GNN Course

Yukuo Cen

GNN Center, Zhipu AI

KEG, Tsinghua University

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

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

Course Logistics

- Wednesday 7:30-8:30pm
- Structure of lectures:
  - 45 minutes of a lecture
  - 15 minutes of a live Q&A/discussion session
- Slides will be shared before each lecture

Course Outline

Date	Description
13th July, Wed. 19：30-20：30	Basic Network Embedding (DeepWalk, LINE, node2vec)
20th July, Wed. 19：30-20：30	Advanced Network Embedding (NetMF, ProNE, NetSMF, LightNE)
27th July, Wed. 19：30-20：30	Basic Graph Neural Networks (GCN, GAT, GraphSAGE)
3rd Aug., Wed. 19：30-20：30	Panel Discussion / Invited Talk
10th Aug., Wed. 19：30-20：30	Simplified Graph Neural Networks (SGC, SIGN, SAGN, GAMLP)
17th Aug., Wed. 19：30-20：30	Training GNNs on Large-scale Graphs (ClusterGCN, GraphSAINT)
24th Aug., Wed. 19：30-20：30	Panel Discussion / Invited Talk
31st Aug., Wed. 19：30-20：30	Graph Semi-supervised Learning (GRAND, GRAND+, SCR)
7th Sept., Wed. 19：30-20：30	Graph Self-supervised Learning (GCC, GraphMAE)
14th Sept., Wed. 19：30-20：30	Applications of Graph Neural Networks

Course Grading

- Attendance: 20%
- Homework: 40%
  - 8 assignments
- Course project: 40%
  - Build a demo based on GNN/CogDL

- Extra credit: code contribution to CogDL

Reading Materials

- Blogs:
  - Introduce to GNNs: https://distill.pub/2021/gnn-intro/
  - Understanding GNNs: https://distill.pub/2021/understanding-gnns/
- Books:
  - https://www.cs.mcgill.ca/~wlh/grl_book/
  - https://web.njit.edu/~ym329/dlg_book/
- Research Papers (DeepWalk, GCN, …)
- CogDL
  - code: https://github.com/THUDM/cogdl
  - doc: https://cogdl.readthedocs.io/

Basic Network Embedding

Yukuo Cen

GNN Center, Zhipu AI

KEG, Tsinghua University

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

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

Networked World

facebook

- 2.7 billion MAU
- 17 billion photos/day

Alibaba Group 阿里巴巴集团

- 2.3 billion trans. on 11/11
- ~500 billion GMV on 11/11

twitter

- 350 million MAU
- 5 million tweets/day

新浪微博 weibo.com

- 550 million MAU
- 12 billion pageview/day

Instagram

- 1.15 billion MAU
- 95 million pics/day

ByteDance 字节跳动

- ~1.9 billion MAU
- 70 minutes/user/day

snapchat

- 500 million MAU
- 30 minutes/user/day

WeChat

- QQ: 600 million MAU
- WeChat: 1.2 billion MAU

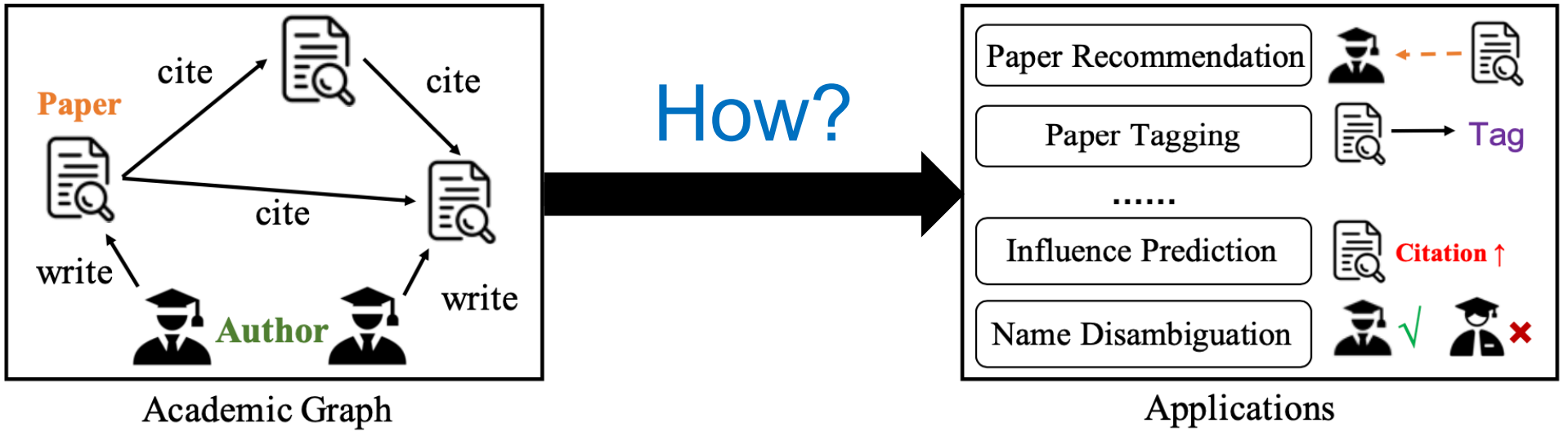
Machine Learning on Graphs

- ML tasks on graphs:
  - Node classification
    - Predict a type of a given node
  - Link prediction
    - Predict whether two nodes are linked
  - Community detection
    - Identify densely linked clusters of nodes
  - Network similarity
    - How similar are two (sub)networks?

Let us start with an application —Academic Graph Mining

Academic Graph Mining

- Input:
  - an academic graph (papers, citation links, …)
- Applications:
  - recommendation, tagging, disambiguation, …

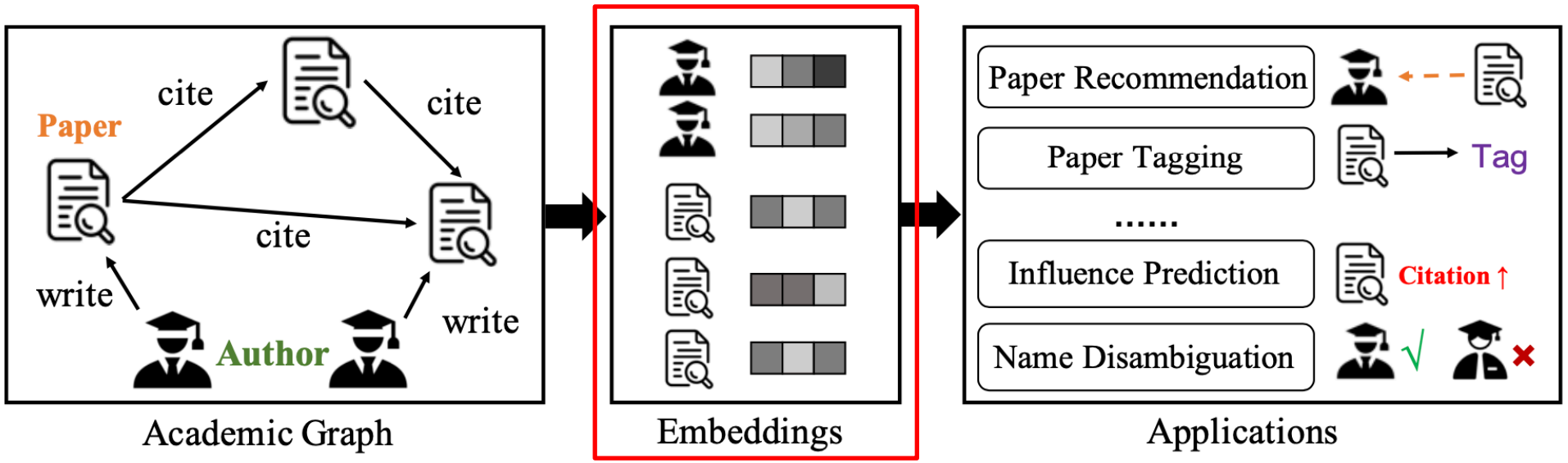


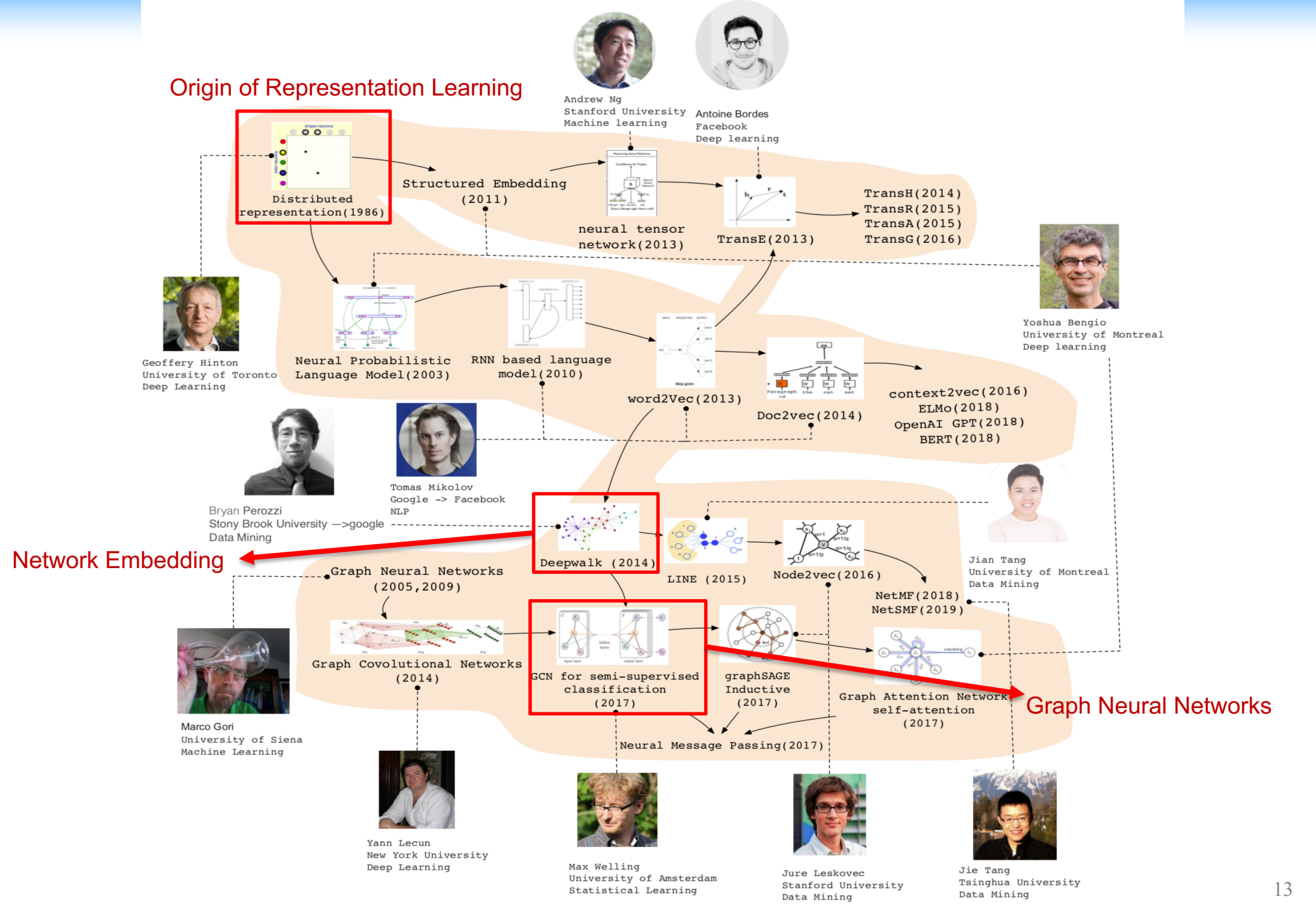
Tasks on Academic Graphs

- Take a deep look at these tasks one by one:
  - Paper recommendation
    - Given past user-paper interactions, the objective is to predict the papers that the user will interact next.
  - Paper Tagging
    - given papers' metadata, attach the most proper concept to each paper
  - Influence Prediction
    - Given publications and paper citations in year yr and before, the goal is to predict the citation number of authors at the end of $yr + \Delta yr$

Question

- How to represent a node in a graph to help downstream tasks?
- **Node Embedding!**




Origin of Representation Learning
Andrew Ng
Stanford University
Machine learning
Antoine Bordes
Facebook
Deep learning
Distributed representation(1986)
Structured Embedding (2011)
neural tensor network(2013)
TransE(2013)
TransH(2014)
TransR(2015)
TransA(2015)
TransG(2016)
Geoffery Hinton
University of Toronto
Deep Learning
Neural Probabilistic Language Model(2003)
RNN based language model(2010)
word2Vec(2013)
Doc2vec(2014)
context2vec(2016)
ELMo(2018)
OpenAI GPT(2018)
BERT(2018)
Yoshua Bengio
University of Montreal
Deep learning
Bryan Perozzi
Stony Brook University —>google
Data Mining
Tomas Mikolov
Google -> Facebook
NLP
Network Embedding
Deepwalk (2014)
LINE (2015)
Node2vec(2016)
NetMF(2018)
NetSMF(2019)
Jian Tang
University of Montreal
Data Mining
Graph Neural Networks (2005,2009)
Graph Covolutional Networks (2014)
GCN for semi-supervised classification (2017)
graphSAGE Inductive (2017)
Graph Attention Network self-attention (2017)
Graph Neural Networks
Marco Gori
University of Siena
Machine Learning
Neural Message Passing(2017)
Yann Lecun
New York University
Deep Learning
Max Welling
University of Amsterdam
Statistical Learning
Jure Leskovec
Stanford University
Data Mining
Jie Tang
Tsinghua University
Data Mining

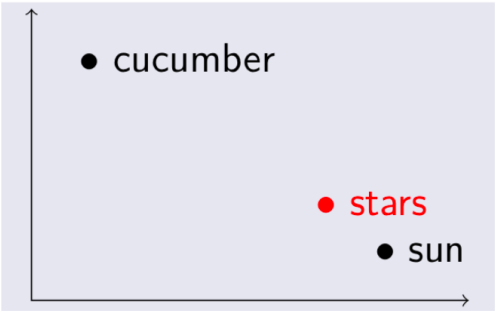
Recall word2vec for NLP

- Learning a representation for words:

$$\phi : v \in V \rightarrow R^d$$

he curtains open and the stars shining in on the barely
ars and the cold , close stars " . And neither of the w
rough the night with the stars shining so brightly , it
made in the light of the stars . It all boils down , wr
surely under the bright stars , thrilled by ice-white
sun , the seasons of the stars ? Home , alone , Jay pla
m is dazzling snow , the stars have risen full and cold

	shining	bright	trees	dark	look
stars	38	45	2	27	12

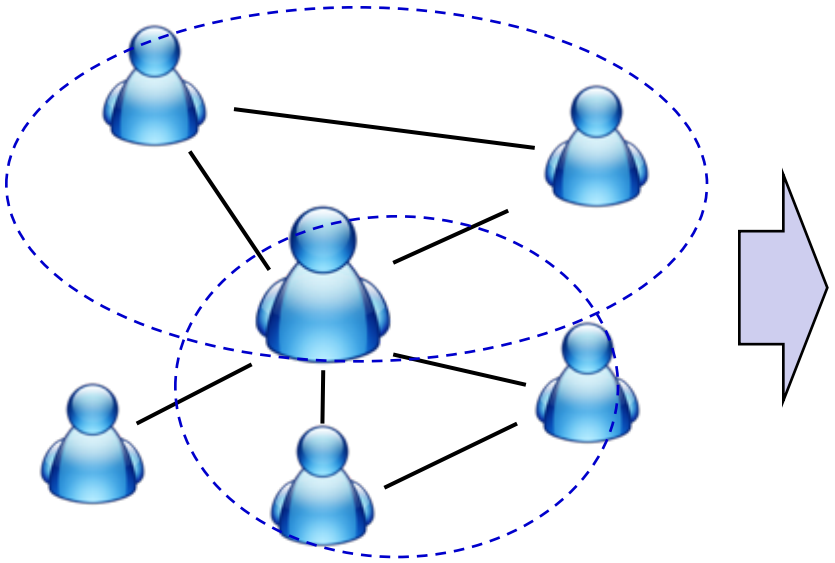


- Ideally, we should have

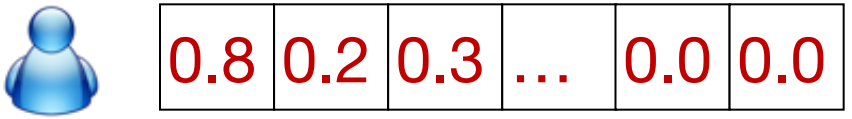
$$\|\phi(\text{sun}) - \phi(\text{stars})\| < \|\phi(\text{cucumber}) - \phi(\text{stars})\|$$

Review Representation Learning for Graphs

Representation Learning/
Graph Embedding

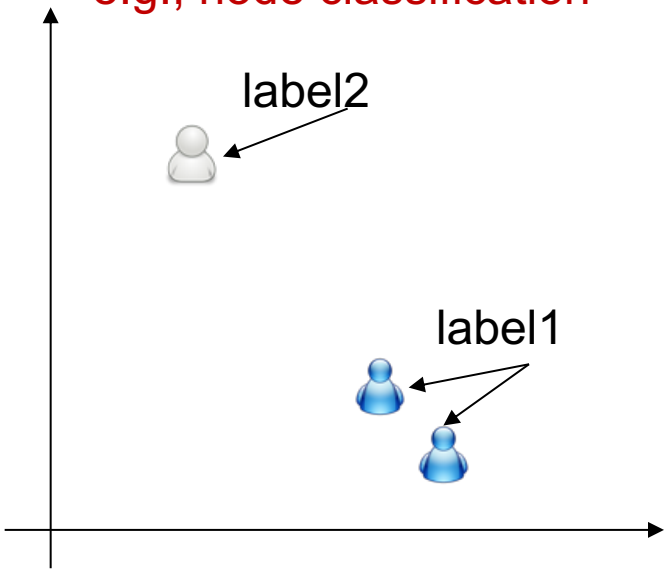


d-dimensional vector, $d << |V|$



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

e.g., node classification

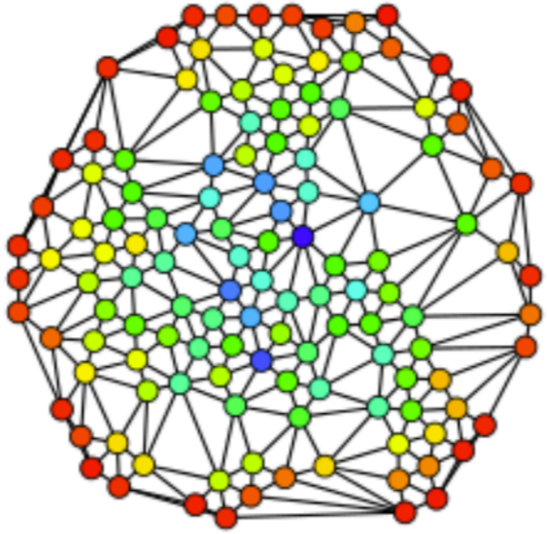


Why is it hard?

- Modern deep learning toolbox is designed for simple sequences or grids.
  - CNNs for fixed-size images/grids...
  - RNNs or word2vec for text/sequences...
- But networks are far more complex!
  - Complex topographical structure (i.e., no spatial locality like grids)
  - No fixed node ordering or reference point (i.e., the isomorphism problem)
  - Often dynamic and have multimodal features.

Node Embeddings

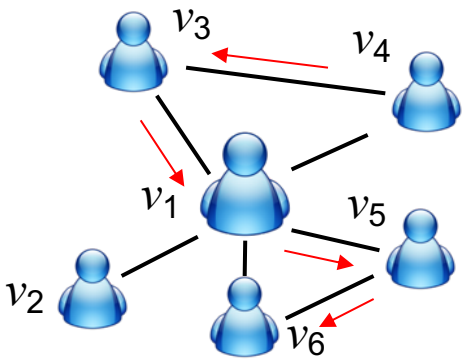
- How to learn a representation for each node of networks?



- Input: $G = (V, E)$ or adjacency matrix
- Problem decomposition:
  - how to generate the context for each vertex?
  - how to learn the representation effectively and efficiently?
  - how to incorporate the other unique network properties?
  - how to combine network and content together?

DeepWalk

Random walk



One example RW path

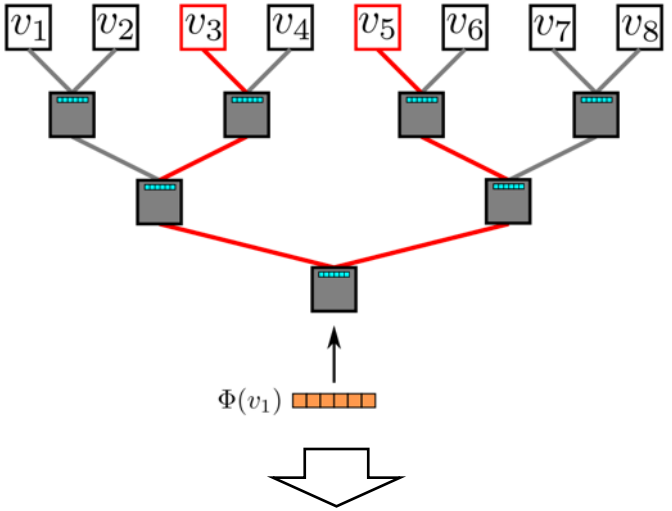
$v_4 \quad v_3 \quad v_1 \quad v_5 \quad v_6$

$\mathcal{W}_{v_4} = 4$

$u_k \begin{bmatrix} 3 \\ 1 \\ 5 \end{bmatrix} v_j \rightarrow$ Φ (d rows, j-th column)

$\begin{matrix} 1 \\ \vdots \end{matrix}$

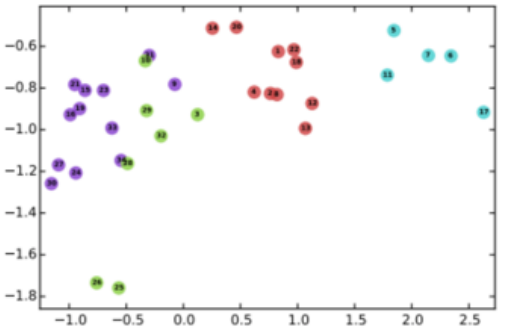
SkipGram with Hierarchical softmax



$$P(\{v_{i-w}, \cdots, v_{i+w}\} \backslash v_i | \Phi(v_i))) = \prod_{j=i-w, j \neq i}^{i+w} P(v_j | \Phi(v_i))$$

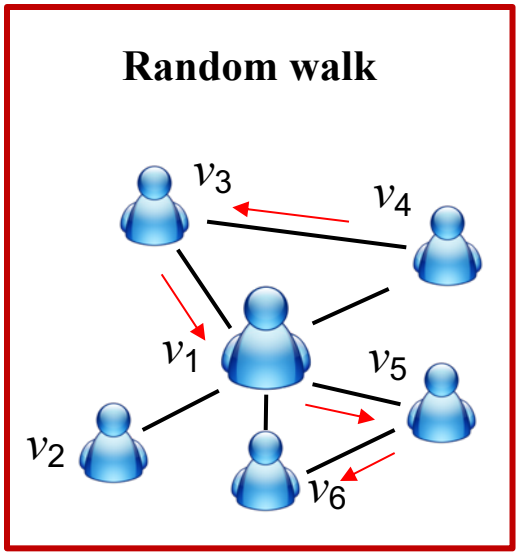
Hierarchical softmax

$$P(v_j | \Phi(v_i)) = \prod_{l=1}^{\log |V|} P(b_l | \Phi(v_i)) = \prod_{l=1}^{\log |V|} 1/(1 + e^{-\Phi(v_i) \cdot \Psi(b_l)})$$



1. B. Perozzi, R. Al-Rfou, and S. Skiena. 2014. Deepwalk: Online learning of social representations. *KDD*, 701–710.

DeepWalk



One example RW path

$v_4 \quad v_3 \quad v_1 \quad v_5 \quad v_6$

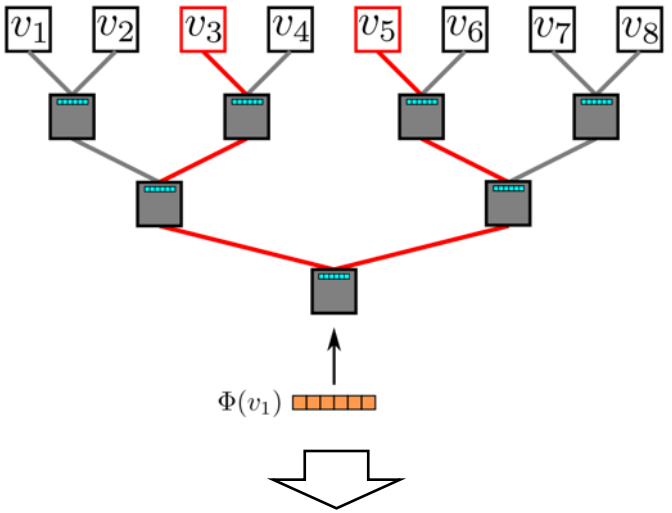
$\mathcal{W}_{v_4} = 4$

$u_k \begin{bmatrix} 3 \\ 1 \\ 5 \end{bmatrix} v_j \rightarrow \Phi \in \mathbb{R}^{|V| \times d}$

1

$\vdots$

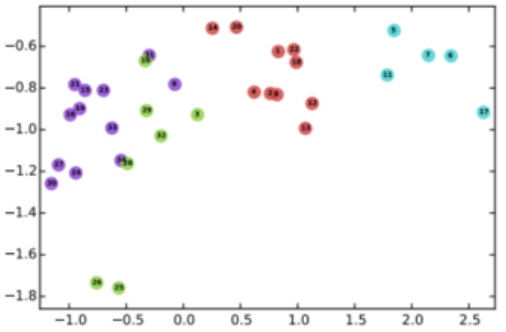
SkipGram with Hierarchical softmax



$$P(\{v_{i-w}, \cdots, v_{i+w}\} \backslash v_i | \Phi(v_i))) = \prod_{j=i-w, j \neq i}^{i+w} P(v_j | \Phi(v_i))$$

Hierarchical softmax

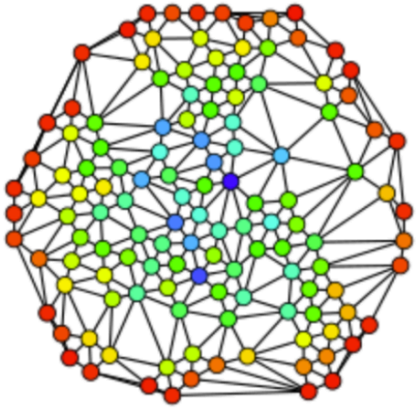
$$P(v_j | \Phi(v_i)) = \prod_{l=1}^{\log |V|} P(b_l | \Phi(v_i)) = \prod_{l=1}^{\log |V|} 1/(1 + e^{-\Phi(v_i) \cdot \Psi(b_l)})$$



1. B. Perozzi, R. Al-Rfou, and S. Skiena. 2014. Deepwalk: Online learning of social representations. *KDD*, 701–710.

Random Walk

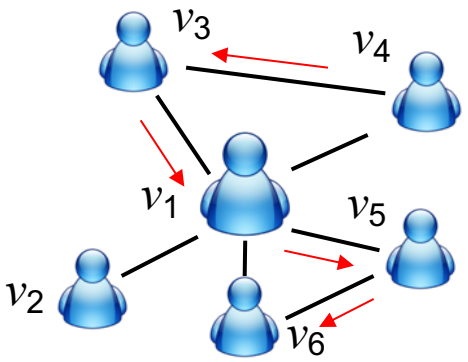
- Generate γ random walks for each vertex
- Each random walk has length t
  - in each random walk step, jump to the next vertex uniformly.
- Example: $v_{46} \rightarrow v_{45} \rightarrow v_{71} \rightarrow v_{24} \rightarrow v_5 \rightarrow v_1 \rightarrow v_{17}$
- Finally, for vertex v_1 in a network, we have



$$
\begin{array}{cccccc}
v_{71} \rightarrow & v_{24} \rightarrow & v_5 \rightarrow & v_1 \rightarrow & v_{17} \rightarrow & v_{80} \rightarrow \\
v_{92} \rightarrow & v_2 \rightarrow & v_3 \rightarrow & v_1 \rightarrow & v_{12} \rightarrow & v_{73} \rightarrow \\
v_{37} \rightarrow & v_{34} \rightarrow & v_9 \rightarrow & v_1 \rightarrow & v_{10} \rightarrow & v_{94} \rightarrow \\
v_{73} \rightarrow & v_{64} \rightarrow & v_5 \rightarrow & v_1 \rightarrow & v_{12} \rightarrow & v_1 \rightarrow \\
v_{75} \rightarrow & v_{14} \rightarrow & v_6 \rightarrow & v_1 \rightarrow & v_{13} \rightarrow & v_{61} \rightarrow
\end{array}
$$

DeepWalk

Random walk



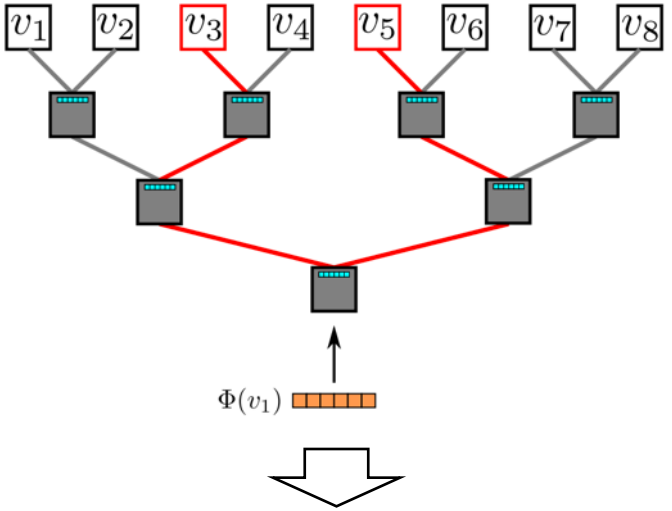
One example RW path

$v_4 \quad v_3 \quad v_1 \quad v_5 \quad v_6$

$\mathcal{W}_{v_4} = 4$

$u_k \begin{bmatrix} 3 \\ 1 \\ 5 \\ 1 \\ \vdots \end{bmatrix} v_j \rightarrow \Phi$ (d × j)

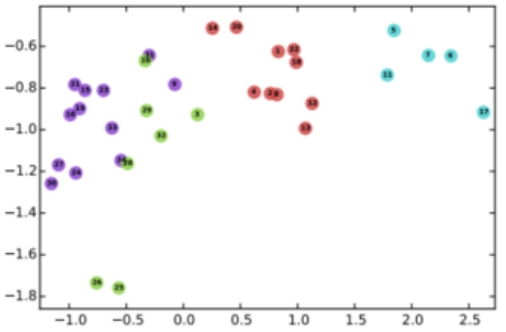
SkipGram with Hierarchical softmax



$$P(\{v_{i-w}, \cdots, v_{i+w}\} \backslash v_i | \Phi(v_i))) = \prod_{j=i-w, j \neq i}^{i+w} P(v_j | \Phi(v_i))$$

Hierarchical softmax

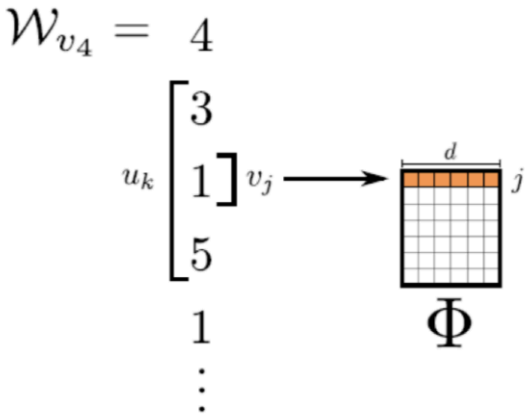
$$P(v_j | \Phi(v_i)) = \prod_{l=1}^{\log |V|} P(b_l | \Phi(v_i)) = \prod_{l=1}^{\log |V|} 1/(1 + e^{-\Phi(v_i) \cdot \Psi(b_l)})$$



1. B. Perozzi, R. Al-Rfou, and S. Skiena. 2014. Deepwalk: Online learning of social representations. *KDD*, 701–710.

Representation Mapping

- For a path $\mathcal{W}_{v_4}$: $v_4 \rightarrow v_3 \rightarrow v_1 \rightarrow v_5 \rightarrow v_1 \rightarrow v_{46}$

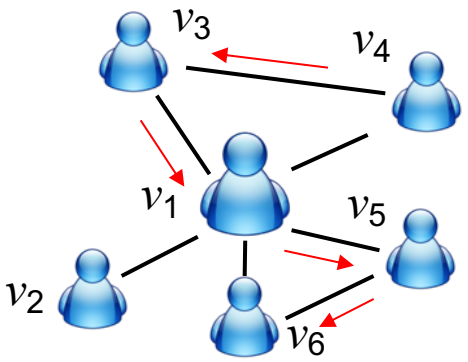


- Define a window size **w**: if $\mathbf{w} = 1$ then $\mathbf{v} = v_1$
- Map the current vertex v_1 to its representation in R^d
- Maximize (skip-gram)

$$P(v_3, v_5 | \phi(v_1))$$

DeepWalk

Random walk



One example RW path

$v_4 \quad v_3 \quad v_1 \quad v_5 \quad v_6$

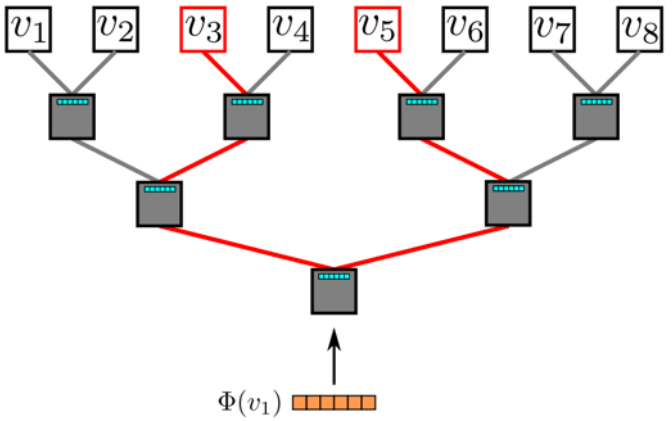
$\mathcal{W}_{v_4} = 4$

$u_k \begin{bmatrix} 3 \\ 1 \\ 5 \end{bmatrix} v_j \rightarrow$ Φ (d dimensions, j)

1

$\vdots$

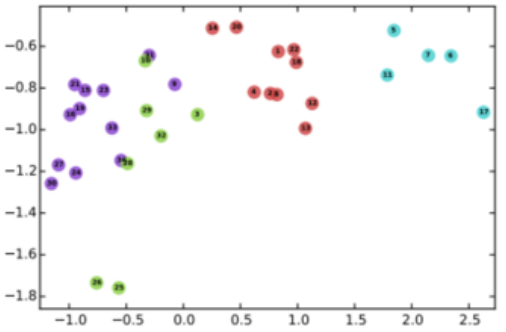
SkipGram with Hierarchical softmax



$$P(\{v_{i-w}, \cdots, v_{i+w}\} \backslash v_i | \Phi(v_i))) = \prod_{j=i-w, j \neq i}^{i+w} P(v_j | \Phi(v_i))$$

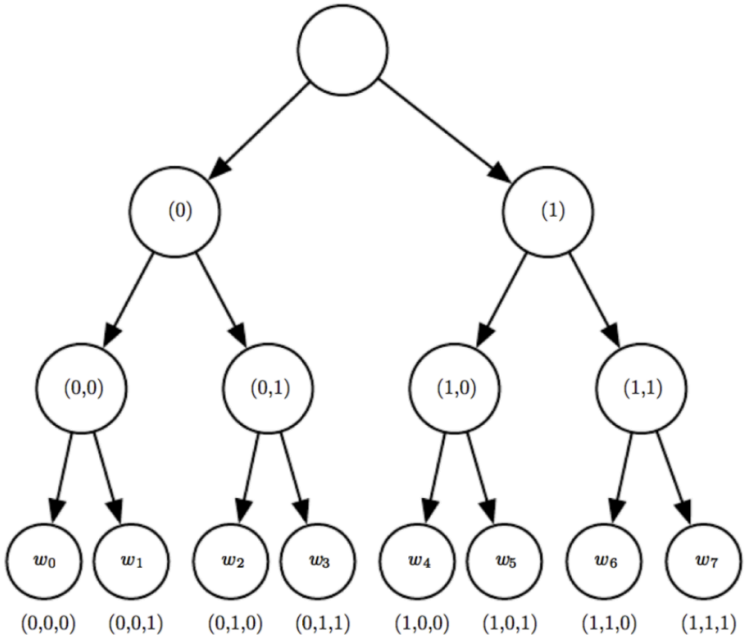
Hierarchical softmax

$$P(v_j | \Phi(v_i)) = \prod_{l=1}^{\log |V|} P(b_l | \Phi(v_i)) = \prod_{l=1}^{\log |V|} 1/(1 + e^{-\Phi(v_i) \cdot \Psi(b_l)})$$



1. B. Perozzi, R. Al-Rfou, and S. Skiena. 2014. Deepwalk: Online learning of social representations. *KDD*, 701–710.

Hierarchical Softmax



- Then

$$p(v_i|C_i) = \prod_{k=1}^{K} p(d_k|q_k, C_i) = \prod_{k=1}^{K} \left(\sigma(q_k \cdot C_i)^{1-d_k} (1 - \sigma(q_k \cdot C_i))^{d_k} \right)$$

where σ is the sigmoid function, q_k is the vector of non-leaf node on the path from root to word leaf, d_k is the corresponding code of q_k.

Parameter Learning

- Randomly initialize the representations
- Each classifier in the hierarchy has a set of weights
- Use SGD (stochastic gradient descent) to update both classifier weights and vertex representations simultaneously

Results: BlogCatalog

Name	BlogCatalog
$\|V\|$	10,312
$\|E\|$	333,983
$\|\mathcal{Y}\|$	39
Labels	Interests

	% Labeled Nodes	10%	20%	30%	40%	50%	60%	70%	80%	90%
Micro-F1(%)	DeepWalk	**36.00**	**38.20**	**39.60**	**40.30**	**41.00**	**41.30**	41.50	41.50	42.00
	SpectralClustering	31.06	34.95	37.27	38.93	39.97	40.99	**41.66**	**42.42**	**42.62**
	EdgeCluster	27.94	30.76	31.85	32.99	34.12	35.00	34.63	35.99	36.29
	Modularity	27.35	30.74	31.77	32.97	34.09	36.13	36.08	37.23	38.18
	wvRN	19.51	24.34	25.62	28.82	30.37	31.81	32.19	33.33	34.28
	Majority	16.51	16.66	16.61	16.70	16.91	16.99	16.92	16.49	17.26
Macro-F1(%)	DeepWalk	**21.30**	**23.80**	25.30	26.30	27.30	27.60	27.90	28.20	28.90
	SpectralClustering	19.14	23.57	**25.97**	**27.46**	**28.31**	**29.46**	**30.13**	**31.38**	**31.78**
	EdgeCluster	16.16	19.16	20.48	22.00	23.00	23.64	23.82	24.61	24.92
	Modularity	17.36	20.00	20.80	21.85	22.65	23.41	23.89	24.20	24.97
	wvRN	6.25	10.13	11.64	14.24	15.86	17.18	17.98	18.86	19.57
	Majority	2.52	2.55	2.52	2.58	2.58	2.63	2.61	2.48	2.62

- Feed the learned representation for multi-label classification
- DeepWalk (vertex representation learning) performs well, especially when labels are sparse.

Results: YouTube

Name	YouTube
$\|V\|$	1,138,499
$\|E\|$	2,990,443
$\|\mathcal{Y}\|$	47
Labels	Groups

	% Labeled Nodes	1%	2%	3%	4%	5%	6%	7%	8%	9%	10%
Micro-F1(%)	DeepWalk	**37.95**	**39.28**	**40.08**	**40.78**	**41.32**	**41.72**	**42.12**	**42.48**	**42.78**	**43.05**
	SpectralClustering	—	—	—	—	—	—	—	—	—	—
	EdgeCluster	23.90	31.68	35.53	36.76	37.81	38.63	38.94	39.46	39.92	40.07
	Modularity	—	—	—	—	—	—	—	—	—	—
	wvRN	26.79	29.18	33.1	32.88	35.76	37.38	38.21	37.75	38.68	39.42
	Majority	24.90	24.84	25.25	25.23	25.22	25.33	25.31	25.34	25.38	25.38
Macro-F1(%)	DeepWalk	**29.22**	**31.83**	**33.06**	**33.90**	**34.35**	**34.66**	**34.96**	**35.22**	**35.42**	**35.67**
	SpectralClustering	—	—	—	—	—	—	—	—	—	—
	EdgeCluster	19.48	25.01	28.15	29.17	29.82	30.65	30.75	31.23	31.45	31.54
	Modularity	—	—	—	—	—	—	—	—	—	—
	wvRN	13.15	15.78	19.66	20.9	23.31	25.43	27.08	26.48	28.33	28.89
	Majority	6.12	5.86	6.21	6.1	6.07	6.19	6.17	6.16	6.18	6.19

- Similar results on YouTube
- Spectral Clustering does not scale to large graphs

Homework 1: DeepWalk

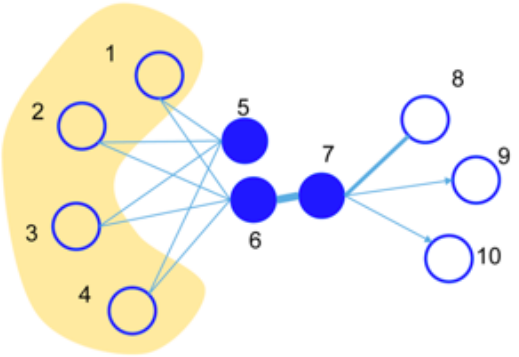
- Implement DeepWalk for citation networks
  - Cora citation dataset;
  - Random walk algorithm;
  - Call gensim.Word2Vec model;
  - Train linear classifier for prediction

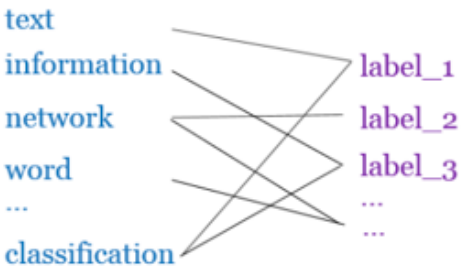
- Requirement:
  - pip install cogdl

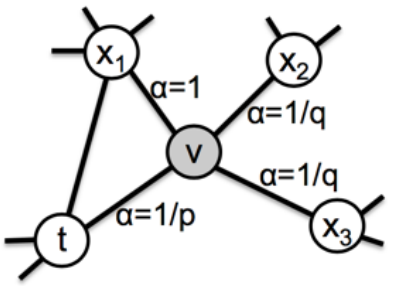
- DeepWalk utilizes fixed-length, unbiased random walks to generate context for each node, can we do better?

Later…

- LINE[1]: explicitly preserves both *first-order* and *second-order* proximities.
- PTE[2]: learn heterogeneous text network embedding via a semi-supervised manner.
- Node2vec[3]: use a biased random walk to better explore node's neighborhood.







1. J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei. 2015. Line: Large-scale information network embedding. *WWW*, 1067–1077.
2. J. Tang, M. Qu, and Q. Mei. 2015. Pte: Predictive text embedding through large-scale heterogeneous text networks. *KDD*, 1165–1174.
3. A. Grover and J. Leskovec. 2016. node2vec: Scalable feature learning for networks. *KDD*, 855–864.

LINE: Large-scale Information Network Embedding

First-order proximity: two adjacent nodes

$$p_1(u,v) = \frac{1}{1+\exp(-z_u^T z_v)}$$

$$\hat{p}_1(u,v) = \frac{w_{u,v}}{W} \; with \; W = \sum_{u,v \in E} w_{u,v}$$

Objective Function (KL divergence):

$$\mathcal{L}_1 = d(\hat{p}_1(.,.), p_1(.,.)) = -\sum_{u,v \in E} w_{u,v} log p_1(u,v)$$

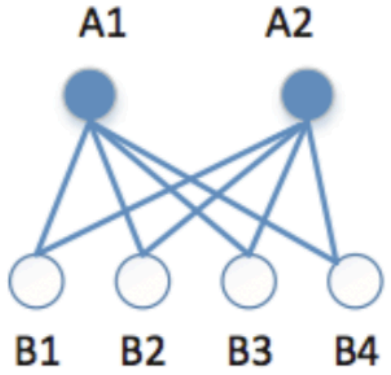
J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei. 2015. Line: Large-scale information network embedding. *WWW'15*, 1067–1077.

LINE: Large-scale Information Network Embedding

Second-order proximity:

$$p_2(v|u) = \frac{\exp(z_v^T z_u)}{\sum_{k \in N(u)} \exp(z_k^T z_u)}$$

$$\hat{p}_2(v|u) = \frac{w_{u,v}}{d_u} \; with \; d_u \; degree \; of \; u$$



Objective Function (KL divergence, $\lambda_u = d_u$)

$$\mathcal{L}_2 = \sum_{u \in V} \lambda_u d(\hat{p}_2(.|u), p_2(.|u)) = -\sum_{u,v \in E} w_{u,v} \log p_2(v|u)$$

Results: YouTube

Table 6: Results of multi-label classification on the YOUTUBE network. The results in the brackets are on the reconstructed network, which adds second-order neighbors (i.e., neighbors of neighbors) as neighbors for vertices with a low degree.

Metric	Algorithm	1%	2%	3%	4%	5%	6%	7%	8%	9%	10%
Micro-F1	GF	25.43 (24.97)	26.16 (26.48)	26.60 (27.25)	26.91 (27.87)	27.32 (28.31)	27.61 (28.68)	27.88 (29.01)	28.13 (29.21)	28.30 (29.36)	28.51 (29.63)
	DeepWalk	39.68	41.78	42.78	43.55	43.96	44.31	44.61	44.89	45.06	45.23
	DeepWalk(256dim)	39.94	42.17	43.19	44.05	44.47	44.84	45.17	45.43	45.65	45.81
	LINE(1st)	35.43 (36.47)	38.08 (38.87)	39.33 (40.01)	40.21 (40.85)	40.77 (41.33)	41.24 (41.73)	41.53 (42.05)	41.89 (42.34)	42.07 (42.57)	42.21 (42.73)
	LINE(2nd)	32.98 (36.78)	36.70 (40.37)	38.93 (42.10)	40.26 (43.25)	41.08 (43.90)	41.79 (44.44)	42.28 (44.83)	42.70 (45.18)	43.04 (45.50)	43.34 (45.67)
	LINE(1st+2nd)	39.01* (**40.20**)	41.89 (**42.70**)	43.14 (**43.94****)	44.04 (**44.71****)	44.62 (**45.19****)	45.06 (**45.55****)	45.34 (**45.87****)	45.69** (**46.15****)	45.91** (**46.33****)	46.08** (**46.43****)
Macro-F1	GF	7.38 (11.01)	8.44 (13.55)	9.35 (14.93)	9.80 (15.90)	10.38 (16.45)	10.79 (16.93)	11.21 (17.38)	11.55 (17.64)	11.81 (17.80)	12.08 (18.09)
	DeepWalk	28.39	30.96	32.28	33.43	33.92	34.32	34.83	35.27	35.54	35.86
	DeepWalk (256dim)	28.95	31.79	33.16	34.42	34.93	35.44	35.99	36.41	36.78	37.11
	LINE(1st)	28.74 (29.40)	31.24 (31.75)	32.26 (32.74)	33.05 (33.41)	33.30 (33.70)	33.60 (33.99)	33.86 (34.26)	34.18 (34.52)	34.33 (34.77)	34.44 (34.92)
	LINE(2nd)	17.06 (22.18)	21.73 (27.25)	25.28 (29.87)	27.36 (31.88)	28.50 (32.86)	29.59 (33.73)	30.43 (34.50)	31.14 (35.15)	31.81 (35.76)	32.32 (36.19)
	LINE(1st+2nd)	**29.85** (29.24)	31.93 (**33.16****)	33.96 (**35.08****)	35.46** (**36.45****)	36.25** (**37.14****)	36.90** **(37.69****)	37.48** **(38.30****)	38.10** **(38.80****)	38.46** **(39.15****)	38.82** **(39.40****)

Significantly outperforms DeepWalk at the: ** 0.01 and * 0.05 level, paired t-test.

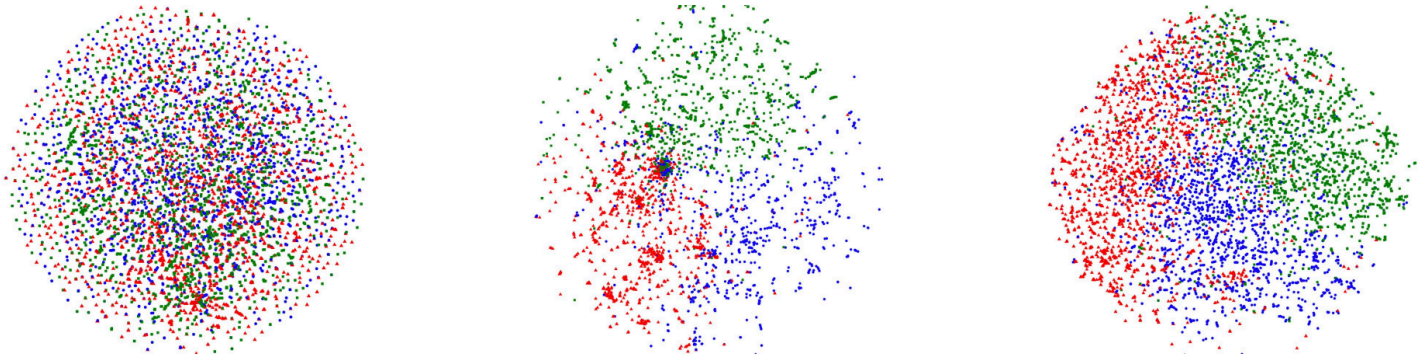
- Better than DeepWalk when adds second-order neighbors

Results: YouTube（cont.）

Table 7: Results of multi-label classification on DBLP(AUTHORCITATION) network.

Metric	Algorithm	10%	20%	30%	40%	50%	60%	70%	80%	90%
Micro-F1	DeepWalk	63.98	64.51	64.75	64.81	64.92	64.99	64.99	65.00	64.90
	LINE-SGD(2nd)	56.64	58.95	59.89	60.20	60.44	60.61	60.58	60.73	60.59
	LINE(2nd)	62.49	63.30	63.63	63.77	63.84	63.94	63.96	64.00	63.77
		(64.69*)	**(65.47**)**	**(65.85**)**	**(66.04**)**	**(66.19**)**	**(66.25**)**	**(66.30**)**	**(66.12**)**	**(66.05**)**
Macro-F1	DeepWalk	63.02	63.60	63.84	63.90	63.98	64.06	64.09	64.11	64.05
	LINE-SGD(2nd)	55.24	57.63	58.56	58.82	59.11	59.27	59.28	59.46	59.37
	LINE(2nd)	61.43	62.38	62.73	62.87	62.93	63.05	63.07	63.13	62.95
		(63.49*)	**(64.42**)**	**(64.84**)**	**(65.05**)**	**(65.19**)**	**(65.26**)**	**(65.29**)**	**(65.14**)**	**(65.14**)**

Significantly outperforms DeepWalk at the: ** 0.01 and * 0.05 level, paired t-test.

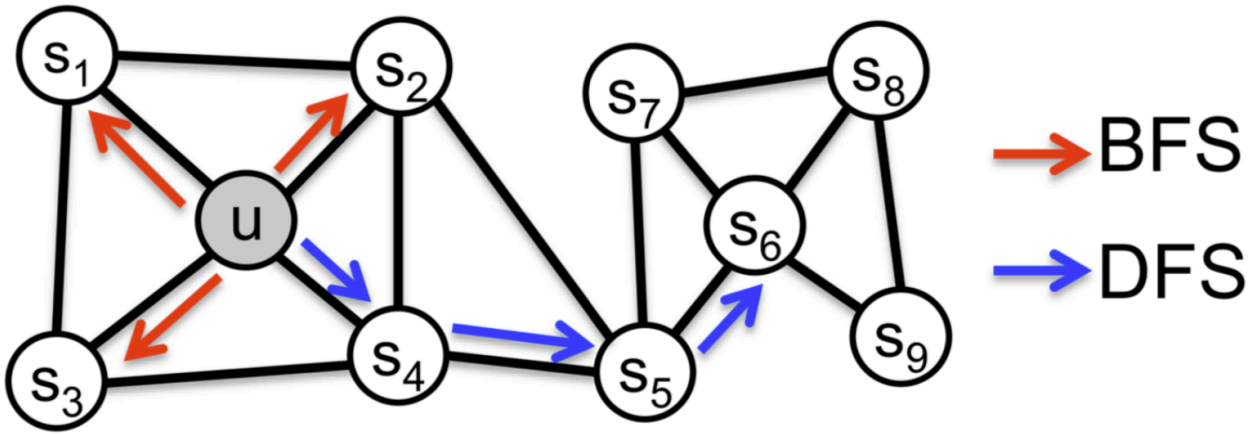


(a) GF (b) DeepWalk (c) LINE(2nd)

Figure 2: Visualization of the co-author network. The authors are mapped to the 2-D space using the t-SNE package with learned embeddings as input. Color of a node indicates the community of the author. Red: "data Mining," blue: "machine learning," green: "computer vision."

node2vec

- **Idea**：apply a biased random walk to capture local and global structures

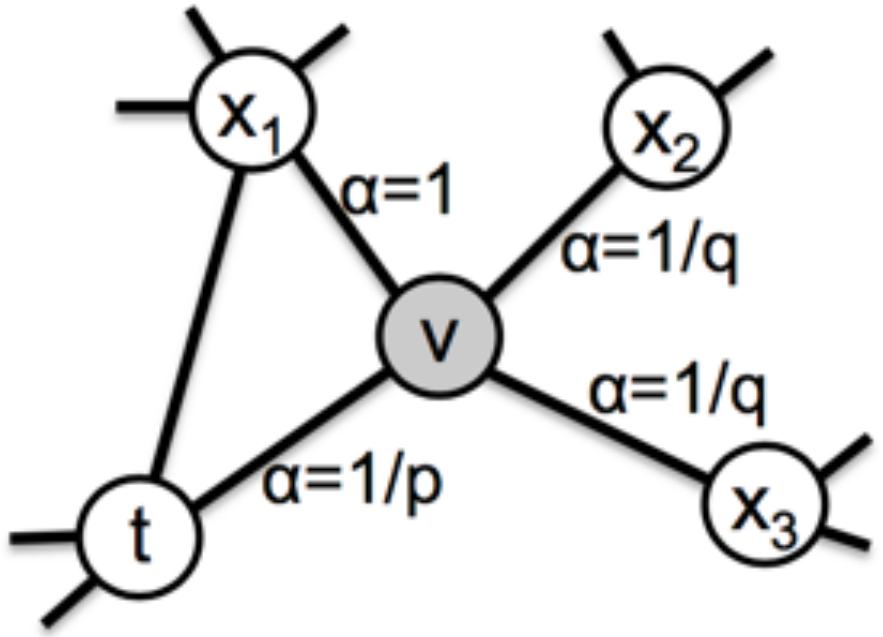


A. Grover and J. Leskovec. 2016. node2vec: Scalable feature learning for networks. *KDD*, 855–864.

Biased random walk in node2vec

- Biased random walk via two parameters：
  - Return parameter p：
    - Revisit the previous node
  - In-out parameter q：
    - inward or outward nodes
    - BFS v.s. DFS

- DeepWalk ≈ node2vec (p=q=1)

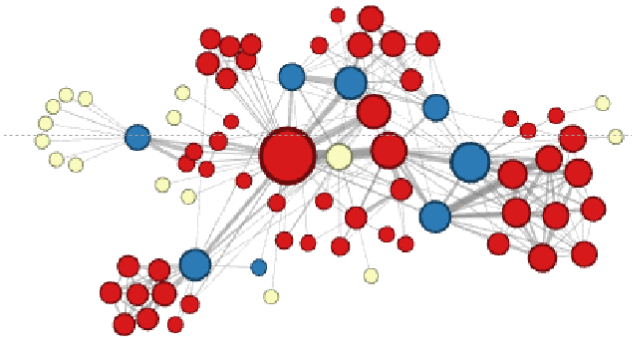


BFS-like： small p
DFS-like： small q

node2vec

1. Preprocess random walk probabilities;
2. Apply biased random walk for each node;
3. Use SGD to optimize the objective $\mathcal{L}^{(u)} = \sum_{v \in N_R(u)} -\log(P(v|\mathbf{z}_u))$

Interactions of characters in a novel:



p=1, q=2

Microscopic view of the network neighbourhood

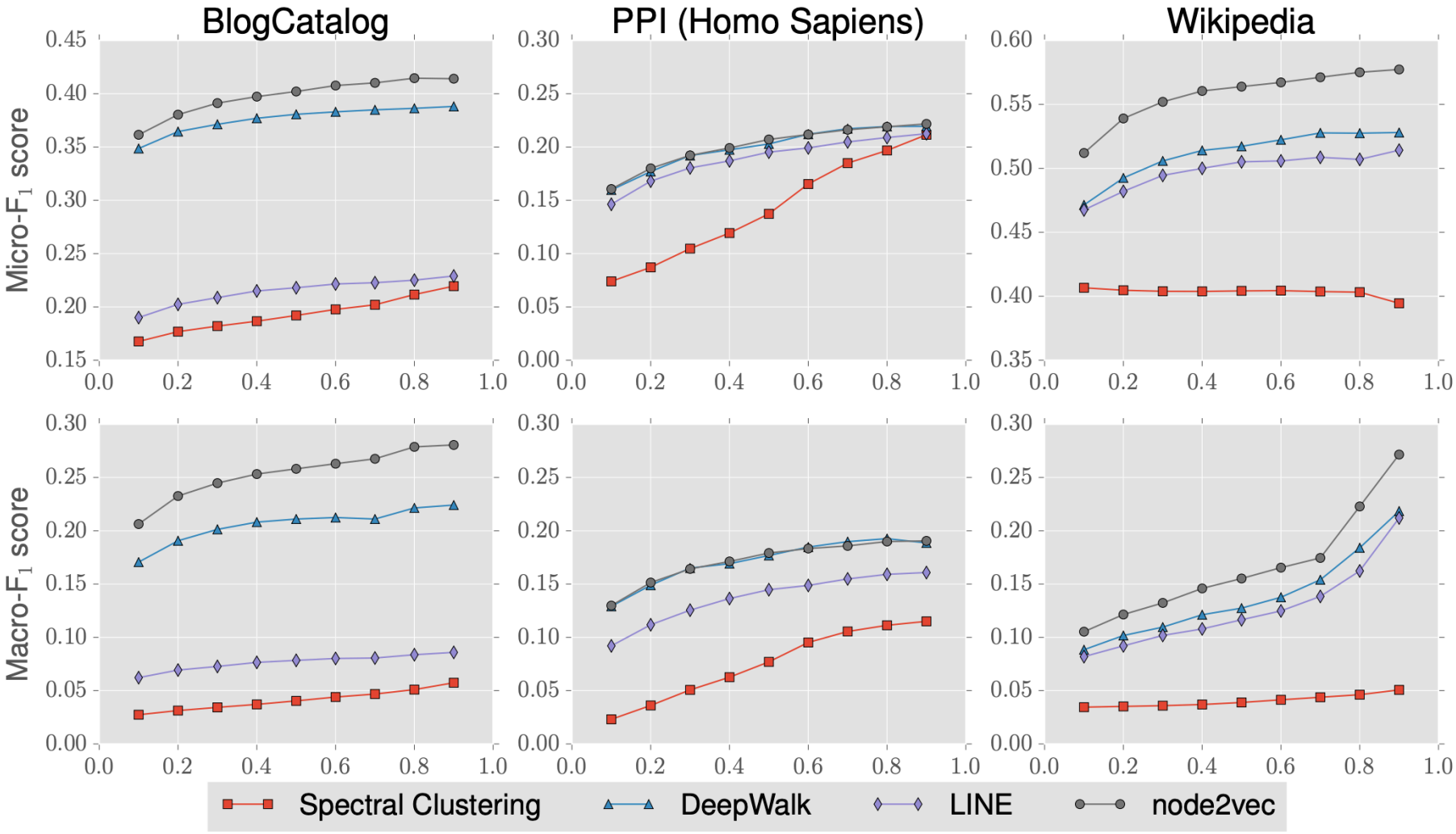
p=1, q=0.5

Macroscopic view of the network neighbourhood

Different p, q
Different result

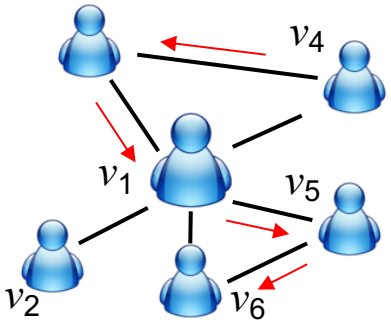
from Prof. Jure Leskovec's slides

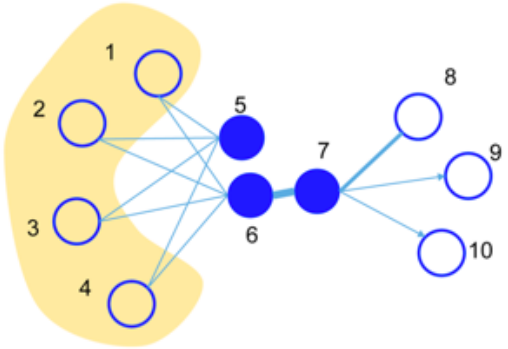
Results: Blog / PPI / Wikipedia

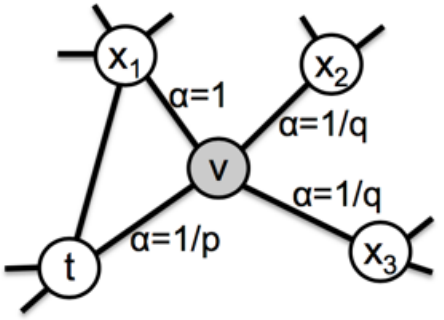


Short Summary

- DeepWalk: random walk + SkipGram
- LINE: both consider first-/second proximities
- Node2vec: Biased random walk







J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei. 2015. Line: Large-scale information network embedding. *WWW'15*, 1067–1077.
J. Tang, M. Qu, and Q. Mei. 2015. Pte: Predictive text embedding through large-scale heterogeneous text networks. *KDD'15*, 1165–1174.
A. Grover and J. Leskovec. 2016. node2vec: Scalable feature learning for networks. *KDD'16*, 855–864.

CogDL Introduction

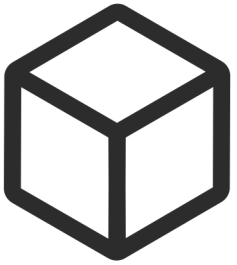
Vision

CogDL provides researchers and developers with a unified trainer, easy-to-use APIs, and high efficiency for most graph tasks and applications.

Philosophy

Unified

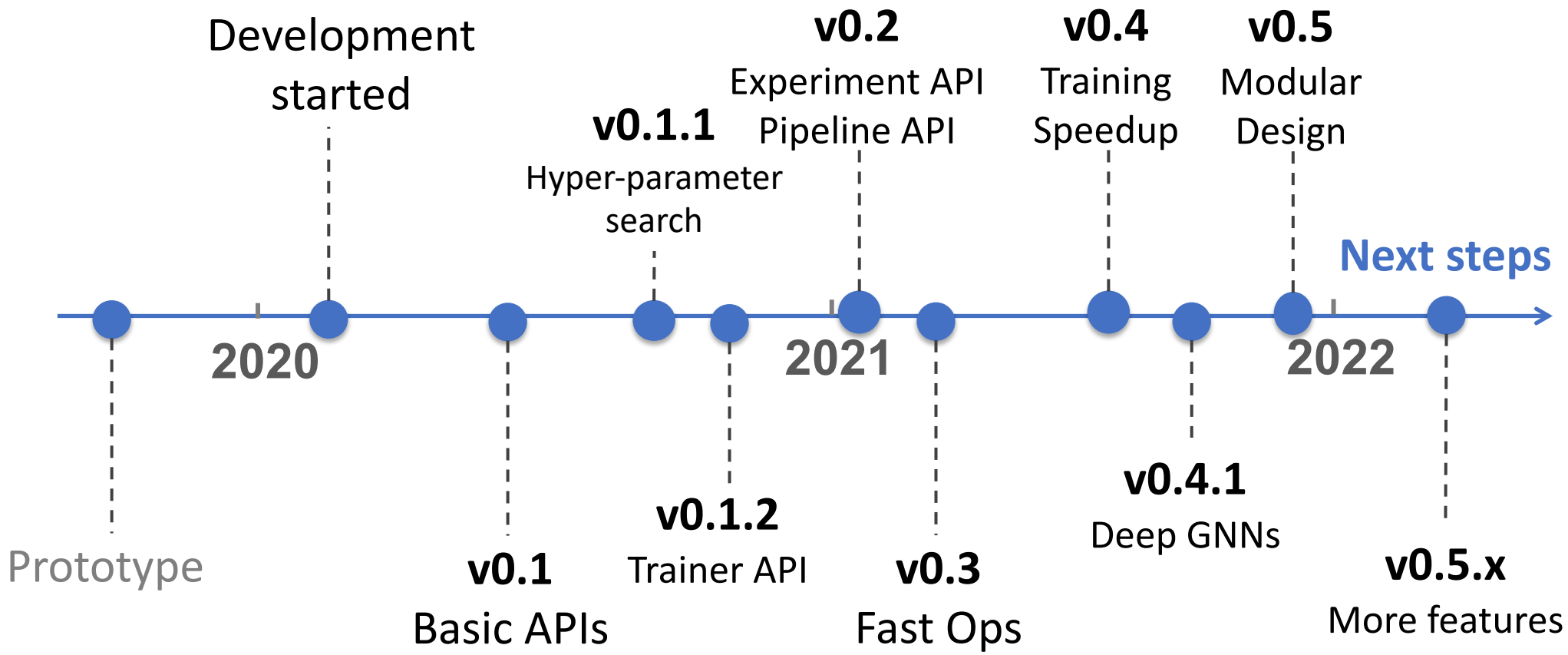
Easy-to-use



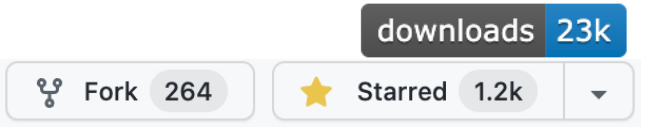
Efficiency

[1] Yukuo Cen, Zhenyu Hou, …, Peng Zhang, Guohao Dai, Yu Wang, Chang Zhou, Hongxia Yang, Jie Tang. CogDL: A Toolkit for Deep Learning on Graphs. In arXiv:2103.00959.

CogDL Development

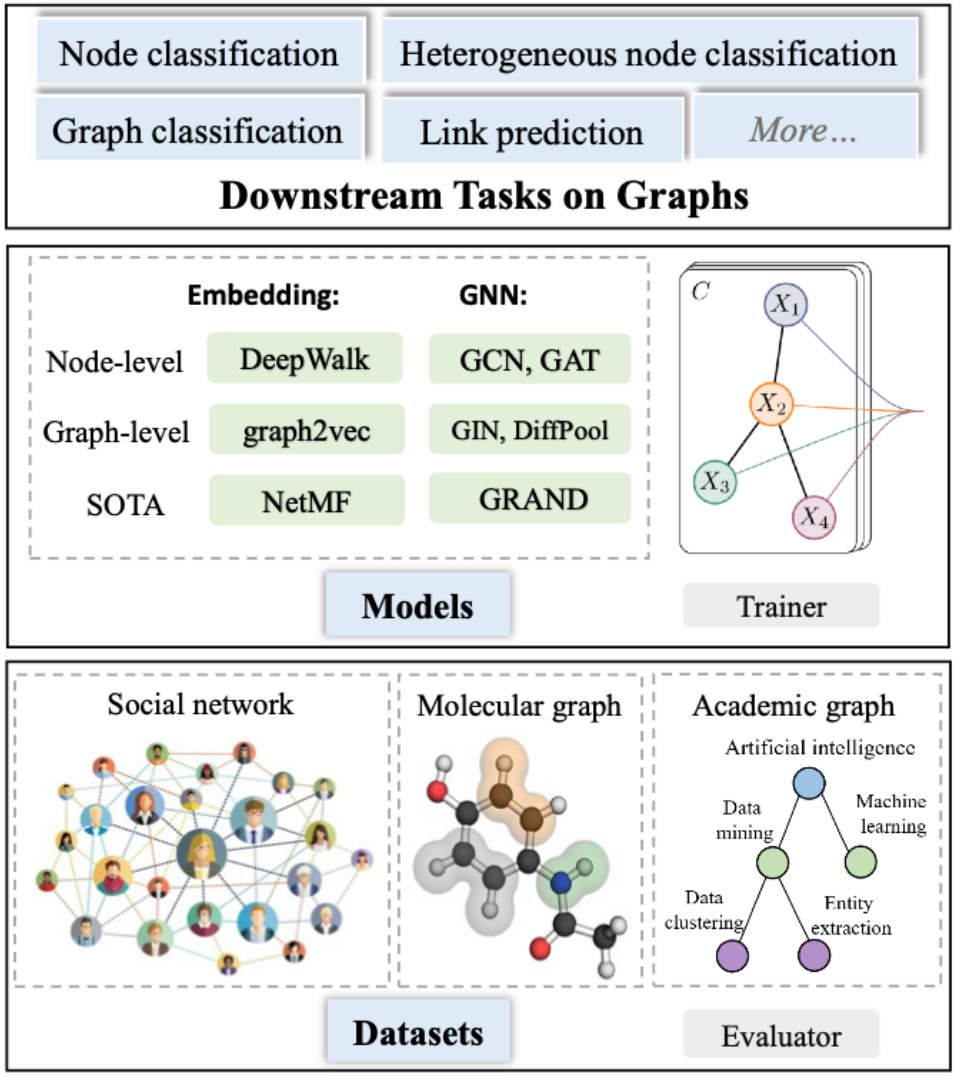


Prerequisite: PyTorch environment
CogDL installation: **pip install cogdl**
or git clone https://github.com/THUDM/cogdl



Tasks, Datasets, Models in CogDL

- >10 Tasks:
  - node classification
  - graph classification
- >60 Datasets:
  - Social networks
  - Academic graphs
  - Molecular graphs
- >70 models:
  - Network embedding
  - Graph Neural Networks



Basic Usage: Experiment API

- Quick start:
  - Dataset, model, (hyper-parameters), (search space)

```python
from cogdl import experiment

# basic usage
experiment(dataset="cora", model="gcn")

# set other hyper-parameters
experiment(dataset="cora", model="gcn", hidden_size=32, epochs=200)

# run over multiple models on different seeds
experiment(dataset="cora", model=["gcn", "gat"], seed=[0, 1])


def search_space(trial):
    return {
        "lr": trial.suggest_categorical("lr", [1e-3, 5e-3, 1e-2]),
        "hidden_size": trial.suggest_categorical("hidden_size", [32, 64, 128]),
        "dropout": trial.suggest_uniform("dropout", 0.5, 0.8),
    }


experiment(dataset="cora", model="gcn", seed=[1, 2], search_space=search_space, n_trials=3)
```

Results of Experiment API

```python
from cogdl import experiment

# basic usage
experiment(dataset="cora", model="gcn")
```

✓ 11.9s Python

```
Namespace(activation='relu', actnn=False, checkpoint_path='./checkpoints/model.pt', cpu=False, cpu_inference=False, dataset=['cora'], devices=[0],
distributed=False, dropout=0.5, dw='node_classification_dw', epochs=500, eval_step=1, hidden_size=64, load_emb_path=None, local_rank=0, log_path='.',
logger=None, lr=0.01, master_addr='localhost', master_port=13425, max_epoch=None, model=['gcn'], mw='node_classification_mw', n_trials=3,
n_warmup_steps=0, no_test=False, norm=None, nstage=1, num_classes=None, num_features=None, num_layers=2, patience=100, progress_bar='epoch',
project='cogdl-exp', residual=False, resume_training=False, rp_ratio=1, save_emb_path=None, seed=[1], split=[0], unsup=False, use_best_config=False,
weight_decay=0)

|-----------------------------------------------------------------------------------------|
    *** Running (`cora`, `gcn`, `node_classification_dw`, `node_classification_mw`)
|-----------------------------------------------------------------------------------------|
Downloading https://cloud.tsinghua.edu.cn/d/6808093f7f8042bfa1f0/files/?p=%2Fcora.zip&dl=1
unpacking cora.zip
Processing...
Done!
Model Parameters: 92231

Epoch: 146, train_loss:  0.0109, val_acc:  0.7860:  29%|███       | 147/500 [00:01<00:04, 74.92it/s]

Using time  0.0039
Saving 47-th model to ./checkpoints/model.pt ...
Loading model from ./checkpoints/model.pt ...
{'test_acc': 0.817, 'val_acc': 0.798}
| Variant         | test_acc      | val_acc       |
|-----------------|---------------|---------------|
| ('cora', 'gcn') | 0.8170±0.0000 | 0.7980±0.0000 |
```

Summary

- Machine Learning on Graphs
  - node-/link-/graph-level tasks
- Academic Graph Mining
- Basic Network Embedding
  - DeepWalk (random walk + skip-gram)
  - LINE (first-/second-proximities)
  - node2vec (biased random walk)
- Basic Introduction to CogDL
- Homework 1: DeepWalk (due by 17th July)
  - Find the homework material from the course website

Thank you !

Collaborators:

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

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

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

Yang Yang (**ZJU**)

Yukuo Cen, KEG, Tsinghua U.

https://github.com/THUDM/cogdl

Online Discussion Forum https://discuss.cogdl.ai/