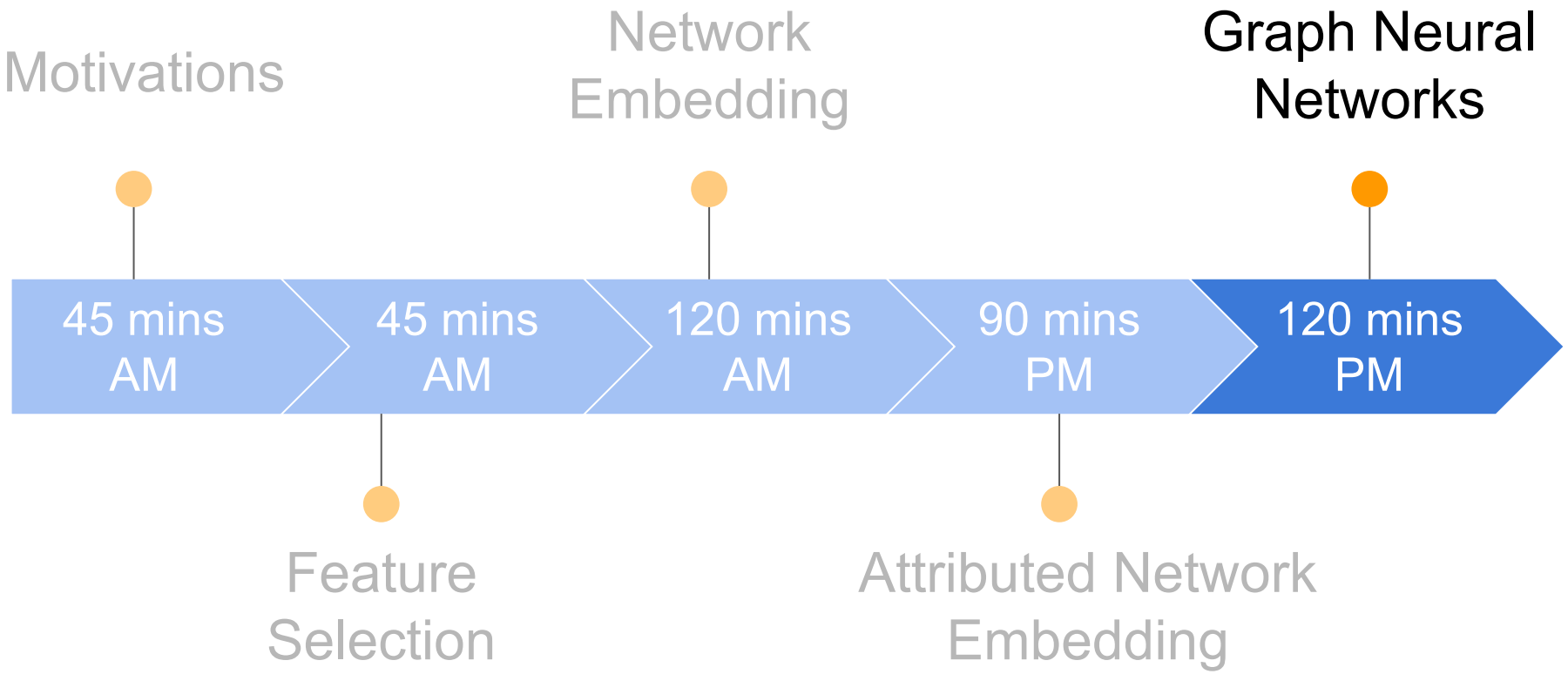
Learning From Networks

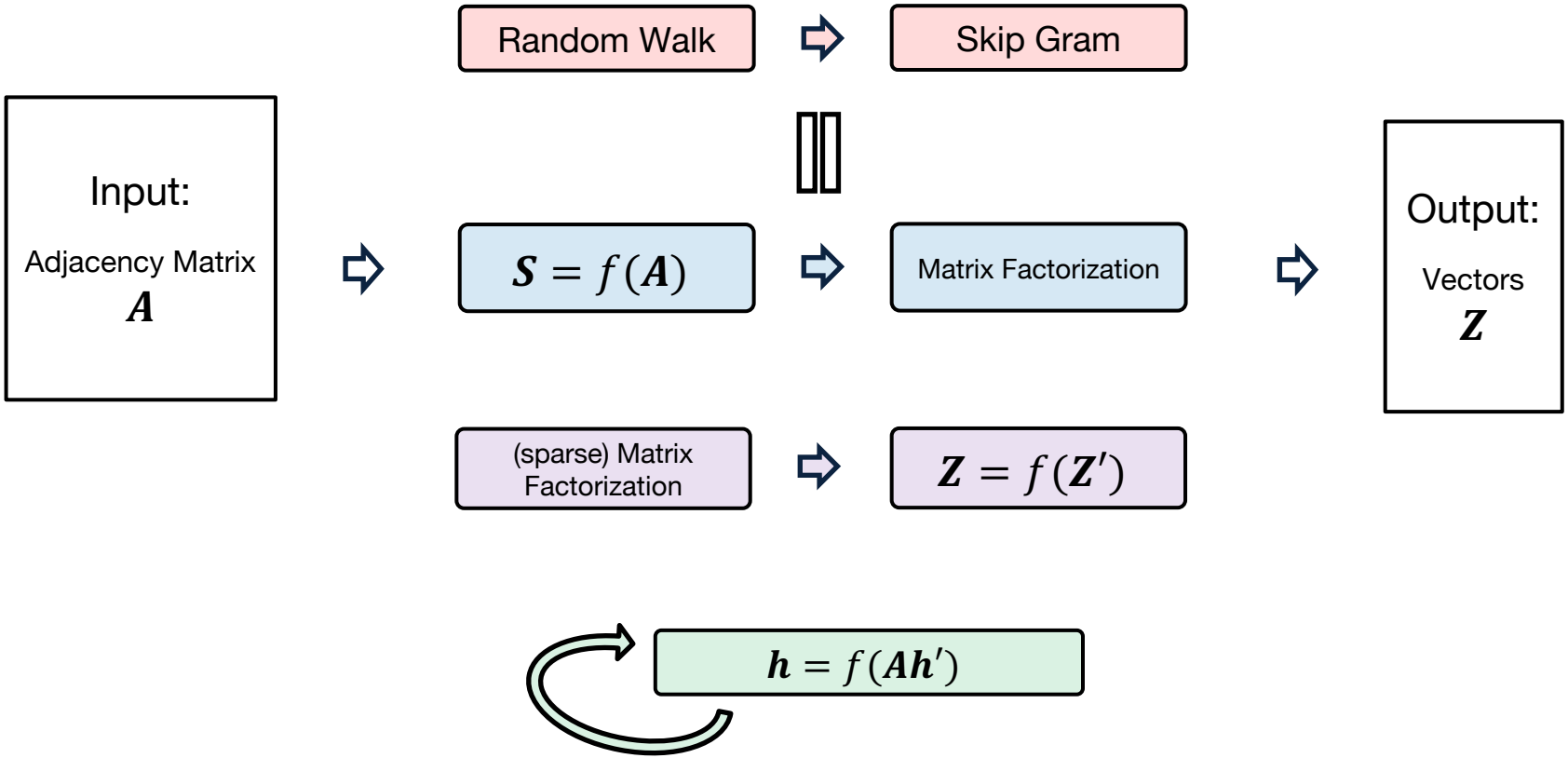
——*Algorithms, Theory, & Applications*

Xiao Huang, Peng Cui, Yuxiao Dong, Jundong Li, Huan Liu, Jian Pei, Le Song,
Jie Tang, Fei Wang, Hongxia Yang, Wenwu Zhu

xhuang@tamu.edu; cuip@tsinghua.edu.cn; yuxdong@microsoft.com; jundongl@asu.edu;
huan.liu@asu.edu; jpei@cs.sfu.ca; le.song@antfin.com; jietang@tsinghua.edu.cn;
few2001@med.cornell.edu; yang.yhx@alibaba-inc.com; wwzhu@tsinghua.edu.cn;


Motivations
Network Embedding
Graph Neural Networks
45 mins AM
45 mins AM
120 mins AM
90 mins PM
120 mins PM
Feature Selection
Attributed Network Embedding

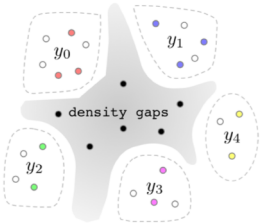
Connecting NE with graph neural networks



Graph neural networks

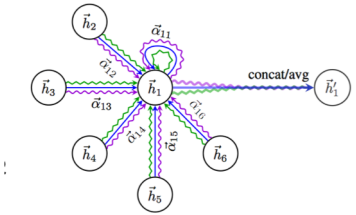
FastGCNs, Graph attention — 2018: Velickovic et al., ICLR'18, Chen et al., ICLR 2018

Neural message passing, GraphSage — 2017: Gilmer et al., ICML'17; Hamilton et al., NIPS'17



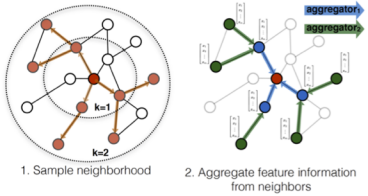
Gated graph neural network — 2016: Li et al., ICLR'16

structure2vec — 2016: Dai et al., ICML'16

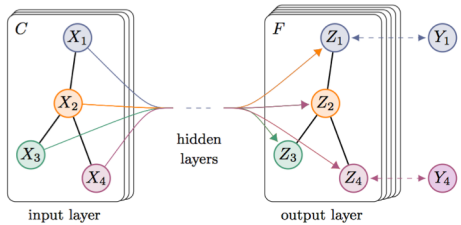


Graph convolutional network — 2015: Duvenaud et al., NIPS'15; Kipf & Welling ICLR'17

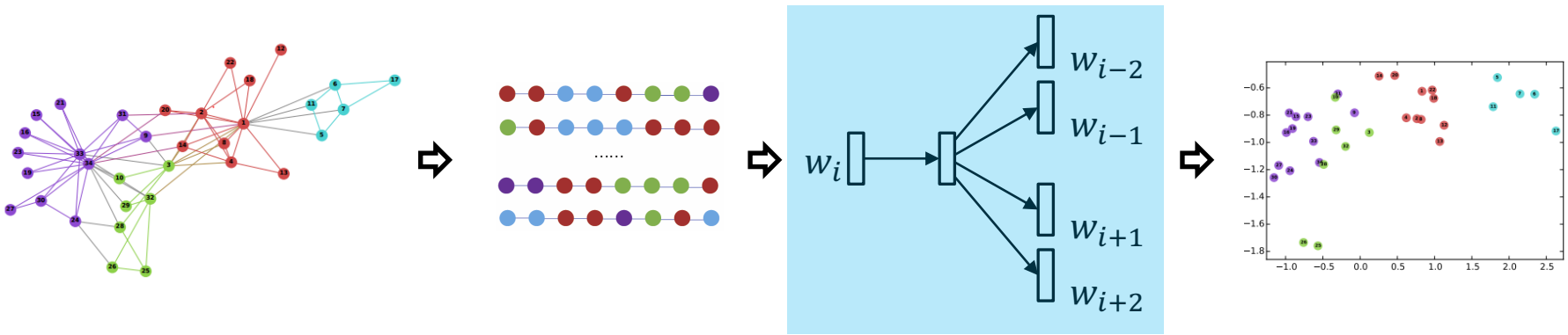
Spectral graph convolution — 2014: Bruna et al., ICLR'14



Graph neural network — 2005: Gori et al., IJCNN'05



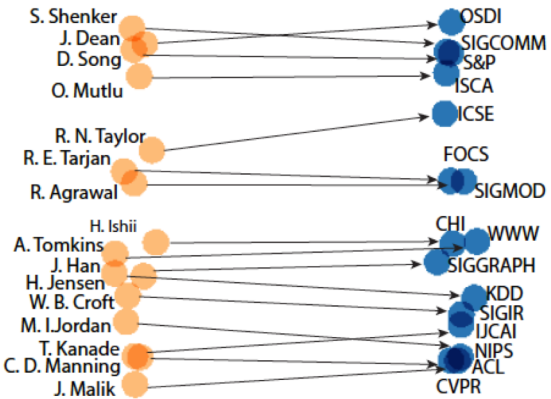
Network embedding: DeepWalk



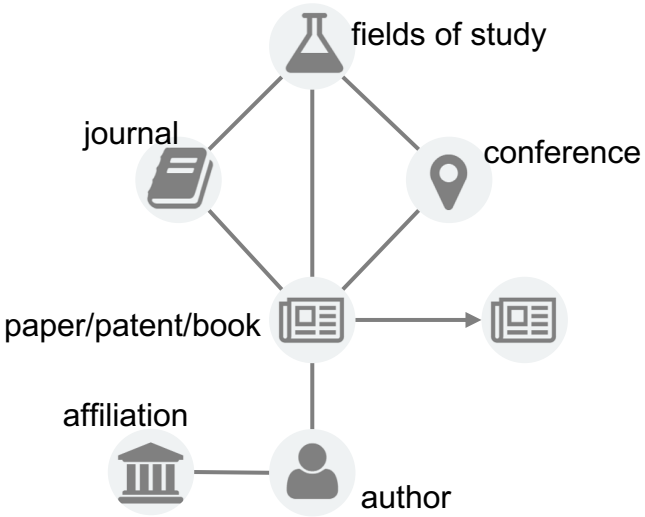
Perozzi et al. DeepWalk: Online learning of social representations. In *KDD' 14*, pp. 701–710. **Most Cited Paper in KDD'14.**

Random walk strategies

- Random Walk
  - DeepWalk (walk length > 1)
  - LINE (walk length = 1)
- Biased Random Walk
  - node2vec (2-order random walk)
  - metapath2vec (heterogeneous random walk)

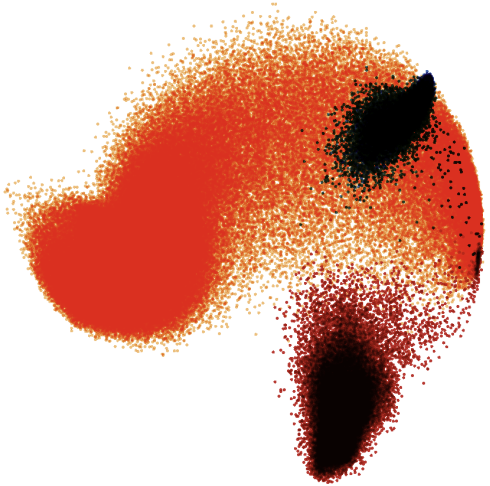


Application: Embedding Heterogeneous Academic Graph



Microsoft Academic Graph
&
AMiner

metapath2vec



Application 1: Related Venues

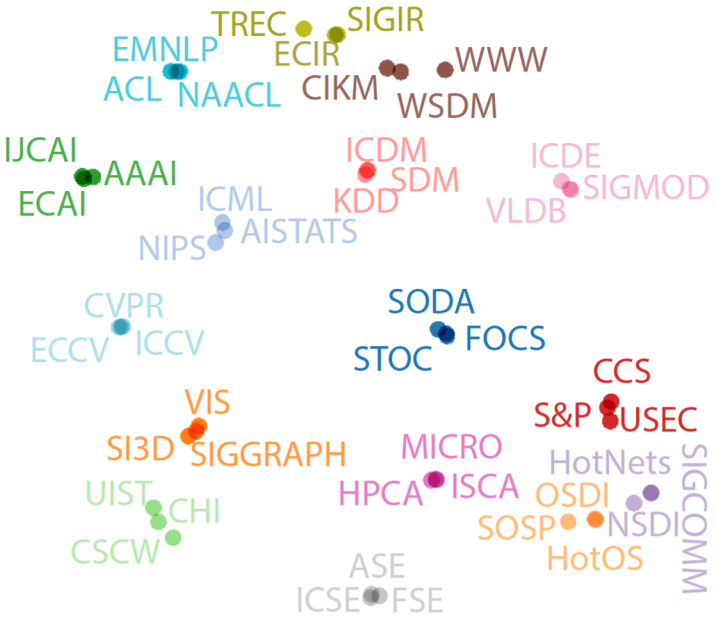
Science

Science, also widely referred to as Science Magazine, is the peer-reviewed academic journal of the American Association for the Advancement of Science (AAAS) and one of the world's top academic journals. It was first published in 1880, is currently circulated weekly and has a subscriber base of around 130,000. Because institutional subscriptions and online access serve a larger audience, its estimated readership is 570,400 people.

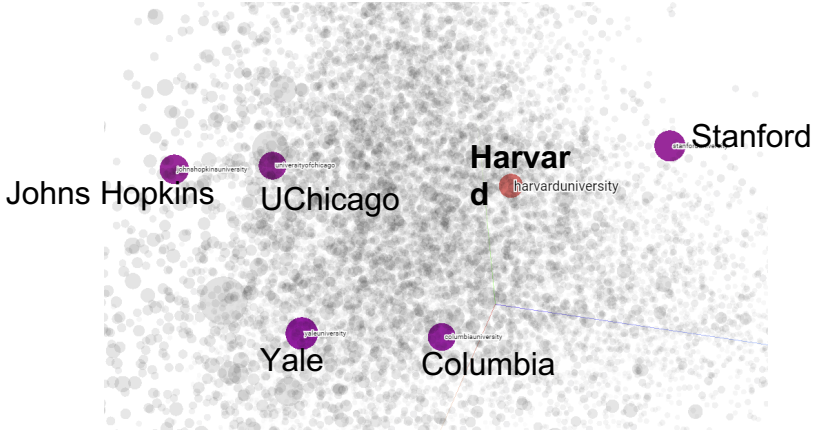
Website links: sciencemag.org, en.wikipedia.org

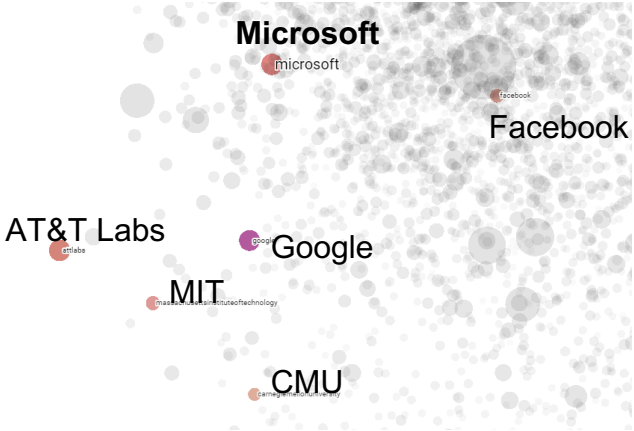
RELATED JOURNALS

Nature | Proceedings of the National Academy of Sciences of the United States of America | Nature Communications



Application 2: Similarity Search (Institution)





What are the fundamentals

underlying random-walk + skip-gram based network embedding models?

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

- DeepWalk $\log\left(\frac{\mathrm{vol}(G)}{b}\left(\frac{1}{T}\sum_{r=1}^{T}\left(\boldsymbol{D}^{-1}\boldsymbol{A}\right)^{r}\right)\boldsymbol{D}^{-1}\right)$ — The most cited paper in KDD14
- LINE $\log\left(\frac{\mathrm{vol}(G)}{b}\boldsymbol{D}^{-1}\boldsymbol{A}\boldsymbol{D}^{-1}\right)$ — The most cited paper in WWW15
- PTE $\log\left(\begin{bmatrix}\alpha\,\mathrm{vol}(G_{\mathsf{ww}})(\boldsymbol{D}^{\mathsf{ww}}_{\mathsf{row}})^{-1}\boldsymbol{A}_{\mathsf{ww}}(\boldsymbol{D}^{\mathsf{ww}}_{\mathsf{col}})^{-1}\\ \beta\,\mathrm{vol}(G_{\mathsf{dw}})(\boldsymbol{D}^{\mathsf{dw}}_{\mathsf{row}})^{-1}\boldsymbol{A}_{\mathsf{dw}}(\boldsymbol{D}^{\mathsf{dw}}_{\mathsf{col}})^{-1}\\ \gamma\,\mathrm{vol}(G_{\mathsf{lw}})(\boldsymbol{D}^{\mathsf{lw}}_{\mathsf{row}})^{-1}\boldsymbol{A}_{\mathsf{lw}}(\boldsymbol{D}^{\mathsf{lw}}_{\mathsf{col}})^{-1}\end{bmatrix}\right)-\log b$ — The 5th most cited paper in KDD15
- node2vec $\log\left(\frac{\frac{1}{2T}\sum_{r=1}^{T}\left(\sum_u \boldsymbol{X}_{w,u}\underline{\boldsymbol{P}}^{r}_{c,w,u}+\sum_u \boldsymbol{X}_{c,u}\underline{\boldsymbol{P}}^{r}_{w,c,u}\right)}{b\left(\sum_u \boldsymbol{X}_{w,u}\right)\left(\sum_u \boldsymbol{X}_{c,u}\right)}\right)$ — The 2nd most cited paper in KDD16

$\boldsymbol{A}$ Adjacency matrix

$\boldsymbol{D}$ Degree matrix

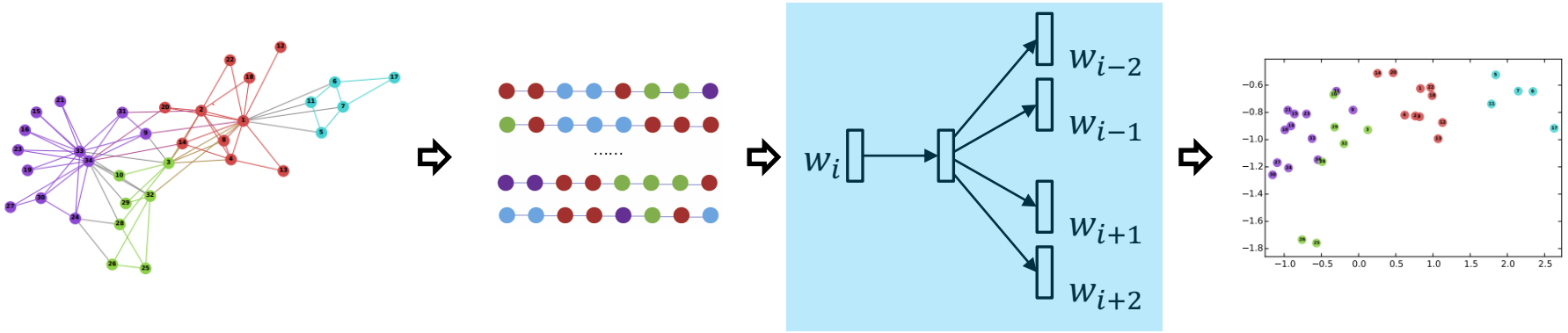
$vol(G)=\sum_i\sum_j A_{ij}$

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*. **The most cited paper in WSDM'18 as of May 2019**

Understanding random walk + skip gram



$G = (V, E)$

- Adjacency matrix $\boldsymbol{A}$
- Degree matrix $\boldsymbol{D}$
- Volume of G: $vol(G)$

?

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

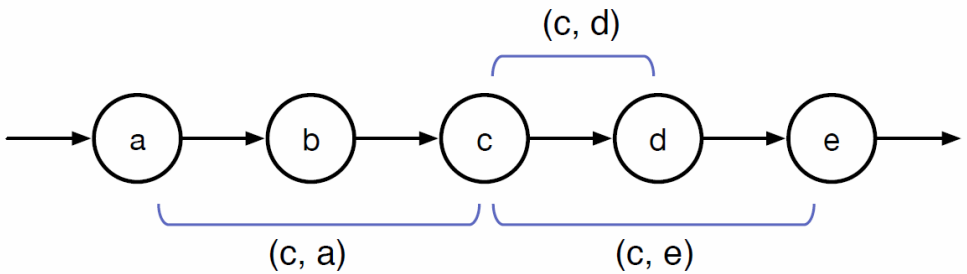
- #(w,c): co-occurrence of w & c
- #(w): occurrence of word w
- #(c): occurrence of context c
- $|\mathcal{D}|$: number of word-context pairs

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

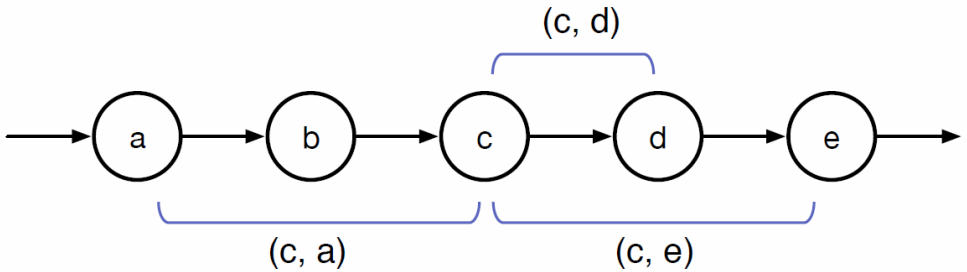
Understanding random walk + skip gram

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

Understanding random walk + skip gram

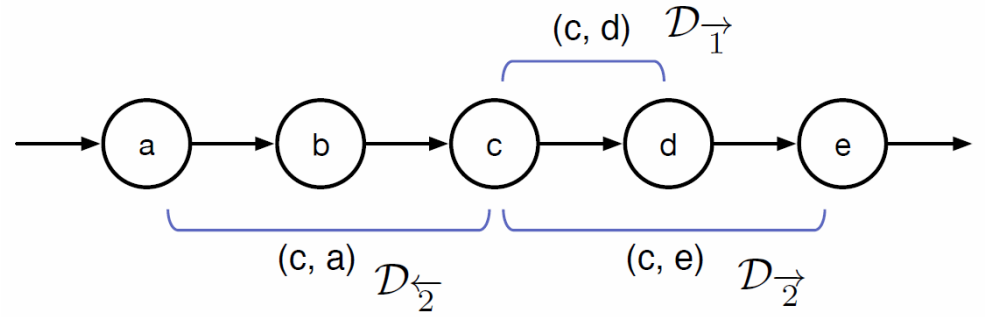


Understanding random walk + skip gram



Suppose the multiset $\mathcal{D}$ is constructed based on random walk on graphs, can we interpret $log\frac{\#(w,c)|\mathcal{D}|}{b\#(w)\#(c)}$ with graph structures?

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

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

Distinguish direction and distance

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

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

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

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 $L\to\infty$

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

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

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

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

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 $L\to\infty$

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

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

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

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

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

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 $L\to\infty$

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

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

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

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

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

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

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

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 $L \to \infty$

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

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

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

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

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

Understanding random walk + skip gram

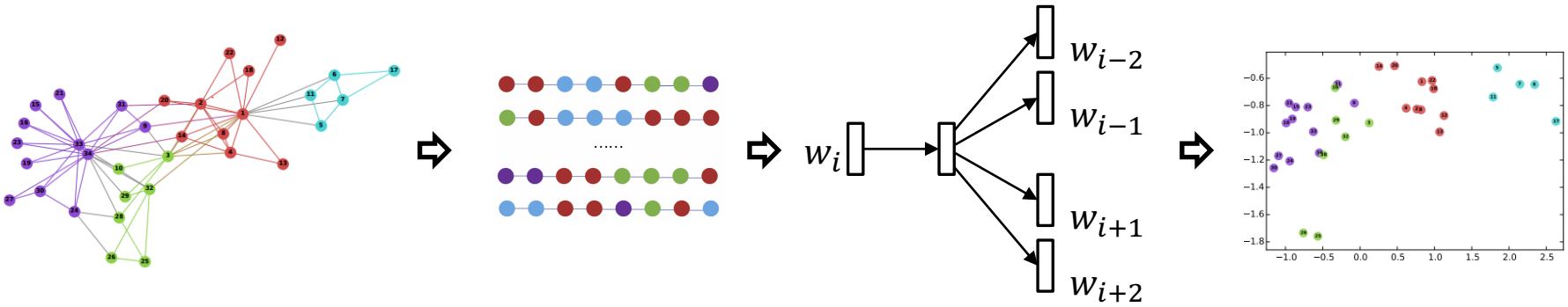
$$\frac{\#(w,c)\,|\mathcal{D}|}{\#(w)\cdot\#(c)} \xrightarrow{p} \frac{\mathrm{vol}(G)}{2T}\left(\frac{1}{d_c}\sum_{r=1}^{T}\left(\boldsymbol{P}^r\right)_{w,c}+\frac{1}{d_w}\sum_{r=1}^{T}\left(\boldsymbol{P}^r\right)_{c,w}\right)$$

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

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

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

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.* **The most cited paper in WSDM'18 as of May 2019**

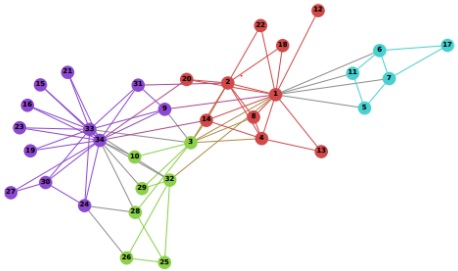
Unifying DeepWalk, LINE, PTE, & node2vec as Matrix Factorization

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

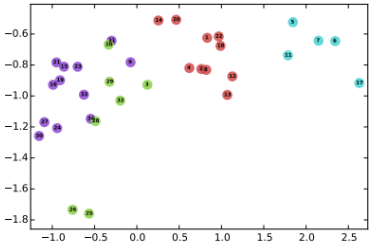
Qiu et al. Network embedding as matrix factorization: unifying deepwalk, line, pte, and node2vec. In *WSDM'18.* **The most cited paper in WSDM'18 as of May 2019**

Can we directly factorize the derived matrices for learning embeddings?

NetMF: explicitly factorizing the DW 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.*

NetMF

How can we solve this issue?

1. Construction
2. 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**

NetMF

- DeepWalk is asymptotically and implicitly factorizing

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

Recall that in random walk + skip gram based network embedding models:
$\boldsymbol{z}_v^{\top}\boldsymbol{z}_c$ → the probability that node v and context c appear on a random walk path

- NetMF is explicitly factorizing

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

$\boldsymbol{z}_v^{\top}\boldsymbol{z}_c$ → the similarity score $\boldsymbol{S}_{vc}$ between node v and context c defined by this matrix

Experimental Setup

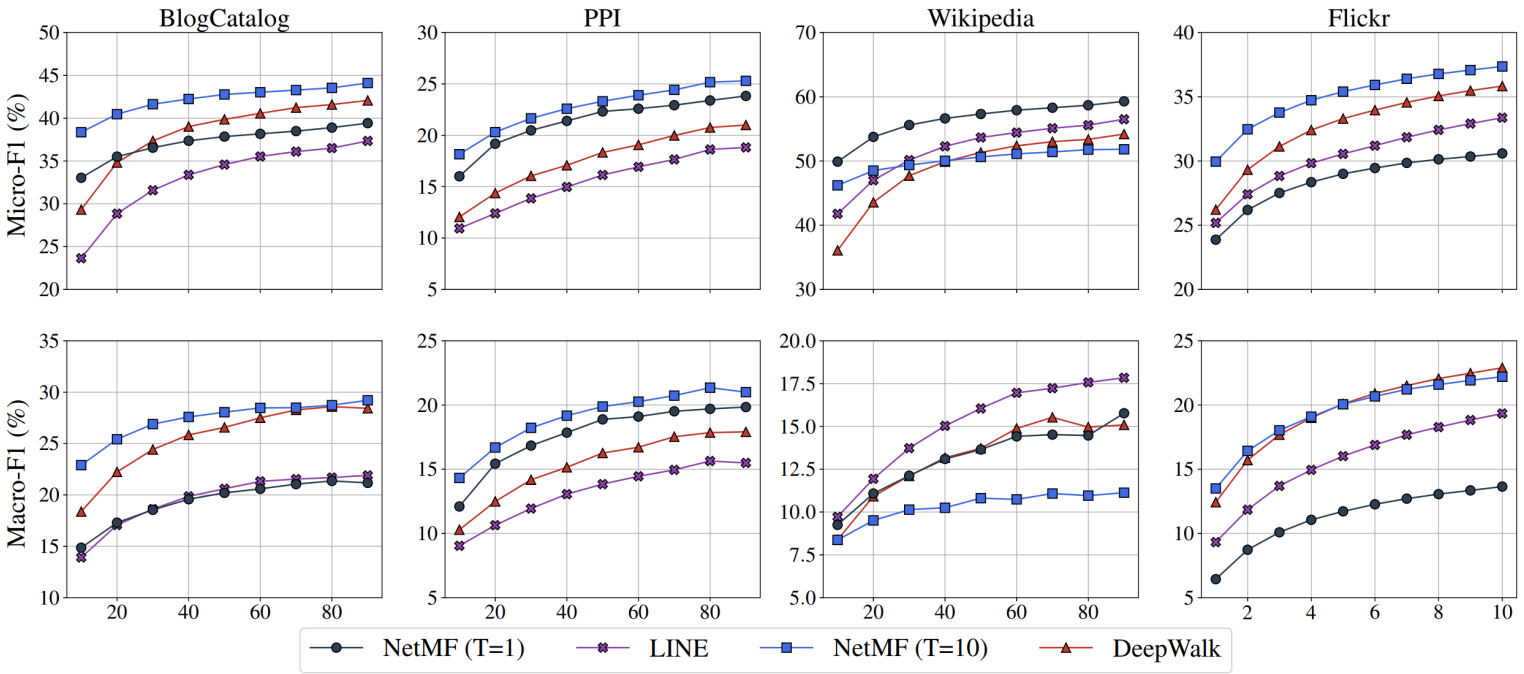
Label Classification:

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

Table 1: Statistics of Datasets.

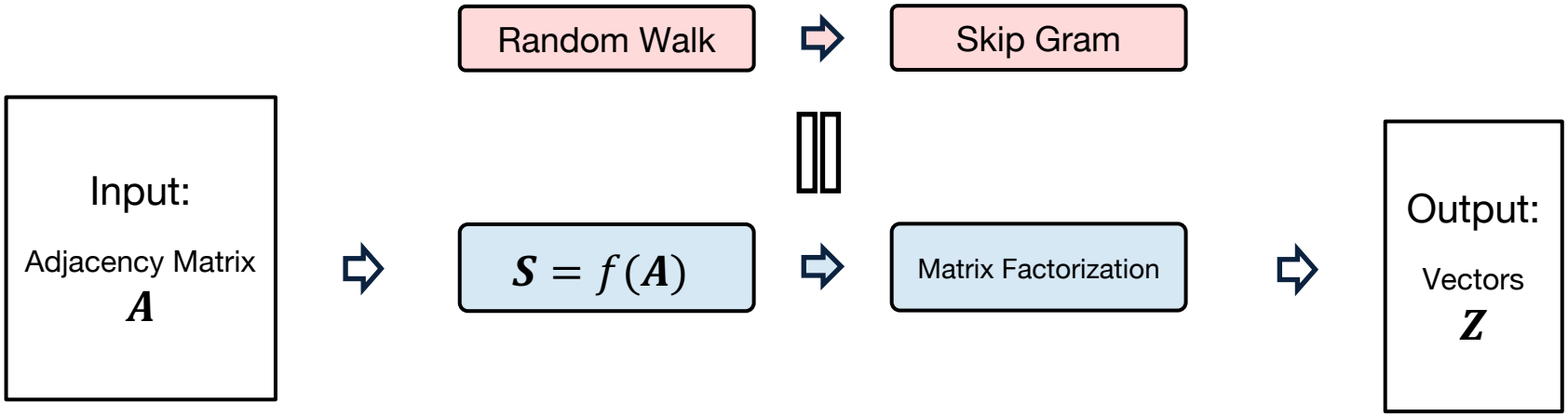
Dataset	BlogCatalog	PPI	Wikipedia	Flickr
$\|V\|$	10,312	3,890	4,777	80,513
$\|E\|$	333,983	76,584	184,812	5,899,882
#Labels	39	50	40	195

Experimental Results



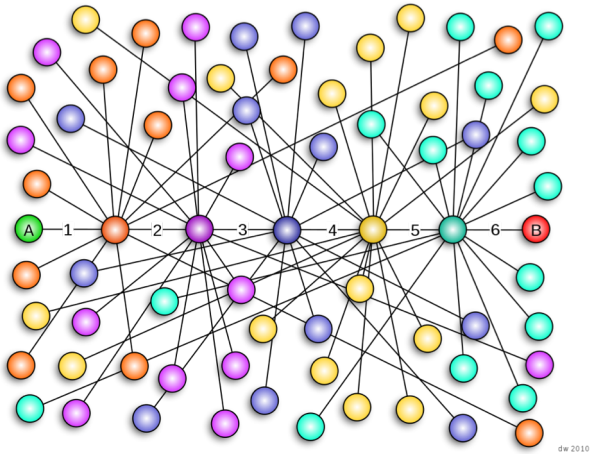
Predictive performance on varying the ratio of training data;
The *x*-axis represents the ratio of labeled data (%)

Connecting NE with Graph Neural Networks



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

Challenges



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

dense

NetMF is not practical for very large networks

NetMF

How can we solve this issue?

1. Construction
2. 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**

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**

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

- Dehua Cheng, Yu Cheng, Yan Liu, Richard Peng, and Shang-Hua Teng, Efficient Sampling for Gaussian Graphical Models via Spectral Sparsification, COLT 2015.
- 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 $O(T^2 m \epsilon^{-2} \log^2 n)$

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

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

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

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

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 $O(n \log^{c} n \epsilon^{-2})$ non-zeros

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

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

$\alpha_1 = \cdots = \alpha_T = \frac{1}{T}$ ⇨

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

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

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

	Time	Space
Step 1	$O(MT\log n)$ for weighted networks $O(MT)$ for unweighted networks	$O(M+n+m)$
Step 2	$O(M)$	$O(M+n)$
Step 3	$O(Md+nd^2+d^3)$	$O(M+nd)$

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

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

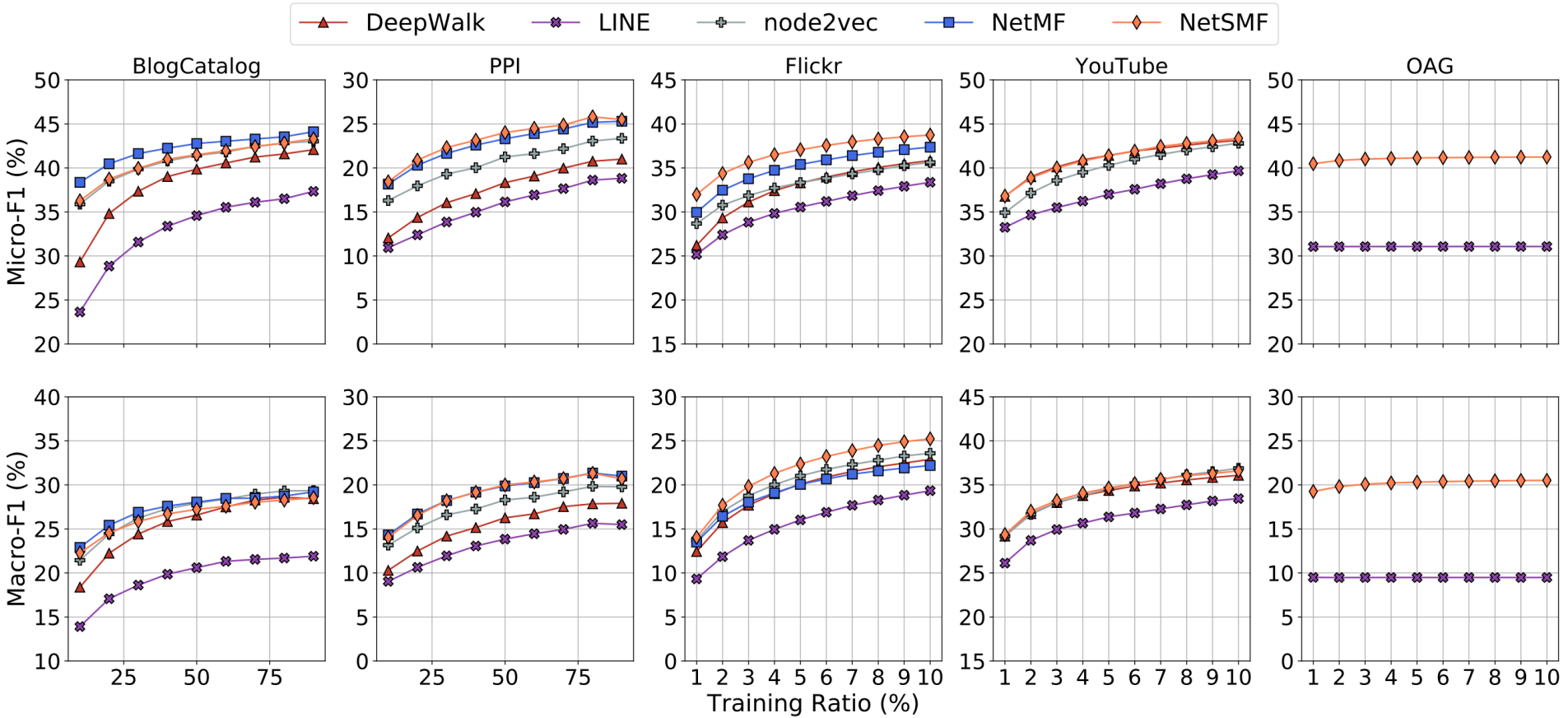
Theorem

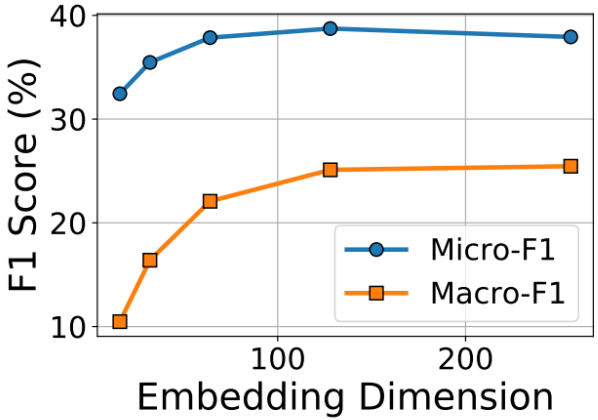
Let $\|\cdot\|_F$ be the matrix Frobenius norm. Then

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

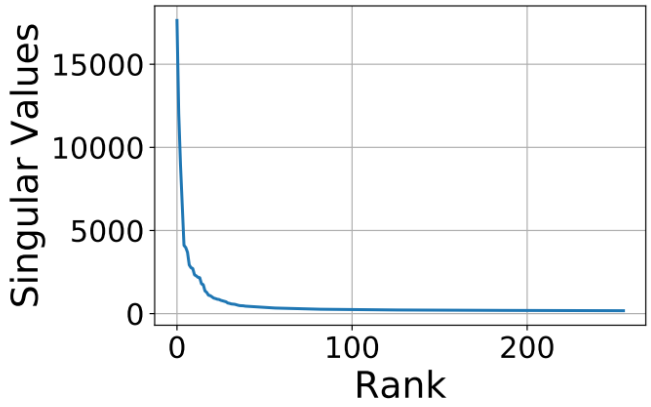
Dataset	BlogCatalog	PPI	Flickr	YouTube	OAG
$\|V\|$	10,312	3,890	80,513	1,138,499	67,768,244
$\|E\|$	333,983	76,584	5,899,882	2,990,443	895,368,962
#labels	39	50	195	47	19

	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


DeepWalk
LINE
node2vec
NetMF
NetSMF
BlogCatalog
PPI
Flickr
YouTube
OAG
Micro-F1 (%)
Macro-F1 (%)
Training Ratio (%)

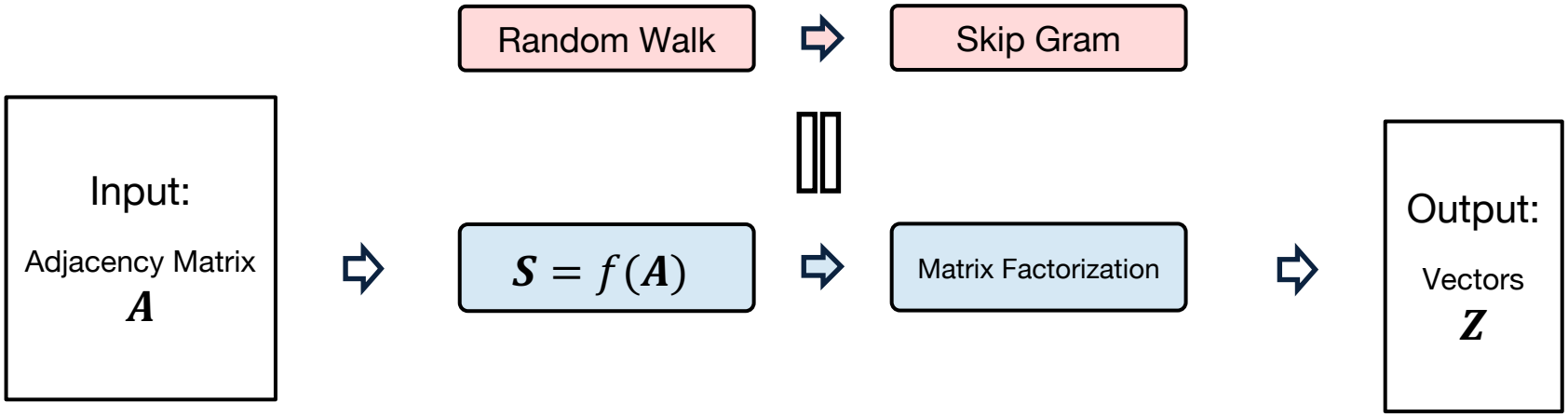

40
30
20
10
F1 Score (%)
100
200
Embedding Dimension
Micro-F1
Macro-F1

(a)


15000
10000
5000
0
Singular Values
0
100
200
Rank

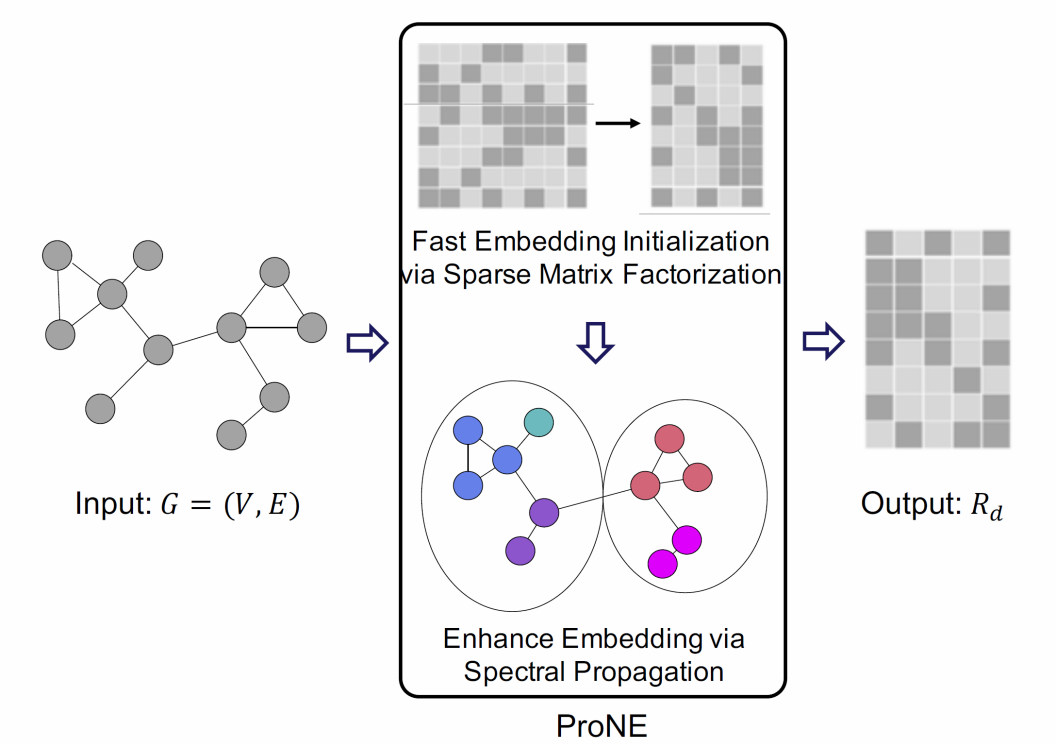
(b)

Connecting NE with graph neural networks



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

ProNE: More fast & scalable network embedding



Embedding enhancement via spectral propagation

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

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

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

• Chebyshev Expansion for Efficiency

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

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

$$\approx U \sum_{i=0}^{k-1} c_i(\theta) T_i(\bar{\Lambda}) U^T$$

$$= \sum_{i=0}^{k-1} c_i(\theta) T_i(\bar{L})$$

⇨

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

Efficiency

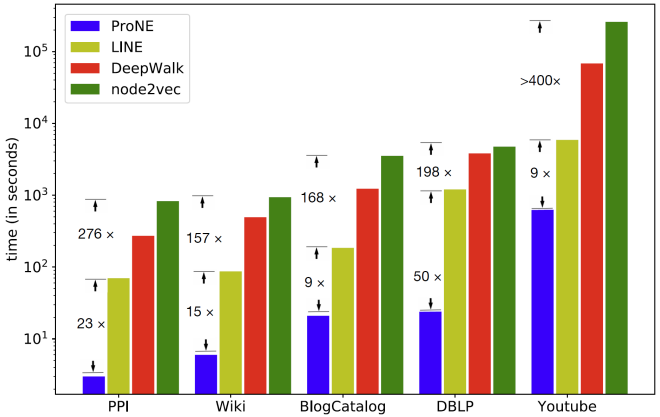
20 Threads

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**

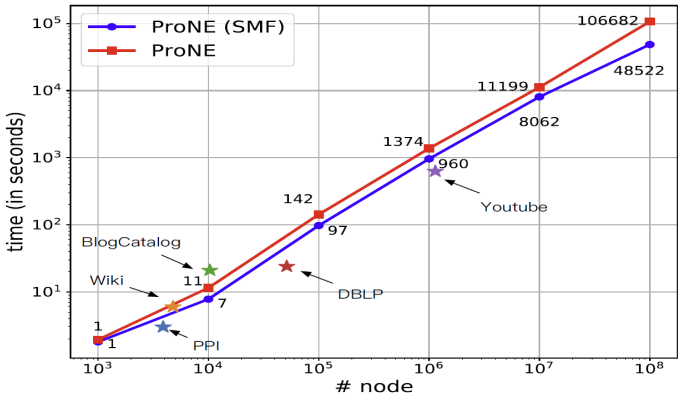
19hours 98mins 10mins

1.1M nodes



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

Scalability & Effectiveness

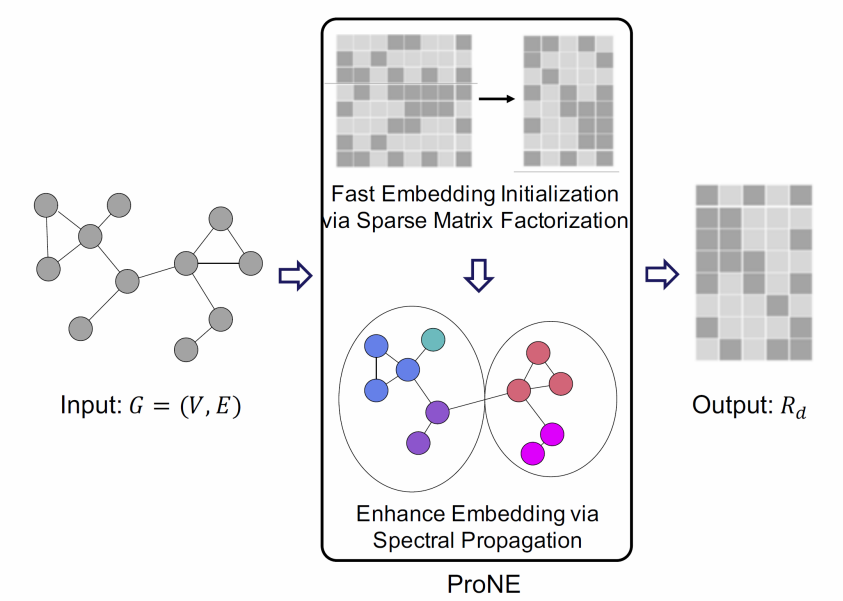


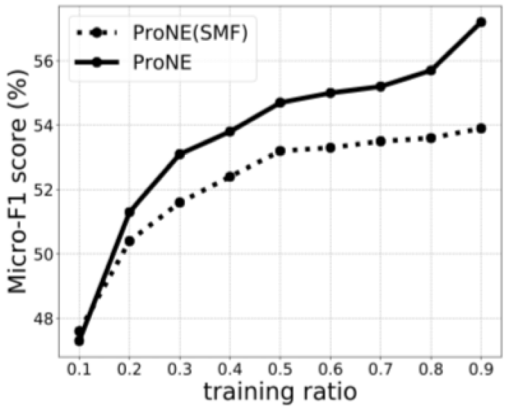
(a) The node degree is fixed to 10 and #nodes grows

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$1.3)
BlogCatalog	DeepWalk	36.2	39.6	40.9	41.4	42.2
	LINE	28.2	30.6	33.2	35.5	36.8
	node2vec	**36.3**	39.7	41.1	42.0	42.1
	GraRep	34.0	32.5	33.3	33.7	34.1
	HOPE	30.7	33.4	34.3	35.0	35.3
	ProNE (SMF)	34.6	37.6	38.6	39.3	39.0
	ProNE	36.0	40.5	42.3	43.2	**42.7**
	($\pm\sigma$)	($\pm$0.8)	($\pm$0.5)	($\pm$0.8)	($\pm$0.9)	($\pm$1.2)

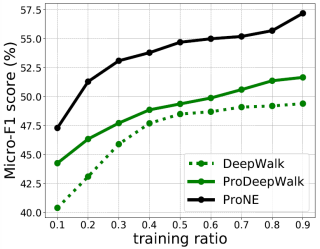
Embed 100,000,000 nodes by one thread:
29 hours with **performance superiority**

Embedding enhancement

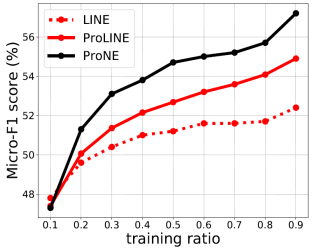




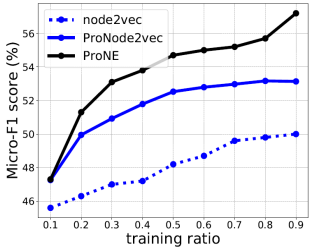
A general embedding enhancement framework



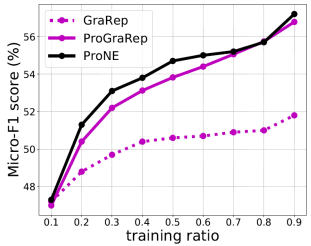
(a) ProDeepWalk



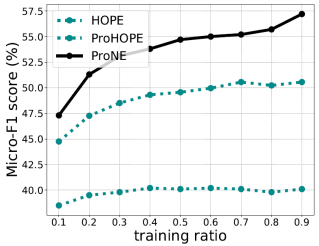
(b) ProLINE



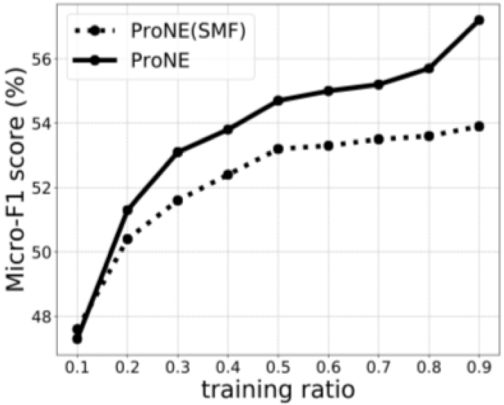
(c) ProNode2vec



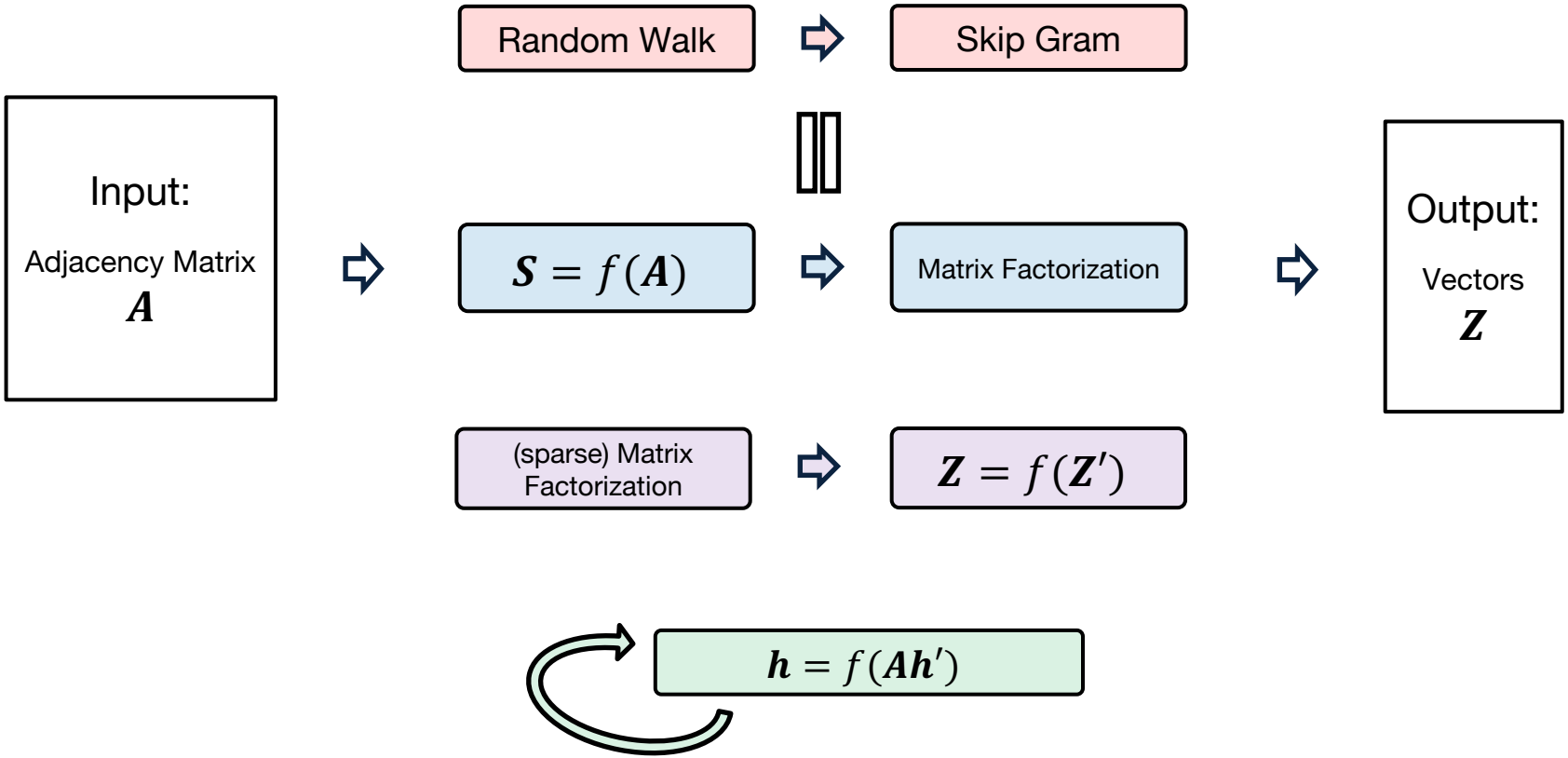
(d) ProGraRep



(e) ProHOPE

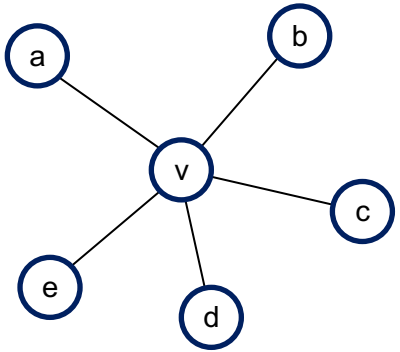


Connecting NE with graph neural networks



Graph neural networks

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

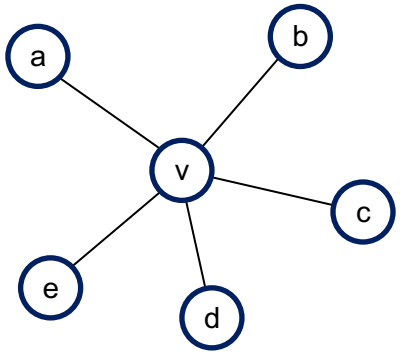


$$\boldsymbol{h}_v = f(\boldsymbol{h}_a, \boldsymbol{h}_b, \boldsymbol{h}_c, \boldsymbol{h}_d, \boldsymbol{h}_e)$$

Graph Neural Networks

- Input: an undirected weighted network $G = (V, E)$ with $|V| = n$ & $|E| = \mathrm{m}$
  - Adjacency matrix $\boldsymbol{A} \in \mathbb{R}_{+}^{n \times \mathrm{n}}$
  - $A_{i,j} = \begin{cases} a_{i,j} > 0 & (i,j) \in E \\ 0 & (i,j) \notin E \end{cases}$
  - Degree matrix $\boldsymbol{D} = diag(d_1, d_2, \cdots, d_n)$
  - Node feature matrix $\boldsymbol{X} \in \mathbb{R}^{n \times q}$
- Output: for each node, its k-dimension latent feature representation vector $\mathbf{Z}^{n \times k}$
  - –Latent feature embedding matrix $\mathbf{Z} \in \mathbb{R}^{n \times k}$

The Core of Graph Neural Networks

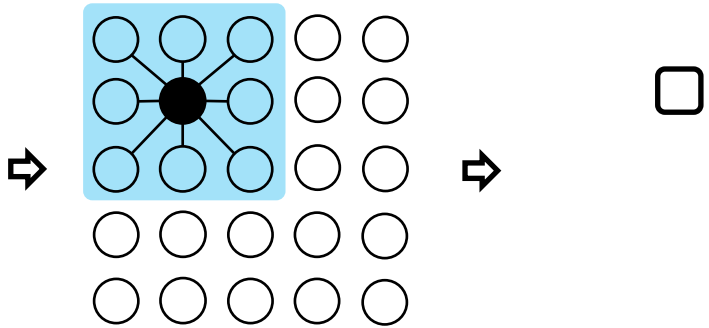


$$\boldsymbol{h}_v = f(\boldsymbol{h}_a, \boldsymbol{h}_b, \boldsymbol{h}_c, \boldsymbol{h}_d, \boldsymbol{h}_e)$$

Neighborhood Aggregation:

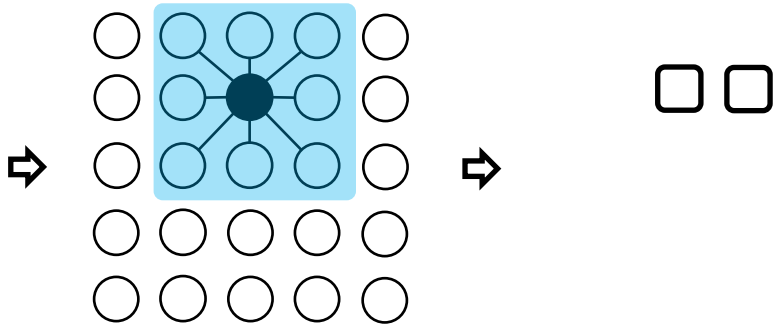
Aggregate neighbor information and pass into a neural network

Convolutional neural network



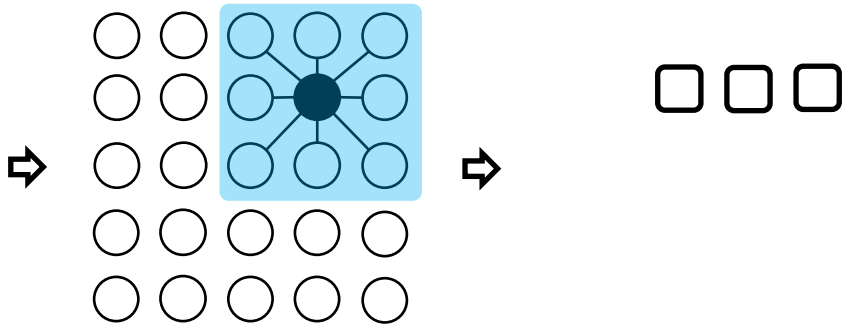
$$\sum WX$$

Convolutional neural network



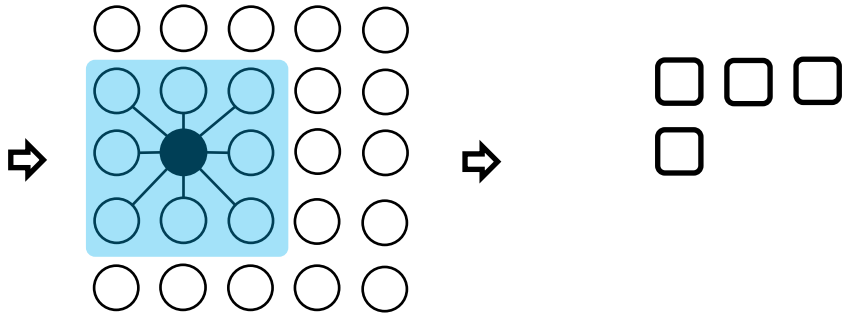
$$\sum WX$$

Convolutional neural network



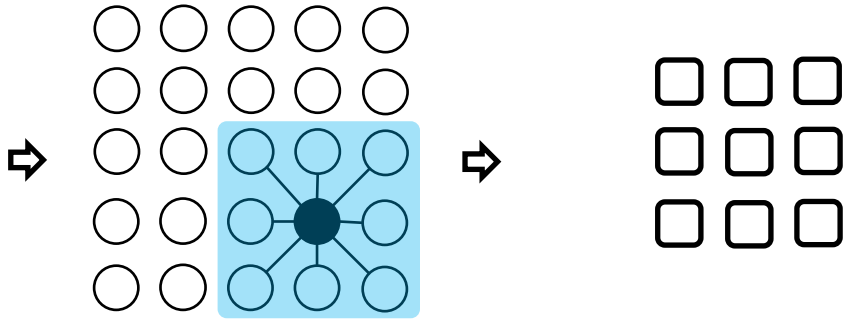
$$\sum WX$$

Convolutional neural network



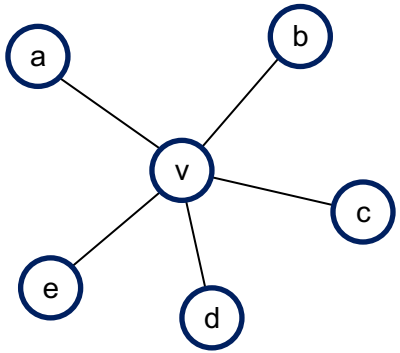
$$\sum WX$$

Convolutional neural network



$$\sum WX$$

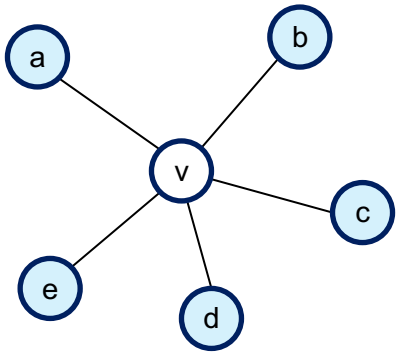
Graph neural networks



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

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

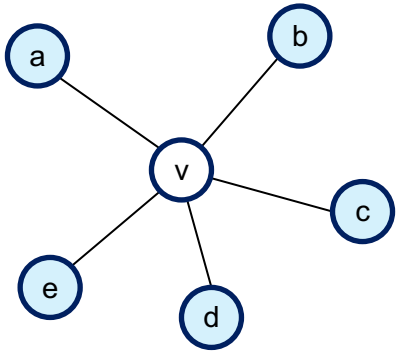
GNN: Graph convolutional networks



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

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

GNN: Graph convolutional networks



parameters in layer k

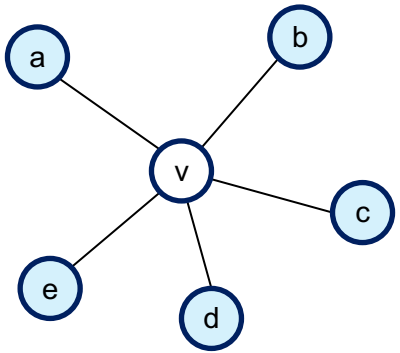
Non-linear activation function (e.g., ReLU)

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

node v's embedding at layer k

the neighbors of node v

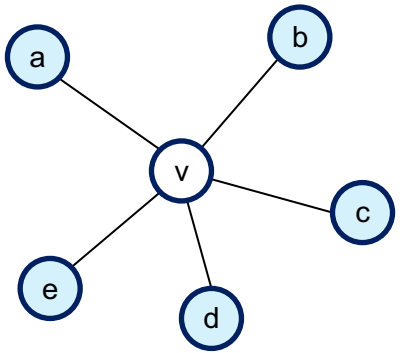
GNN: Graph convolutional networks



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

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

GNN: Graph convolutional networks



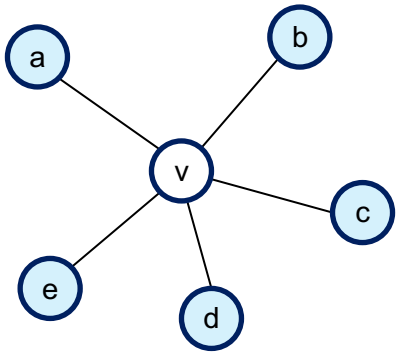
Aggregate from v's neighbors

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

Aggregate from itself

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

GNN: Graph convolutional networks

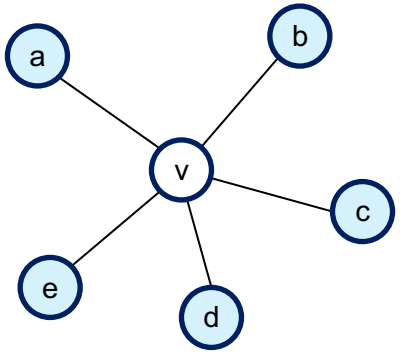


The same parameters for both its neighbors & itself

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

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

GNN: Graph Convolutional Networks



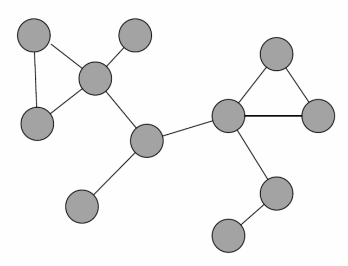
$$\boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{A}\boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{H}^{(k-1)}\boldsymbol{W}^{(k)}$$

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

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

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

GNN: Graph convolutional networks



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

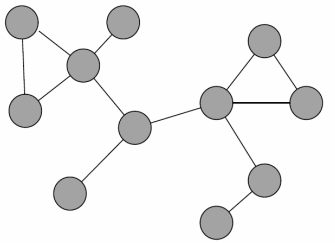
$\boldsymbol{H}^0 = \boldsymbol{X}$

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

Input

Output

GNN: Graph convolutional networks



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

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

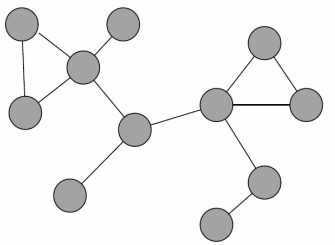
$$\Rightarrow \boldsymbol{H}^k = \sigma\left(\boldsymbol{D}^{-\frac{1}{2}}(\boldsymbol{A}+\boldsymbol{I})\boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{H}^{(k-1)}\boldsymbol{W}^{(k)}\right) \Rightarrow \quad \boldsymbol{Z} = \boldsymbol{H}^K$$

Input

Output

- 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 $\boldsymbol{Z}$

GNN: Graph convolutional networks



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

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

Input

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

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

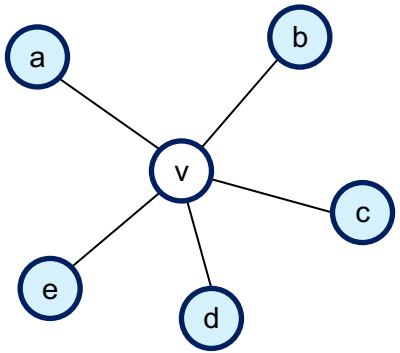
Output

GNN: Graph convolutional networks

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

- GCN is one way of neighbor aggregations
- **GraphSage**
- Graph Attention
- … …

GraphSage



GCN

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

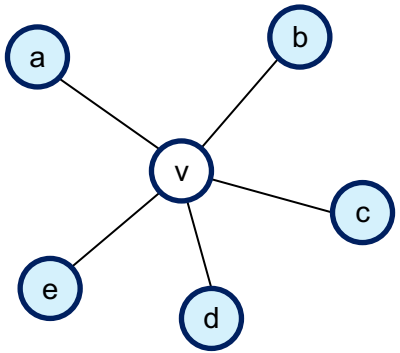
GraphSage

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

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
Hamiltion & Tang, AAAI 2019 Tutorial on Graph Representation Learning

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

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

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
Hamiltion & Tang, AAAI 2019 Tutorial on Graph Representation Learning

GraphSage

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

- 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

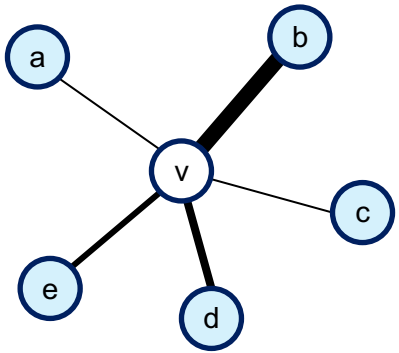
$$\mathrm{AGG} = \gamma(\{\mathbf{Q}\mathbf{h}_u^{k-1}, \forall u \in N(v)\})$$

- 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 Networks



Realistically, neighbors are of different importance to each node

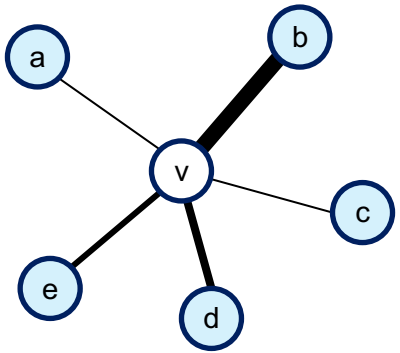
Velickovic et al. Graph Attention Networks. ICLR 2018

Graph Neural Networks

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

- GCN is one way of neighbor aggregations
- GraphSage
- **Graph Attention**
- … …

GNN: Graph Attention



GCN

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

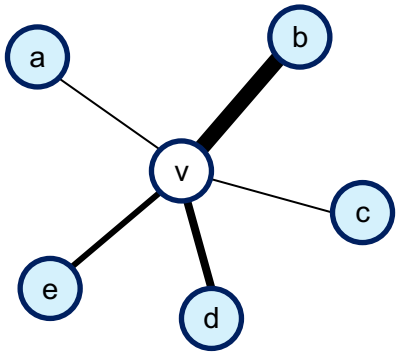
Graph Attention

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

Learned attention weights

Velickovic et al. Graph Attention Networks. ICLR 2018

GNN: Graph Attention



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

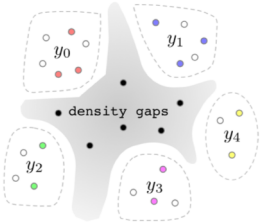
Various ways to define attention!

Velickovic et al. Graph Attention Networks. ICLR 2018

Graph neural networks

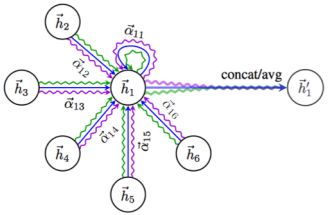
FastGCNs, Graph attention — 2018: Velickovic et al., ICLR'18, Chen et al., ICLR 2018

Neural message passing, GraphSage — 2017: Gilmer et al., ICML'17; Hamilton et al., NIPS'17



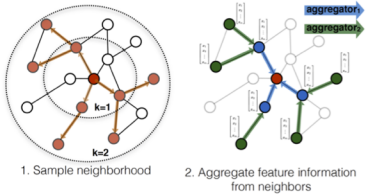
Gated graph neural network — 2016: Li et al., ICLR'16

structure2vec — 2016: Dai et al., ICML'16

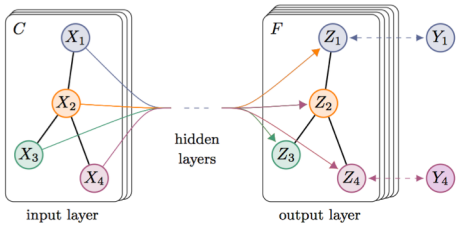


Graph convolutional network — 2015: Duvenaud et al., NIPS'15; Kipf & Welling ICLR'17

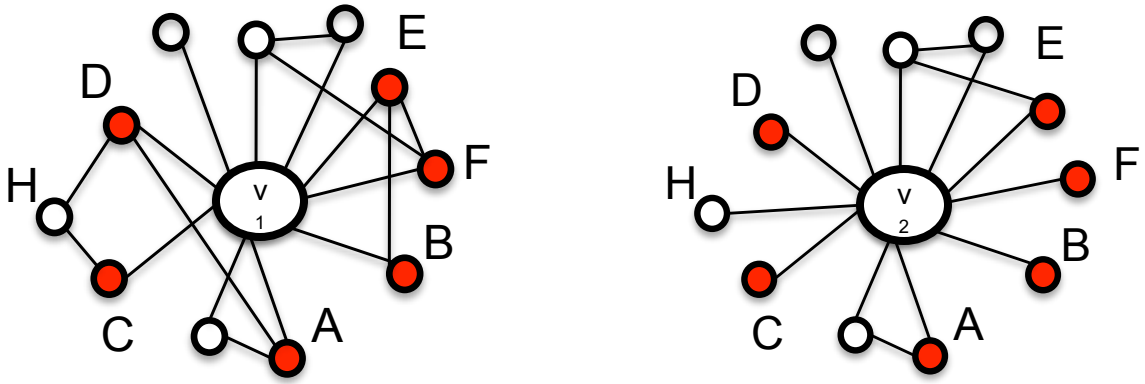
Spectral graph convolution — 2014: Bruna et al., ICLR'14



Graph neural network — 2005: Gori et al., IJCNN'05



DeepInf: Modeling social influence with graph neural networks



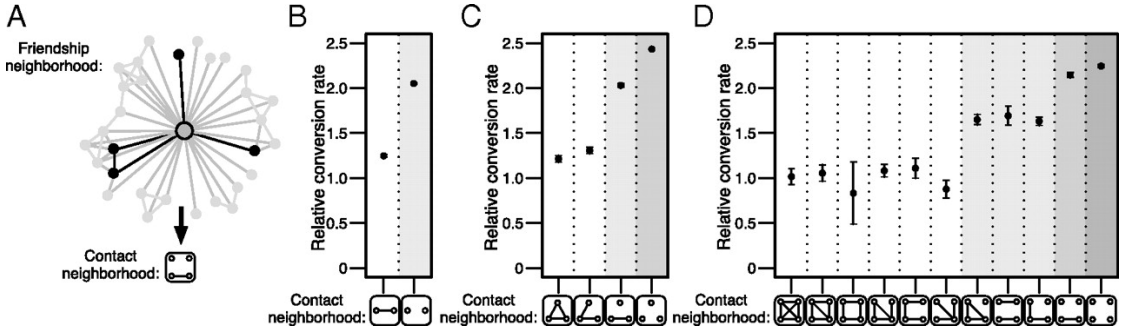
Who are more likely to be "active", v_1 or v_2?

e.g., active = "to attend CAAI AIDL"

Active neighbor | Inactive neighbor | v User to be influenced

1. Qiu et al. DeepInf: Social Influence Prediction with Deep Learning. In KDD'18.

Previous Solution



Name	Description
Vertex	Coreness [3].
	Pagerank [30].
	Hub score and authority score [8].
	Eigenvector Centrality [5].
	Clustering Coefficient [46].
	Rarity (reciprocal of ego user's degree) [1].
	Network embedding (DeepWalk [31], 64-dim).
Ego	The number/ratio of active neighbors [2].
	Density of subgnetwork induced by active neighbors [40].
	#Connected components formed by active neighbors [40].

J. Ugander, L. Backstrom, C. Marlow, and J. Kleinberg. Structural diversity in social contagion. PNAS'12.

Graph attention

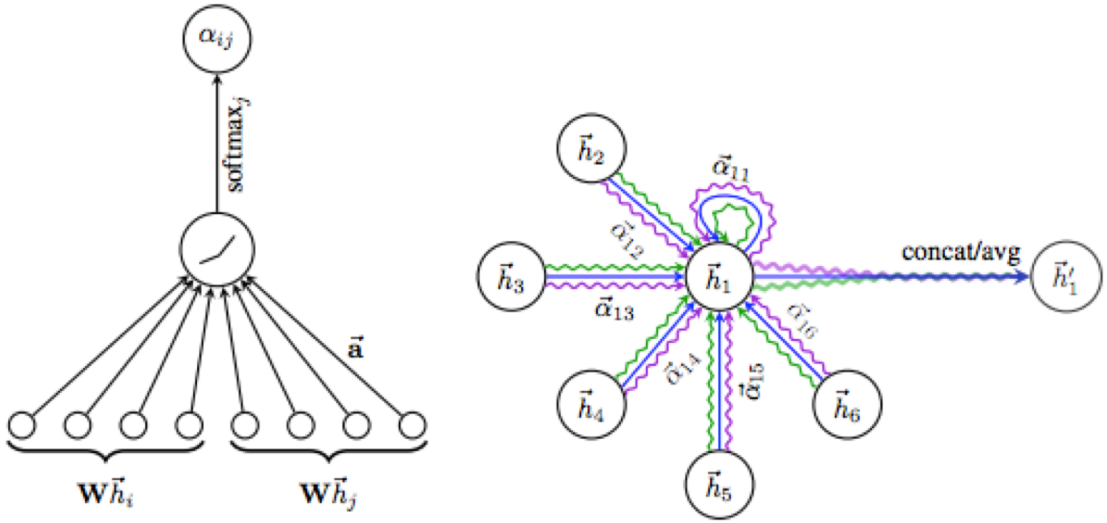
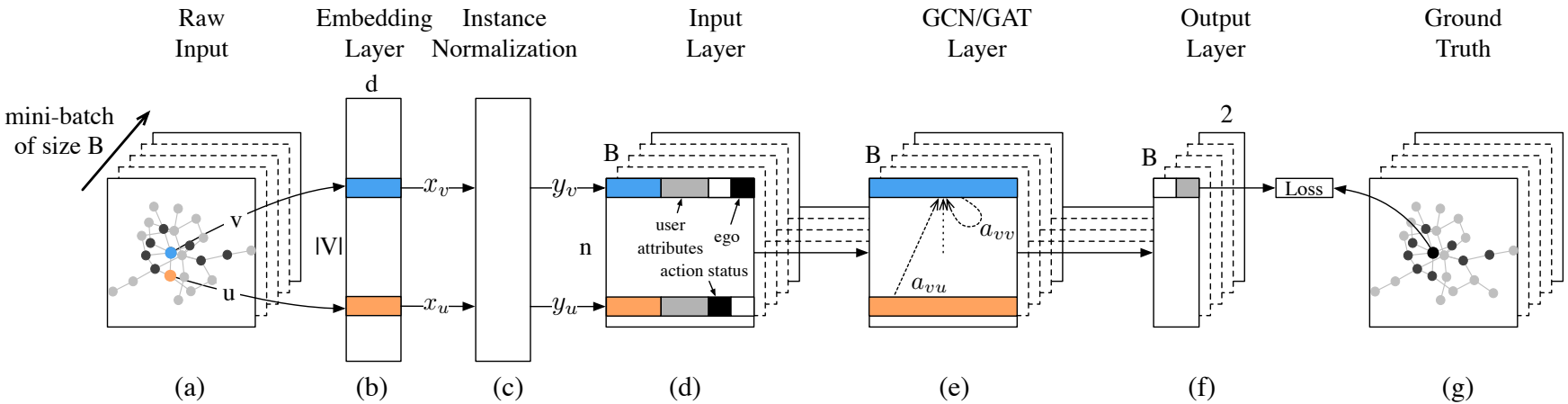


Figure 1: **Left:** The attention mechanism $a(\mathbf{W}\vec{h}_i, \mathbf{W}\vec{h}_j)$ employed by our model, parametrized by a weight vector $\vec{\mathbf{a}} \in \mathbb{R}^{2F'}$, applying a LeakyReLU activation. **Right:** An illustration of multi-head attention (with $K = 3$ heads) by node 1 on its neighborhood. Different arrow styles and colors denote independent attention computations. The aggregated features from each head are concatenated or averaged to obtain $\vec{h}'_1$.

Graph Attention Networks



1. Qiu et al. DeepInf: Social Influence Prediction with Deep Learning. In KDD'18.

Experiments --- Datasets

	OAG	Digg	Twitter	Weibo
$\|V\|$	953,675	279,630	456,626	1,776,950
$\|E\|$	4,151,463	1,548,126	12,508,413	308,489,739
N	499,848	24,428	499,160	779,164

- OAG (Open Academic Graph): predict paper citations
- Digg (a Reddit-like platform): predict vote-up
- Twitter & Weibo: predict retweet

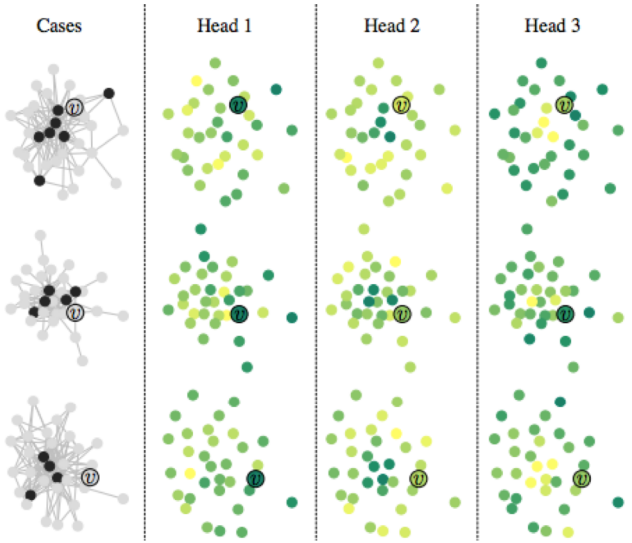
Experiments --- Results

- LR/SVM: Logistic regression/Support vector machine
- PSCN: SOTA deep graph kernel model (ICML'16)
- DeepInf-GAT: Our solution with GAT as bulding blocks

Data	Model	AUC	Prec.	Rec.	F1
OAG	LR	65.55	32.26	69.97	44.16
	SVM	65.48	32.17	**69.82**	44.04
	PSCN	67.70	36.24	60.46	45.32
	DeepInf-GAT	**70.59**	**38.93**	61.29	**47.61**
Digg	LR	84.72	56.78	73.12	63.92
	SVM	86.01	63.42	67.34	65.32
	PSCN	83.96	62.16	67.34	64.65
	DeepInf-GAT	**88.97**	**68.80**	**73.79**	**71.21**
Twitter	LR	78.07	45.86	**69.81**	55.36
	SVM	79.42	49.12	67.31	56.79
	PSCN	79.40	48.43	68.06	56.59
	DeepInf-GAT	**80.01**	**49.39**	67.47	**57.03**
Weibo	LR	77.10	42.34	72.88	53.56
	SVM	77.11	43.27	70.79	53.71
	PSCN	79.54	44.89	73.48	55.73
	DeepInf-GAT	**82.75**	**48.86**	**74.13**	**58.90**

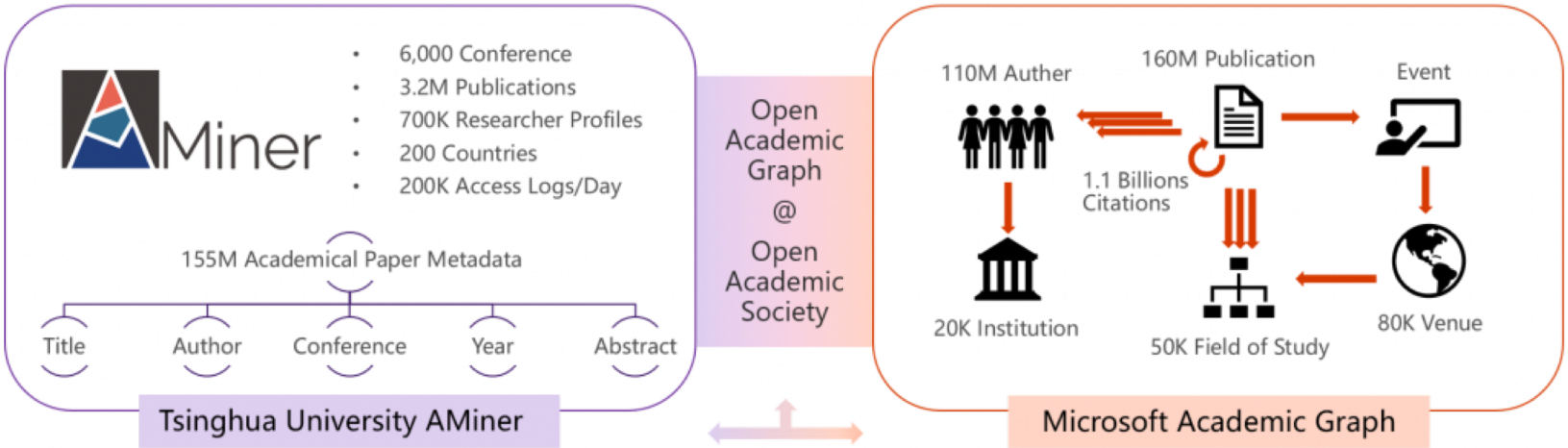
Case Study

- How different graph attention heads highlight different areas of the network.
- Head 1: Focus on the ego-user
- Head 2: Highlight active users
- Head 3: Highlight inactive users



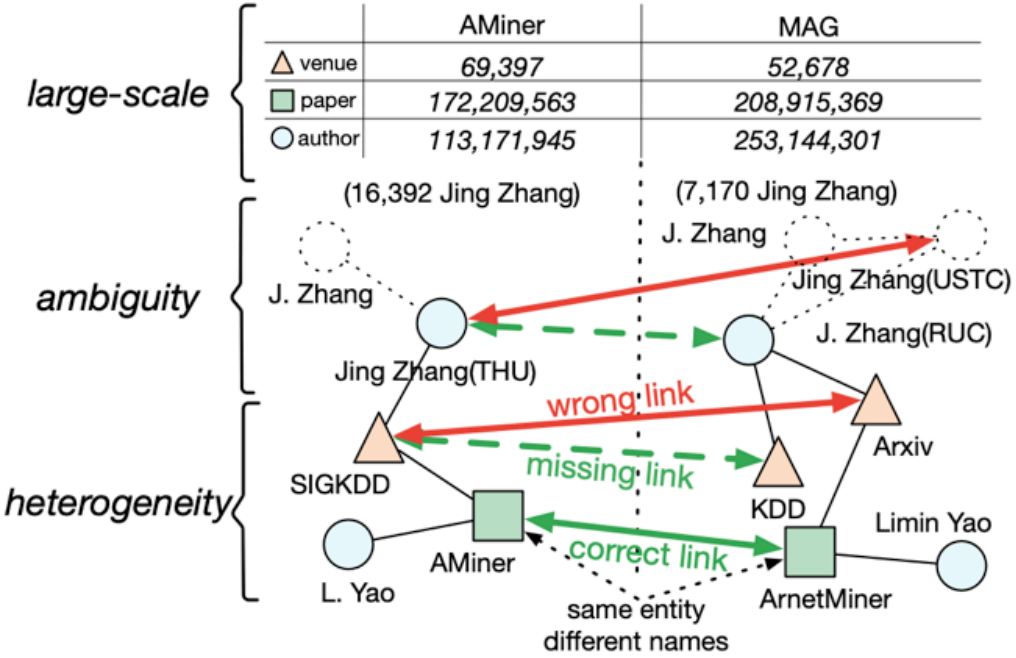
LinKG: Knowledge graph linking with heterogeneous graph attention

- **Input**: two heterogeneous entity graphs HG_1 and HG_2.
- **Output**: entity linkings $L = \{(e_1, e_2) | e_1 \in HG_1, e_2 \in HG_2\}$ such that e_1 and e_2 represent exactly the same entity.



1. Zhang, et al. OAG: Toward Linking Large-scale Heterogeneous Entity Graphs. In KDD'19.

Linking large-scale heterogeneous academic graphs

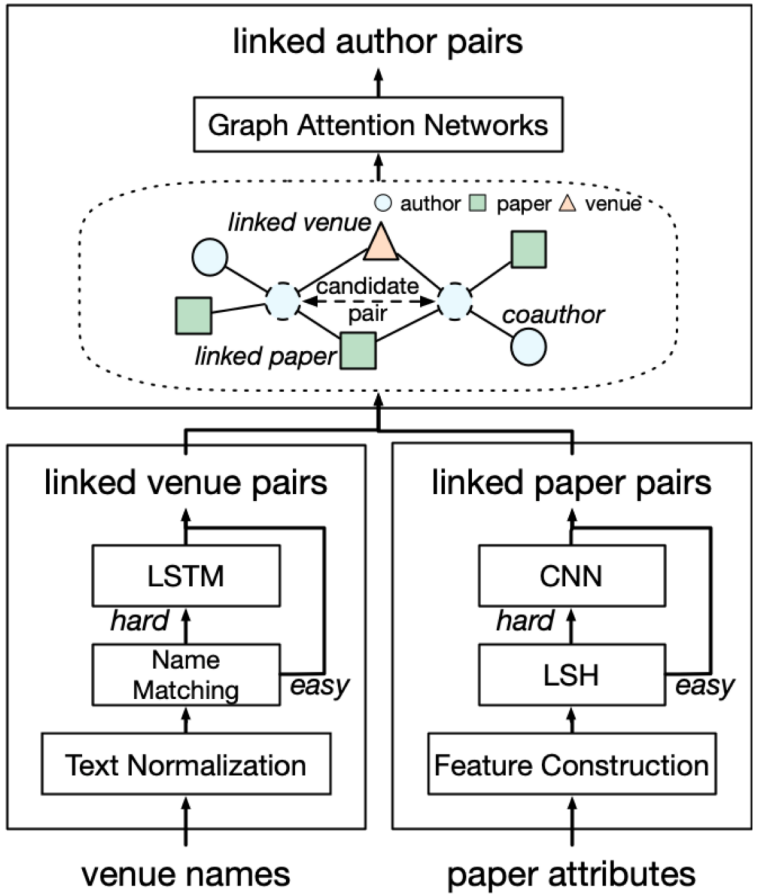


Solution -- LinKG

Author linking module

Venue linking module

Paper linking module



Author linking model — Heterogenous Graph Attention

- Encoder layers
  - attention coefficient attn (e_i, e_j) learnt by self-attention mechanism

$$o_{ij} = \text{attn}(Wh_i, Wh_j)$$

  - Normalized attention coefficient: differentiate different types of entities

$$\alpha_{ij} = \frac{\exp(\text{LeakyReLU}(c_{\tau(e_i)}^{\top} Wh_i + c_{\tau(e_j)}^{\top} Wh_j))}{\sum_{k \in \mathcal{N}_i} \exp(\text{LeakyReLU}(c_{\tau(e_i)}^{\top} Wh_i + c_{\tau(e_k)}^{\top} Wh_k))}$$

aggregation weight of source entity e_j 's embedding on target entity e_i

Author linking model — Heterogenous Graph Attention

- Encoder layers (cont.)
  - Multi-head attention

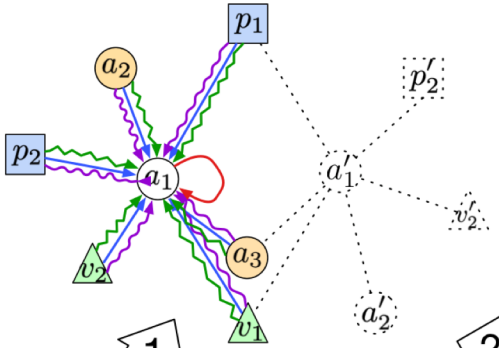
$$h_i' = \Big\Vert_{k=1}^{K} \sigma\Big(\sum_{j \in \mathcal{N}_i} \alpha_{ij}^k W^k h_j\Big)$$

  - Two graph attention layers in the encoder
- Decoder layers
  - Fuse embeddings of candidate pairs, and use fully-connected layers to produce the final matching score.

$$y = fc(\hat{h}_{MAG} \otimes \hat{h}_{AMiner})$$

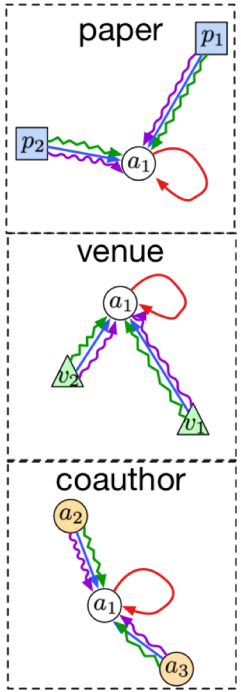
Author linking model — Heterogenous Graph Attention

Heterogeneous subgraph
for a candidate author pair



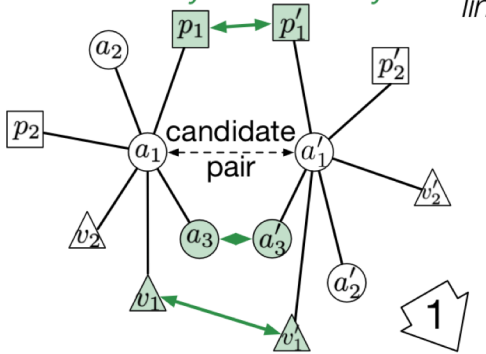
1 merge linked nodes

Split 2



Different attention parameters for different entity types

already linked entity



1

$$\alpha_{ij} = \frac{\exp(\text{LeakyReLU}(c^{\top}_{\tau(e_i)} W h_i + c^{\top}_{\tau(e_j)} W h_j))}{\sum_{k \in \mathcal{N}_i} \exp(\text{LeakyReLU}(c^{\top}_{\tau(e_i)} W h_i + c^{\top}_{\tau(e_k)} W h_k))}$$

Attention coefficient

Experimental Results

Table 1: Results of linking heterogeneous entity graphs. "–" indicates the method does not support the entity linking.

Methods		Keyword	SVM	Dedupe	COSNET	MEgo2Vec	LinKG$_C$	LinKG$_L$	LinKG
Venue	Prec.	80.15	81.69	84.25	–	–	84.67	**91.16**	**91.16**
	Rec.	83.76	83.45	80.92			85.81	**87.58**	**87.58**
	F1	81.91	82.56	82.55			85.23	**89.33**	**89.33**
Paper	Prec.	91.01	96.93	**99.30**	–	–	98.68	86.72	98.68
	Rec.	80.53	96.78	87.09			**98.10**	86.59	**98.10**
	F1	85.45	96.86	92.80			**98.39**	86.66	**98.39**
Author	Prec.	44.48	84.70	50.65	91.73	91.03	81.30	84.92	**95.37**
	Rec.	80.63	92.22	85.46	85.33	90.82	84.95	**94.75**	93.48
	F1	57.33	88.30	63.60	88.42	90.92	83.09	89.57	**94.42**
Overall	Prec.	74.80	92.36	82.26	91.73	91.03	92.38	86.21	**97.36**
	Rec.	80.64	94.89	86.38	85.33	90.82	93.29	89.41	**96.26**
	F1	77.61	93.61	84.27	88.42	90.92	92.83	87.78	**96.81**

OAG: Open Academic Graph

https://www.openacademic.ai/oag/

Data set	#Pairs/Venues	Date
Linking relations	29,841	2018.12
AMiner venues	69,397	2018.07
MAG venues	52,678	2018.11

Table 1: statistics of OAG venue data

Data set	#Pairs/Papers	Date
Linking relations	91,137,597	2018.12
AMiner papers	172,209,563	2019.01
MAG papers	208,915,369	2018.11

Table 2: statistics of OAG paper data

Data set	#Pairs/Authors	Date
Linking relations	1,717,680	2019.01
AMiner authors	113,171,945	2018.07
MAG authors	253,144,301	2018.11

Open Academic Graph

Open Academic Graph (OAG) is a large knowledge graph unifying two billion-scale academic graphs: Microsoft Academic Graph (MAG) and AMiner. In mid 2017, we published OAG v1, which contains 166,192,182 papers from MAG and 154,771,162 papers from AMiner (see below) and generated 64,639,608 linking (matching) relations between the two graphs. This time, in OAG v2, author, venue and newer publication data and the corresponding matchings are available.

Overview of OAG v2

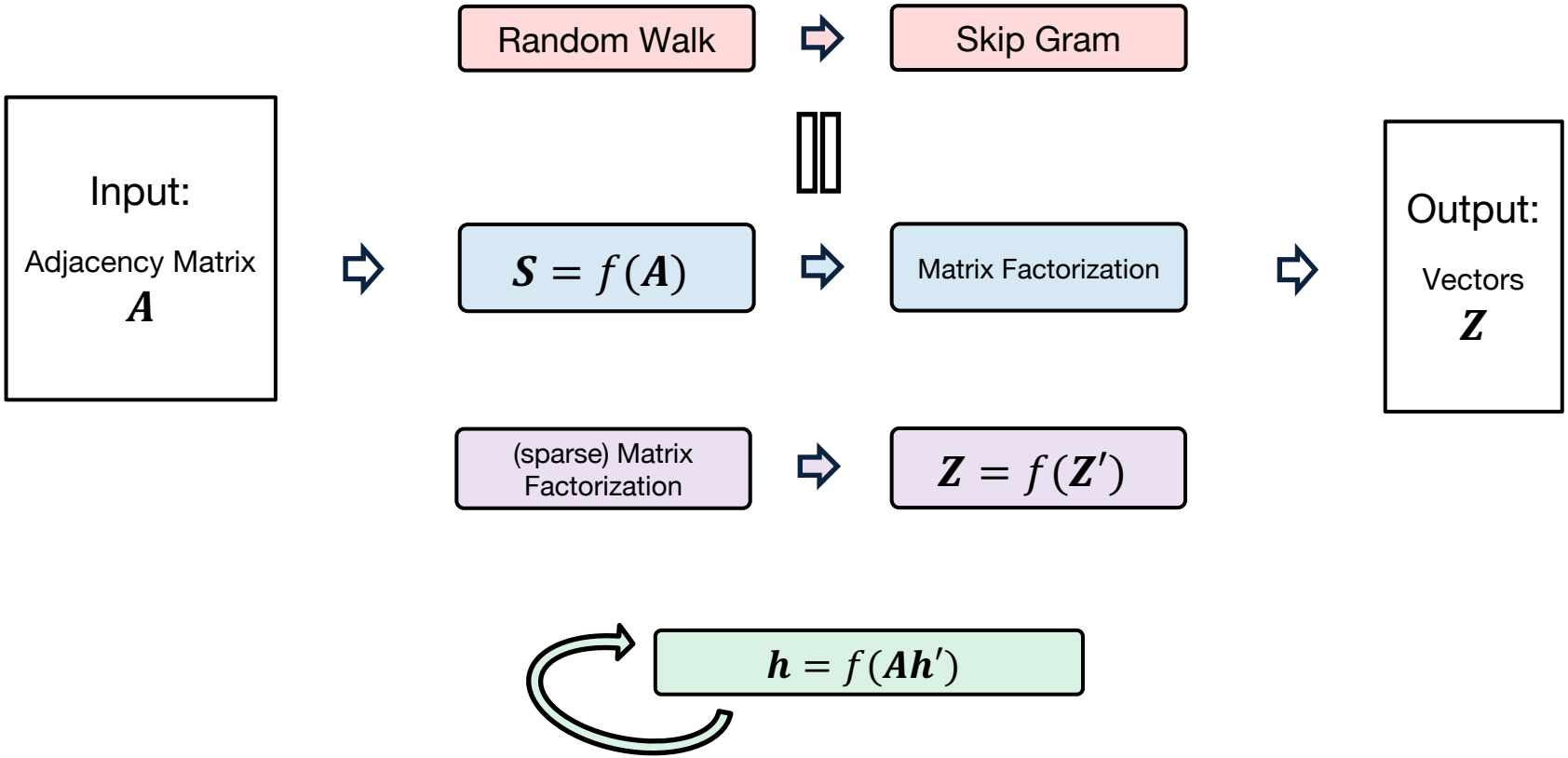
The statistics of OAG v2 is listed as the three tables below. The two large graphs are both evolving and we take MAG November 2018 snapshot and AMiner July 2018 or January 2019 snapshot for this version.

1. Zhang, et al. OAG: Toward Linking Large-scale Heterogeneous Entity Graphs. In KDD'19.

GNN applications

- DeepInf: Modeling social influence with graph neural networks
- LinKG: Knowledge graph linking with heterogeneous graph attention

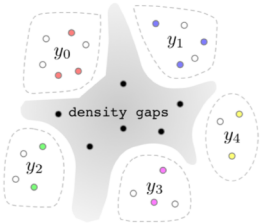
Connecting NE with graph neural networks



Graph neural networks

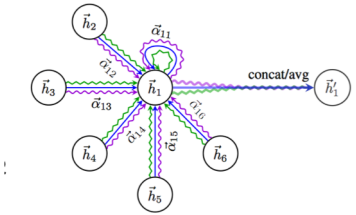
FastGCNs, Graph attention — 2018: Velickovic et al., ICLR'18, Chen et al., ICLR 2018

Neural message passing, GraphSage — 2017: Gilmer et al., ICML'17; Hamilton et al., NIPS'17



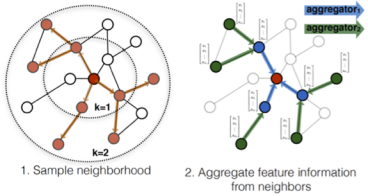
Gated graph neural network — 2016: Li et al., ICLR'16

structure2vec — 2016: Dai et al., ICML'16



Graph convolutional network — 2015: Duvenaud et al., NIPS'15; Kipf & Welling ICLR'17

Spectral graph convolution — 2014: Bruna et al., ICLR'14



Graph neural network — 2005: Gori et al., IJCNN'05