Graph Representation Learning and Applications

Jiezhong Qiu, Jie Tang
Tsinghua University

Yuxiao Dong
Microsoft Research, Redmond

Microsoft Research

Updated Grading Policy

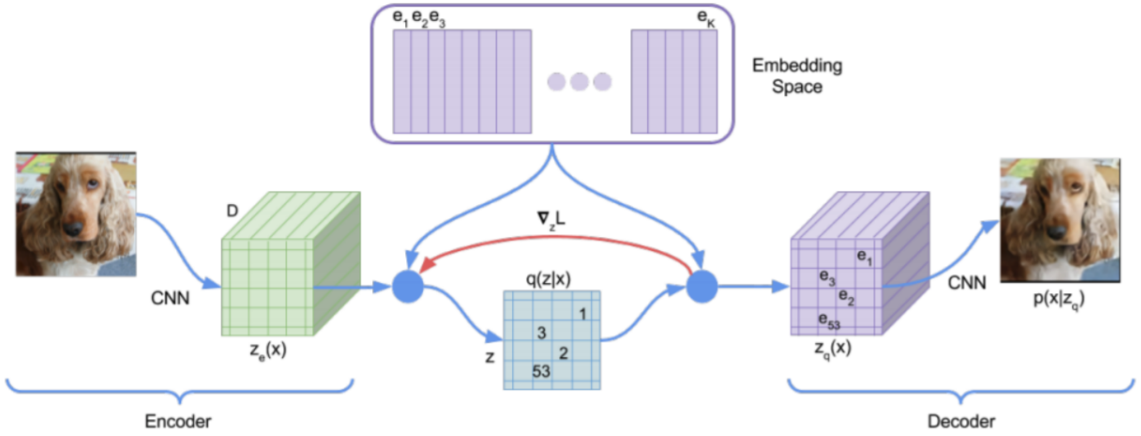
- Participation (10%)
  - Collect information from rain classroom (10 points)
- Homework (40%)
  - ~~4 homework (10 points each)~~
  - 2 homework (20 point each)
  - HW2 will be released in today's course.
- Final Project (50%)
  - A proposal + A midterm report (will not be graded)
  - Poster presentation (Week 15)
  - A final report in NeurIPS format + code (Week 17)

Frontier in ML

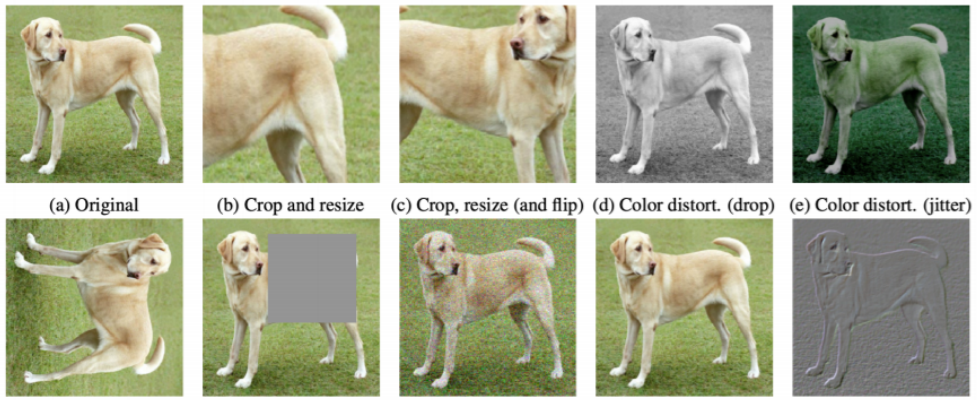
- Recurrent Neural Networks (Week 9)
- Representation Learning (Week 9)
- Network Embedding and GNN (Week 10)
- AutoML (Week 11)
- Reinforcement Learning (Week 12)
- Pre-Training and Self-Supervised Learning (Week 13)
- Cognitive Graph and Reasoning (Week 14)
- plus 2-3 invited speakers
  - You are welcome to suggest potential candidates

Debate (Week 13/14)

- Generative
  - GPT-2/3
  - VQ-VAE
  - ...



- Contrastive
  - SimCLR
  - MoCo
  - ...



Gifts + 5 points bonus

Self-supervised Learning: Generative or Contrastive, https://arxiv.org/abs/2006.08218

Graph Representation Learning

Graph Benchmarks (HW2)

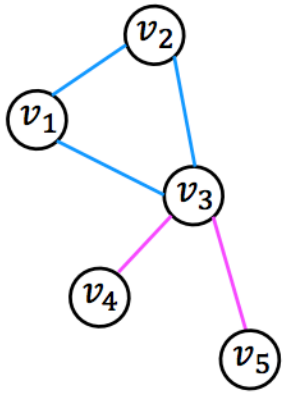
CogDL

github.com/thudm/cogdl

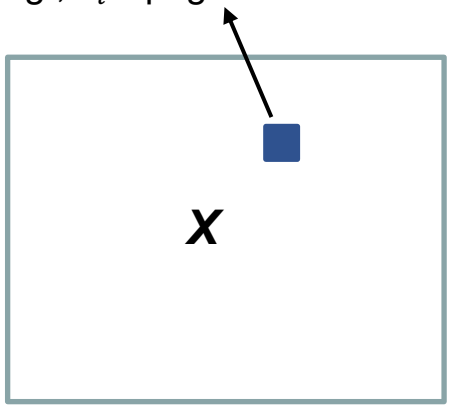
Representation Learning on Networks

- **The first part:**
  - Network embeddings
    - Embedding models
    - Theoretical understanding
    - Large-scale embedding
- **The second part:**
  - Graph neural networks
    - GCN/GAT/GraphSAGE…
    - Graph Neural Network pretraining
    - GNN for heterogeneous graph

The classic network analysis framework



x_{ij}: node v_i's j^{th} feature,
e.g., v_i's pagerank value



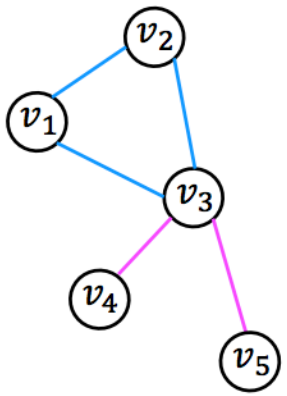
hand-crafted feature matrix

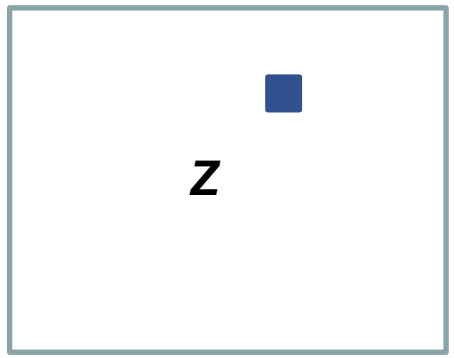
Graph & network applications
- Node label inference;
- Link prediction;
- User behavior… …

feature **engineering**

machine learning models

Representation learning for networks?





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

Graph & network applications
- Node label inference;
- Node clustering;
- Link prediction;
- … …

Feature ~~engineering~~ **learning**

machine learning models

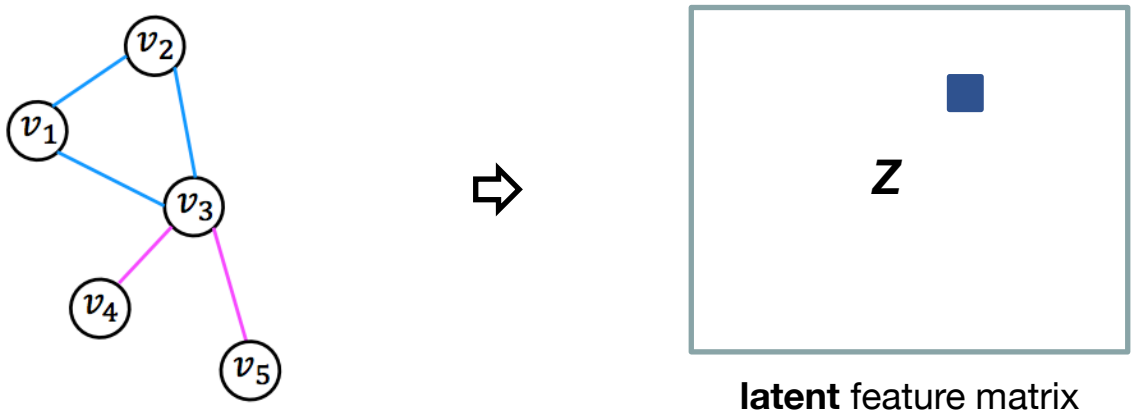
- Bengio, Courville, Vincent. Representation learning: A review and new perspectives. *IEEE TPAMI* 2013.
- LeCun, Bengio, Hinton. Deep learning. *Nature,* 521(7553):436–444, 2015.

Representation learning for networks?

Problem (Graph representation learning, network embedding, graph embedding)

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

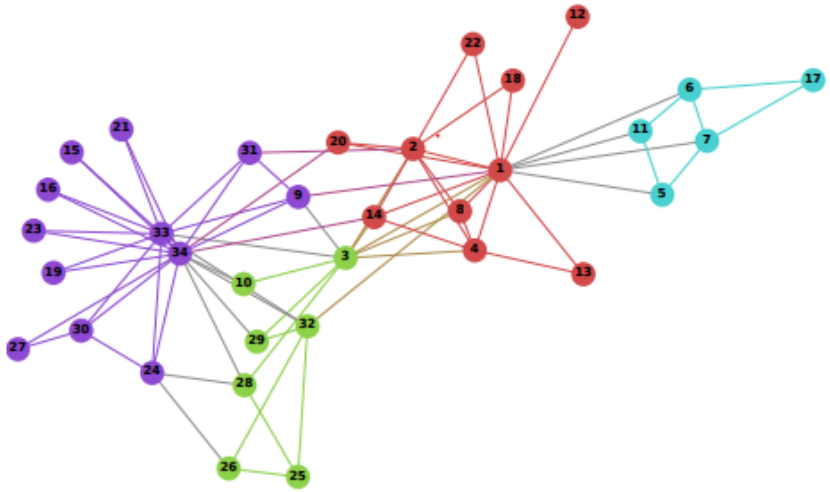
The goal is to map each node into a latent low-dimension space such that network structure information is encoded into distributed node representations



latent feature matrix

Feature **learning**

Graph representation learning



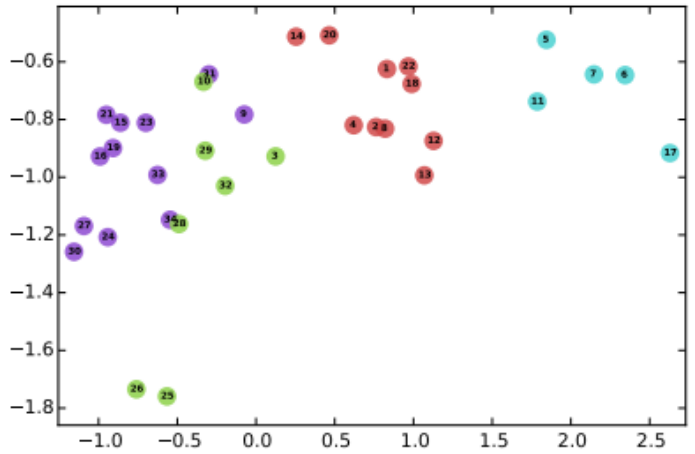
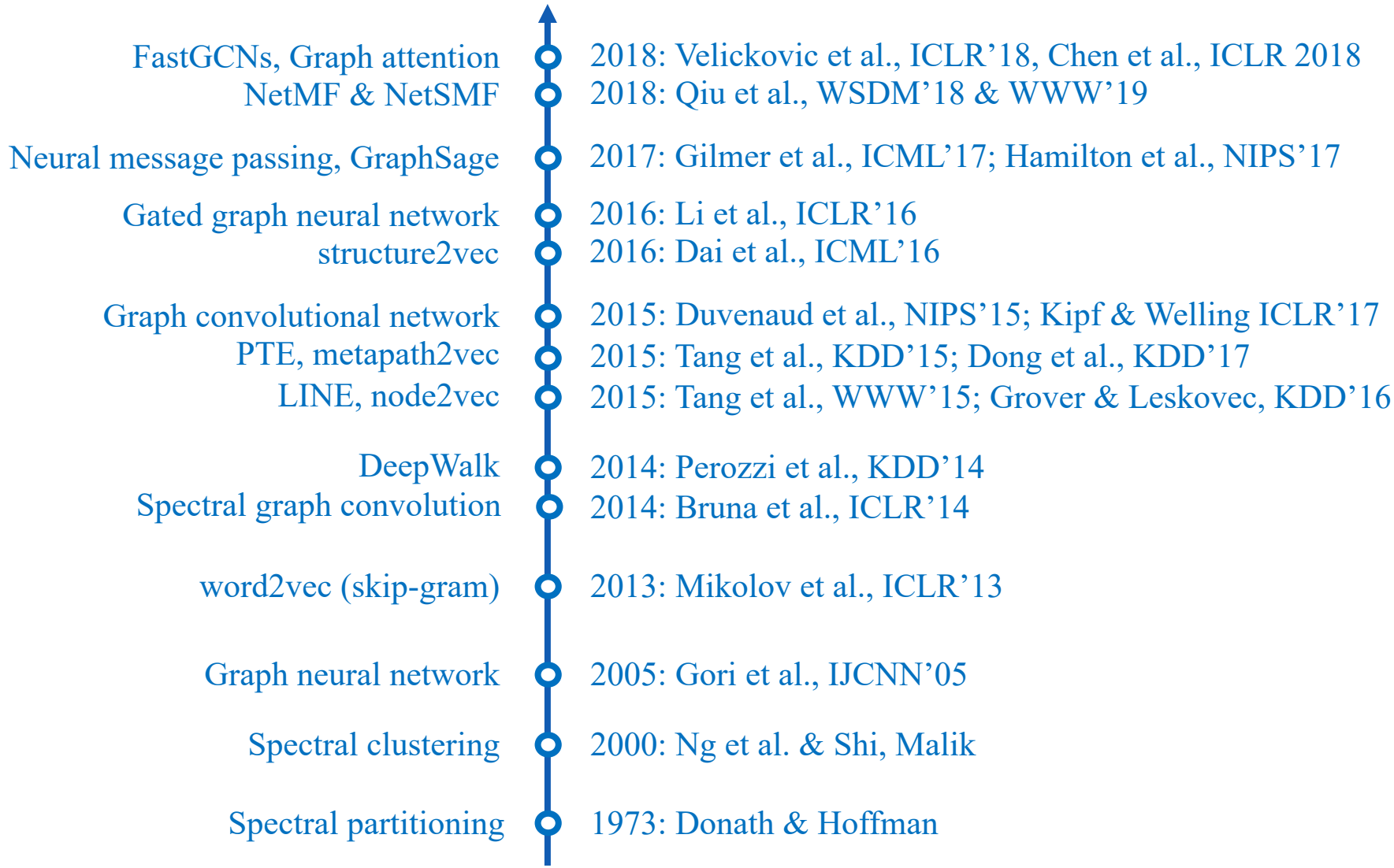


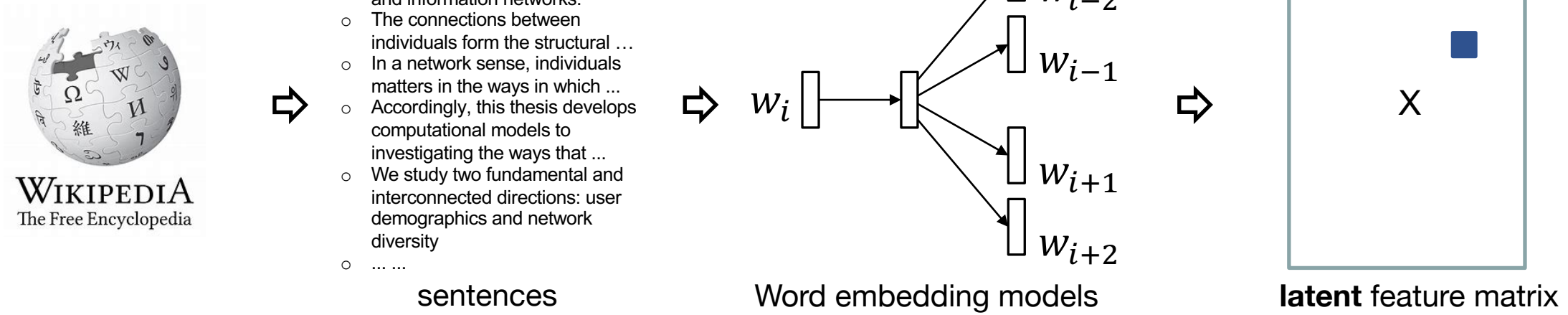
Image Credit: Perozzi et al. DeepWalk: Online learning of social representations. In *KDD' 14*, pp. 701–710.

Network Representation Learning / Network Embedding



Word2vec in NLP

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



sentences | Word embedding models | **latent** feature matrix

- Harris' distributional hypothesis: words in similar contexts have similar meanings.
- Key idea: try to predict the words that surrounding each one.

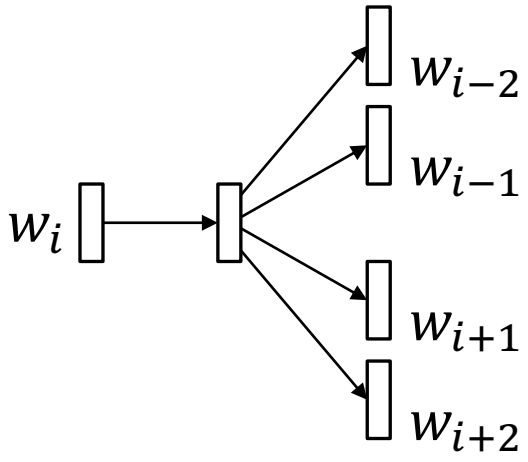
1. Bengio, et al. Representation learning: A review and new perspectives. In *IEEE TPAMI 2013.*
2. Mikolov, et al. Efficient estimation of word representations in vector space. In *ICLR 2013*.

The learning problem

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

word2vec addresses the $O(|V|^2)$ complexity of skip-gram by

- Hierarchical softmax
- Negative sampling



Skip gram with negative sampling

Skip-gram with negative sampling (SGNS)

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

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

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

$$x_w^T x_c \approx \log \frac{\#(w,c)|\mathcal{D}|}{b\#(w)\#(c)}$$

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

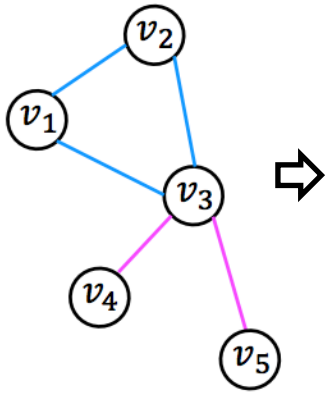
Distributional Hypothesis of Harris

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

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

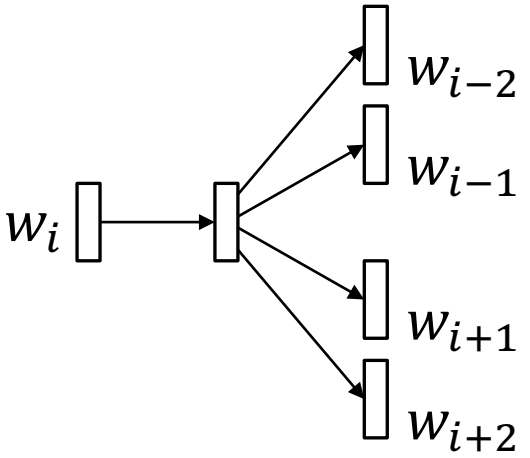
Network Representation Learning

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

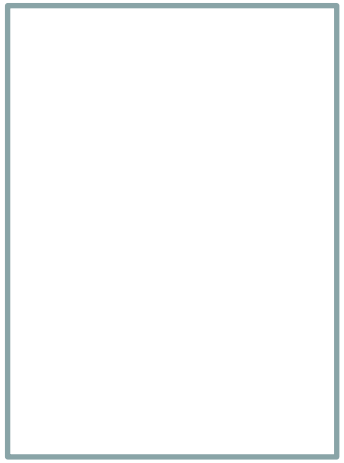


- Computational lens on big social and information networks.
- The connections between individuals form the structural …
- In a network sense, individuals matters in the ways in which ...
- Accordingly, this thesis develops computational models to investigating the ways that ...

sentences



skip-gram

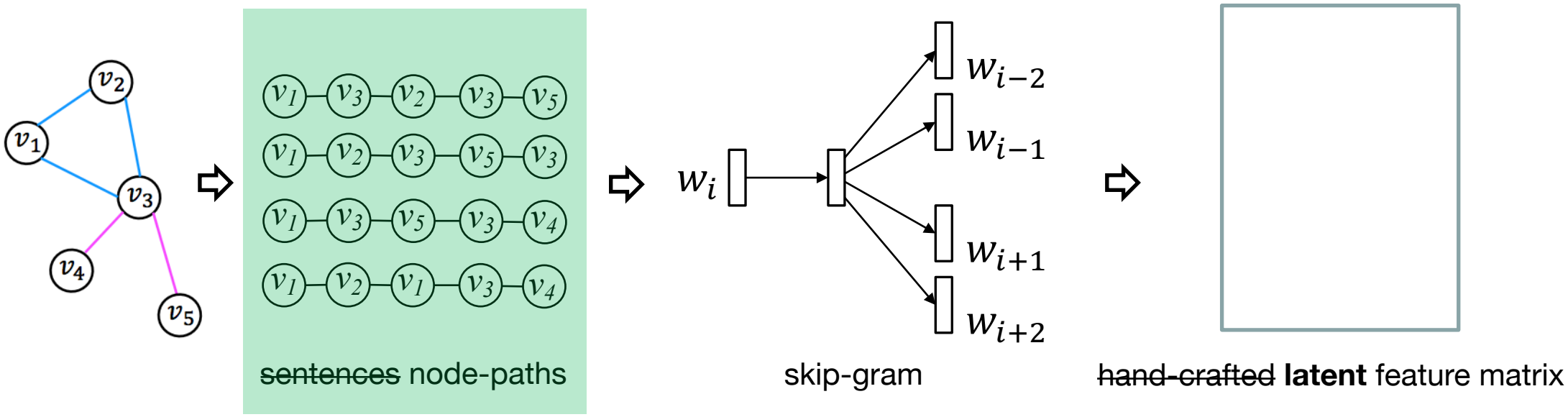


latent feature matrix

Feature **learning**

Network embedding: DeepWalk

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



Feature **learning**

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

Network Embedding

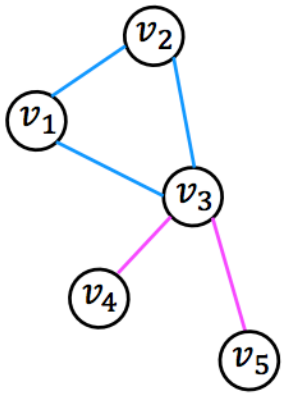
Random Walk

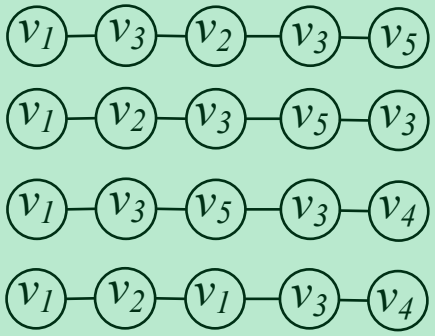
Skip Gram

Input:

Adjacency Matrix

$\boldsymbol{A}$





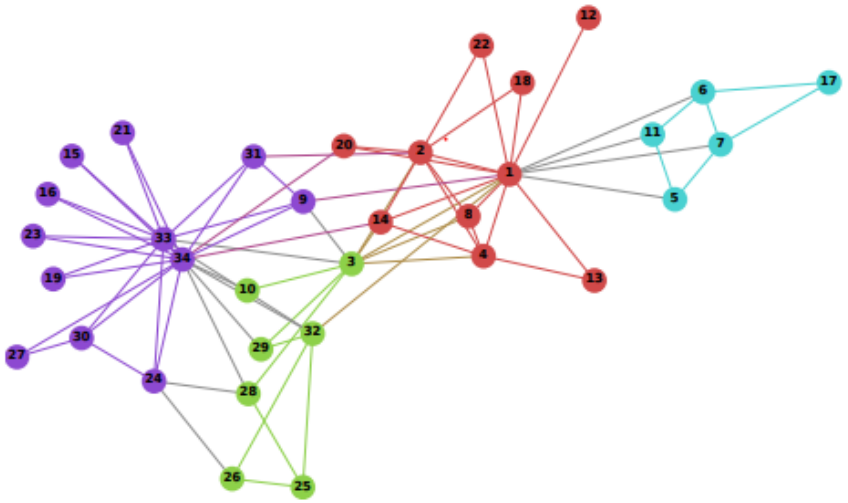
~~sentences~~ node-paths

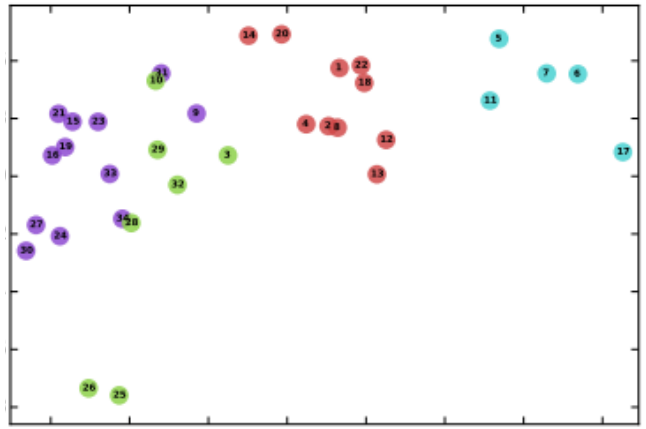
Output:

Vectors

$\boldsymbol{Z}$

Network embedding: DeepWalk





Graph & network applications

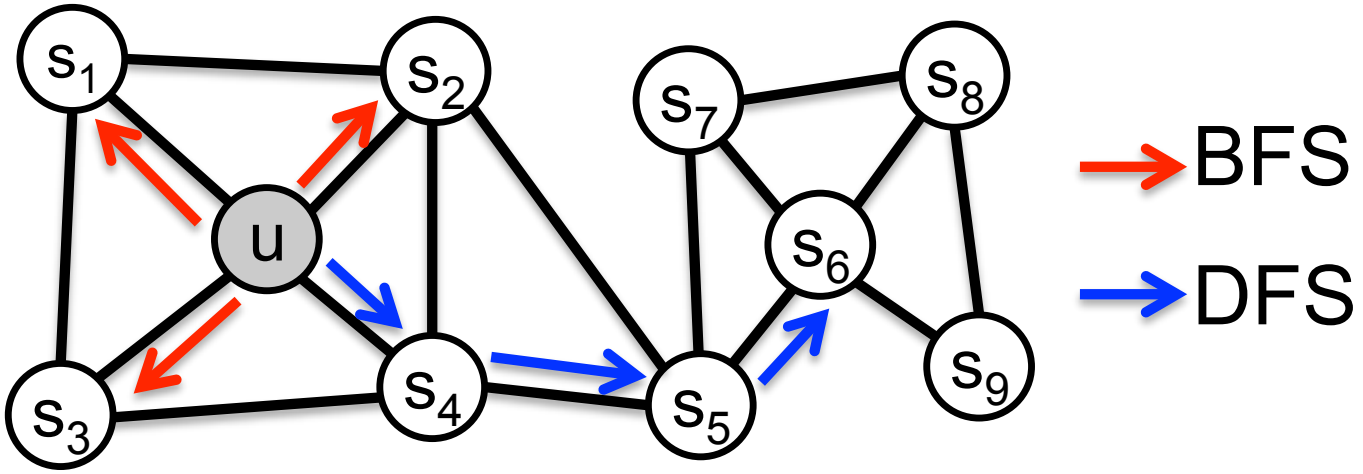
- Node label inference;
- Node clustering;
- Link prediction;
- … …

Image Credit: Perozzi et al. DeepWalk: Online learning of social representations. In *KDD' 14*, pp. 701–710.

Random Walk Strategies

- Random Walk
  - DeepWalk
- Biased Random Walk
  - 2nd order Random Walk
    - node2vec
  - Metapath guided Random Walk
    - metapath2vec

node2vec



$$N_{BFS}(u) = \{ s_1, s_2, s_3 \}$$

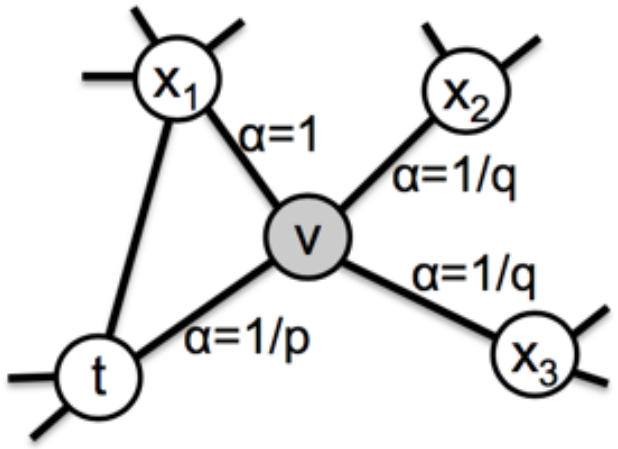
$$N_{DFS}(u) = \{ s_4, s_5, s_6 \}$$

A. Grover and J. Leskovec. 2016. node2vec: Scalable feature learning for networks. *KDD*, 855–864.

node2vec

Biased random walk R that given a node v generates random walk neighborhood $N_{rw}(v)$

- Return parameter p:
  - Return back to the previous node
- In-out parameter q:
  - Moving outwards (DFS) vs. inwards (BFS)



Return Edge: $P(v_{t+1} = t | v_t = v, v_{t-1} = t) = \frac{1}{p}$

Closed Triad: $P(v_{t+1} = x_1 | v_t = v, v_{t-1} = t) = 1$

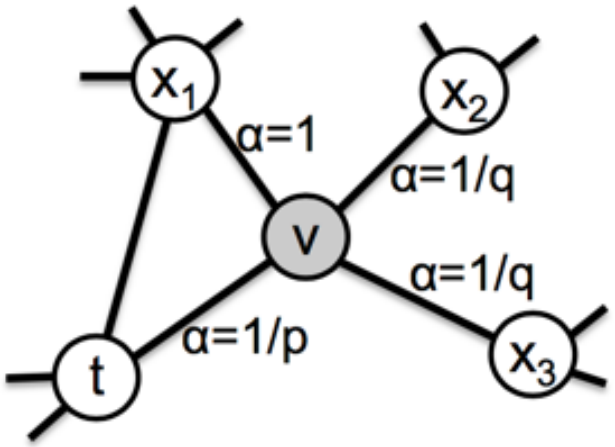
Open Triad: $P(v_{t+1} = x_2 | v_t = v, v_{t-1} = t) = \frac{1}{q}$

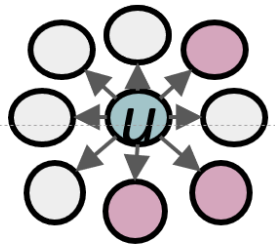
Picture snipped from Leskovec

node2vec

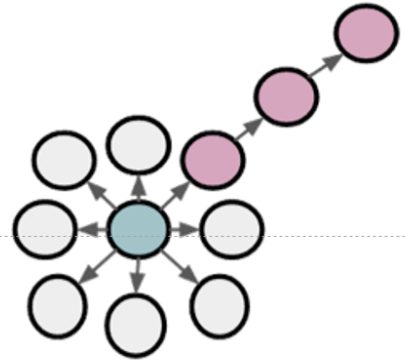
Biased random walk R that given a node v generates random walk neighborhood $N_{rw}(v)$

- Return parameter p:
  - Return back to the previous node
- In-out parameter q:
  - Moving outwards (DFS) vs. inwards (BFS)





BFS:
Micro-view of neighbourhood



DFS:
Macro-view of neighbourhood

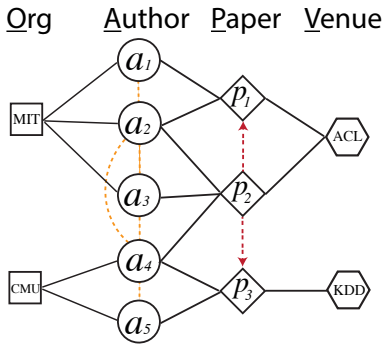
Picture snipped from Leskovec

Random Walk Strategies

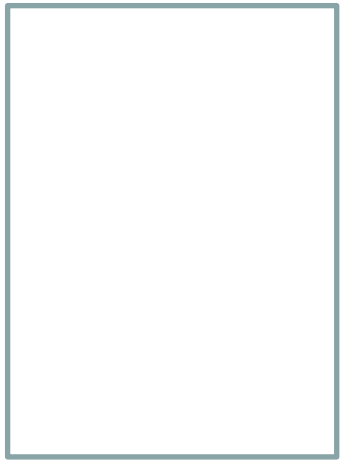
- Random Walk
  - DeepWalk
- Biased Random Walk
  - 2nd order Random Walk
    - node2vec
  - Metapath guided Random Walk
    - metapath2vec

Heterogeneous random walk

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



- How do we random walk over heterogeneous networks?
- How do we apply skip-gram over different types of nodes?



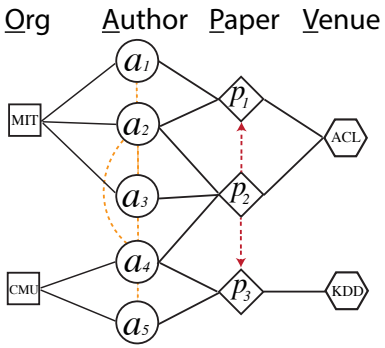
~~hand-crafted~~ **latent** feature matrix

Feature **learning**

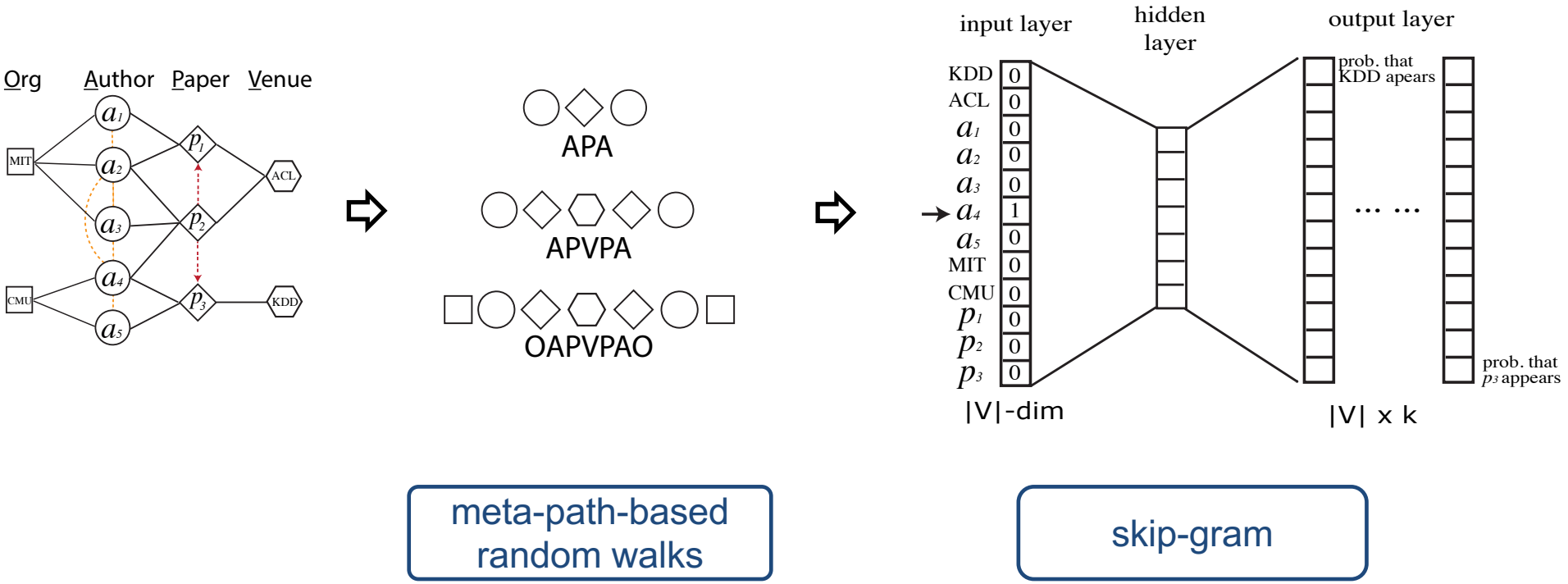
Microsoft Academic Graph

- Paper vertices and Conference vertices are of different orders of magnitude.

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

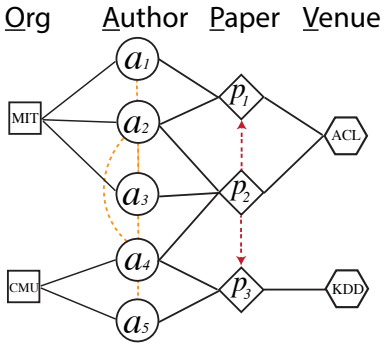


Heterogeneous graph embedding



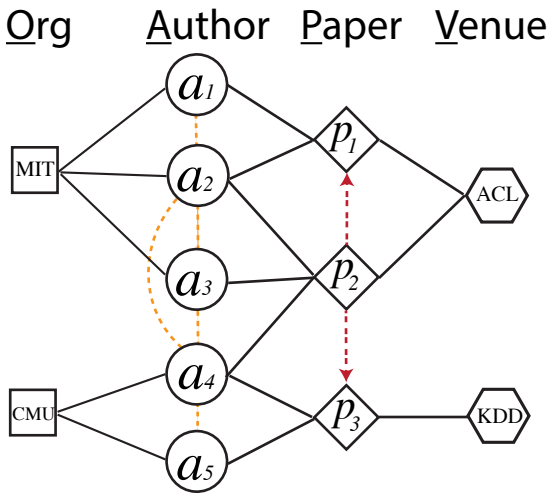
1. Sun and Han. Mining heterogeneous information networks: Principles and Methodologies. Morgan & Claypool Publishers, 2012.
2. Dong et al. metapath2vec: scalable representation learning for heterogeneous networks. In *ACM KDD* 2017. **The most cited paper in KDD'17 as of Aug 2018.**

Heterogeneous graph embedding



Goal: to generate paths that are able to capture both the semantic and structural correlations between different types of nodes, facilitating the transformation of heterogeneous network structures into skip-gram.

Heterogeneous graph embedding: Meta-Path-Based Random Walks



- Given a meta-path scheme (Example)

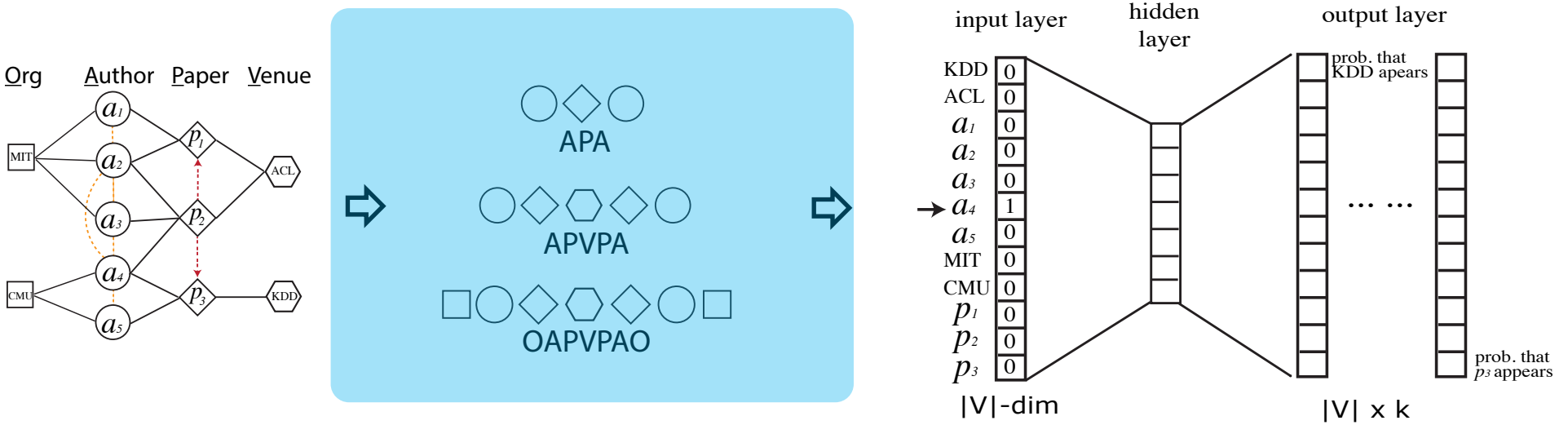
  OAPVPAO

- In a traditional random walk procedure, in the toy example, the next step of a walker on node a_4 transitioned from node O_{CMU} can be all types of nodes surrounding it— a_2, a_3, a_5, p_2, p_3 and O_{CMU}.

- Under the meta-path scheme 'OAPVPAO', for example, the walker is biased towards paper nodes (P) given its previous step on an organization node O_{CMU}(O), following the semantics of this meta-path.

Heterogeneous graph embedding

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

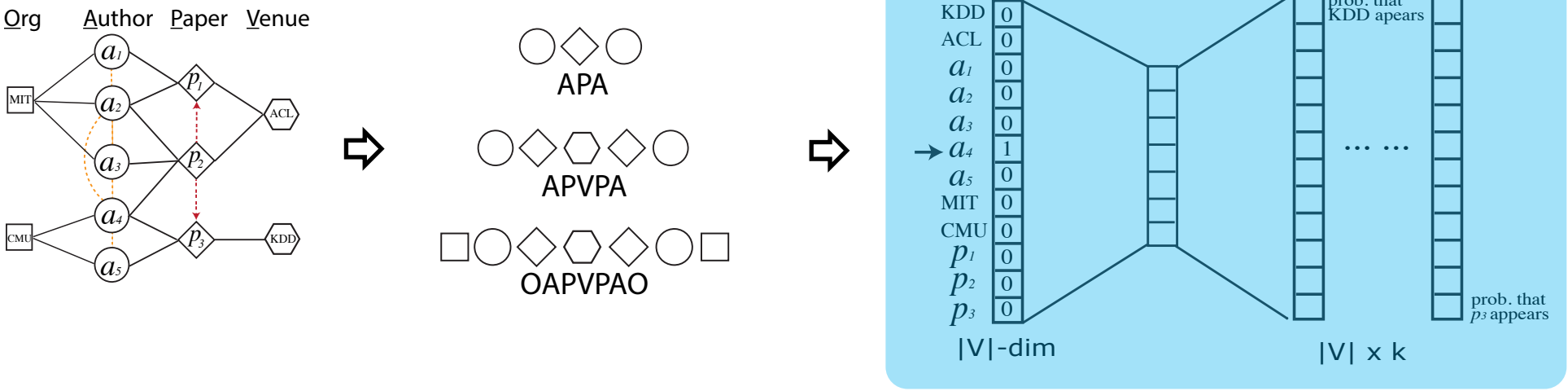


meta-path-based random walks

skip-gram

Heterogeneous graph embedding

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

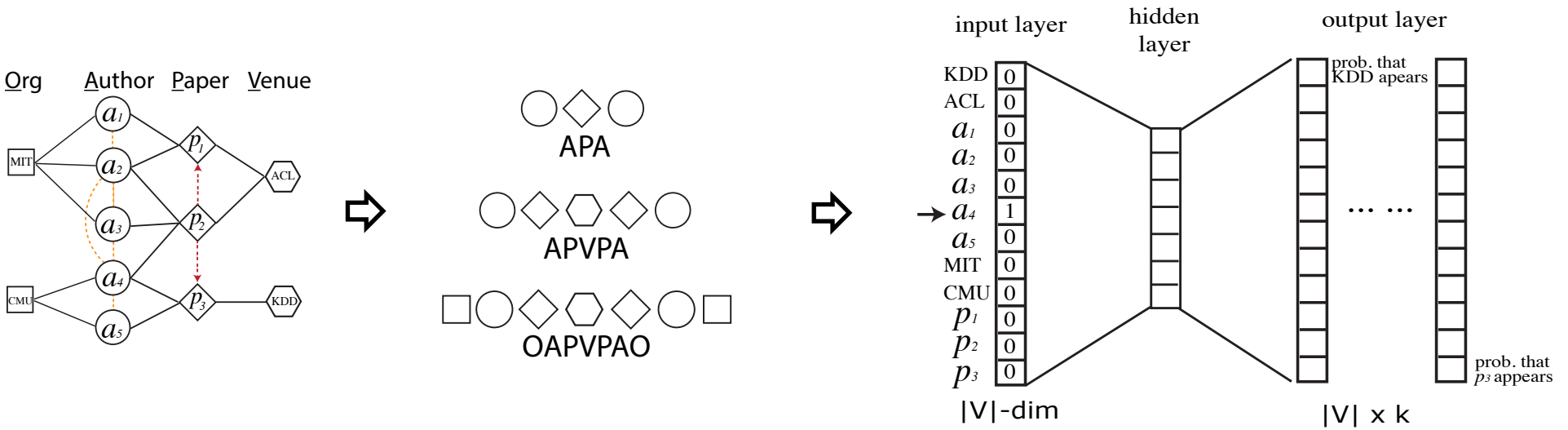


meta-path-based random walks

skip-gram

Heterogeneous graph embedding

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



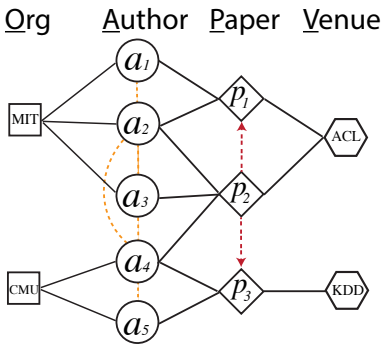
meta-path-based random walks

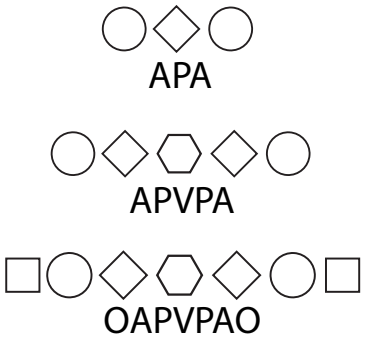
skip-gram

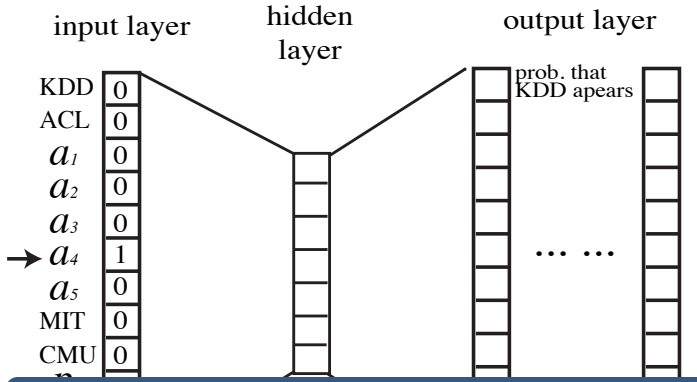
Heterogeneous graph embedding

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

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





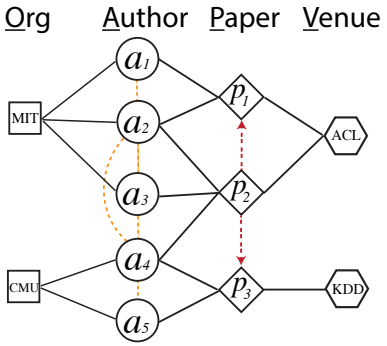


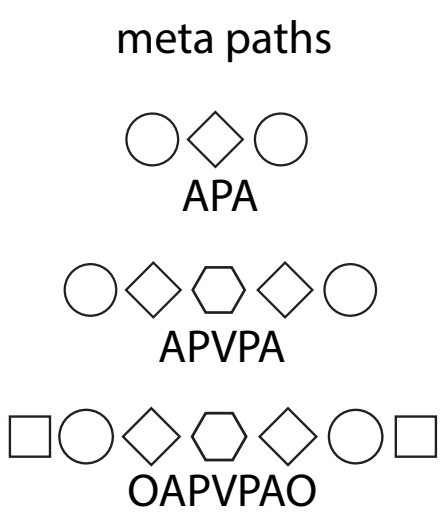
meta-path-based random walks

The potential issue of skip-gram for heterogeneous network embedding:

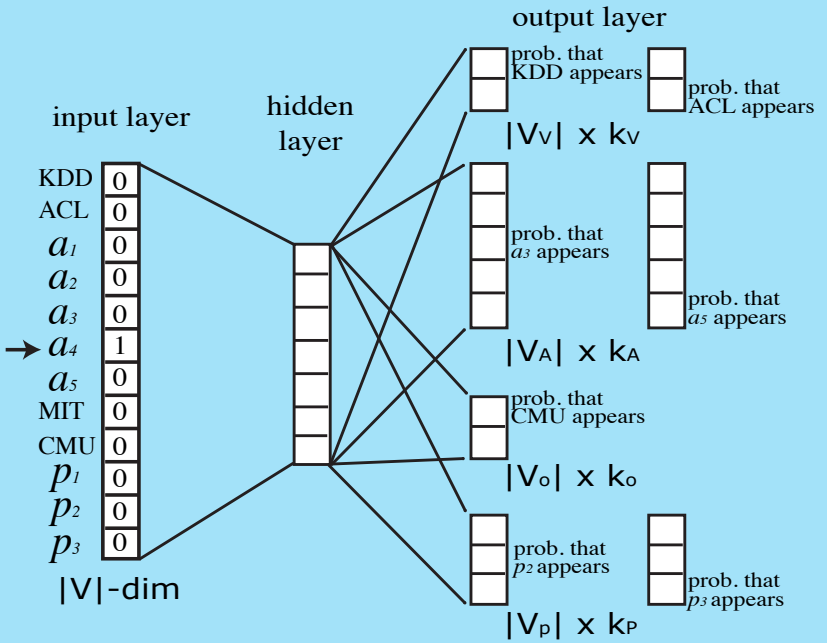
To predict the context node c_t (type t) given a node v, skip-gram assumes all types of nodes to appear in this context position

Heterogeneous graph embedding





meta-path-based random walks

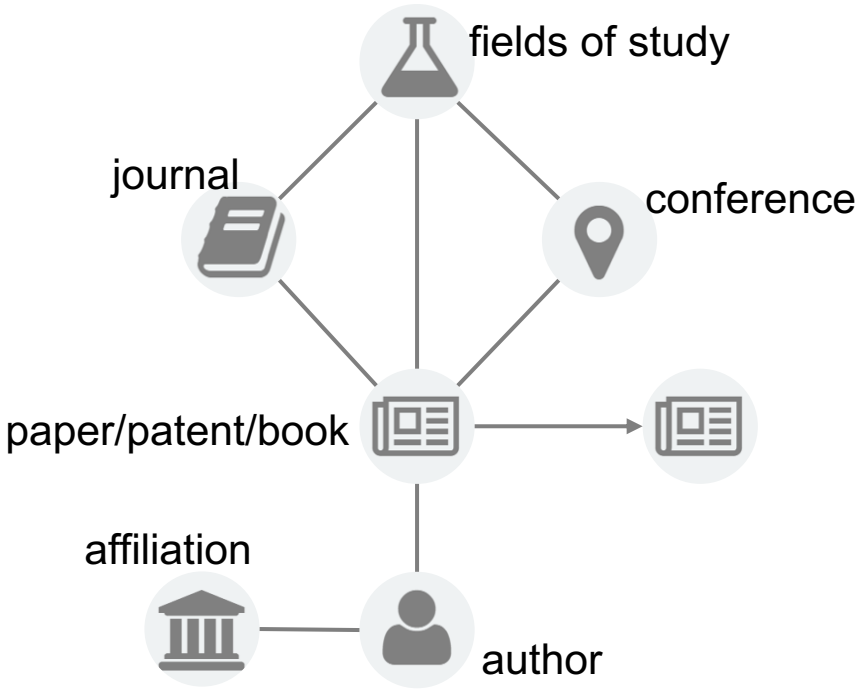


heterogeneous skip-gram

Heterogeneous graph embedding

1. Sun and Han. Mining heterogeneous information networks: Principles and Methodologies. Morgan & Claypool Publishers, 2012.
2. Dong et al. metapath2vec: scalable representation learning for heterogeneous networks. In *ACM KDD* 2017. **The most cited paper in KDD'17 as of Aug 2018.**

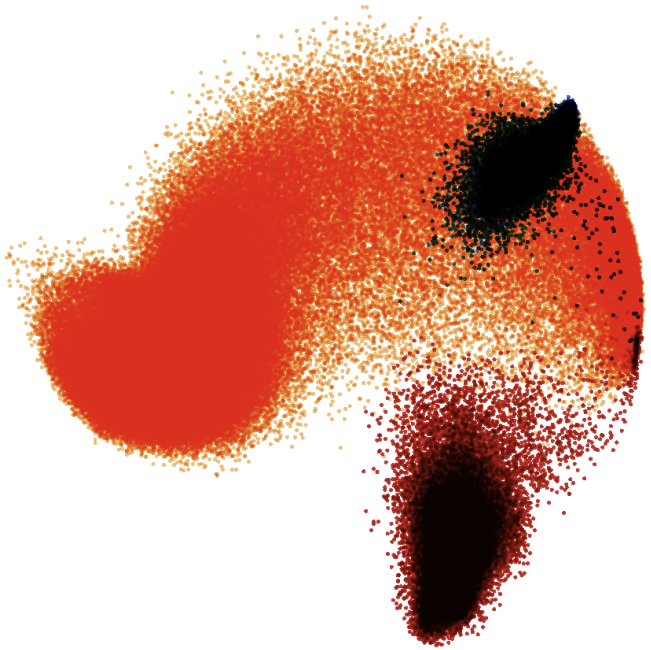
Application: Embedding Heterogeneous Academic Graph



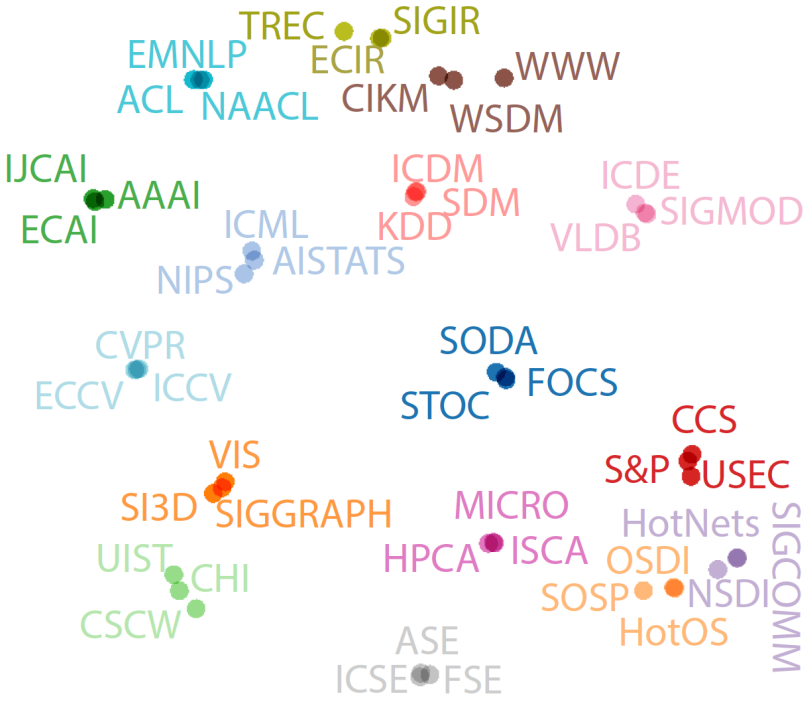
Microsoft Academic Graph
&
AMiner

metapath2vec

metapath2vec++



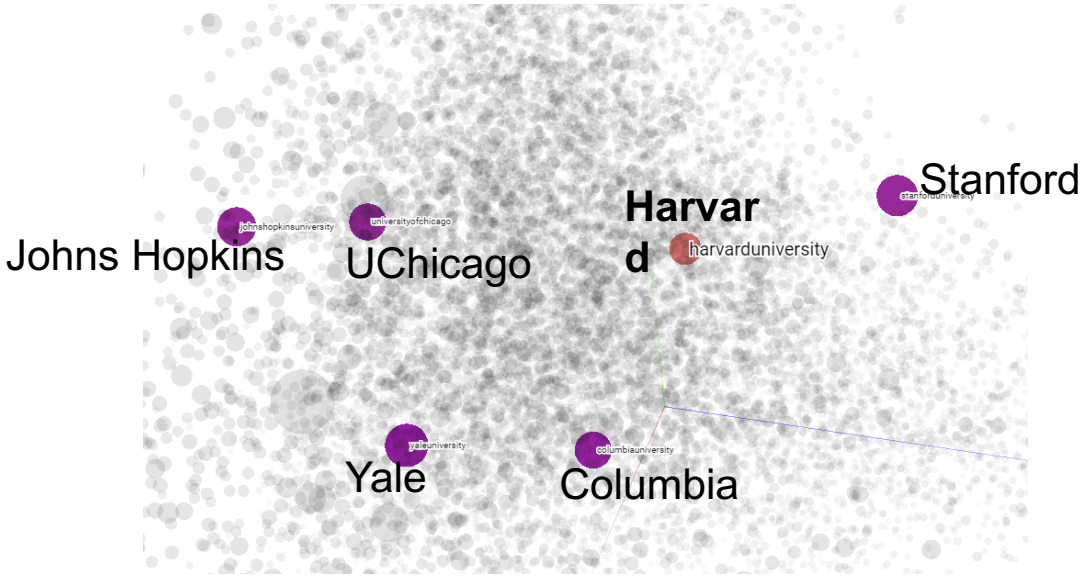
Application 2: Node Clustering

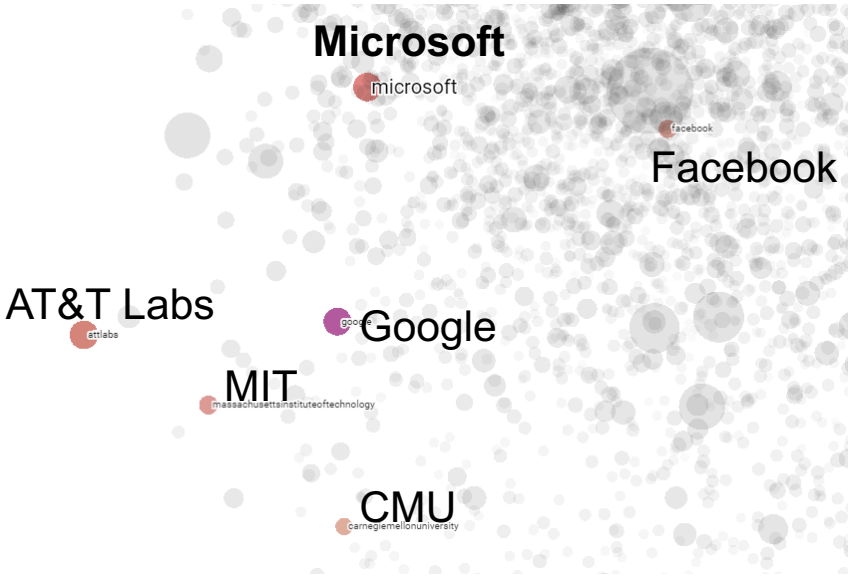


Application 3: Similarity Search (Conference)

Rank	ACL	NIPS	IJCAI	CVPR	FOCS	SOSP	ISCA	S&P	ICSE	SIGGRAPH	SIGCOMM	CHI	KDD	SIGMOD	SIGIR	WWW
0	ACL	NIPS	IJCAI	CVPR	FOCS	SOSP	ISCA	S&P	ICSE	SIGGRAPH	SIGCOMM	CHI	KDD	SIGMOD	SIGIR	WWW
1	EMNLP	ICML	AAAI	ECCV	STOC	TOCS	HPCA	CCS	TOSEM	TOG	CCR	CSCW	SDM	PVLDB	ECIR	WSDM
2	NAACL	AISTATS	AI	ICCV	SICOMP	OSDI	MICRO	NDSS	FSE	SI3D	HotNets	TOCHI	TKDD	ICDE	CIKM	CIKM
3	CL	JMLR	JAIR	IJCV	SODA	HotOS	ASPLOS	USENIX S	ASE	RT	NSDI	UIST	ICDM	DE Bull	IR J	TWEB
4	CoNLL	NC	ECAI	ACCV	A-R	SIGOPS E	PACT	ACSAC	ISSTA	CGF	CoNEXT	DIS	DMKD	VLDBJ	TREC	ICWSM
5	COLING	MLJ	KR	CVIU	TALG	ATC	ICS	JCS	E SE	NPAR	IMC	HCI	KDD E	EDBT	SIGIR F	HT
6	IJCNLP	COLT	AI Mag	BMVC	ICALP	NSDI	HiPEAC	ESORICS	MSR	Vis	TON	MobileHCI	WSDM	TODS	ICTIR	SIGIR
7	NLE	UAI	ICAPS	ICPR	ECCC	OSR	PPOPP	TISS	ESEM	JGT	INFOCOM	INTERACT	CIKM	CIDR	WSDM	KDD
8	ANLP	KDD	CI	EMMCVPR	TOC	ASPLOS	ICCD	ASIACCS	A SE	VisComp	PAM	GROUP	PKDD	SIGMOD R	TOIS	TIT
9	LREC	CVPR	AIPS	T on IP	JAlG	EuroSys	CGO	RAID	ICPC	GI	MobiCom	NordiCHI	ICML	WebDB	IPM	WISE
10	EACL	ECML	UAI	WACV	ITCS	SIGCOMM	ISLPED	CSFW	WICSA	CG	IPTPS	UbiComp	PAKDD	PODS	AIRS	WebSci

Application 3: Similarity Search (Institution)





Application 3: Related Venues

https://academic.microsoft.com/#/detail/137773608

Microsoft Academic | Nature

Nature

Description: Nature is a British multidisciplinary scie
the Science Edition of the 2010 Journal Citation Rep
of the few remaining academic journals that publish

Websites: www.nature.com, en.wikipedia.org,

Related Journals

- Science
- Proceedings of the National Academy of Sciences of the United States of America
- Nature Communications
- PLOS Biology
- Philosophical Transactions of the Royal Society B
- Current Biology
- BioEssays
- Nature Methods
- EMBO Reports
- PLOS ONE

Secure | https://academic.microsoft.com/#/detail/41523882

Microsoft Academic | ACM Transactions on Knowledge Discovery From Data

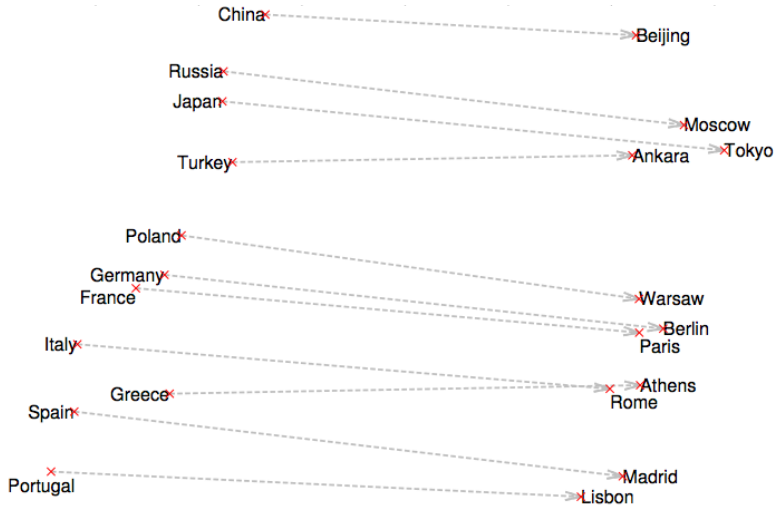
ACM Transactions on Knowledge Discovery From Data

Websites: bing.com

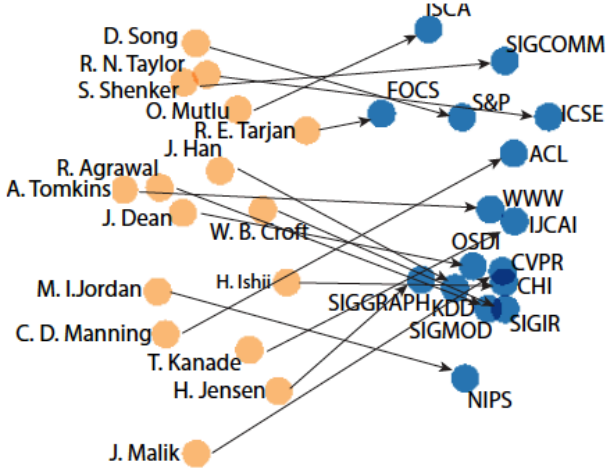
Related Journals

- Data Mining and Knowledge Discovery
- IEEE Transactions on Knowledge and Data Engineering
- Knowledge and Information Systems
- Sigkdd Explorations
- Proceedings of The Vldb Endowment
- World Wide Web
- ACM Transactions on Intelligent Systems and Technology
- Statistical Analysis and Data Mining
- arXiv: Learning
- The Vldb Journal

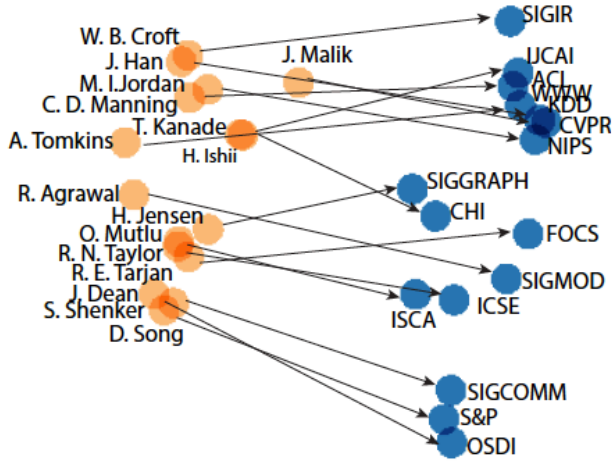
Visualization



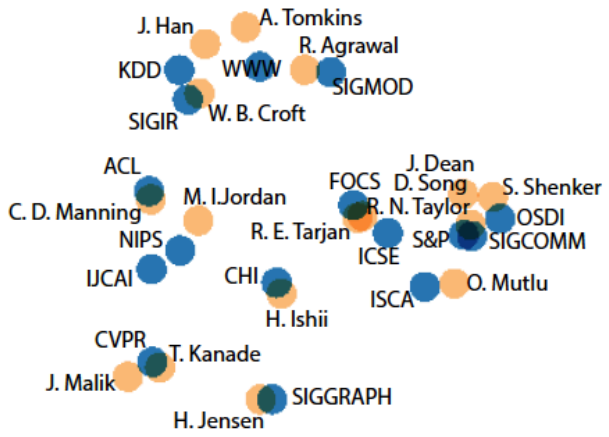
word2vec [Mikolov, 2013]



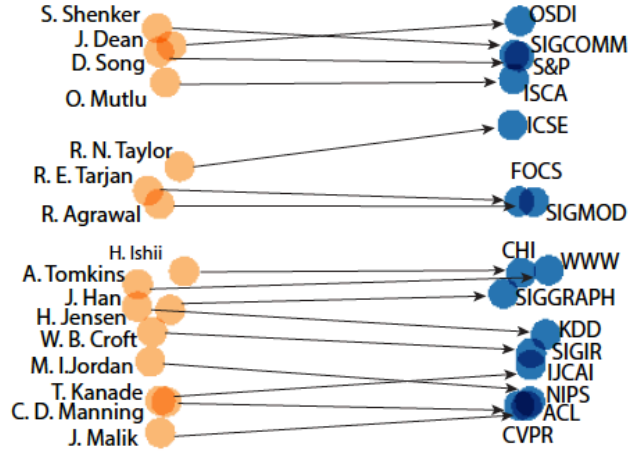
(a) DeepWalk/node2vec



(b) PTE



(c) *metapath2vec*



(d) *metapath2vec++*

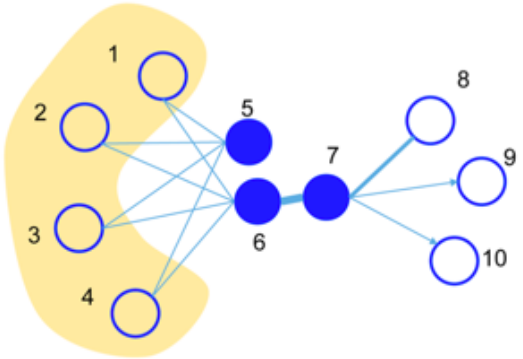
http://projector.tensorflow.org/

Random Walk Strategies

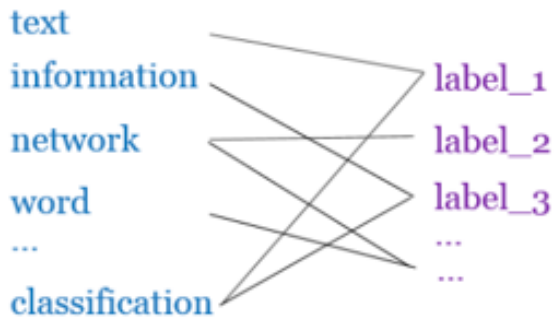
- Random Walk
  - DeepWalk
- Biased Random Walk
  - 2nd order Random Walk
    - node2vec
  - Metapath guided Random Walk
    - metapath2vec

LINE and PTE

- **LINE:** explicitly preserves both first-order and second-order proximities.

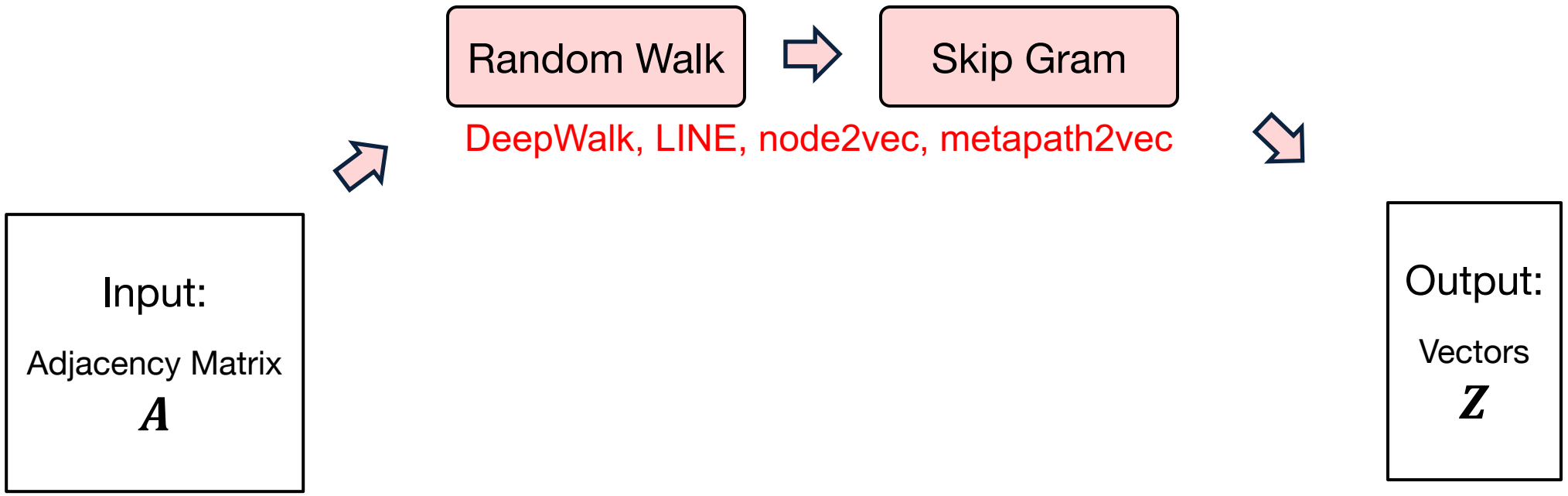


- **PTE:** learn heterogeneous text network embedding via a semi-supervised manner.



J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei. 2015. Line: Large-scale information network embedding. *WWW*. ***Most Cited Paper in WWW'15.***
J. Tang, M. Qu, and Q. Mei. 2015. Pte: Predictive text embedding through large-scale heterogeneous text networks. *KDD*, 1165–1174.

Network Embedding



What are the fundamentals

underlying skip-gram based network embedding models?

Network Embedding as Matrix Factorization: Unifying Deepwalk, LINE, PTE, and node2vec

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 Nov. 2020

Skip-gram based network embedding

- 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 $\boldsymbol{D} = diag(d_1, d_2, \cdots, d_n)$
  - Volume of G: $vol(G) = \sum_i \sum_j A_{ij}$
- Output: for each node, its k-dimension latent feature representation vector $\mathbf{Z}^{n \times k}$
  - Latent feature embedding matrix $\mathbf{Z} \in \mathbb{R}^{n \times k}$

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

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

$\boldsymbol{A}$ Adjacency matrix

$\boldsymbol{D}$ Degree matrix

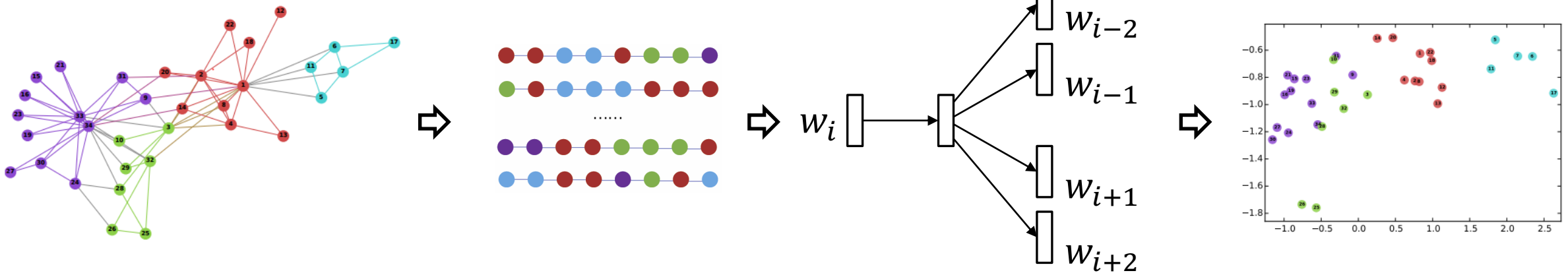
$vol(G)=\sum_i\sum_j A_{ij}$

b: #negative samples

T: context window size

1. Qiu et al. Network embedding as matrix factorization: unifying deepwalk, line, pte, and node2vec. In *WSDM'18*. **The most cited paper in WSDM'18 as of Sept 2020**

Understanding random walk + skip gram



Skip gram with negative sampling

Skip-gram with negative sampling (SGNS)

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

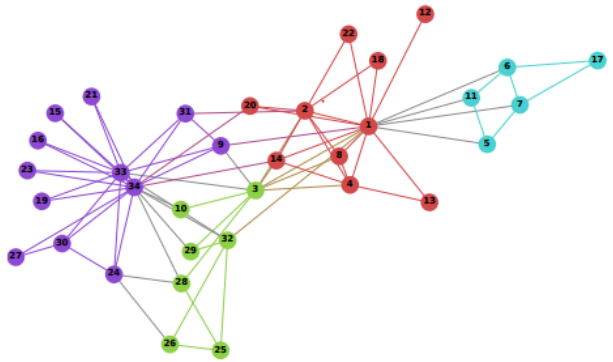
$$\mathcal{L} = \sum_{w} \sum_{c} (\#(w,c) \log g(x_w^T x_c) + \frac{b\#(w)\#(c)}{|\mathcal{D}|} \log g(-x_w^T x_c))$$

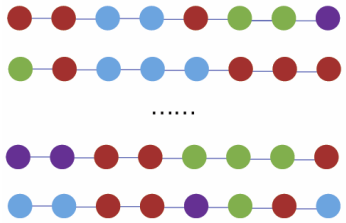
- For sufficiently large dimension *d*, the objective above is equivalent to factorizing the PMI matrix

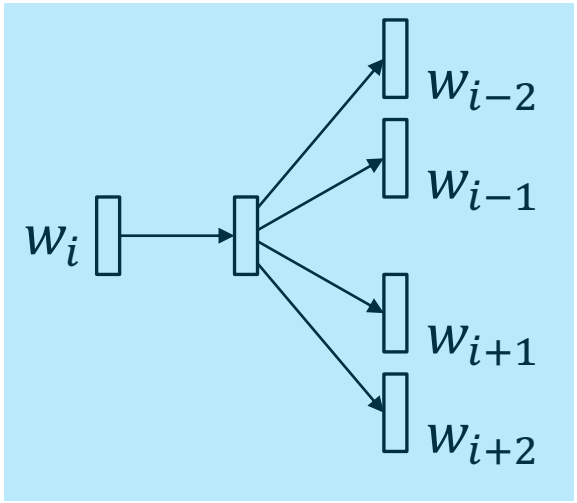
$$x_w^T x_c \approx \log \frac{\#(w,c)|\mathcal{D}|}{b\#(w)\#(c)}$$

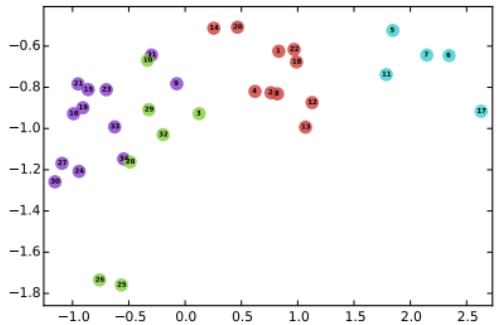
Levy and Goldberg. Neural word embeddings as implicit matrix factorization. In *NIPS 2014*

Understanding random walk + skip gram









$G = (V, E)$

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

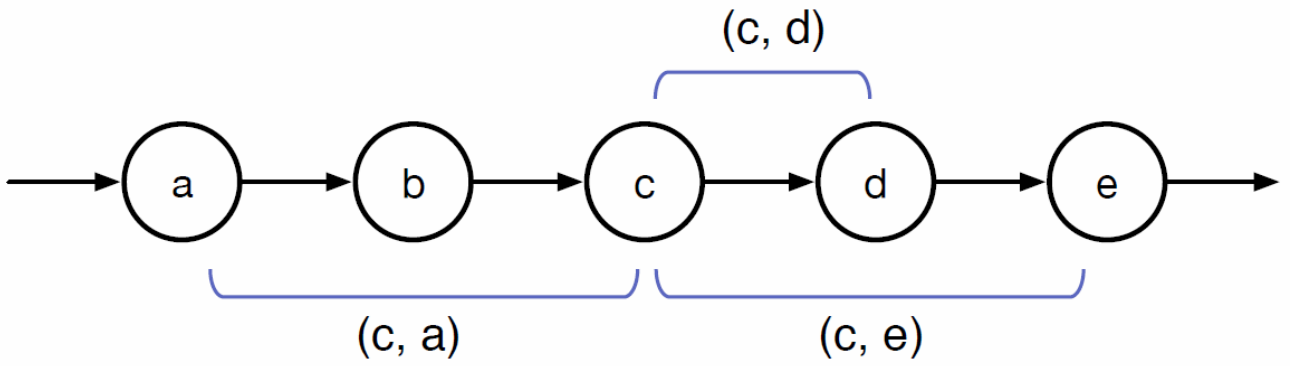
?

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

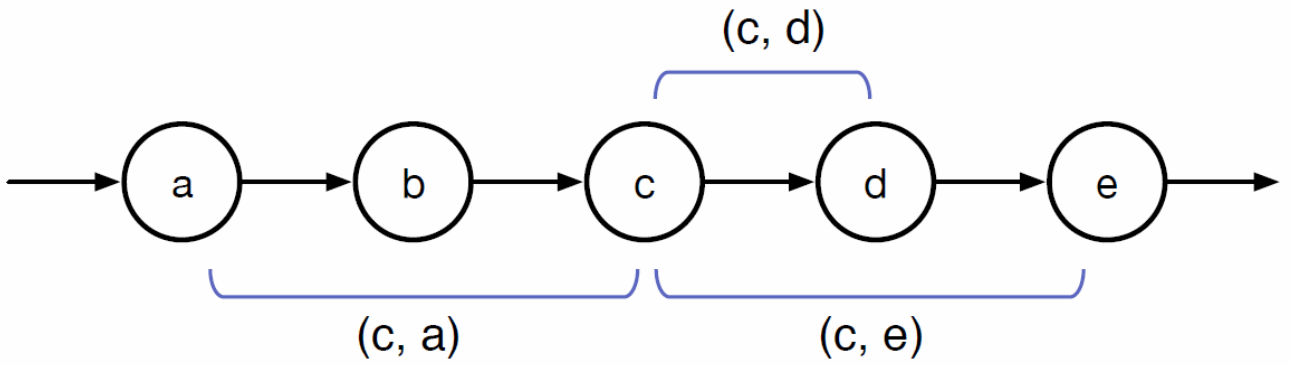
Understanding random walk + skip gram

```
for n = 1, 2, ..., N do
    Pick w_1^n according to a probability distribution P(w_1);
    Generate a vertex sequence (w_1^n, ··· , w_L^n) of length L by a
     random walk on network G;
    for j = 1, 2, ..., L − T do
        for r = 1, ..., T do
            Add vertex-context pair (w_j^n, w_{j+r}^n) to multiset D;
            Add vertex-context pair (w_{j+r}^n, w_j^n) to multiset D;
Run SGNS on D with b negative samples.
```

Understanding random walk + skip gram

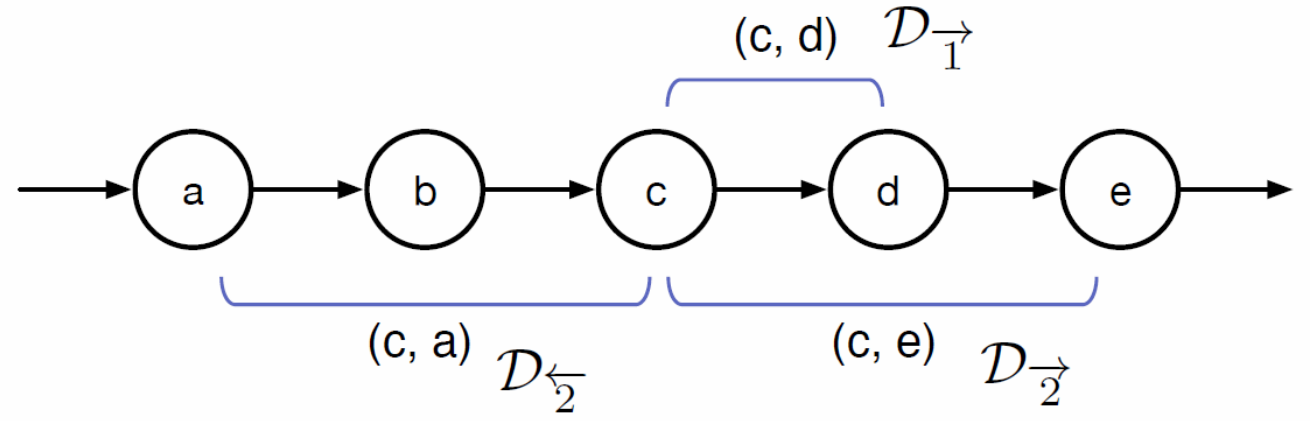


Understanding random walk + skip gram



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

Law of Large Numbers

Definition [edit]

A sequence $\{X_n\}$ of random variables **converges in probability** towards the random variable X if for all $\varepsilon > 0$

$$\lim_{n\to\infty} \Pr\left(|X_n - X| > \varepsilon\right) = 0.$$

The probability that X_n is outside the ball of radius ε centered at X goes to zero.

- $X_1, X_2, \ldots$ is an infinite sequence of i.i.d. Random Variables s.t. $E(X_1) = E(X_2) = \overline{X}_n \xrightarrow{P} \mu \quad \text{when } n \to \infty.$

- (S.N. Bernstein LLN) Let $X_1, X_2, \cdots$ be a sequence of random variables with finite expectation and variance, and covariances are s.t. $\mathrm{Cov}(X_i, X_j) \to 0$ as $|i - j| \to \infty$. Then the law of large numbers holds.

Convergence in Probability of Co-occurrence

- Consider a stationary random walk $w_1, w_2, \ldots$
- Co-occurrence can be written in the form of sample average:

$$\frac{\#(w,c)_{\overrightarrow{r}}}{|\mathcal{D}_{\overrightarrow{r}}|} = \frac{1}{L-T}\sum_{j=1}^{L-T} \mathbf{1}[w_j = w, w_{j+r} = c]$$

- Denote $X_i = \mathbf{1}[w_j = w, w_{j+r} = c]$
  - $\mathrm{Cov}(X_i, X_j) \to 0$ as $|i-j| \to \infty$
  - $E[X_i] = Prob(w_j = w, w_{j+r} = c) = \frac{d_w}{\mathrm{vol}(G)}(\boldsymbol{P}^r)_{w,c}$ $\qquad \boldsymbol{P} = \boldsymbol{D}^{-1}\boldsymbol{A}$
- Apply S.N. Bernstein LLN:

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

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

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

LEMMA 6.1. *(S.N. Bernstein Law of Large Numbers [34]) Let $Y_1, Y_2 \cdots$ be a sequence of random variables with finite expectation $\mathbb{E}[Y_j]$ and variance $\mathrm{Var}(Y_j) < K, j \geq 1$, and covariances are s.t. $\mathrm{Cov}(Y_i, Y_j) \to 0$ as $|i-j| \to \infty$. Then the law of large numbers (LLN) holds.*

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

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

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

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

Definition [edit]

A sequence $\{X_n\}$ of random variables **converges in probability** towards the random variable X if for all $\varepsilon > 0$

$$\lim_{n\to\infty}\Pr\left(|X_n - X| > \varepsilon\right) = 0.$$

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

The probability that X_n is outside the ball of radius ε centered at X goes to zero.

or every continuous function $g(\cdot)$, if , then also .

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 $L\to\infty$

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

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

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

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

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

Continuous mapping theorem
For every continuous function $g(\cdot,\cdot)$, if $X_n\xrightarrow{p}X$ and $Y_n\xrightarrow{p}Y$
then also $g(X_n,Y_n)\xrightarrow{p}g(X,Y)$

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 $L \to \infty$

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

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

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

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

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

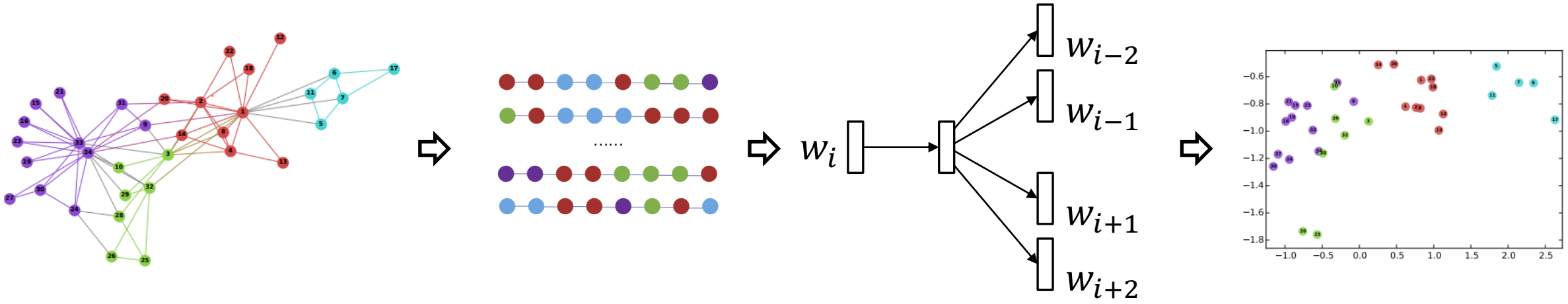
$$\frac{\#(w,c)\,|\mathcal{D}|}{\#(w)\cdot\#(c)} \xrightarrow{p} \frac{\mathrm{vol}(G)}{2T}\left(\frac{1}{d_c}\sum_{r=1}^{T}\left(\boldsymbol{P}^r\right)_{w,c}+\frac{1}{d_w}\sum_{r=1}^{T}\left(\boldsymbol{P}^r\right)_{c,w}\right)$$

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

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

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

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**

Understanding 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

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

J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei. 2015. Line: Large-scale information network embedding. *WWW*. ***Most Cited Paper in WWW'15.***

Understanding PTE

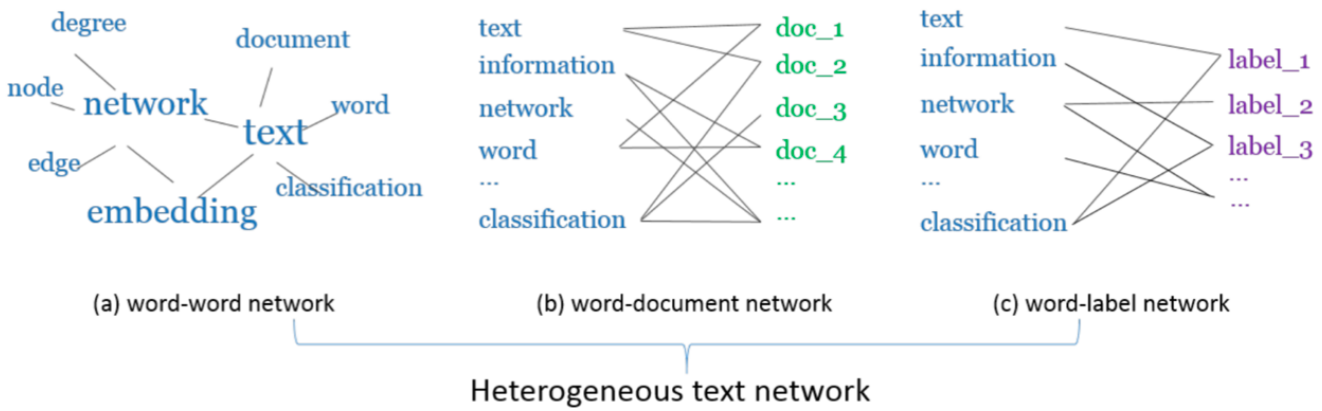


Figure 2: Heterogeneous Text Network.

- word-word network G_{ww}, $\boldsymbol{A}_{\text{ww}} \in \mathbb{R}^{\#\text{word} \times \#\text{word}}$.
- document-word network G_{dw}, $\boldsymbol{A}_{\text{dw}} \in \mathbb{R}^{\#\text{doc} \times \#\text{word}}$.
- label-word network G_{lw}, $\boldsymbol{A}_{\text{lw}} \in \mathbb{R}^{\#\text{label} \times \#\text{word}}$.

J. Tang, M. Qu, and Q. Mei. 2015. Pte: Predictive text embedding through large-scale heterogeneous text networks. *KDD*, 1165–1174.

Understanding PTE

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

- The matrix is of shape $(\#\mathsf{word}+\#\mathsf{doc}+\#\mathsf{label})\times\#\mathsf{word}$.
- b is the number of negative samples in training.
- $\{\alpha,\beta,\gamma\}$ are hyper-parameters to balance the weights of the three networks. In PTE, $\{\alpha,\beta,\gamma\}$ satisfy

$$\alpha\,\mathrm{vol}(G_{\mathsf{ww}})=\beta\,\mathrm{vol}(G_{\mathsf{dw}})=\gamma\,\mathrm{vol}(G_{\mathsf{lw}})$$

J. Tang, M. Qu, and Q. Mei. 2015. Pte: Predictive text embedding through large-scale heterogeneous text networks. *KDD*, 1165–1174.

Understanding node2vec

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

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

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.

A. Grover and J. Leskovec. 2016. node2vec: Scalable feature learning for networks. *KDD*, 855–864.

Understanding node2vec

Theorem

node2vec is asymptotically and implicitly factorizing a matrix whose entry at w-th row, c-th column is

$$\log\left(\frac{\frac{1}{2T}\sum_{r=1}^{T}\left(\sum_u \boldsymbol{X}_{w,u}\underline{\boldsymbol{P}}^r_{c,w,u}+\sum_u \boldsymbol{X}_{c,u}\underline{\boldsymbol{P}}^r_{w,c,u}\right)}{b\left(\sum_u \boldsymbol{X}_{w,u}\right)\left(\sum_u \boldsymbol{X}_{c,u}\right)}\right)$$

A. Grover and J. Leskovec. 2016. node2vec: Scalable feature learning for networks. *KDD*, 855–864.

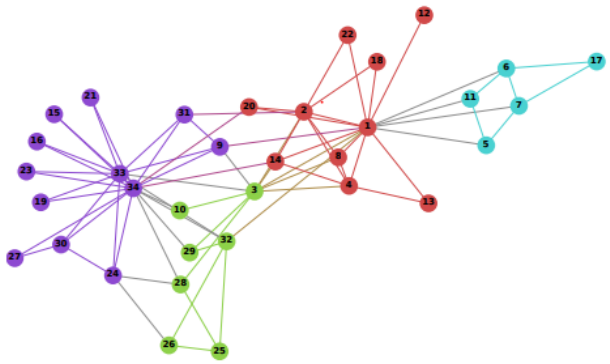
Unifying DeepWalk, LINE, PTE, & node2vec as Matrix Factorization

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

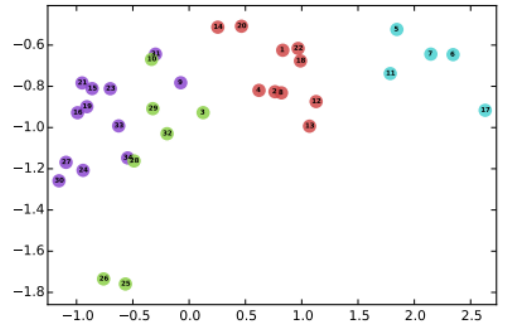
1. Qiu et al. Network embedding as matrix factorization: unifying deepwalk, line, pte, and node2vec. In *WSDM'18*. **The most cited paper in WSDM'18 as of May 2019**

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

NetMF

- DeepWalk is implicitly factorizing

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

- NetMF is explicitly factorizing

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

NetMF

- DeepWalk is asymptotically and implicitly factorizing

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

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

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

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

The NetMF Algorithm

- Factorize the DeepWalk matrix:

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

- For numerical reason, we use truncated logarithm — $\widetilde{\log}(x) = \log\left(\max(1, x)\right)$

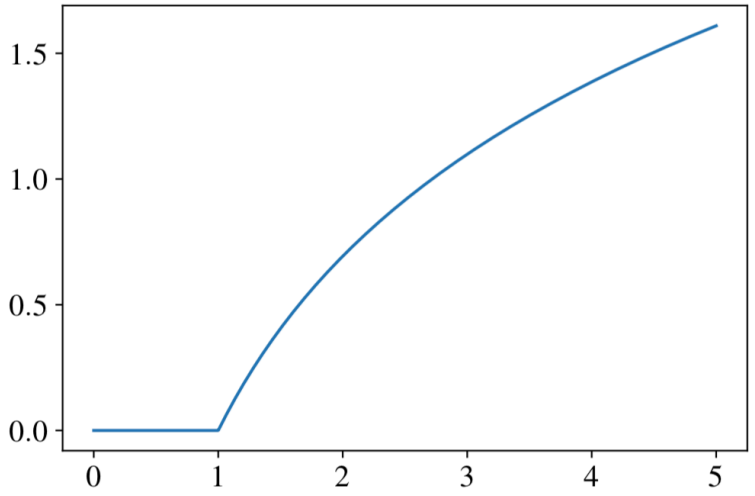


Figure 3: Truncated Logarithm

NetMF for a small window size T

Algorithm 2: NetMF for a Small Window Size T

Compute $\boldsymbol{P}^1, \cdots, \boldsymbol{P}^T$;

Compute $\boldsymbol{M} = \frac{\mathrm{vol}(G)}{bT} \left(\sum_{r=1}^{T} \boldsymbol{P}^r \right) \boldsymbol{D}^{-1}$;

Compute $\boldsymbol{M}' = \max(\boldsymbol{M}, 1)$;

Rank-d approximation by SVD: $\log \boldsymbol{M}' = \boldsymbol{U}_d \boldsymbol{\Sigma}_d \boldsymbol{V}_d^\top$;

return $\boldsymbol{U}_d \sqrt{\boldsymbol{\Sigma}_d}$ as network embedding.

NetMF for a LARGE window size T

Algorithm 2: 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.

Expensive

NetMF for a large window size *T*

- Factorize the DeepWalk matrix explicitly, e.g., using singular-value decomposition (SVD)

$$\log\left(\frac{\mathrm{vol}(G)}{b}\left(\frac{1}{T}\sum_{r=1}^{T}\left(\boldsymbol{D}^{-1}\boldsymbol{A}\right)^{r}\right)\boldsymbol{D}^{-1}\right)$$ This may be computationally challenging with a large T

- But we know the property of normalized graph Laplacian $\boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{A}\boldsymbol{D}^{-\frac{1}{2}} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^{\top}$ where $\boldsymbol{\Lambda} = diag(\lambda_1, \lambda_2, \cdots, \lambda\_n)$ and $\forall\lambda_i \in [-1, 1]$.

$$\left(\frac{1}{T}\sum_{r=1}^{T}\left(\boldsymbol{D}^{-1}\boldsymbol{A}\right)^{r}\right)\boldsymbol{D}^{-1} = \left(\boldsymbol{D}^{-1/2}\right)\left(\frac{1}{T}\sum_{r=1}^{T}\left(\boldsymbol{D}^{-1/2}\boldsymbol{A}\boldsymbol{D}^{-1/2}\right)^{r}\right)\left(\boldsymbol{D}^{-1/2}\right)$$

$$= \left(\boldsymbol{D}^{-1/2}\right)\left(\boldsymbol{U}\underbrace{\left(\frac{1}{T}\sum_{r=1}^{T}\boldsymbol{\Lambda}^{r}\right)}_{\text{A polynomial}}\boldsymbol{U}^{\top}\right)\left(\boldsymbol{D}^{-1/2}\right)$$

NetMF for a large window size T

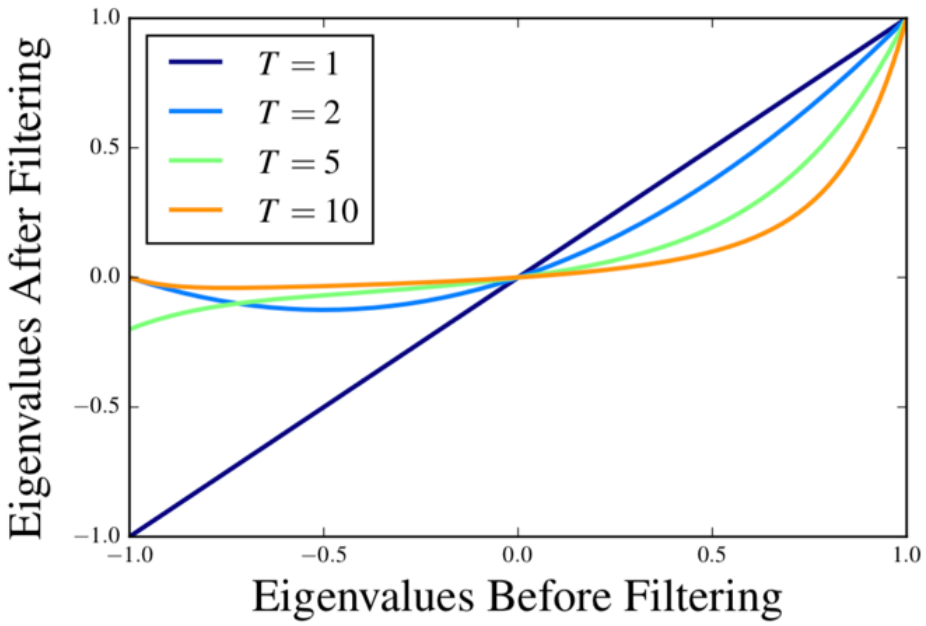


Figure 4: $f(\lambda) = \frac{1}{T}\sum_{r=1}^{T}\lambda^r$

- This polynomial implicitly filters out negative eigenvalues and small positive eigenvalues
- We can do it explicitly for efficiency

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 $\boldsymbol{U}_h\boldsymbol{\Lambda}_h\boldsymbol{U}_h^\top$

The Arnoldi algorithm [1] for significant time reduction

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

Error Bound for NetMF for a large window size *T*

- According to Frobenius norm's property

$$\left|\log M'_{i,j} - \log \hat{M}'_{i,j}\right| = \log \frac{\hat{M}'_{i,j}}{M'_{i,j}} = \log\left(1 + \frac{\hat{M}'_{i,j} - M'_{i,j}}{M'_{i,j}}\right)$$

$$\leq \frac{\hat{M}'_{i,j} - M'_{i,j}}{M'_{i,j}} \leq \hat{M}'_{i,j} - M'_{i,j} = \left|\hat{M}'_{i,j} - M'_{i,j}\right|$$

- and because $M'_{i,j}=\max(M_{i,j}, 1)>=1$, we have

$$\left|M'_{i,j} - \hat{M}'_{i,j}\right| = \left|\max(M_{i,j}, 1) - \max(\hat{M}_{i,j}, 1)\right| \leq \left|M_{i,j} - \hat{M}_{i,j}\right|$$

- Also because the property of NGL,

$$\sigma_s\left(\left(\frac{1}{T}\sum_{r=1}^{T} P^r\right) D^{-1}\right) \leq \frac{\left|\frac{1}{T}\sum_{r=1}^{T} \lambda_{p_s}^r\right|}{d_{q_1}} \Rightarrow \left\|M - \hat{M}\right\|_F \leq \frac{\mathrm{vol}(G)}{b d_{\min}} \sqrt{\sum_{j=k+1}^{n} \left|\frac{1}{T}\sum_{r=1}^{T} \lambda_j^r\right|^2}$$

Experimental Setup

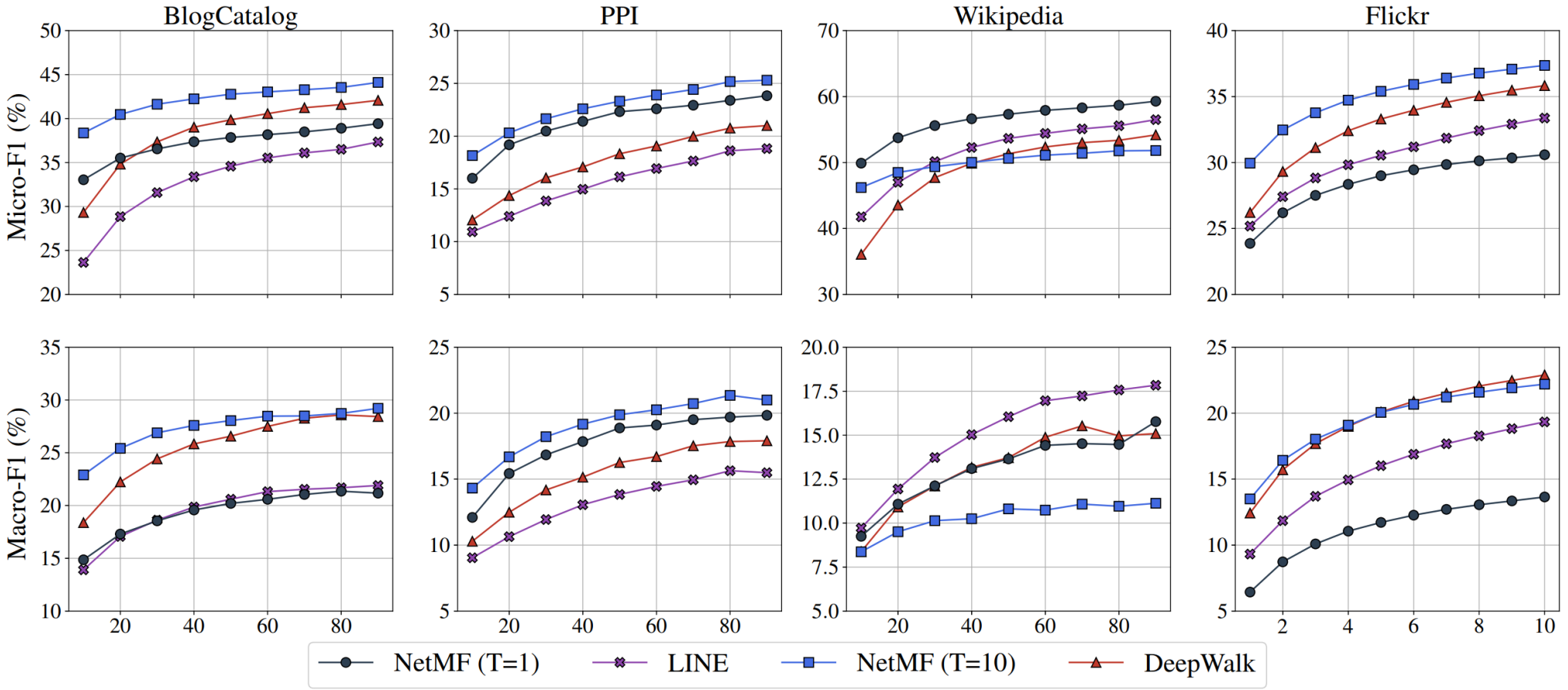
Label Classification:

- 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



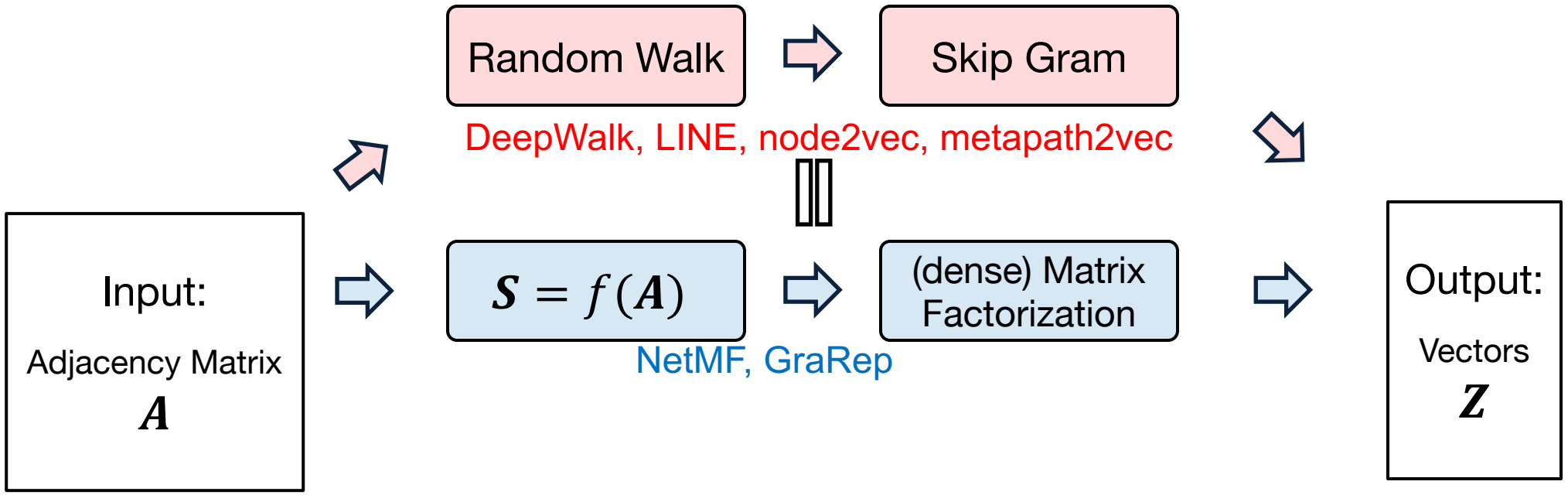
Predictive performance on varying the ratio of training data;
The *x*-axis represents the ratio of labeled data (%)

Network Embedding as Matrix Factorization

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

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 Aug 2018**

Network Embedding



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

A Matrix Chernoff Bound for Markov Chains and Its Application to Co-occurrence Matrices

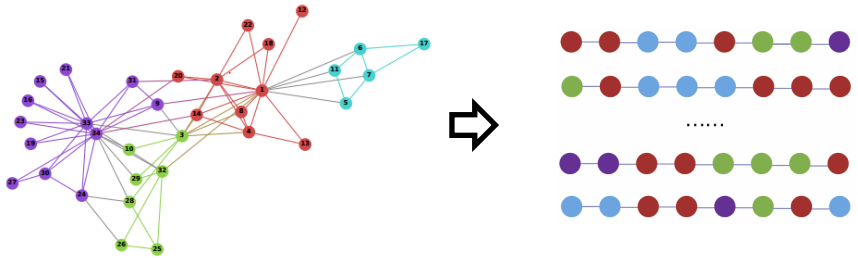
https://arxiv.org/abs/2008.02464 (NeurIPS 20)

Jiezhong Qiu, Chi Wang, Ben Liao, Richard Peng, Jie Tang

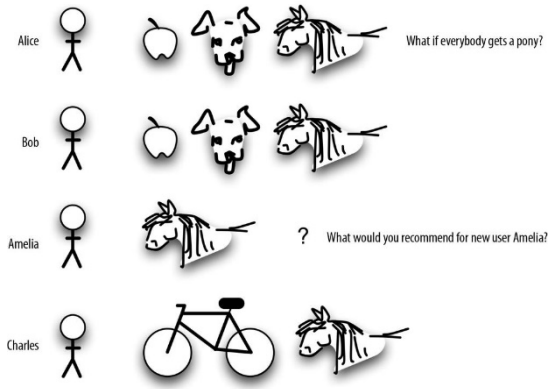
The Application to Co-occurrence Matrices

counts	I	like	enjoy	deep	learning	NLP	flying	.
I	0	2	1	0	0	0	0	0
like	2	0	0	1	0	1	0	0
enjoy	1	0	0	0	0	0	1	0
deep	0	1	0	0	1	0	0	0
learning	0	0	0	1	0	0	0	1
NLP	0	1	0	0	0	0	0	1
flying	0	0	1	0	0	0	0	1
.	0	0	0	0	1	1	1	0

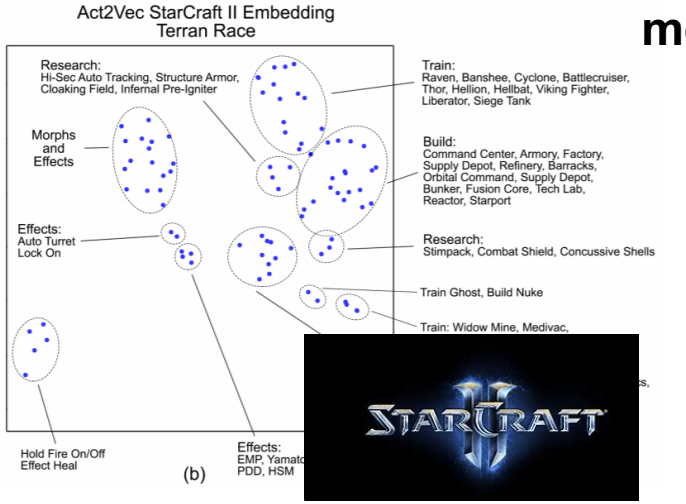
NLP
(LDA, Word2vec, Glove)



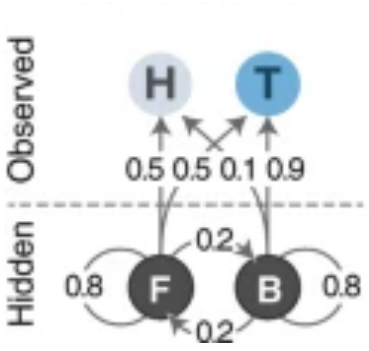
Graph Learning
(DeepWalk, node2vec, metapath2vec)



Recommendation System
(Pin2Vec, Item2vec)



Reinforcement Learning
(Act2Vec)

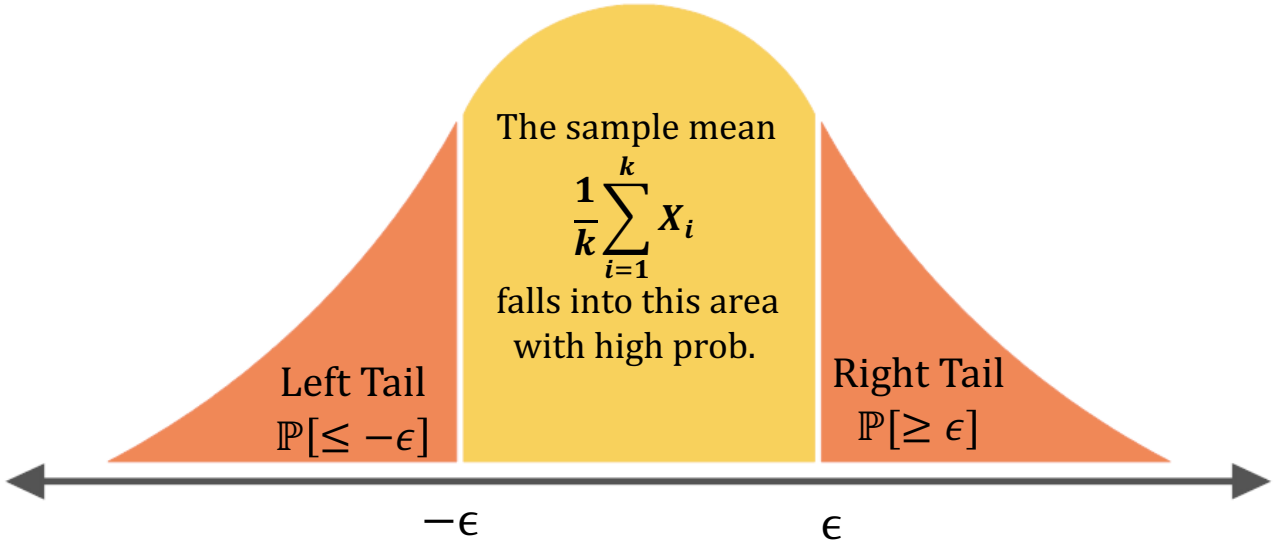


Hidden Markov Models
(Emission Co-occurrence)

Chernoff Bounds

Theorem (Chernoff Bound, 1952): If $X_1, X_2, \cdots, X_k$ are independent zero-mean scaler-valued random variables with $|X_i| \leq 1$. Then for $\epsilon \in (0, 1)$

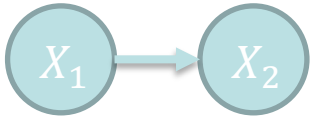
$$\mathbb{P}\left(\left|\frac{1}{k}\sum_{i=1}^{k} X_i\right| \geq \epsilon\right) \leq 2\exp(-k\epsilon^2/4)$$



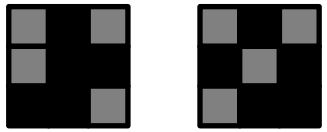
A Matrix Chernoff Bound for Markov Chains

~~Independence~~
Markov Dependence

$P(X_2|X_1)$



~~Scalar-valued Random Variables~~
Matrix-valued Random Variables



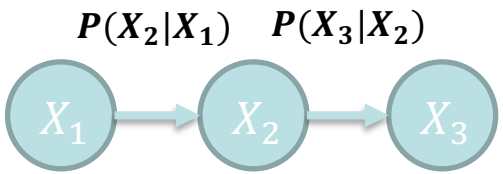
$f(X_1)$ $f(X_2)$

Sample Mean Matrix

$$\frac{1}{2}(f(X_1) + f(X_2))$$

A Matrix Chernoff Bound for Markov Chains

~~Independence~~
Markov Dependence



~~Scalar-valued Random Variables~~
Matrix-valued Random Variables

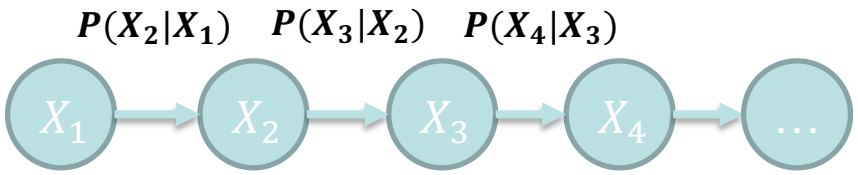
$f(X_1)$ $f(X_2)$ $f(X_3)$

Sample Mean Matrix

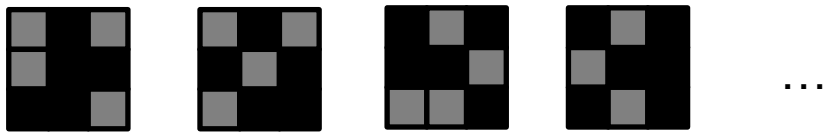
$$\frac{1}{3}(f(X_1) + f(X_2) + f(X_3))$$

A Matrix Chernoff Bound for Markov Chains

~~Independence~~
Markov Dependence



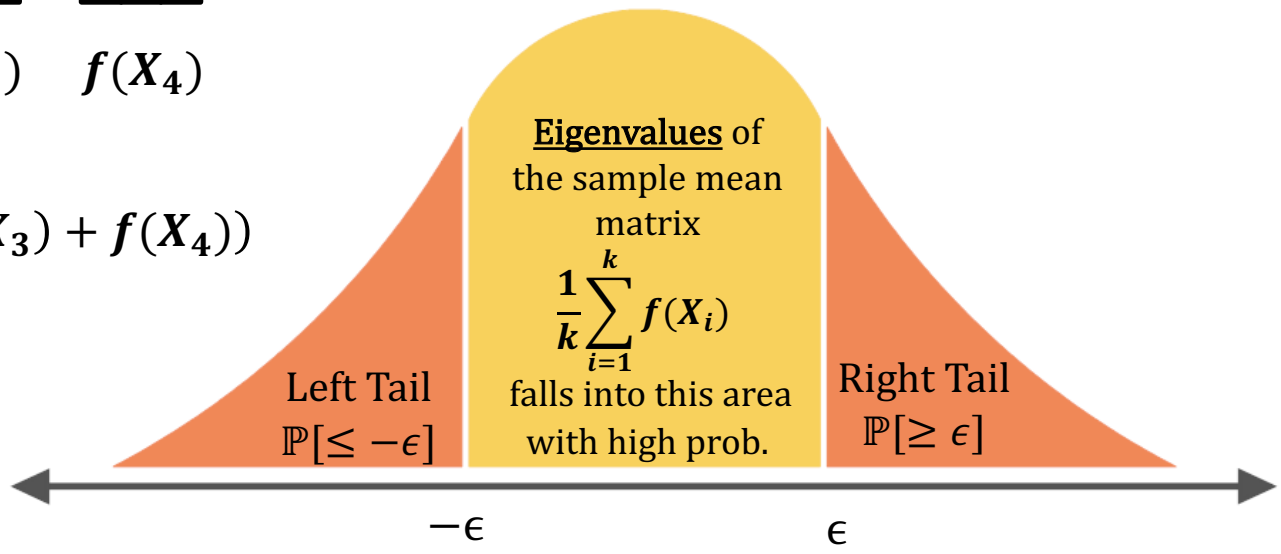
~~Scalar-valued Random Variables~~
Matrix-valued Random Variables



$f(X_1)$ $f(X_2)$ $f(X_3)$ $f(X_4)$

Sample Mean Matrix

$$\frac{1}{4}(f(X_1) + f(X_2) + f(X_3) + f(X_4))$$



A Matrix Chernoff Bound for Markov Chains

$$\mathbb{P}\left[\lambda_{\min}\left(\frac{1}{k}\sum_{i=1}^{k} f(X_i)\right) \le -\epsilon\right] \quad \text{and} \quad \mathbb{P}\left[\lambda_{\max}\left(\frac{1}{k}\sum_{i=1}^{k} f(X_i)\right) \ge \epsilon\right]$$

Comparison	Chernoff `52	Tropp`12	GLSS`18	Our Result
X	i.i.d scalars	i.i.d matrices	Stationary random walk on an undirected regular graph with spectral expansion λ	Non-stationary random walk on a regular Markov chain with spectral expansion λ
$f(X)$	X	X	$d \times d$ matrix	$d \times d$ matrix
tail prob.	$\exp(-\Omega(k\epsilon^2))$	$d\exp(-\Omega(k\epsilon^2))$	$d\exp(-\Omega(k(1-\lambda)\epsilon^2))$	$d\exp(-\Omega(k(1-\lambda)\epsilon^2))$

A Matrix Chernoff Bound for Markov Chains

Theorem: Let P be an regular Markov chain with state space $[N]$, stationary distribution π and spectral expansion λ. Let $f\colon [N] \to \mathbb{C}^{d\times d}$ be a matrix-valued function such that

1. $\forall X \in [N], f(X)$ is Hermitian and $\|f(X)\|_2 \leq 1$;
2. $\sum_{X\in[N]} \pi_X f(X) = 0$.

Let $(X_1, X_2, \cdots, X_k)$ denote a k-step random walk on P starting from an initial distribution ϕ. Then for $\epsilon \in (0,1)$:

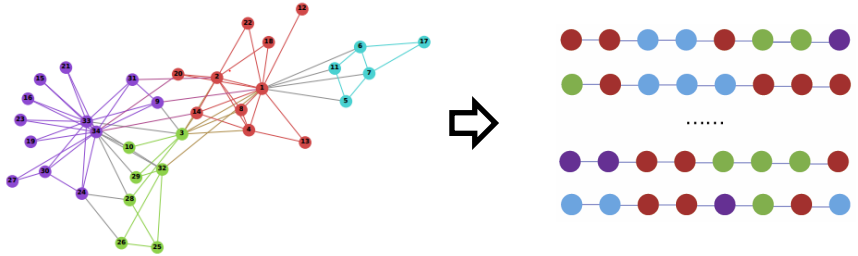
$$\mathbb{P}\left[\lambda_{\min}\left(\frac{1}{k}\sum_{i=1}^{k} f(X_i)\right) \leq -\epsilon\right] \leq \|\phi\|_\pi d^2 \exp(-k(1-\lambda)\epsilon^2/72)$$

$$\mathbb{P}\left[\lambda_{\max}\left(\frac{1}{k}\sum_{i=1}^{k} f(X_i)\right) \geq \epsilon\right] \leq \|\phi\|_\pi d^2 \exp(-k(1-\lambda)\epsilon^2/72)$$

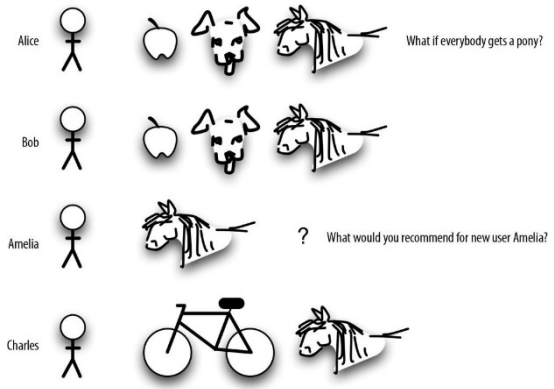
The Application to Co-occurrence Matrices

counts	I	like	enjoy	deep	learning	NLP	flying	.
I	0	2	1	0	0	0	0	0
like	2	0	0	1	0	1	0	0
enjoy	1	0	0	0	0	0	1	0
deep	0	1	0	0	1	0	0	0
learning	0	0	0	1	0	0	0	1
NLP	0	1	0	0	0	0	0	1
flying	0	0	1	0	0	0	0	1
.	0	0	0	0	1	1	1	0

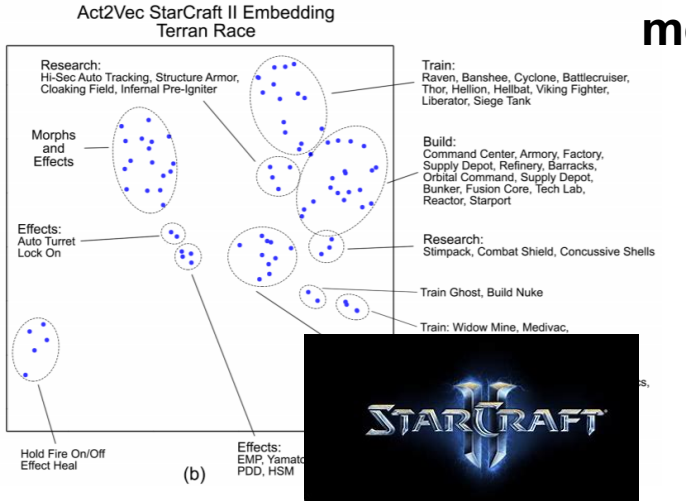
NLP
(LDA, Word2vec, Glove)



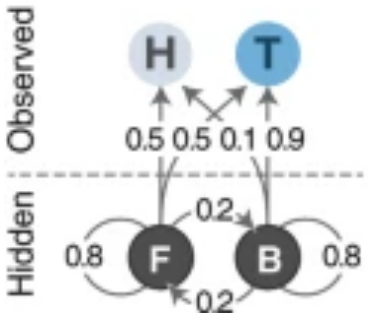
Graph Representation Learning
(DeepWalk, node2vec, metapath2vec)



Recommendation System
(Pin2Vec, Item2vec)

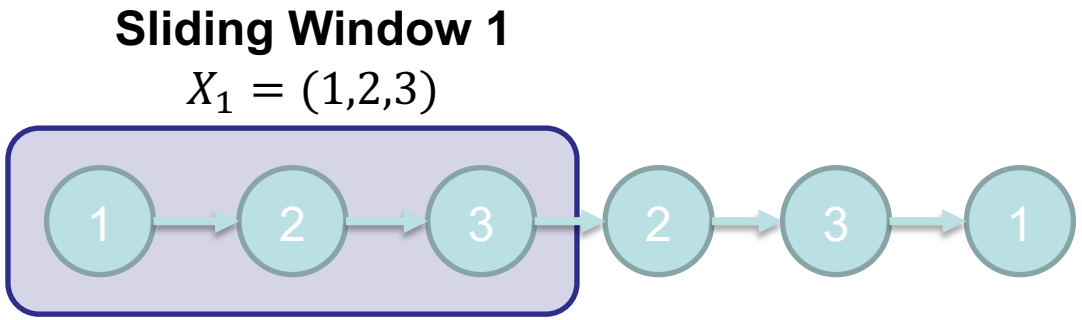


Reinforcement Learning
(Act2Vec)



Hidden Markov Models
(Emission Co-occurrence)

Co-occurrence Matrix of Sequential Data

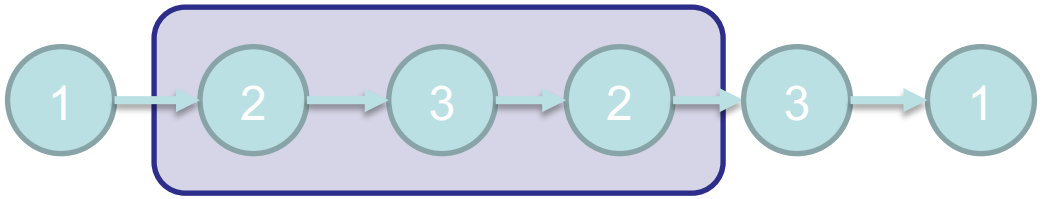


$$\boldsymbol{C} = \frac{1}{4}\begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$

Co-occurrence Matrix of Sequential Data

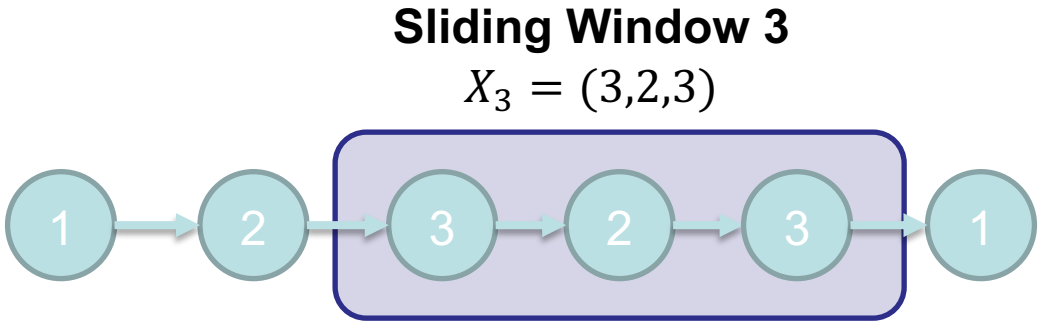
Sliding Window 2

$X_2 = (2,3,2)$



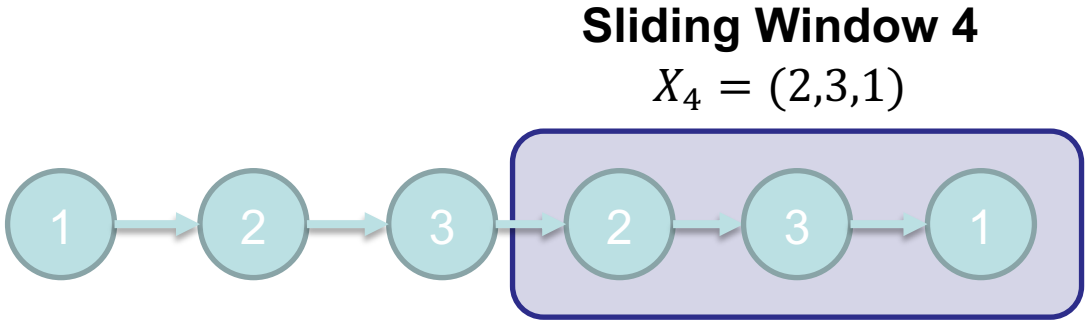
$$\boldsymbol{C} = \frac{1}{2}\left[\frac{1}{4}\begin{pmatrix}0 & 1 & 1\\ 1 & 0 & 0\\ 1 & 0 & 0\end{pmatrix} + \frac{1}{4}\begin{pmatrix}0 & 0 & 0\\ 0 & 2 & 1\\ 0 & 1 & 0\end{pmatrix}\right]$$

Co-occurrence Matrix of Sequential Data



$$\boldsymbol{C} = \frac{1}{3}\left[\frac{1}{4}\begin{pmatrix}0 & 1 & 1\\ 1 & 0 & 0\\ 1 & 0 & 0\end{pmatrix} + \frac{1}{4}\begin{pmatrix}0 & 0 & 0\\ 0 & 2 & 1\\ 0 & 1 & 0\end{pmatrix} + \frac{1}{4}\begin{pmatrix}0 & 0 & 0\\ 0 & 0 & 1\\ 0 & 1 & 2\end{pmatrix}\right]$$

Markov chain Matrix Chernoff Bound!



$$\boldsymbol{C} = \frac{1}{4}\left[\frac{1}{4}\begin{pmatrix}0 & 1 & 1\\ 1 & 0 & 0\\ 1 & 0 & 0\end{pmatrix} + \frac{1}{4}\begin{pmatrix}0 & 0 & 0\\ 0 & 2 & 1\\ 0 & 1 & 0\end{pmatrix} + \frac{1}{4}\begin{pmatrix}0 & 0 & 0\\ 0 & 0 & 1\\ 0 & 1 & 2\end{pmatrix} + \frac{1}{4}\begin{pmatrix}0 & 1 & 0\\ 1 & 0 & 1\\ 0 & 1 & 0\end{pmatrix}\right]$$

$$= \frac{1}{4}(\boldsymbol{f}(\boldsymbol{X_1}) + \boldsymbol{f}(\boldsymbol{X_2}) + \boldsymbol{f}(\boldsymbol{X_3}) + \boldsymbol{f}(\boldsymbol{X_4}))$$

Observation 1:

Let $\boldsymbol{X_1}, \boldsymbol{X_2}, \cdots, \boldsymbol{X_{L-T}}$ be the sequence of sliding windows, and $\boldsymbol{f}$ maps a sliding window to the co-occurrence matrix within this window. The co-occurrence matrix C can be written as the **sample mean** of $\boldsymbol{f}(\boldsymbol{X_1}), \boldsymbol{f}(\boldsymbol{X_2}), \cdots, \boldsymbol{f}(\boldsymbol{X_{L-T}})$:

$$\boldsymbol{C} = \frac{1}{L-T}\sum_{k=1}^{L-T} \boldsymbol{f}(\boldsymbol{X_k})$$

Observation 2: If the input sequence $v_1, v_2, \cdots$ is a Markov Chain, then $X_1, X_2, \cdots$ is a Markov Chain, too.

Convergence Rate of Co-occurrence Matrices

- **The co-occurrence matrix:**

$$\boldsymbol{C = \frac{1}{L-T}\sum_{k=1}^{L-T} f(X_k)}$$

- **The asymptotic expectation of C (denote $\Pi = \mathbf{diag}(\pi)$):**

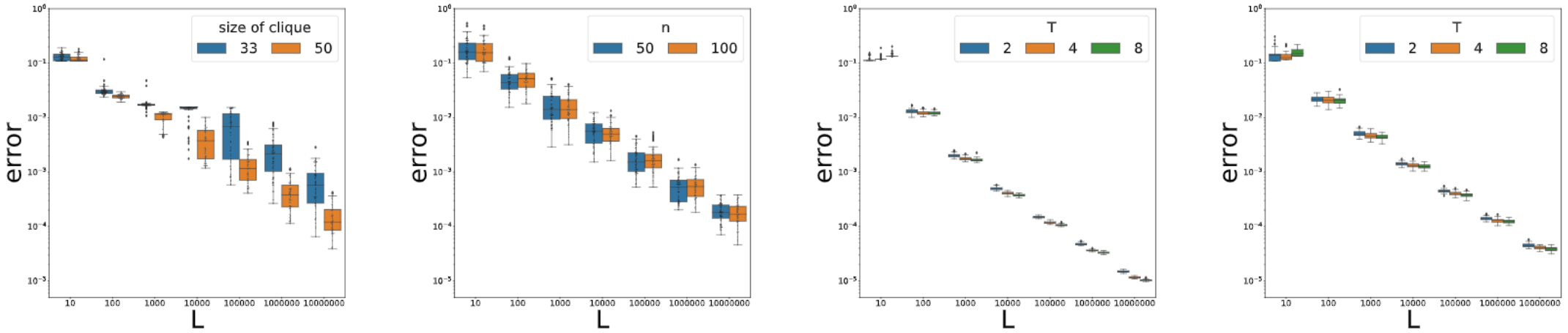
$$\mathbb{AE}[\boldsymbol{C}] = \lim_{L\to+\infty} \mathbb{E}[\boldsymbol{C}] = \sum_{r=1}^{T} \frac{1}{2T}(\boldsymbol{\Pi P^r} + (\boldsymbol{\Pi P^r})^\top)$$

Theorem: Let P be a regular Markov chain with state space $[n]$, stationary distribution π and mixing time τ. Let $(v_1, \cdots, v_L)$ be a L-step random walk on P starting from a distribution ϕ. Given $\epsilon \in (0, 1)$, the probability that the co-occurrence matrix C deviates from its asymptotic expectation $\mathbb{AE}[C]$ (in 2-norm) is bounded by:

$$\mathbb{P}(\|\boldsymbol{C} - \mathbb{AE}[\boldsymbol{C}]\|_2 \geq \epsilon) \leq 2(\tau + T)\|\phi\|_\pi n^2 \exp\left(-\frac{\epsilon^2(L-T)}{576(\tau+T)}\right)$$

Roughly, one needs $L = O(\tau(\log n + \log\tau)/\epsilon^2)$ samples to guarantee good estimation to the co-occurrence matrix.

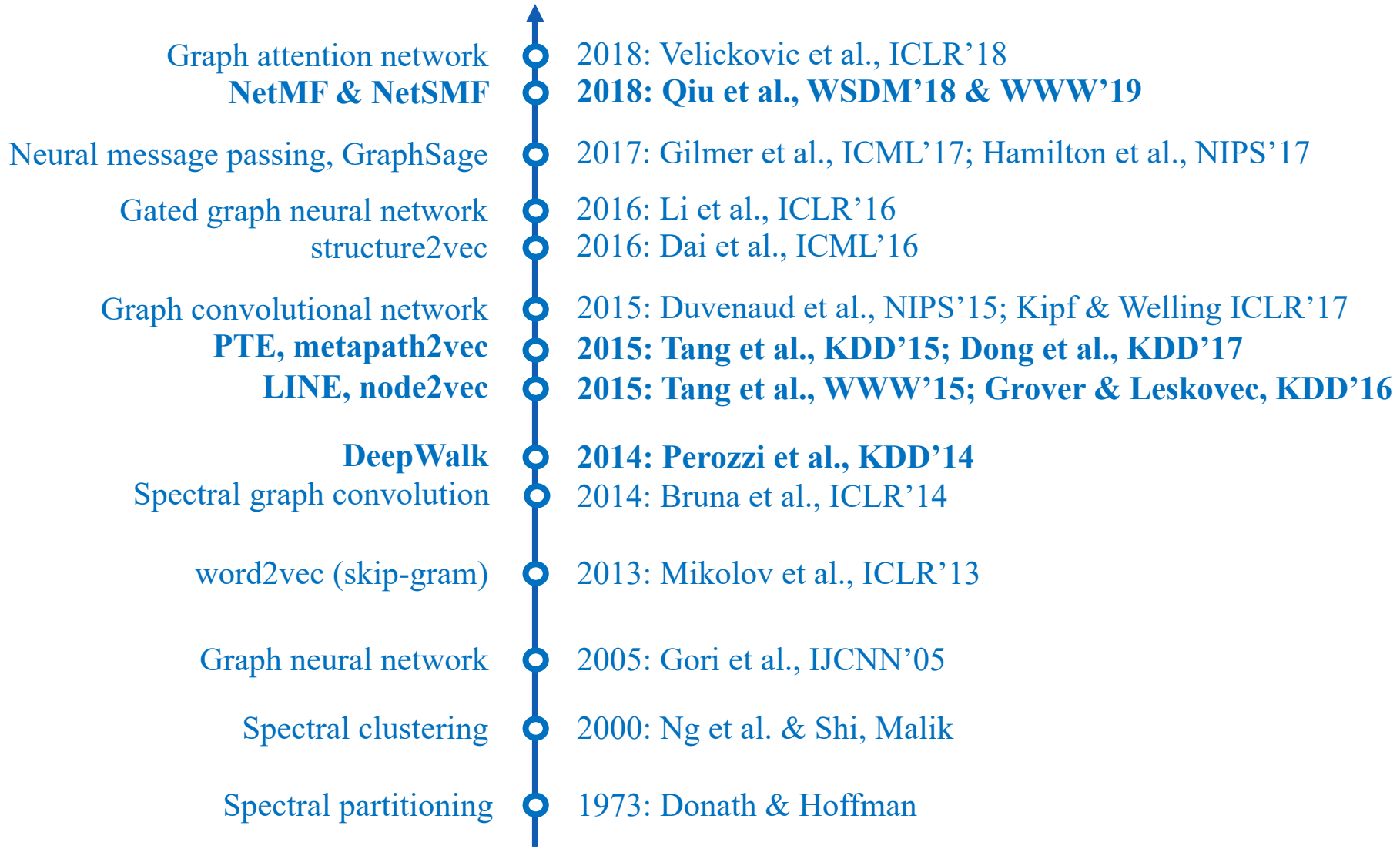
Numerical Experiments



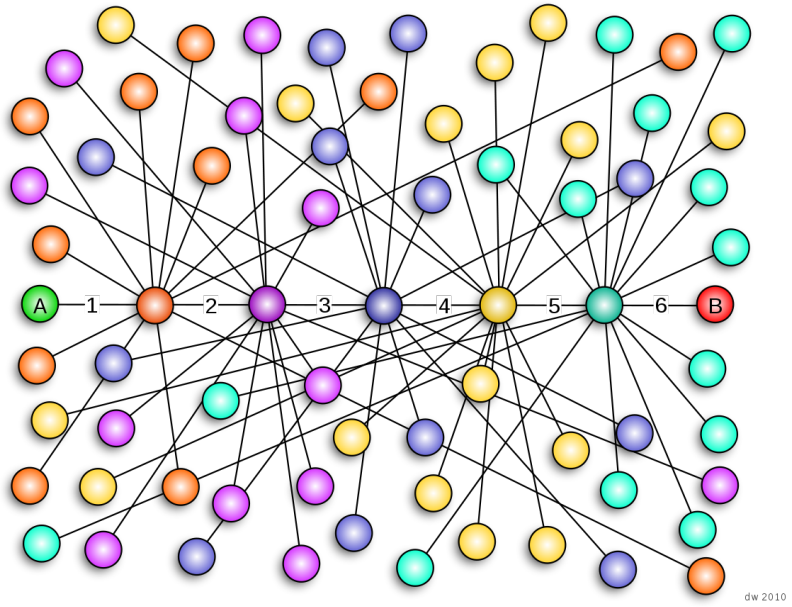
(a) Barbell Graph (b) Winning Streak Chain (c) BlogCatalog (d) Random Graph

Figure 1: The convergence rate of co-occurrence matrices on Barbell graph, winning streak chain, BlogCatalog graph , and random graph (in log-log scale). The x-axis is the trajectory length L and the y-axis is the approximation error $\|\boldsymbol{C} - \mathbb{AE}[\boldsymbol{C}]\|_2$. Each experiment contains 64 trials, and the error bar is presented.

Network Representation Learning / Network Embedding



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

Recall 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 $\boldsymbol{U}_h\boldsymbol{\Lambda}_h\boldsymbol{U}_h^\top$

The Arnoldi algorithm [1] for significant time reduction

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

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**

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**

NetSMF: Network Embedding as Sparse Matrix Factorization

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

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

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

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

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

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

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

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

NetSMF --- Sparse

Input : A social network $G = (V, E, \boldsymbol{A})$ which we want to learn network embedding; The number of non-zeros M in the sparsifier; The dimension of embedding d.

Output: An embedding matrix of size $n \times d$, each row corresponding to a vertex.

$\widetilde{G} \leftarrow (V, \emptyset, \widetilde{\boldsymbol{A}} = \boldsymbol{0})$

/* Create an empty network with $E = \emptyset$ and $\widetilde{\boldsymbol{A}} = 0$. */

for $i \leftarrow 1$ **to** M **do**

  Uniformly pick an edge $e = (u, v) \in E$

  Uniformly pick an integer $r \in [1 : T]$

  $u', v', Z \leftarrow$ PathSampling(*e*, *r*)

  Add an edge $\left(u', v', \frac{2rm}{MZ}\right)$ to $\widetilde{G}$

  /* Parallel edges will be merged into one edge, with their weights summed up together. */

end

Compute $\widetilde{\boldsymbol{L}}$ to be the unnormalized graph Laplacian of $\widetilde{G}$

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

$\boldsymbol{U}_d, \boldsymbol{\Sigma}_d, \boldsymbol{V}_d \leftarrow$ RandomizedSVD$\left(\text{trunc_log}^{\circ}\left(\frac{\text{vol}(G)}{b}\widetilde{\boldsymbol{M}}\right), d\right)$

return $\boldsymbol{U}_d\sqrt{\boldsymbol{\Sigma}_d}$ as network embeddings

Algorithm 5: PathSampling algorithm as described in [CCL+15].

Procedure `PathSampling`$(e = (u, v),\ r)$

Uniformly pick an integer $k \in [1 : r]$

Perform $(k-1)$-step random walk from u to u_0

Perform $(r-k)$-step random walk from v to u_r

Keep track of $Z(\boldsymbol{p}) = \sum_{i=1}^{r} \frac{2}{\boldsymbol{A}_{u_{i-1},u_i}}$ along the length-r path $\boldsymbol{p}$ between u_0 and u_r

return $u_0, u_r, Z(\boldsymbol{p})$

- 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.

	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

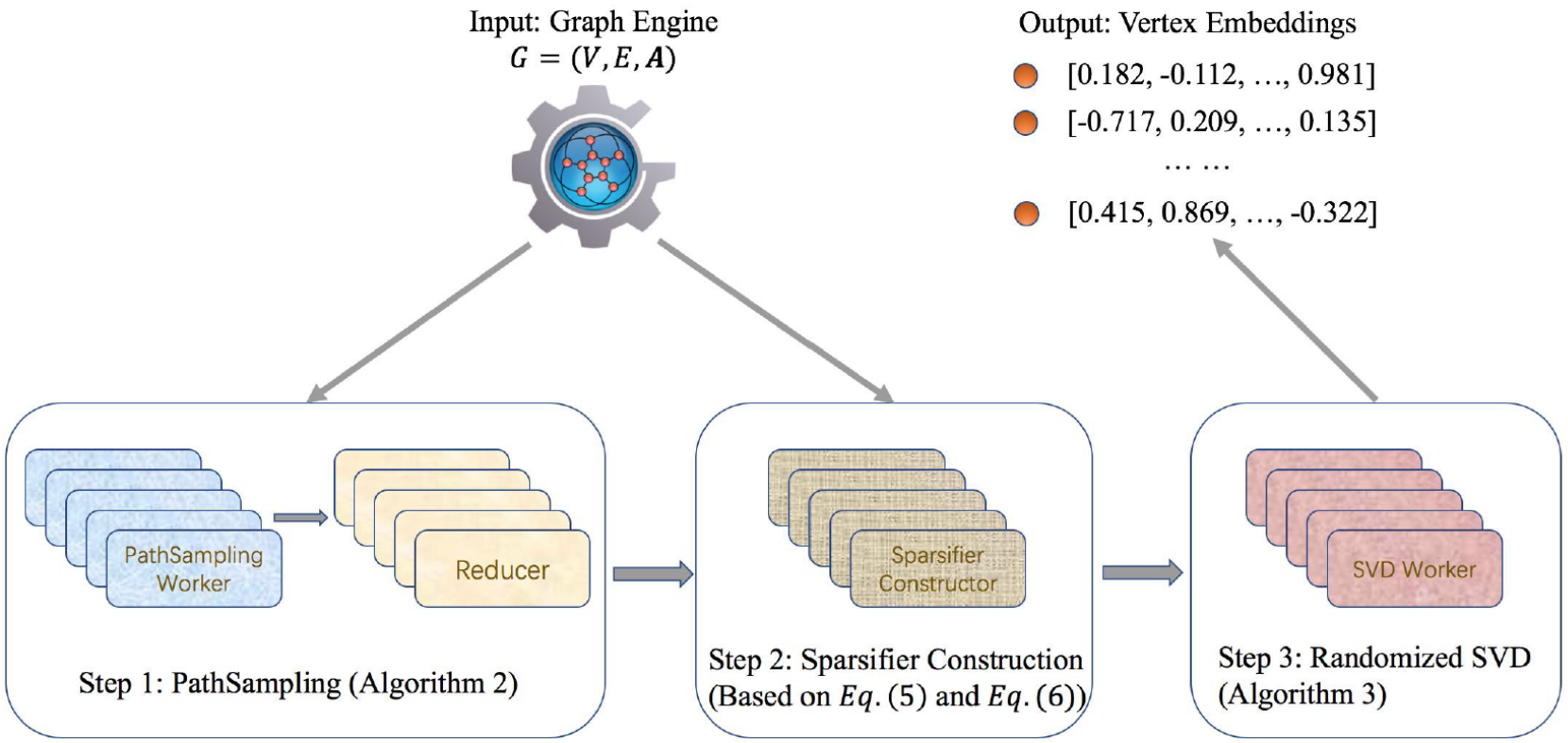
$$\sigma_i(\widetilde{\boldsymbol{M}}-\boldsymbol{M})\leq\frac{4\epsilon}{\sqrt{d_i d_{\min}}},\forall i\in[n].$$

Theorem

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

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

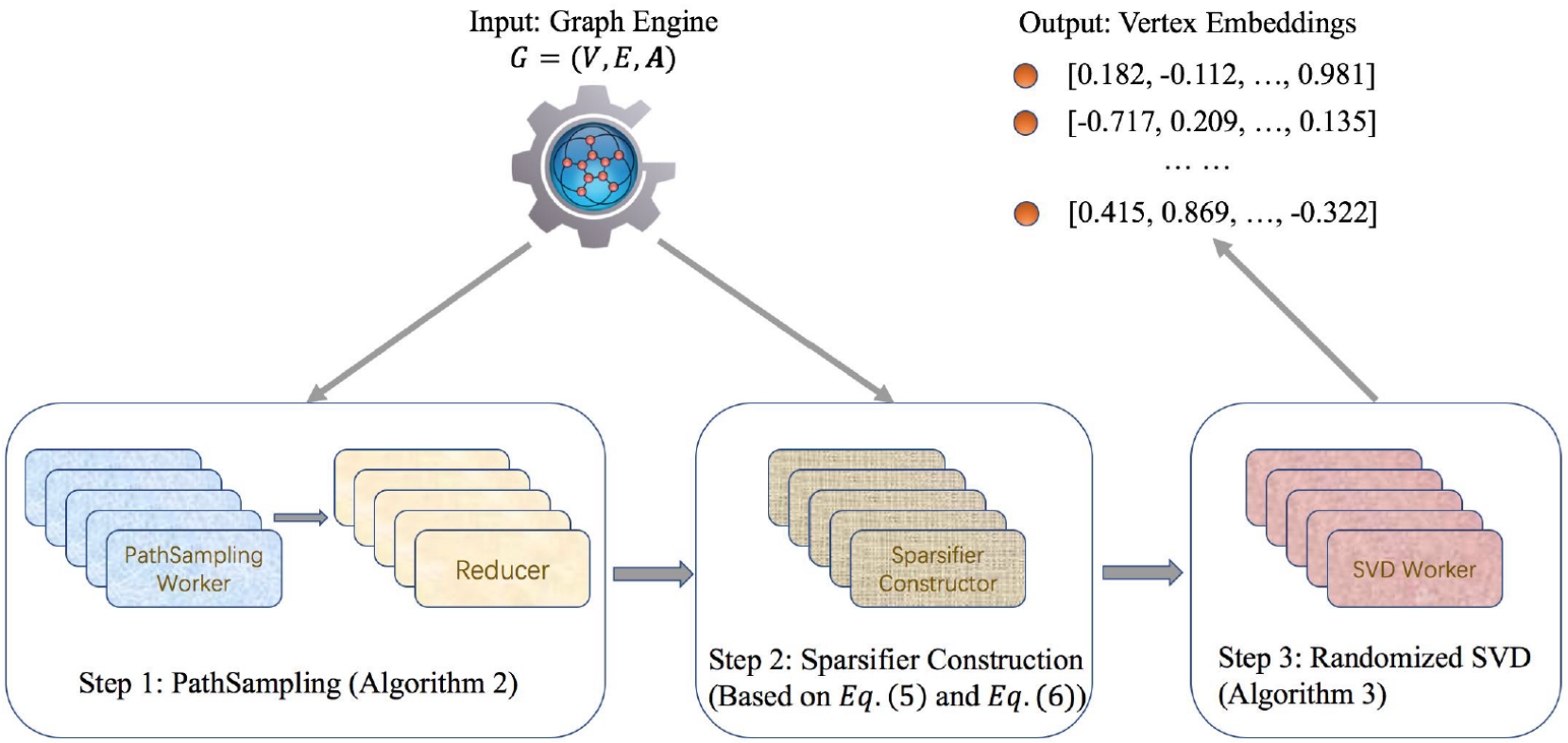
Distributed system design

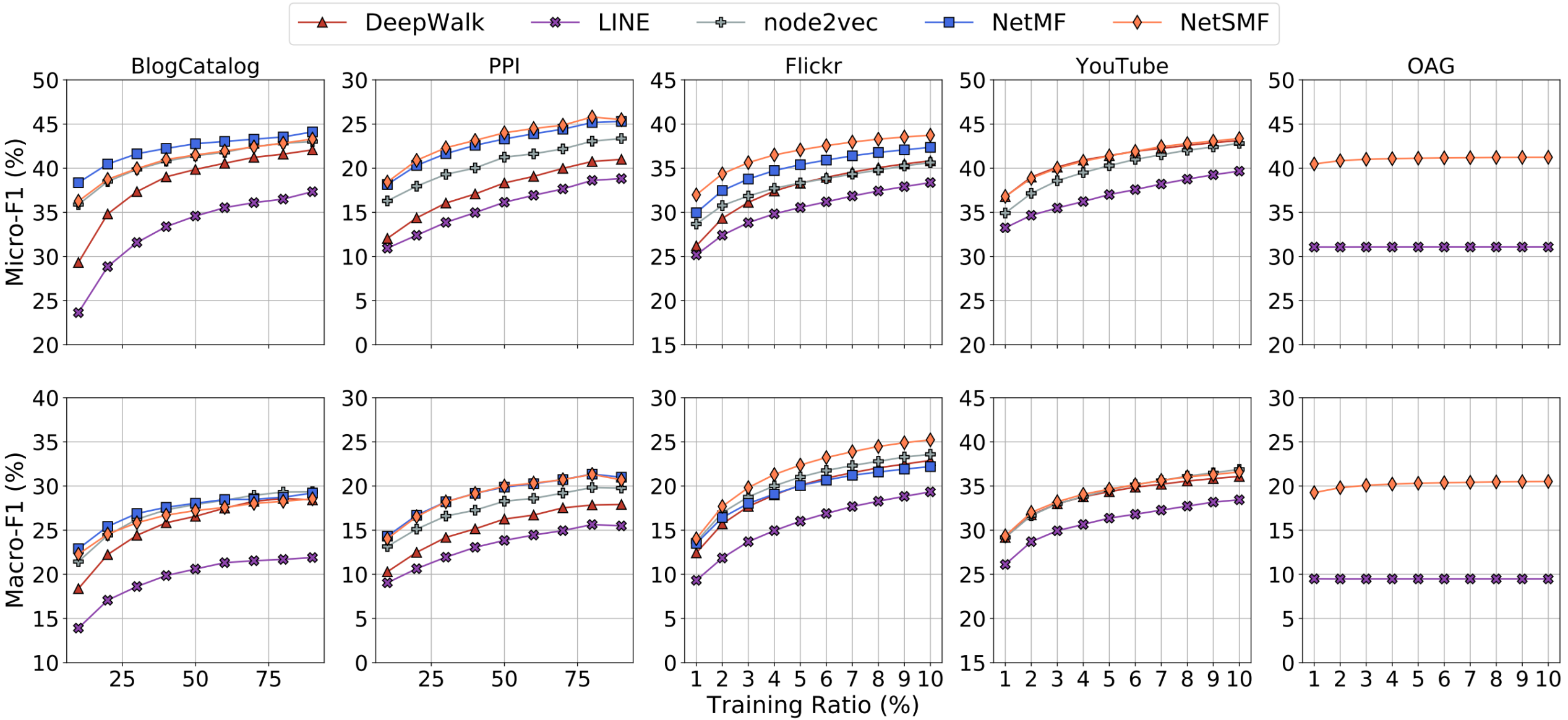


Dataset	BlogCatalog	PPI	Flickr	YouTube	OAG
$\|V\|$	10,312	3,890	80,513	1,138,499	67,768,244
$\|E\|$	333,983	76,584	5,899,882	2,990,443	895,368,962
#labels	39	50	195	47	19

	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

Distributed system design




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

Network Embedding