcal{D}\end{cases}$$

- Sparsity (local structure and local negative samples)→ much faster and scalable (e.g., randomized tSVD, O(|*E*|))
- The optimization (single thread) is much faster than SGD used in DeepWalk, LINE, etc. and is still scalable!!!
- Challenge: may lose high order information!
- Improvement via spectral propagation

Higher-order Cheeger's inequality

- $L = U \Lambda U^{-1}$, where $\Lambda = diag([\lambda_1, ..., \lambda_n])$ with $0 = \lambda_1 \leq \cdots \leq \lambda_n$
- Bridge graph spectrum and graph partitioning

$$\frac{\lambda_k}{2} \leq \rho_G(k) \leq O(k^2)\sqrt{\lambda_k}$$

- *k*-way Cheeger constant $\rho_G(k)$: reflects the effect of the graph partitioned into *k* parts. A smaller value of $\rho_G(k)$ means a better partitioning effect.

1. J. R. Lee, S. O. Gharan, L. Trevisan. Multiway spectral partitioning and higher-order cheeger inequalities. Journal of the ACM (JACM), 2014, 61(6): 37.

Spectral Propagation of ProNE

- Spectral propagation only involves sparse matrix multiplication! The complexity is linear!

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

- where $$\widetilde{L} = U diag([g(\lambda_1), ..., g(\lambda_n)])U^T$$

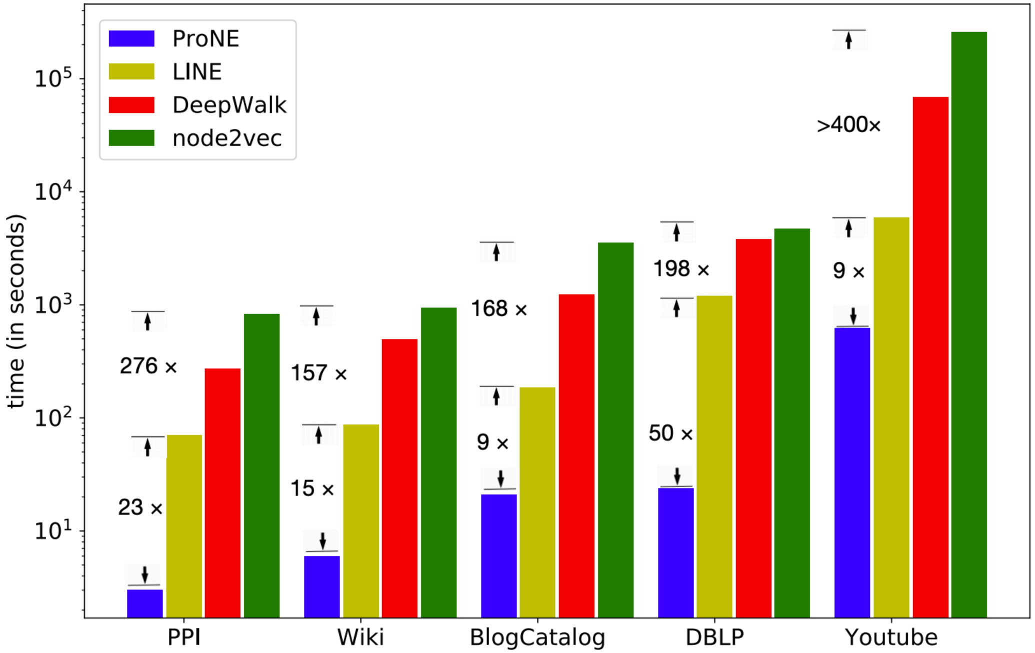
$$g(\lambda) = e^{-\frac{1}{2}[(\lambda-\mu)^2-1]\theta}$$

- To avoid explicit eigendecomposition, use Chebyshev expansion:

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

- sparse matrix factorization + spectral propagation = $O(|V|d^2 + k|E|)$

Results



* ProNE (1 thread) v.s. Others (20 threads)

* 10 minutes on Youtube (~1M nodes)

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**

** Code available at https://github.com/THUDM/ProNE

Effectiveness experiments

Dataset	training ratio	0.1	0.3	0.5	0.7	0.9
PPI	DeepWalk	16.4	19.4	21.1	22.3	22.7
	LINE	16.3	20.1	21.5	22.7	23.1
	node2vec	16.2	19.7	21.6	23.1	24.1
	GraRep	15.4	18.9	20.2	20.4	20.9
	HOPE	16.4	19.8	21.0	21.7	22.5
	ProNE (SMF)	15.8	20.6	22.7	23.7	24.2
	ProNE	**18.2**	**22.7**	**24.6**	**25.4**	**25.9**
	($\pm\sigma$)	($\pm$0.5)	($\pm$0.3)	($\pm$0.7)	($\pm$1.0)	($\pm$1.1)
Wiki	DeepWalk	40.4	45.9	48.5	49.1	49.4
	LINE	**47.8**	50.4	51.2	51.6	52.4
	node2vec	45.6	47.0	48.2	49.6	50.0
	GraRep	47.2	49.7	50.6	50.9	51.8
	HOPE	38.5	39.8	40.1	40.1	40.1
	ProNE (SMF)	47.6	51.6	53.2	53.5	53.9
	ProNE	47.3	**53.1**	**54.7**	**55.2**	**57.2**
	($\pm\sigma$)	($\pm$0.7)	($\pm$0.4)	($\pm$0.8)	($\pm$0.8)	($\pm$1.3)
BlogCatalog	DeepWalk	36.2	39.6	40.9	41.4	42.2
	LINE	28.2	30.6	33.2	35.5	36.8
	node2vec	**36.3**	39.7	41.1	42.0	42.1
	GraRep	34.0	32.5	33.3	33.7	34.1

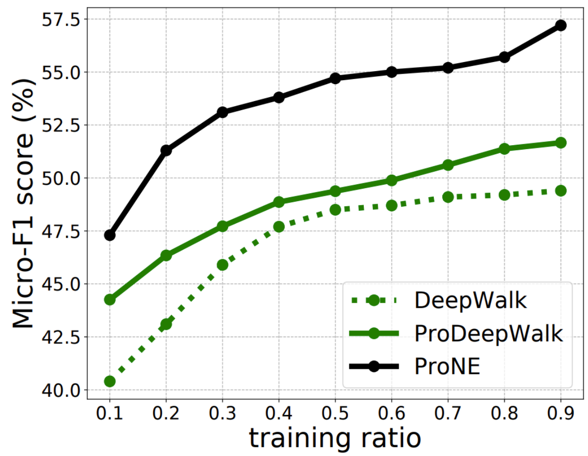
Dataset	training ratio	0.01	0.03	0.05	0.07	0.09
DBLP	DeepWalk	49.3	55.0	57.1	57.9	58.4
	LINE	48.7	52.6	53.5	54.1	54.5
	node2vec	48.9	55.1	57.0	58.0	58.4
	GraRep	50.5	52.6	53.2	53.5	53.8
	HOPE	**52.2**	55.0	55.9	56.3	56.6
	ProNE (SMF)	50.8	54.9	56.1	56.7	57.0
	ProNE	48.8	**56.2**	**58.0**	**58.8**	**59.2**
	($\pm\sigma$)	($\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 (SMF) = ProNE w/ only sparse matrix factorization

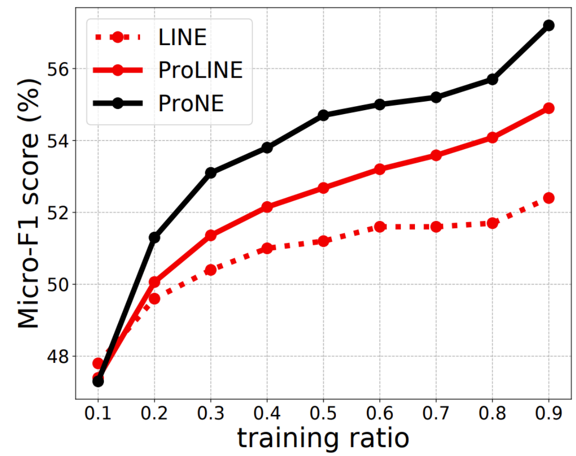
Embed 100,000,000 nodes by one thread:
29 hours with **performance superiority**

** Code available at https://github.com/THUDM/ProNE

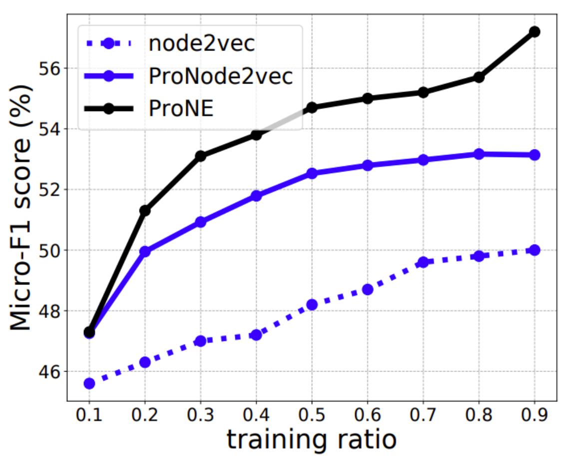
Spectral Propagation for Enhancement



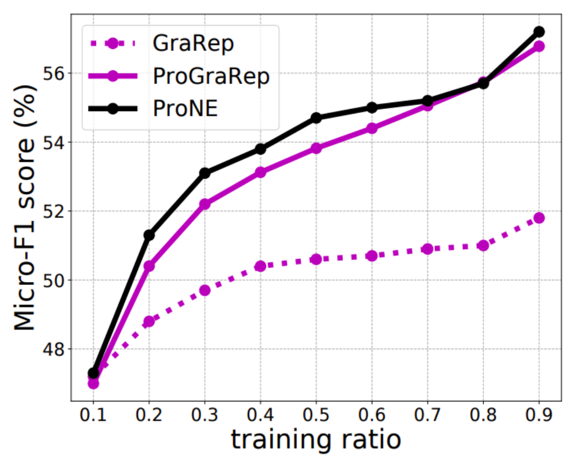
(a) ProDeepWalk



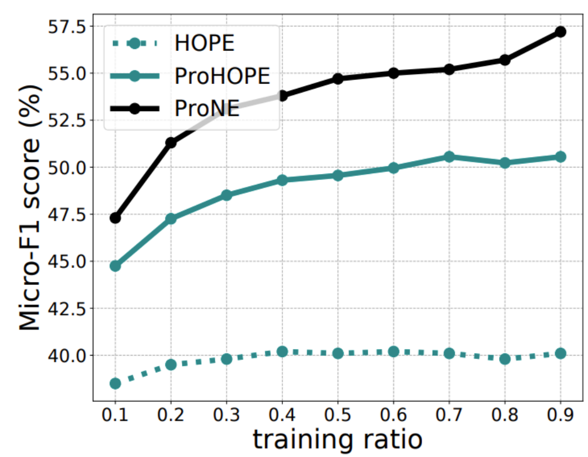
(b) ProLINE



(c) ProNode2vec



(d) ProGraRep



(e) ProHOPE

Net(S)MF vs. ProNE

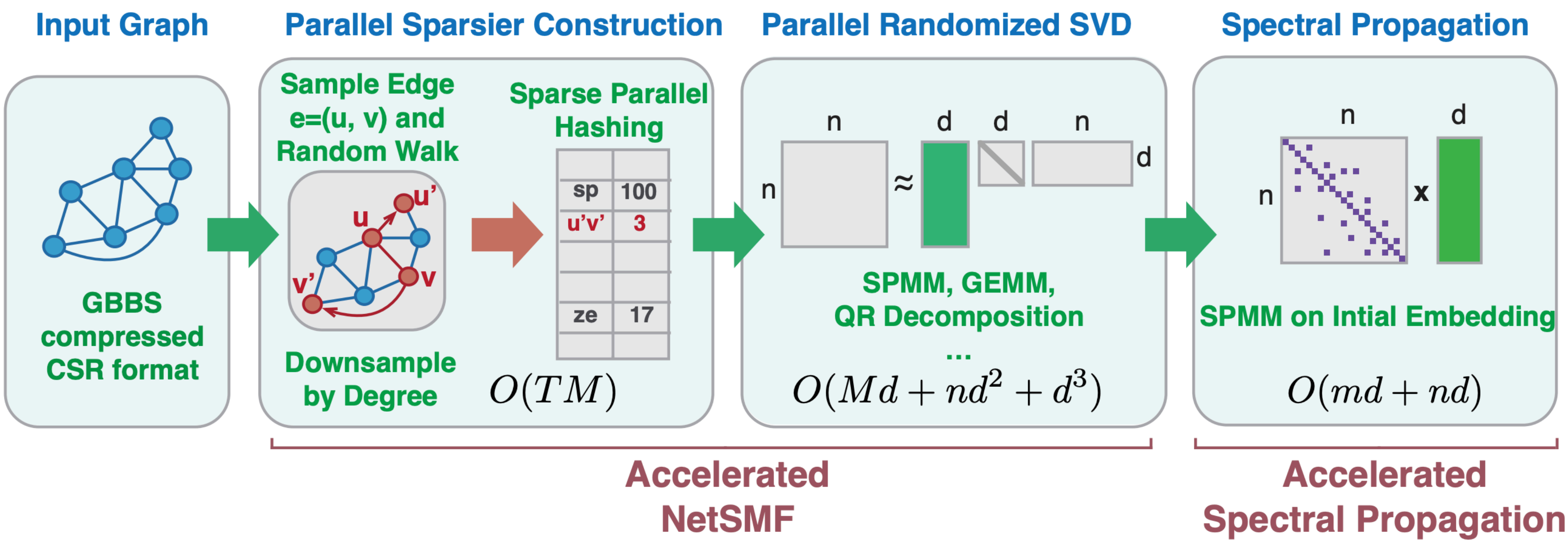
$$\log\left(\frac{\operatorname{vol}(G)}{b}\left(\frac{1}{T}\sum_{r=1}^{T}\left(\boldsymbol{D}^{-1}\boldsymbol{A}\right)^{r}\right)\boldsymbol{D}^{-1}\right)$$

V.S.

$$M_{i,j}=\begin{cases}\ln p_{i,j}-\ln(\lambda P_{D,j}) & ,(v_i,v_j)\in\mathcal{D}\\ 0 & ,(v_i,v_j)\notin\mathcal{D}\end{cases}$$

- NetMF is slow depending on the density of the matrix;
- NetSMF needs to approximate high-order random-walk matrix polynomials
- ProNE=sparse MF + spectral propagation is much faster
- Is that possible? Net(S)MF + ProNE?

LightNE (SIGMOD'21)



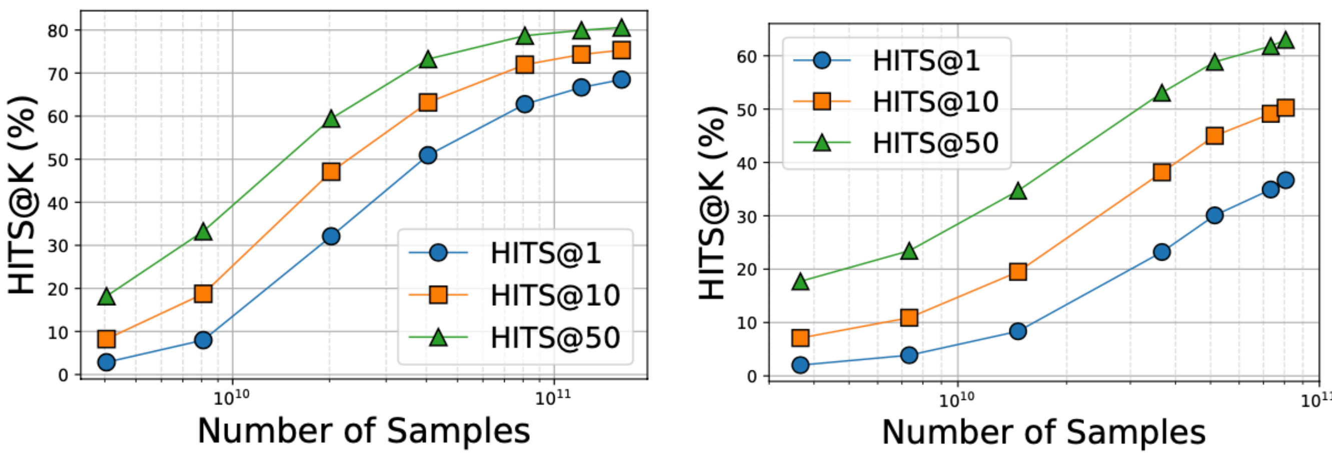
- Scalable: Embed graphs with 1B edges within 1.5 hours.
- Lightweight: Occupy hardware costs below 100 dollars measured by cloud rent to process 1B to 100B edges.
- Accurate: Achieve the highest accuracy in downstream tasks under the same time budget and similar resources.

1. J. Qiu, L. Dhulipala, J. Tang, R. Peng, and C. Wang. Lightne: A lightweight graph processing system for network embedding. SIGMOD'21.

LightNE on Very Large Graphs

- None of the existing network embedding systems can handle such large graphs in a single machine!

	ClueWeb	Hyperlink2014
$\lvert V \rvert$	978,408,098	1,724,573,718
$\lvert E \rvert$	74,744,358,622	124,141,874,032

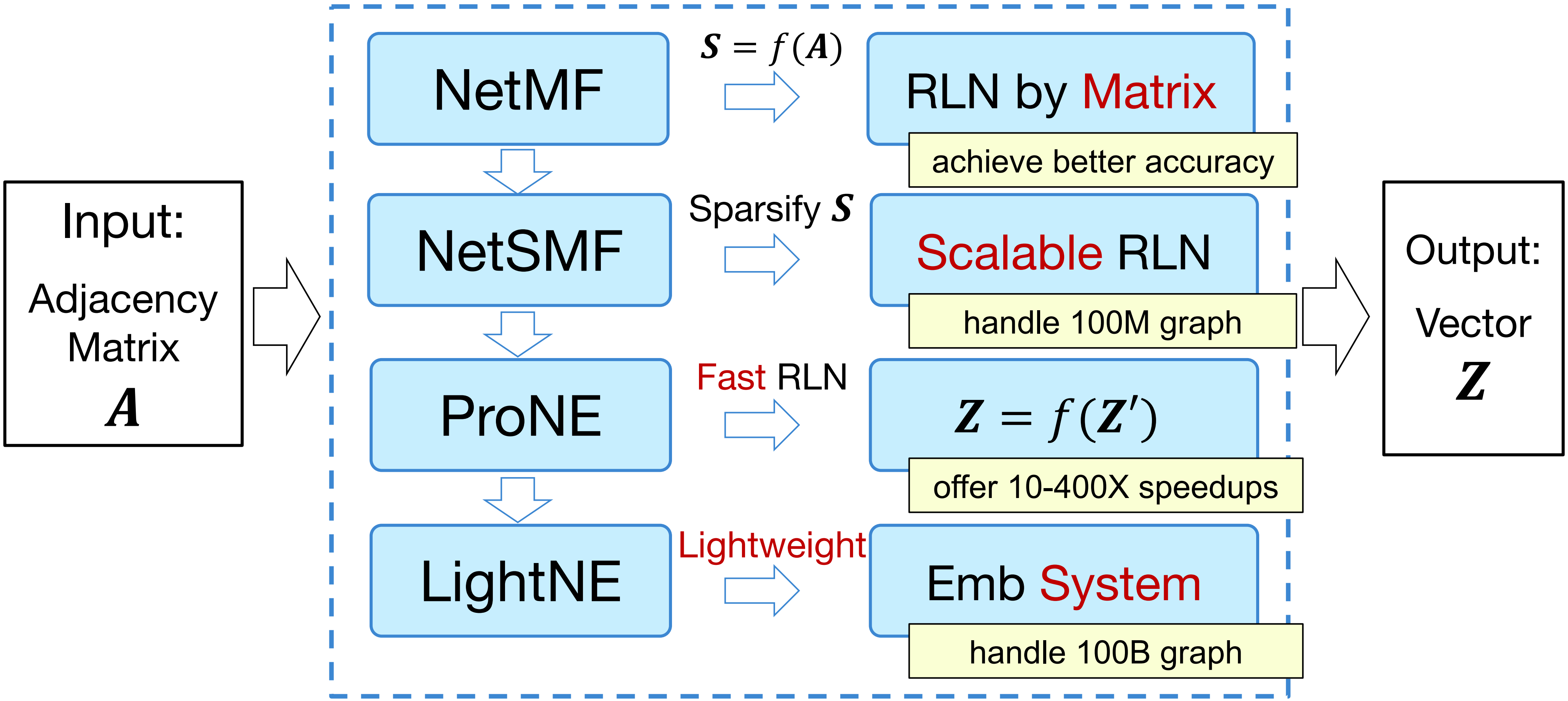


(a) ClueWeb-Sym **(b)** Hyperlink2014-Sym

Figure 3: HITS@K ($K = 1, 10, 50$) of LightNE w.r.t. the number of samples.

** Code available at https://github.com/xptree/LightNE

Representation Learning on Networks



1. Qiu et al. Network embedding as matrix factorization: unifying deepwalk, line, pte, and node2vec. *WSDM'18.* **The most cited paper in WSDM'18 as of May 2019**
2. J. Qiu, Y. Dong, H. Ma, J. Li, C. Wang, K. Wang, and J. Tang. NetSMF: Large-Scale Network Embedding as Sparse Matrix Factorization. WWW'19.
3. J. Zhang, Y. Dong, Y. Wang, J. Tang, and M. Ding. ProNE: Fast and Scalable Network Representation Learning. IJCAI'19.
4. J. Qiu, L. Dhulipala, J. Tang, R. Peng, and C. Wang. Lightne: A lightweight graph processing system for network embedding. SIGMOD'21.

Homework 2

- Experiments on different network embedding methods
  - Due by 24th July
  - Compare the performance of four methods (including DeepWalk, NetMF, NetSMF, ProNE)
  - Directly import the models from CogDL (but read the implementations in CogDL if possible)
  - Carefully select the hyper-parameter setting for methods
  - Visualize the experimental results
  - Give the analysis of the results

- Find the homework material from the course website: https://cogdl.ai/gnn2022/

Thank you！

Collaborators:

Zhenyu Hou, Yuxiao Dong, Jie Tang et al. (**THU**)

Qingfei Zhao, Xinije Zhang, Peng Zhang (**Zhipu AI**)

Hongxiao Yang, Chang Zhou, et al. (**Alibaba**)

Yang Yang (**ZJU**)

Yukuo Cen, KEG, Tsinghua U.	https://github.com/THUDM/cogdl
Online Discussion Forum	https://discuss.cogdl.ai/