KDD2017

Learning Representations of Large-Scale Networks

Jian Tang[1], Cheng Li[2], Qiaozhu Mei[2]

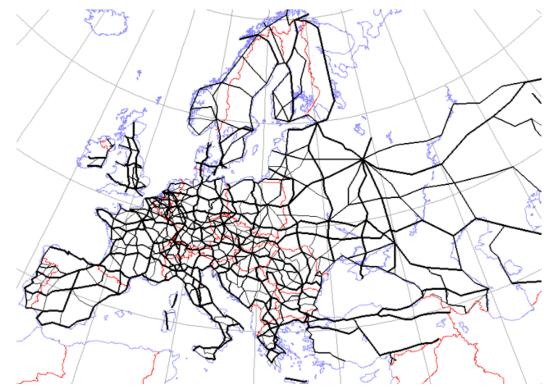
[1]HEC Montréal & Montréal Institute of Learning Algorithms (MILA)

[2]School of Information, University of Michigan

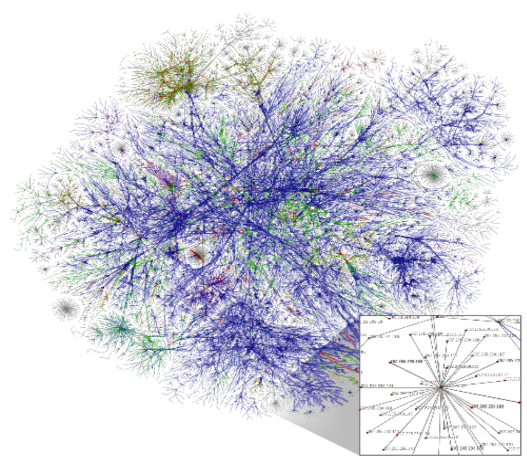
Networks

- Ubiquitous in real world

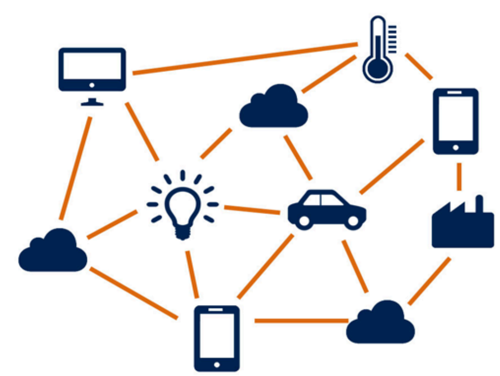
Social Network



Road Network



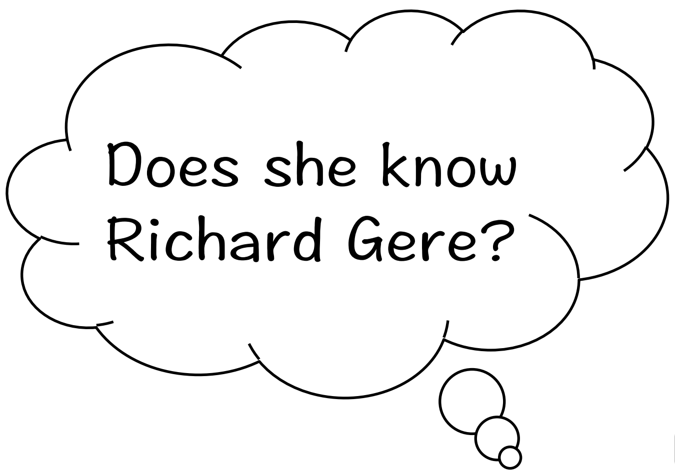
World Wide Web

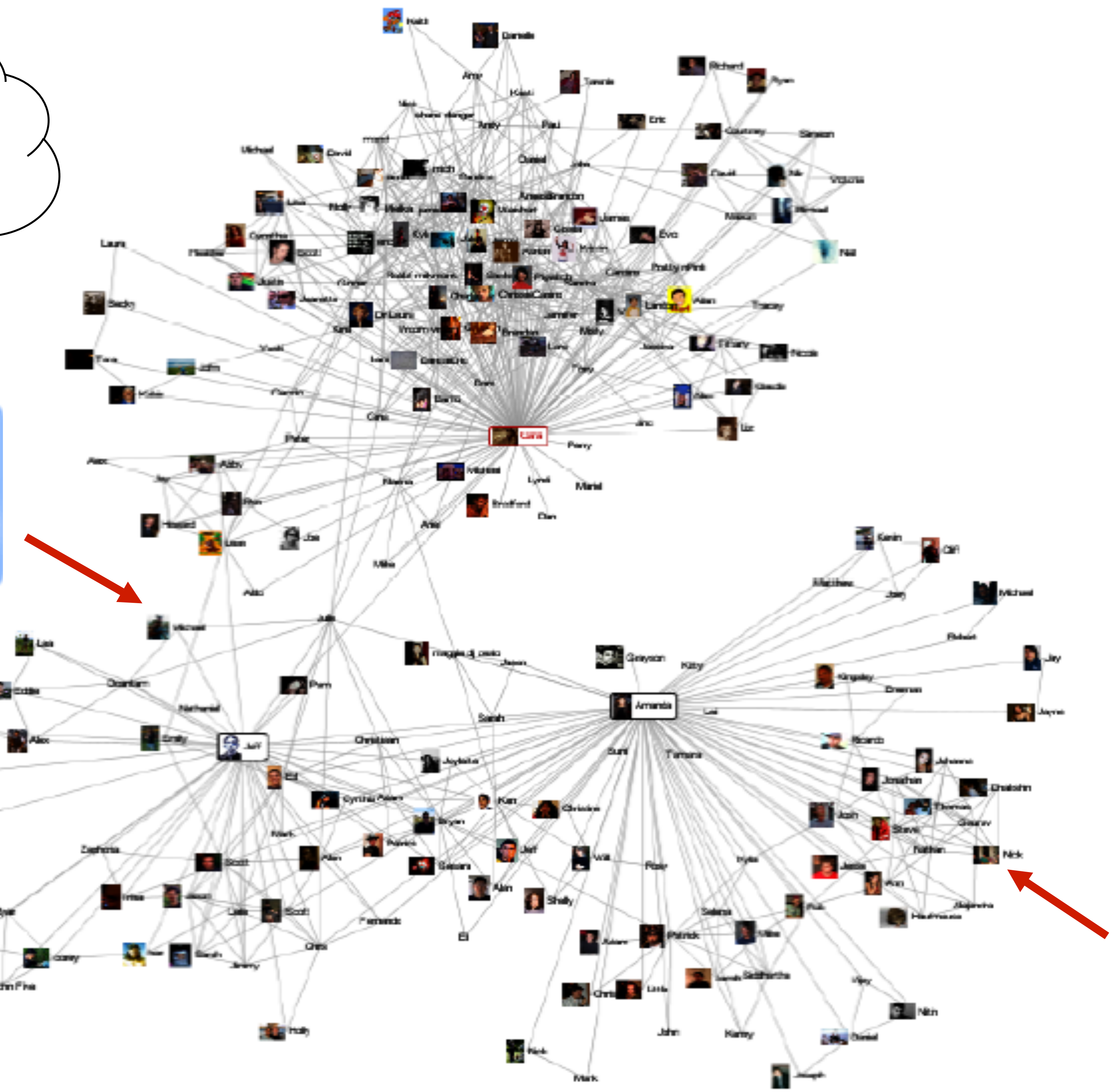


Internet-of-Things

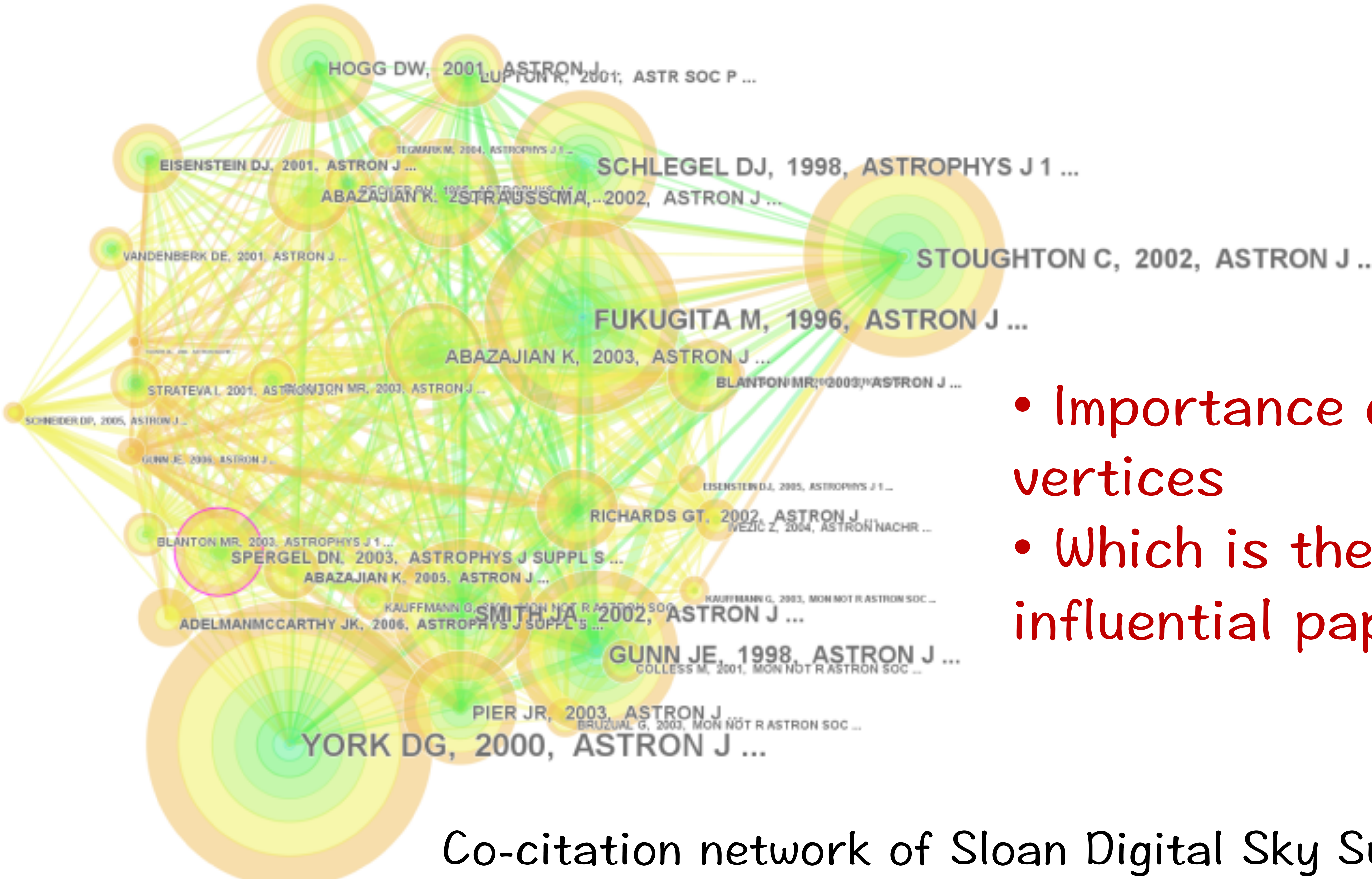
- A flexible and general data structure
  - Many types of data can be formulated as networks

Network Mining: Link Prediction





Network Mining: Ranking

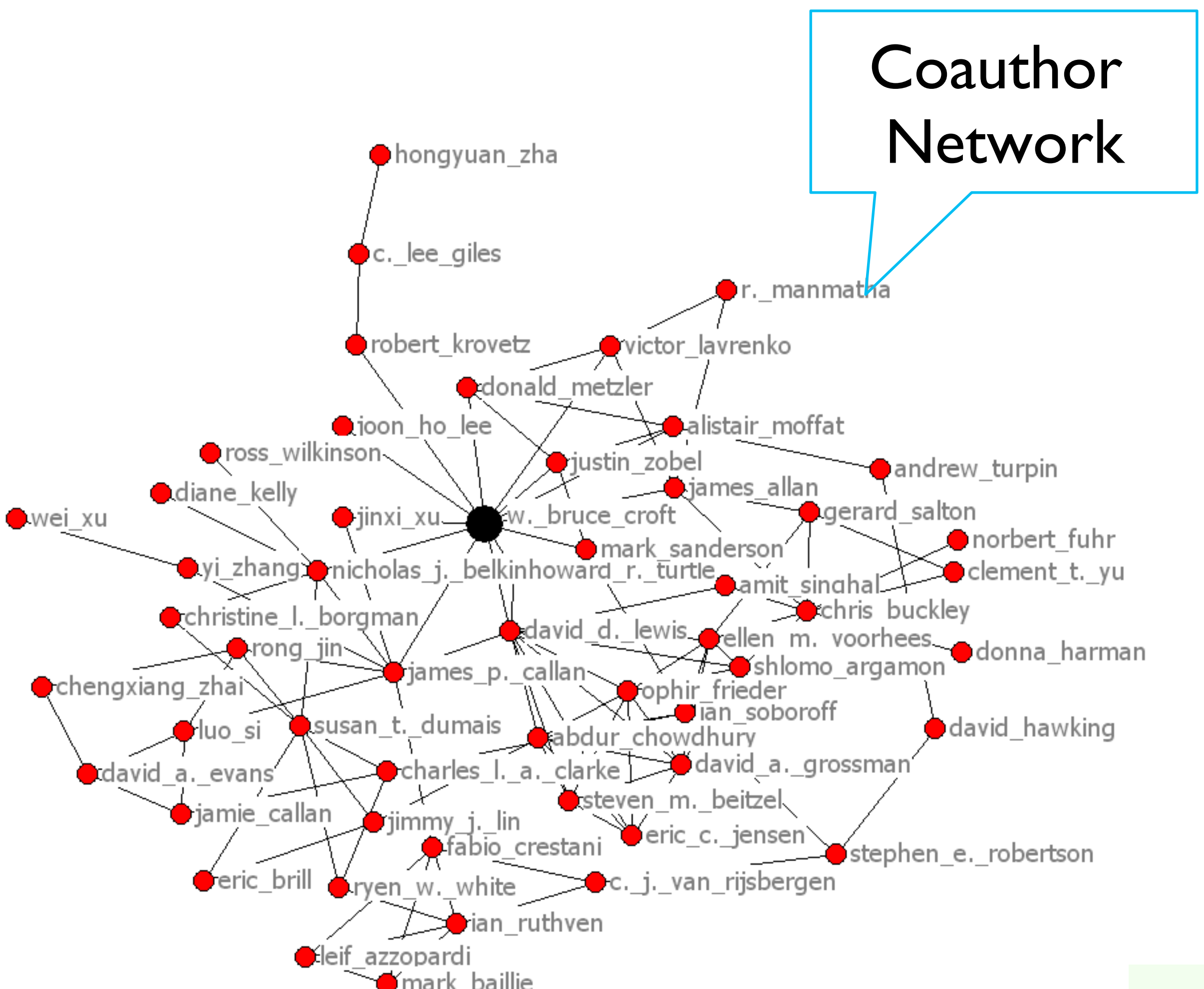


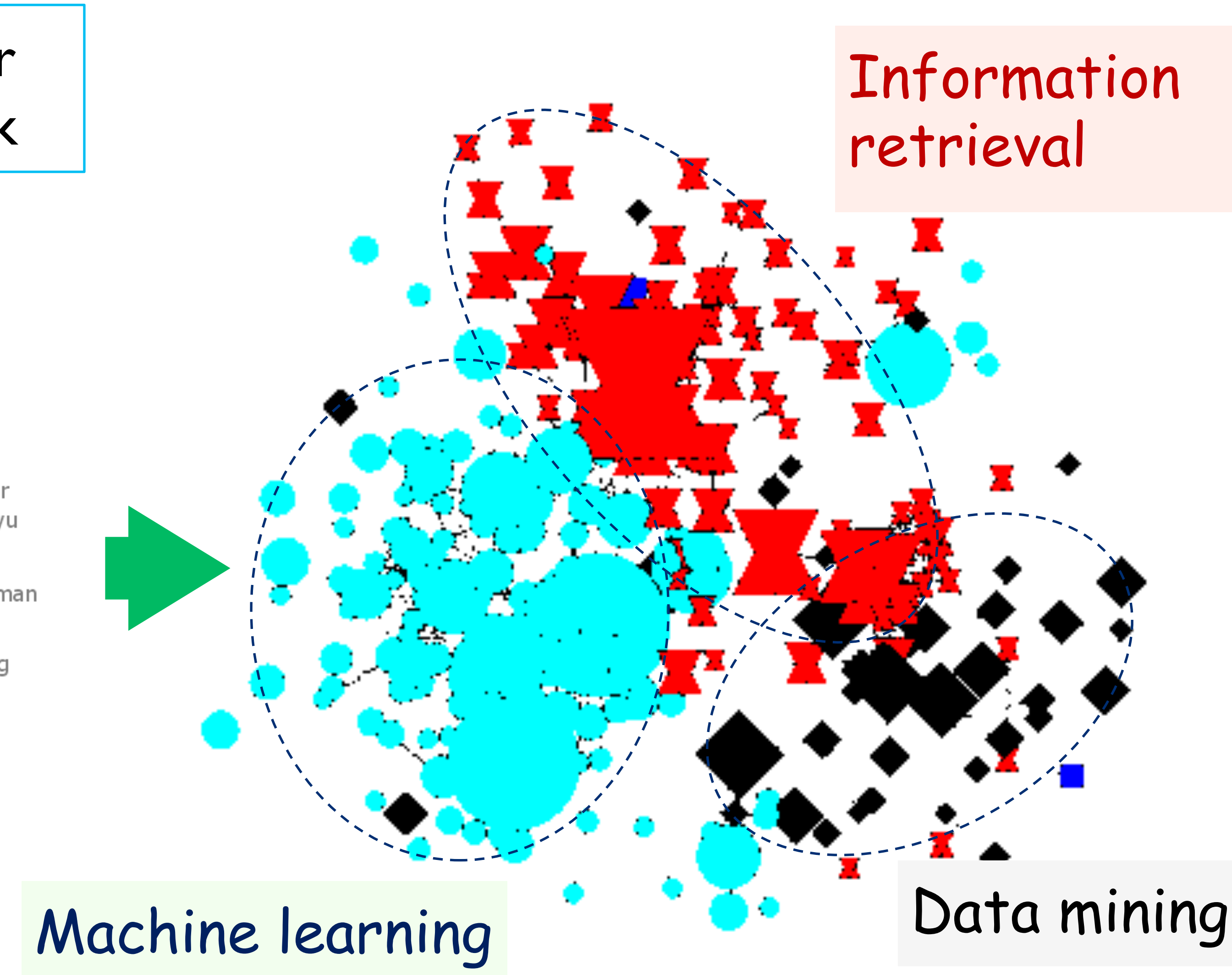
- Importance of vertices
- Which is the most influential paper?

Co-citation network of Sloan Digital Sky Survey

- http://nevac.ischool.drexel.edu/~james/infovis09/FP-tree-visual.html

Network Mining: Community Detection

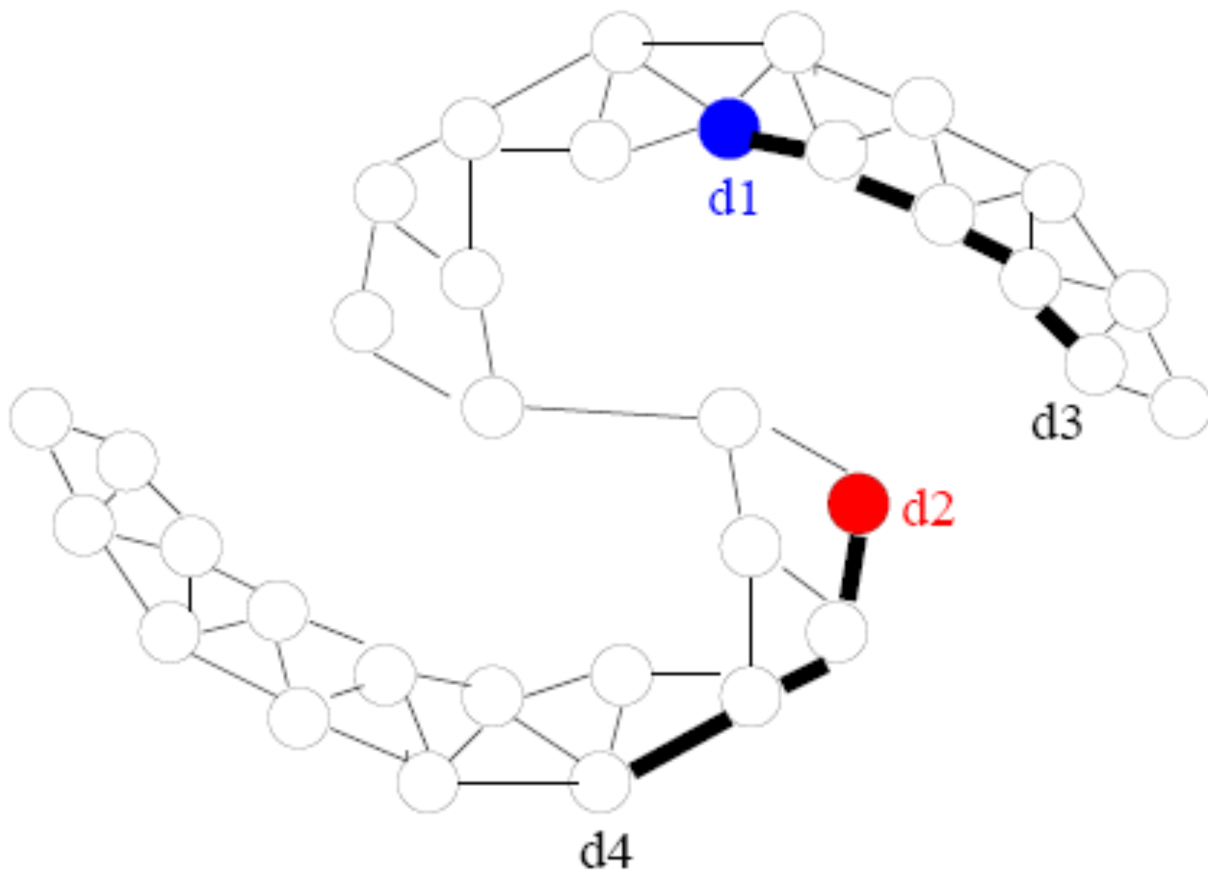




Who tend to work together?

- Q.Mei, D.Cai, D.Zhang, and C.Zhai, Topic Modeling with Hitting Time, WWW 2008

Network Mining: Classification

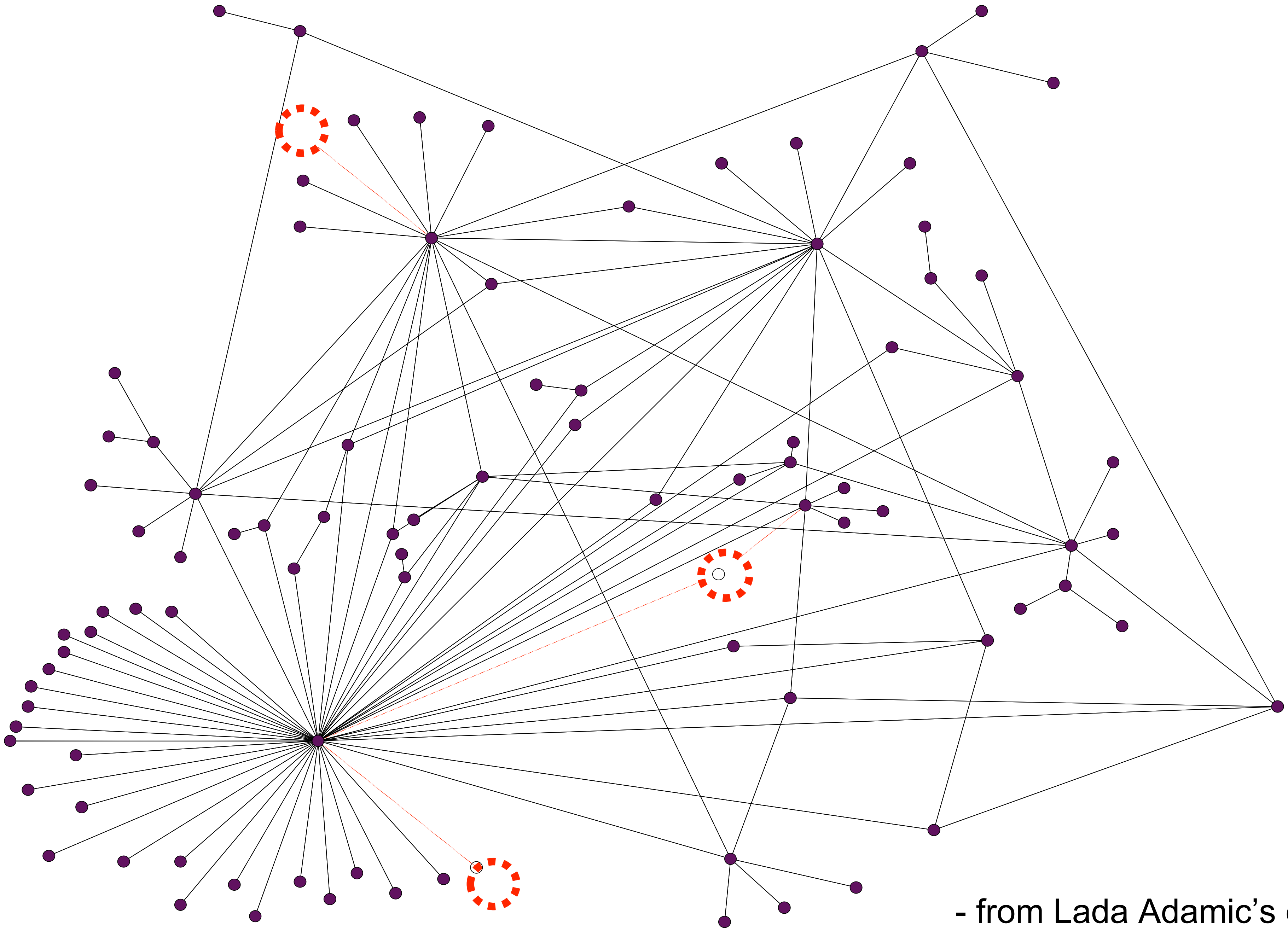


- d1 is democratic
- d2 is republican
- What can we say about d3 and d4?

- Graph from Jerry Zhu's tutorial in ICML 07

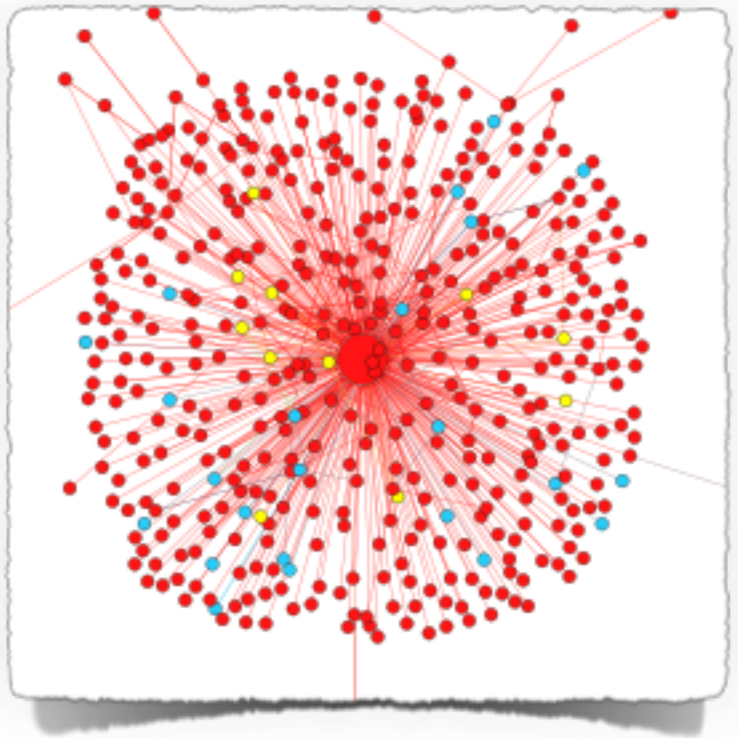
Network Mining: Resilience

How robust are networks to random/targeted attacks?

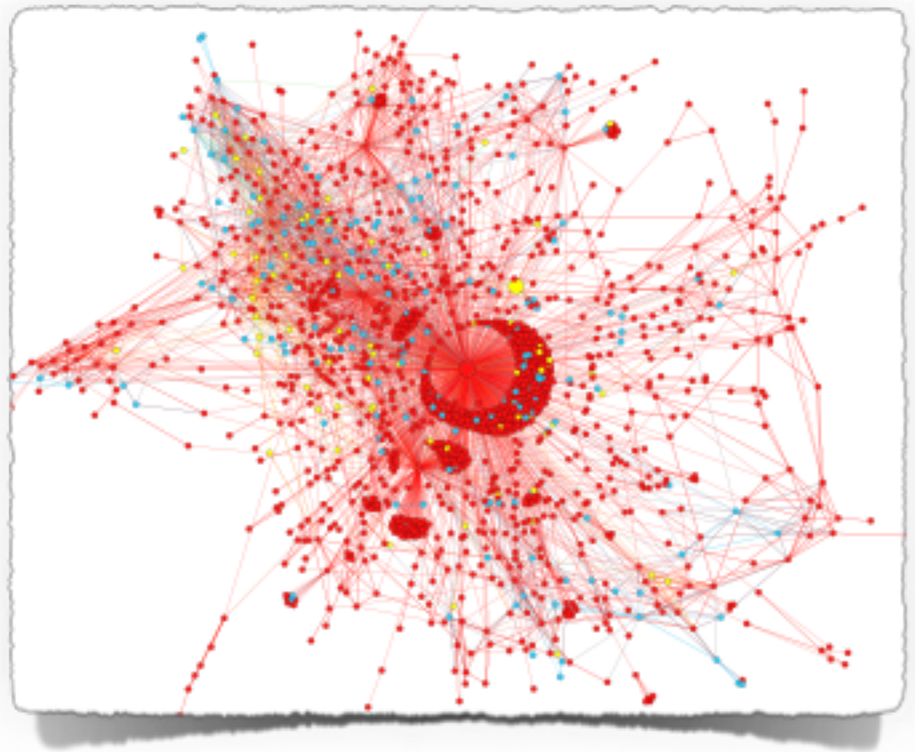


- from Lada Adamic's course

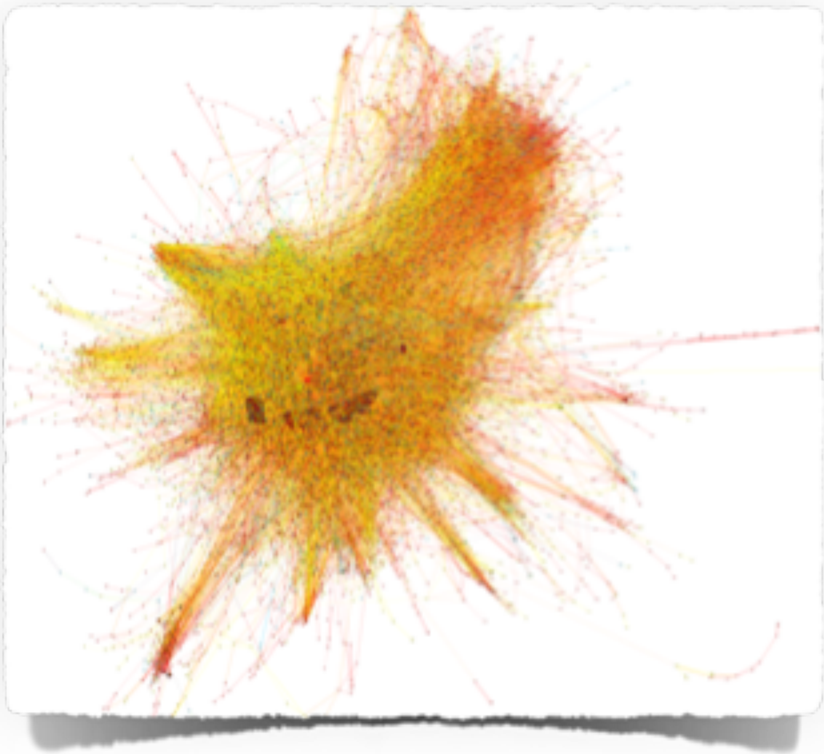
Network Mining: Information Cascades



60 seconds after



Two minutes before the official denial



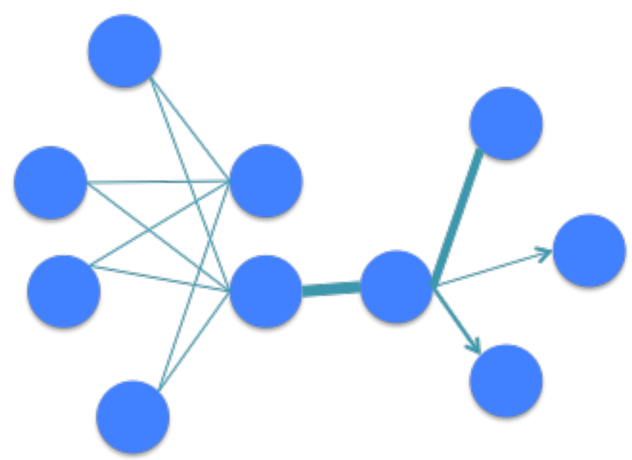
Three hours after

Cascade of the "white house bombing rumor" - Zhao et al., WWW 2015

Network Mining: Many Other Tasks

- Sampling
- Recommendation
- Structure analysis (e.g., structural holes)
- Evolution
- Matching
- Visualization
- ...

Traditional Representations of Networks



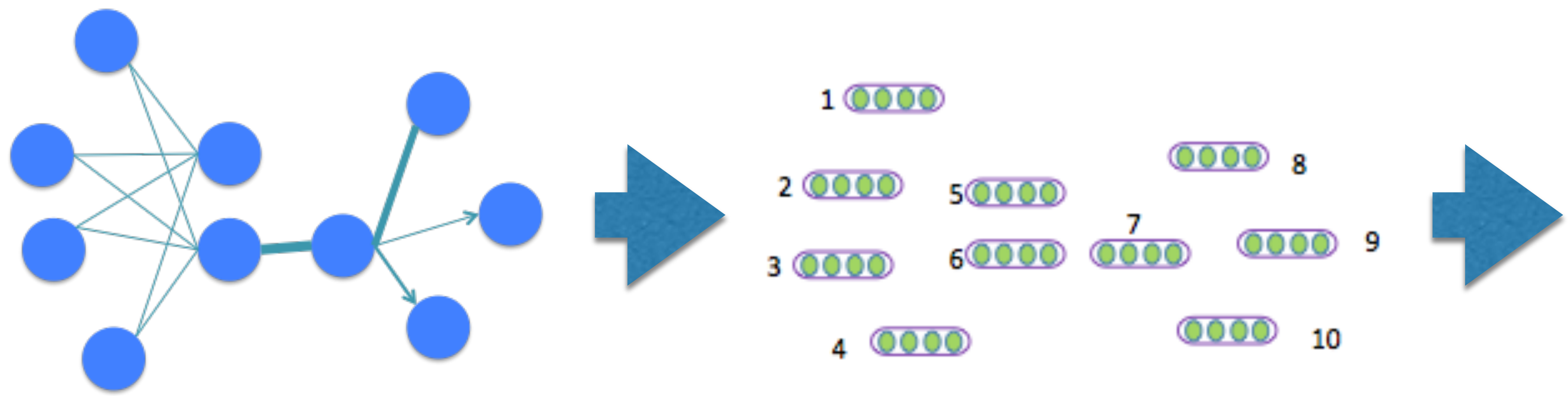
$$\begin{pmatrix} 0 & 0 & 1 & . & . & . & . & 2 & 1 & 1 \\ 1 & 0 & 1 & . & . & . & . & 0 & 0 & 1 \\ 3 & 2 & 0 & . & . & . & . & 0 & 0 & . \\ 1 & . & . & . & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & . & . & . & . & . & . & . & . & 1 \\ 0 & . & . & . & . & . & . & . & 1 & . \\ 0 & . & 0 & . & . & . & 0 & . & . & . \\ 0 & . & . & 1 & . & . & . & 0 & . & . \\ 0 & 0 & . & . & . & . & . & . & 0 & . \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

- Suffer from data sparsity
- Suffer from high dimensionality
- Does not facilitate computation
- Does not represent “semantics”
- ...

Research Question and Challenges

- How to effectively and efficiently **represent** networks?
- Challenges:
  - Large-scale: millions of nodes and billions of edges
  - Heterogeneous: directed/undirected, and binary/weighted

Learning Node Representations for Networks



Network

Node representations

- Node Classification
- Node Clustering
- Link Prediction
- Recommendation
- …

- E.g., Facebook social network -> user representations (features)-> friend recommendation

Text representation, e.g., word and document representation, …

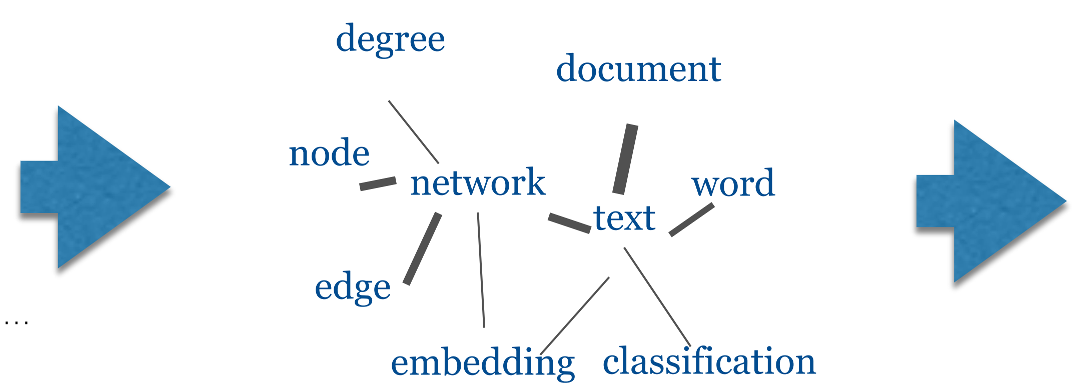
Deep learning has been attracting increasing attention …

A future direction of deep learning is to integrate unlabeled data …

…

The Skip-gram model is quite effective and efficient …

…

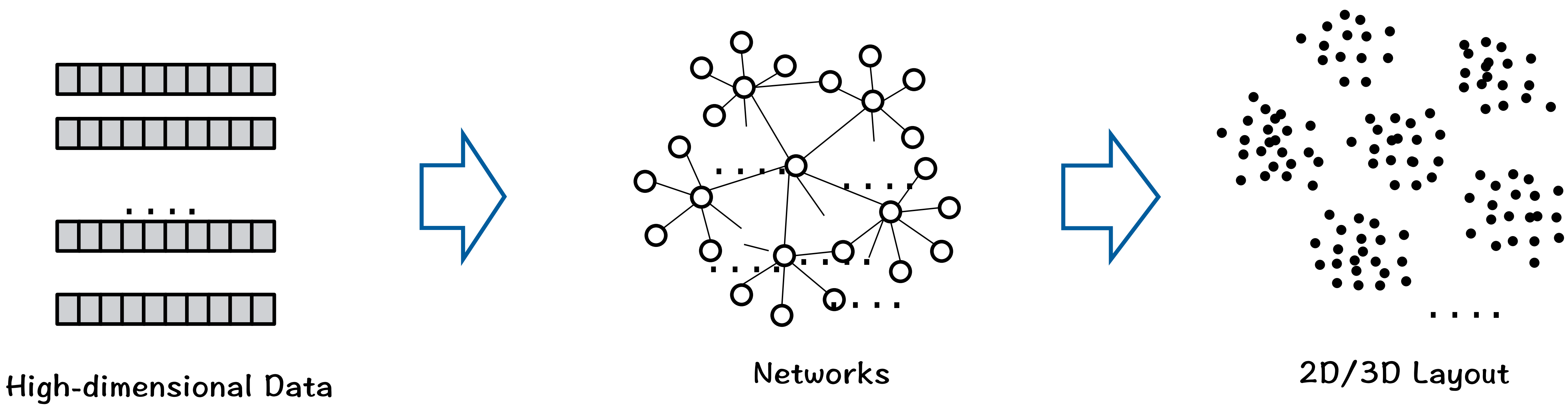


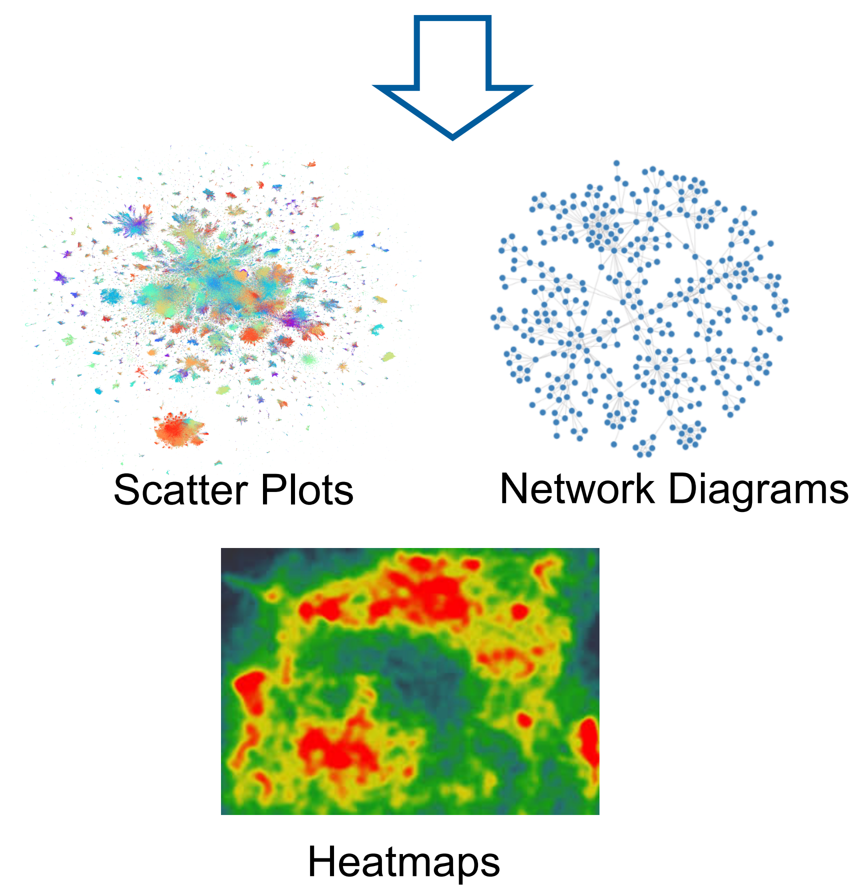
Unstructured text

Word co-occurrence network

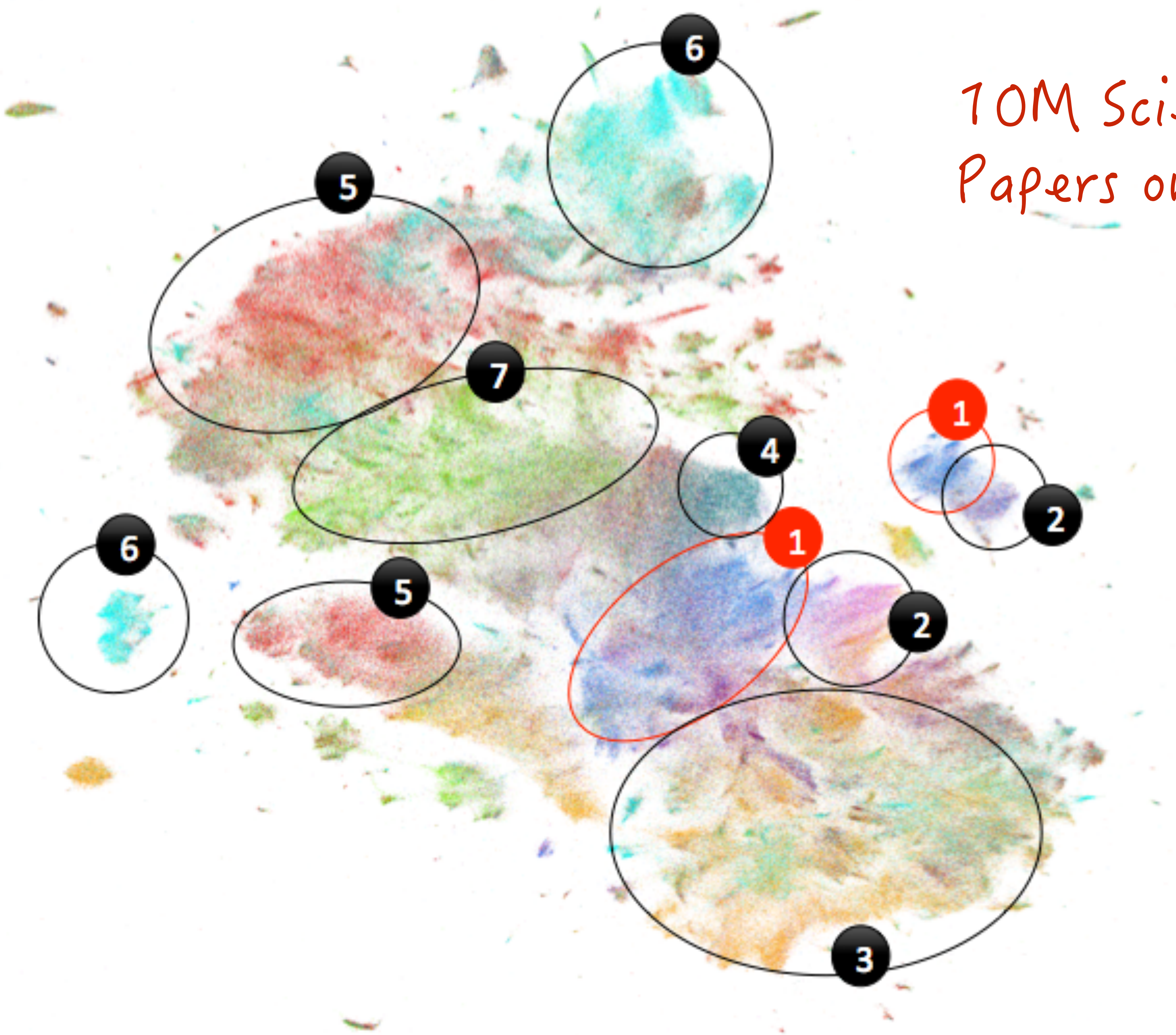
- Word representation

Extremely Low-dimensional Representations: 2D/3D for Visualizing Networks





Visualizing Scientific Papers



10M Scientific Papers on One Slide

1 Computer Science
2 Mathematics
3 Physics
4 Economics
5 Biology
6 Chemistry
7 Medicine

From Node Representation to Graph Representation

- Node representations are good for
  - Node classification
  - Recommendation
  - Link prediction
  - …
- How about …
  - Information cascades
  - Community detection
  - Protein function prediction
- We want to learn **graph** representations

Outline

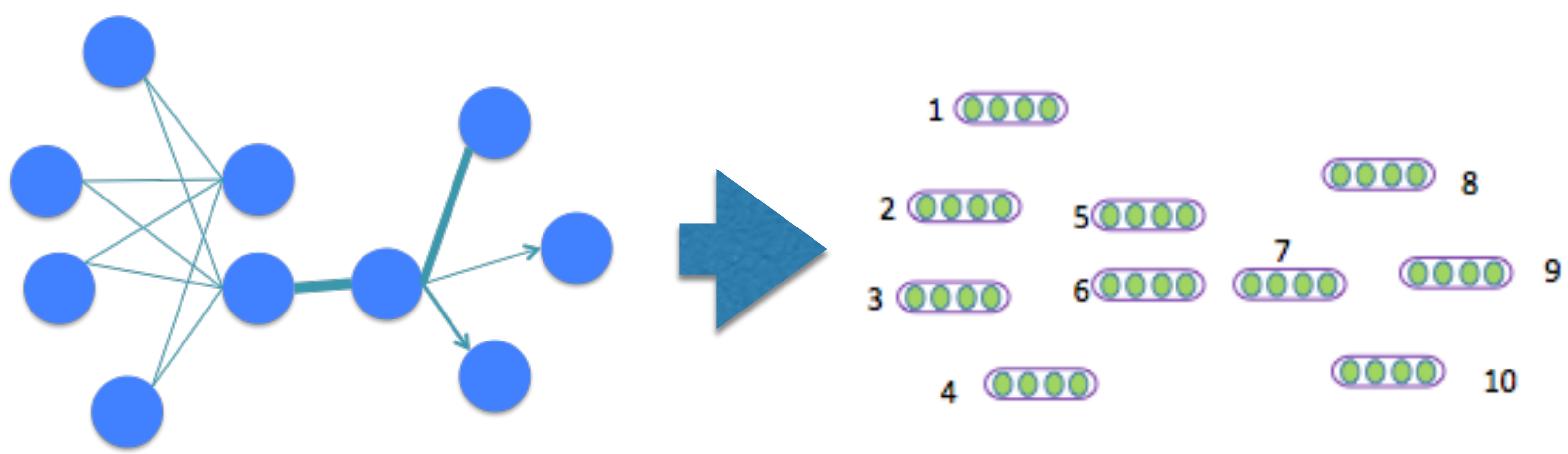
- **Part I: Learning Node Representations of Networks**
  - Related Work: Laplacian Eigenmap, Word2Vec
  - LINE, DeepWalk, and Node2Vec
  - Extensions
- **Part II: Visualizing Networks and High-Dimensional Data**
  - t-SNE
  - LargeVis
- **Pat III: Learning Representations of Entire Networks**
  - Graph kernels
  - End-to-end methods
- **Part IV: Summary, Challenges & Future Work**

Outline

- Part I: Learning Node Representations of Networks
  - Related Work: Laplacian Eigenmap, Word2Vec
  - LINE, DeepWalk, and Node2Vec
  - Extensions
- Part II: Visualizing Networks and High-Dimensional Data
  - t-SNE
  - LargeVis
- Pat III: Learning Representations of Entire Networks
  - Graph kernels
  - End-to-end methods
- Part IV: Summary, Challenges & Future Work

Problem Definition: Node Embedding

- Given a network/graph G=(V, E, W), where V is the set of nodes, E is the set of edges between the nodes, and W is the set of weights of the edges, the goal of ***node embedding*** is to represent each node *i* with ***a vector*** $\vec{u}_i \in R^d$, which preserves the structure of networks.



Networks

Node representations

Related Work

- Classical graph embedding algorithms
  - MDS, IsoMap, LLE, Laplacian Eigenmap, …
  - Hard to scale up
- Graph factorization (Ahmed et al. 2013)
  - Not specifically designed for network representation
  - Undirected graphs only
- Neural word embeddings (Bengio et al. 2003)
  - Neural language model
  - word2vec (skipgram), paragraph vectors, etc.

Laplacian Eigenmap (Belkin and Niyogi, 2003)

- Intuition: the embeddings of similar nodes should be close to each other
- Objective:

$$O = \frac{1}{2} \sum_{(i,j)\in E} w_{ij} (\vec{u}_i - \vec{u}_j)^2 = tr(U^T L U)$$

- Where $U = [\vec{u}_1, \vec{u}_2, \cdots, \vec{u}_N]$, L is the Laplacian matrix $L = D - W$, and $D_{ii} = \sum_j w_{ij}$
- Optimization by finding the eigenvectors of smallest eigenvalues of the Laplacian matrix L:

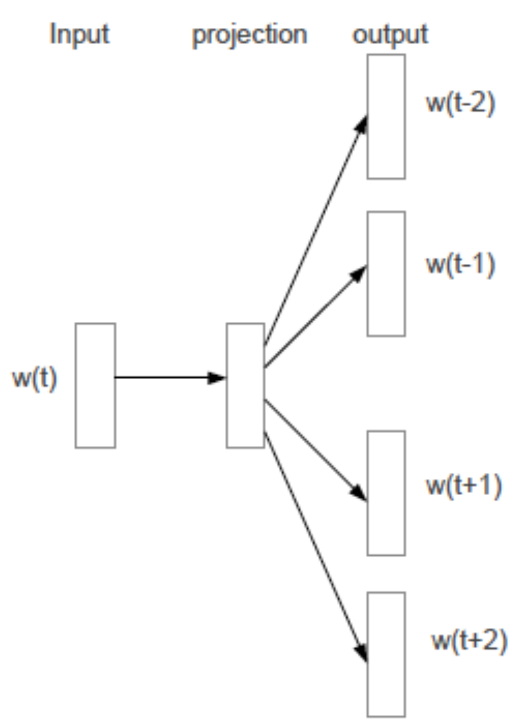
$$Lu = \lambda Du$$

- Computationally expensive for finding eigenvectors when networks are very big

Mikhail Belkin and Partha Niyogi. **Laplacian Eigenmaps for Dimensionality Reduction and Data Representation.** Neural Computation, 2003.

Word2VEC (Mikolov et al. 2014)

- Goal: represent each word *i* with a vector $\vec{v}_i \in R^d$ by training from a sequence $(w_1, w_{2,} \cdots, w_T)$
- Distributional hypothesis (John Rupert Firth): You know a word by the company it keeps
- Skip-gram: learning word representations by predicting the nearby words



$$p(w_O|w_I) = \frac{\exp\left({v'_{w_O}}^\top v_{w_I}\right)}{\sum_{w=1}^{W} \exp\left({v'_w}^\top v_{w_I}\right)}$$

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, Jeffrey Dean. **Distributed Representations of Words and Phrases and their Compositionality.** NIPS 2014

Skipgram

- Objective:

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c,j\neq 0}\log p(w_{t+j}|w_t)$$

- Where c is the window size
- Direct optimization is computationally expensive due to the softmax function

- Negative sampling:

$$\log \sigma({v'_{w_O}}^\top v_{w_I}) + \sum_{i=1}^{k}\mathbb{E}_{w_i\sim P_n(w)}\left[\log \sigma(-{v'_{w_i}}^\top v_{w_I})\right]$$

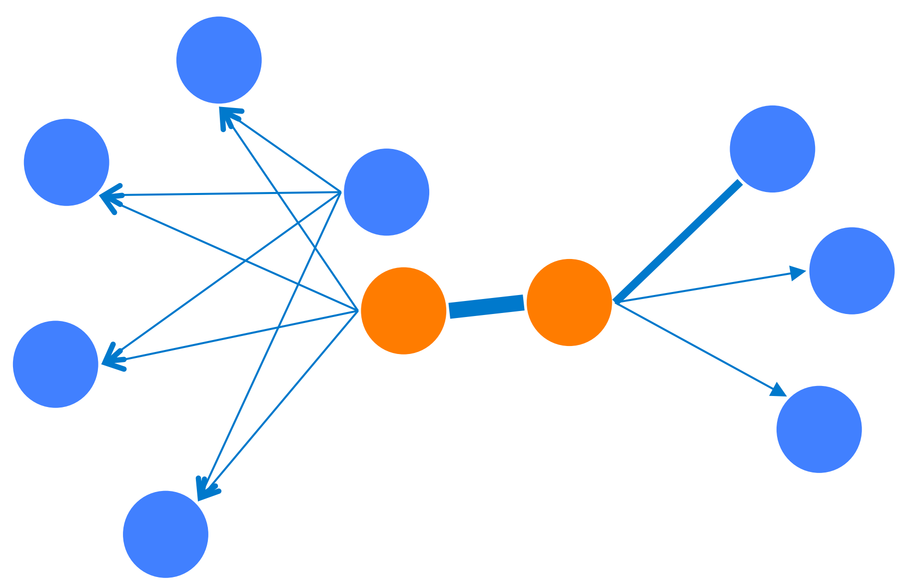
- Where $P_n(w)$ is a noisy distribution

LINE: Large-scale Information Network Embedding (Tang et al., **Most Cited Paper of WWW 2015**)

- Arbitrary types of networks
  - Directed, undirected, and/or weighted
- Clear objective function
  - Preserve the first-order and second-order proximity
- Scalable
  - Asynchronous stochastic gradient descent
  - Millions of nodes and billions of edges: a coupe of hours on a single machine

Jian Tang, Meng Qu, Mingzhe Wang, Jun Yan, and Qiaozhu Mei. **LINE: Large-scale Information Network Embedding**. WWW'15

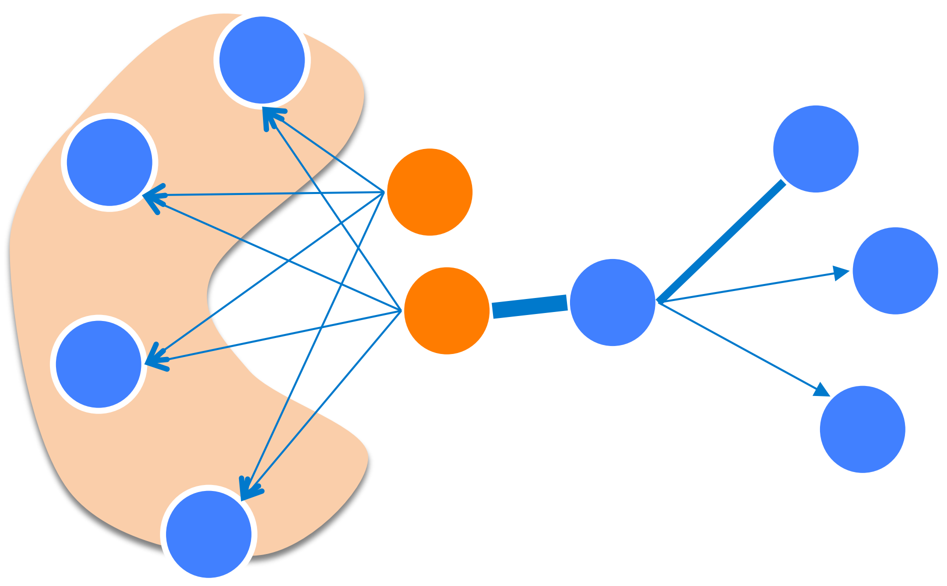
First-order Proximity



- The local pairwise proximity between the nodes
- However, many links between the nodes are not observed
  - Not sufficient for preserving the entire network structure

Second-order Proximity

"The degree of overlap of two people's friendship networks correlates with the strength of ties between them" --Mark Granovetter



"You shall know a word by the company it keeps" --John Rupert Firth

- Proximity between the neighborhood structures of the nodes

Preserving the First-order Proximity (LINE 1st)

- Distributions: : (defined on the undirected edge i - j)

Empirical distribution of first-order proximity:

$$\hat{p}_1(v_i, v_j) = \frac{w_{ij}}{\sum_{(m,n)\in E} w_{mn}}$$

Model distribution of first-order proximity:

$$p_1(v_i, v_j) = \frac{\exp(\vec{u}_i^T \vec{u}_j)}{\sum_{(m,n)\in V\times V} \exp(\vec{u}_m^T \vec{u}_n)}$$

$\vec{u}_i$: Embedding of i

- Objective:

$$O_1 = KL(\hat{p}_1, p_1) = -\sum_{(i,j)\in E} w_{ij} \log p_1(v_i, v_j)$$

Preserving the Second-order Proximity (LINE 2nd)

- Distributions: (defined on the directed edge i -> j)

Empirical distribution of neighborhood structure:

$$\hat{p}_2(v_j \mid v_i) = \frac{w_{ij}}{\sum_{k \in V} w_{ik}}$$

Model distribution of neighborhood structure:

$$p_2(v_j \mid v_i) = \frac{\exp(\vec{u}'^T_i \vec{u}_j)}{\sum_{k \in V} \exp(\vec{u}'^T_k \vec{u}_i)}$$

- Objective:

$$O_2 = \sum_i KL(\hat{p}_2(\cdot \mid v_i), p_2(\cdot \mid v_i)) = -\sum_{(i,j) \in E} w_{ij} \log p_2(v_j \mid v_i)$$

Optimization Tricks

- Stochastic gradient descent + Negative Sampling
  - Randomly sample an edge and multiple negative edges

- The gradient w.r.t the embedding with edge (i, j)

$$\frac{\partial O_2}{\partial \vec{u}_i} = w_{ij} \frac{\partial \log \hat{p}_2(v_j \mid v_i)}{\partial \vec{u}_i}$$

- Problematic when the variances of weights of the edges are large
  - The variance of the gradients are large

- Solution: edge sampling
  - Sample the edges according to their weights and treat the edges as binary

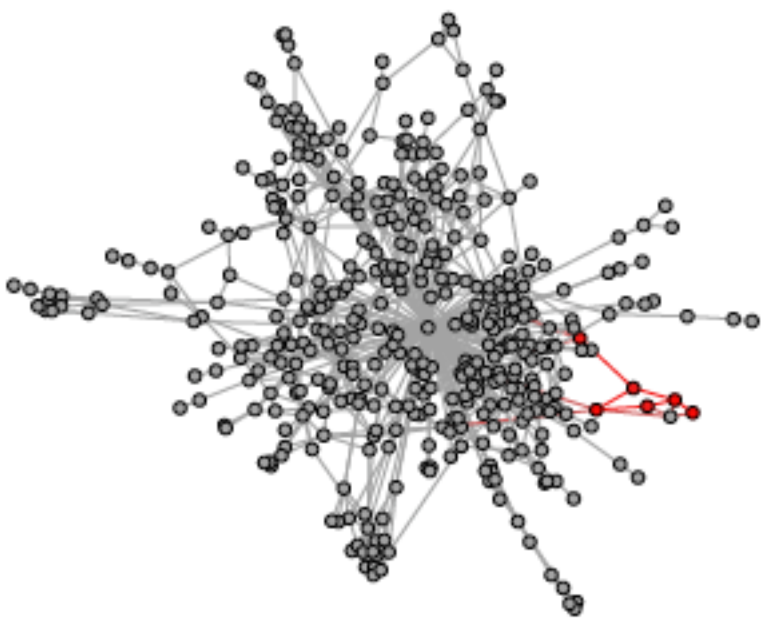
- Complexity: O(d*K*|E|)
  - Linear to the dimensionality d, the number of negative samples K, and the number of edges

Discussion

- Embed nodes with few neighbors
  - Expand the neighbors by adding higher-order neighbors
  - **Breadth-first search (BFS)**
  - Adding only second-order neighbors works well in most cases
- Embed new nodes
  - Fix the embeddings of existing nodes
  - Optimize the objective w.r.t. the embeddings of new nodes

DeepWalk (Perozzi et al. 2014)

- Learning node representations with the technique for learning word representations, i.e., Skipgram
- Treat *random walks on networks* as *sentences*



Random walk generation (generate node contexts through random search)

$$\begin{array}{l} 3 \\ 1\,]\,{\scriptstyle v_j} \\ 5 \\ 1 \\ \vdots \end{array}$$

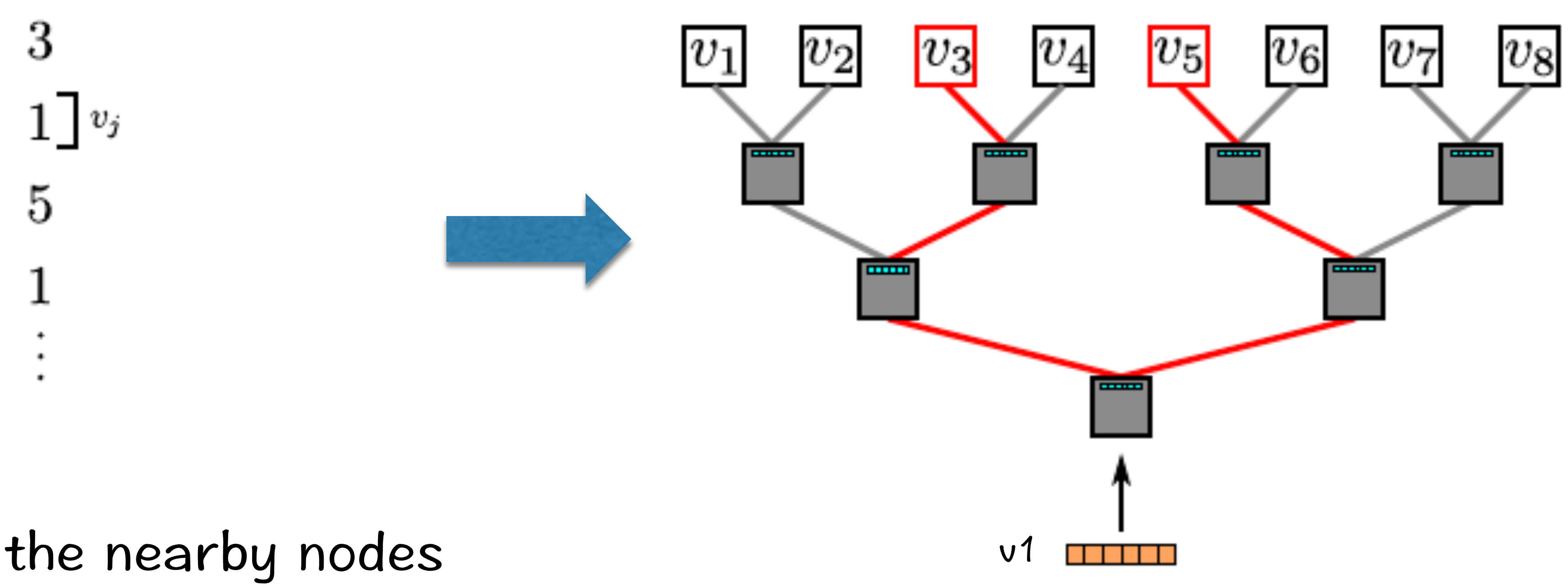
$$p(v_j \mid v_i) = \frac{\exp(\vec{u}'^{T}_{i}\,\vec{u}_j)}{\sum_{k \in V} \exp(\vec{u}'^{T}_{k}\,\vec{u}_i)}$$

Predict the nearby nodes in the random walks

Bryan Perozzi, Rami Al-Rfou, Steven Skiena. DeepWalk: Online Learning of Social Representations. KDD'14

DeepWalk (Perozzi et al. 2014)

- Optimization: hierarchical softmax (Morin, Bengio, 2005)
- Assign the nodes to the leaves of a binary tree
- Predict the node => predict a path in the tree
  - Make binary decisions along the path
- Complexity from |V| to log(|V|)



Predict the nearby nodes
in the random walks
(v1->v3, v1->v5)

Hierarchical softmax

Node2Vec (Grover and Leskovec, 2016)

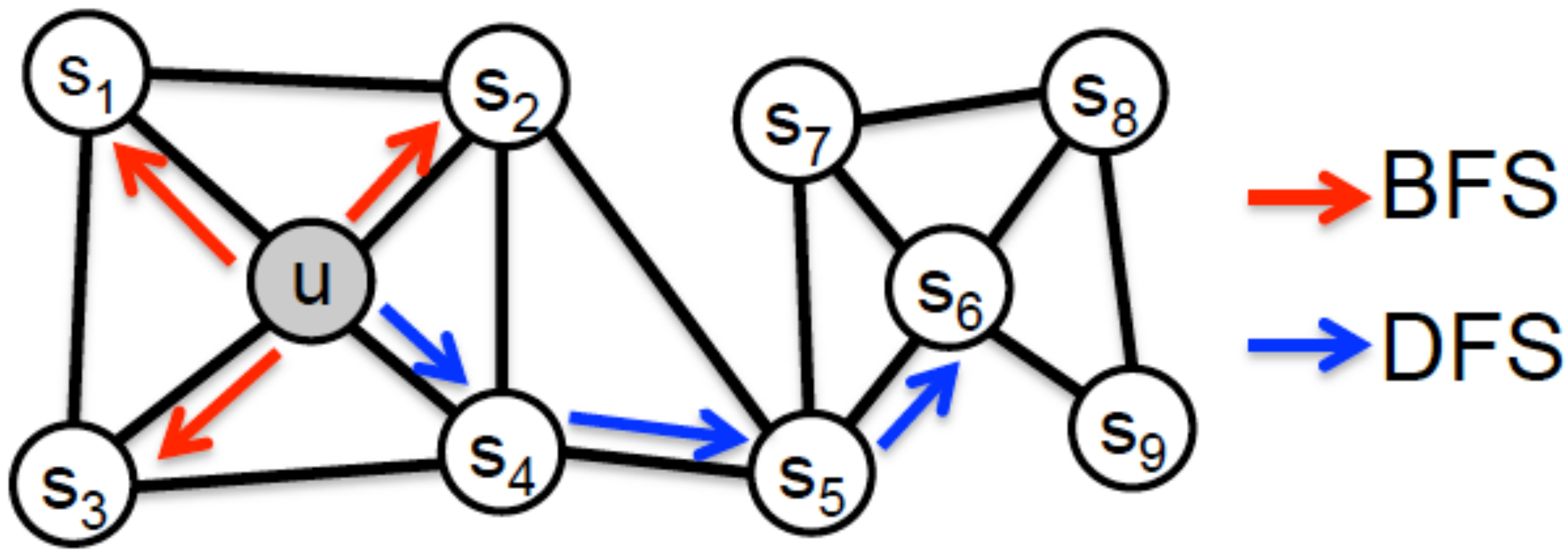
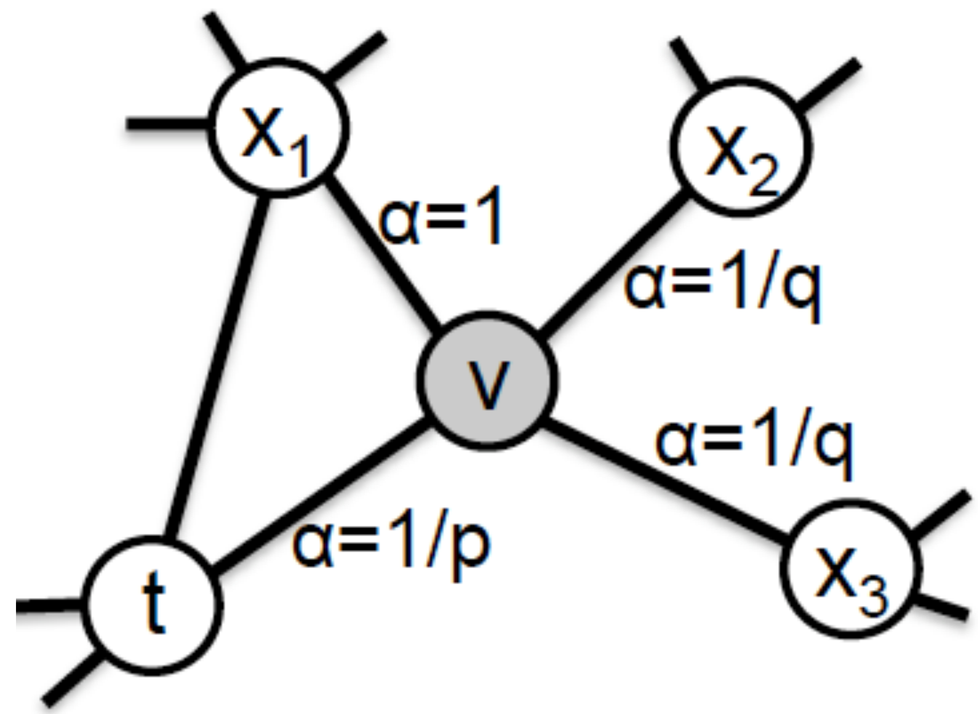


Figure 1: BFS and DFS search strategies from node u ($k = 3$).

- Find the node context by a hybrid strategy of
  - Breadth-first Sampling (BFS): **homophily**
  - Depth-first Sampling (DFS): **structural equivalence**

Aditya Grover and Jure Leskovec. node2vec: Scalable Feature Learning for Networks. KDD'16

Expand Node Contexts with Biased Random Walk



- Biased random walk with two parameters p and q
  - p: controls the probability of revisiting a node in the walk
  - q: controls the probability of exploring "outward" nodes
  - Find optimal p and q through cross-validation on labeled data
- Optimized through similar objective as LINE with first-order proximity

Comparison between LINE, DeepWalk, and Node2Vec

Algorithm	Neighbor Expansion	Proximity	Optimization	Labeled Data
LINE	BFS	1^{st} or 2^{nd}	Negative Sampling	No
DeepWalk	Random	2^{nd}	Hierarchical Softmax	No
Node2Vec	BFS + DFS	1^{st}	Negative Sampling	Yes

Applications

- Node classification (Perozzi et al. 2014, Tang et al. 2015a, Grover et al. 2015)
- Node visualization (Tang et al. 2015a)
- Link prediction (Grover et al. 2015)
- Recommendation (Zhao et al. 2016)
- Text representation (Tang et al. 2015a, Tang et al. 2015b)
- ...

Node Classification

- social network => user representations (features) => classifier
- Community identities as classification labels

	% Labeled Nodes	1%	2%	3%	4%	5%	6%	7%	8%	9%	10%
Micro-F1(%)	DeepWalk	**32.4**	**34.6**	**35.9**	**36.7**	**37.2**	**37.7**	**38.1**	**38.3**	**38.5**	**38.7**
	SpectralClustering	27.43	30.11	31.63	32.69	33.31	33.95	34.46	34.81	35.14	35.41
	EdgeCluster	25.75	28.53	29.14	30.31	30.85	31.53	31.75	31.76	32.19	32.84
	Modularity	22.75	25.29	27.3	27.6	28.05	29.33	29.43	28.89	29.17	29.2
	wvRN	17.7	14.43	15.72	20.97	19.83	19.42	19.22	21.25	22.51	22.73
	Majority	16.34	16.31	16.34	16.46	16.65	16.44	16.38	16.62	16.67	16.71

Table: Results on Flickr Network (Perozzi et al. 2014)

DeepWalk > Laplacian Eigenmap

Node Classification

- social network => user representations (features) => classifier
- Community identities as classification labels

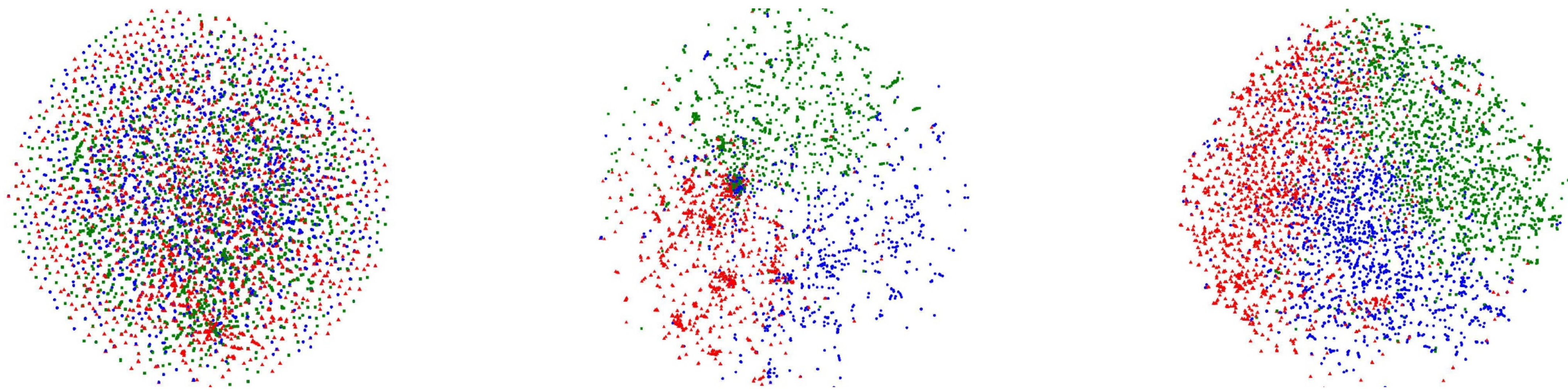
Metric	Algorithm	1%	2%	3%	4%	5%	6%	7%	8%	9%	10%
Micro-F1	GF	25.43 (24.97)	26.16 (26.48)	26.60 (27.25)	26.91 (27.87)	27.32 (28.31)	27.61 (28.68)	27.88 (29.01)	28.13 (29.21)	28.30 (29.36)	28.51 (29.63)
	DeepWalk	39.68	41.78	42.78	43.55	43.96	44.31	44.61	44.89	45.06	45.23
	DeepWalk(256dim)	39.94	42.17	43.19	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****)

Table: Results on Youtube Network(Tang et al. 2015a)

LINE(1^{st} + 2^{nd}) >LINE(2^{nd})> DeepWalk > LINE(1^{st})

Node Visualization (Tang et al. 2015a)

- Coauthor network: authors from three different research fields



● "Data mining" ● "Machine learning" ● "Computer vision"

(a) Graph factorization (b) DeepWalk (c) LINE(2^{nd})

Link Prediction (Grover and Leskovec, 2016)

Op	Algorithm	Dataset		
		Facebook	PPI	arXiv
	Common Neighbors	0.8100	0.7142	0.8153
	Jaccard's Coefficient	0.8880	0.7018	0.8067
	Adamic-Adar	0.8289	0.7126	0.8315
	Pref. Attachment	0.7137	0.6670	0.6996
(a)	Spectral Clustering	0.5960	0.6588	0.5812
	DeepWalk	0.7238	0.6923	0.7066
	LINE	0.7029	0.6330	0.6516
	node2vec	0.7266	0.7543	0.7221
(b)	Spectral Clustering	0.6192	0.4920	0.5740
	DeepWalk	**0.9680**	0.7441	0.9340
	LINE	0.9490	0.7249	0.8902
	node2vec	**0.9680**	**0.7719**	**0.9366**

Table: Results of Link Prediction

Node Embeddings (LINE, DeepWalk, node2vec)
> Jaccard's Coefficient > Adamic-Adar

Unsupervised Text Representation (Tang et al. 2015a)

- Construct text networks from unstructured text

Text representation, e.g., word and document representation, …

Deep learning has been attracting increasing attention …

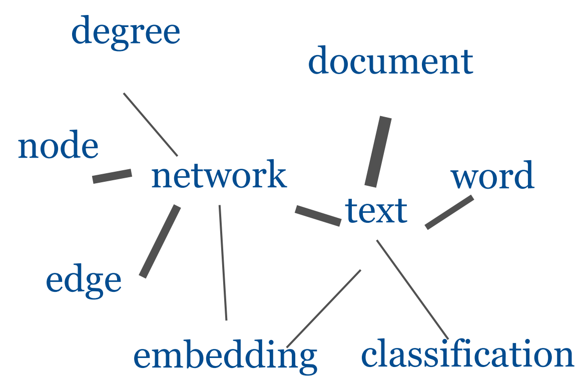
A future direction of deep learning is to integrate unlabeled data …

…

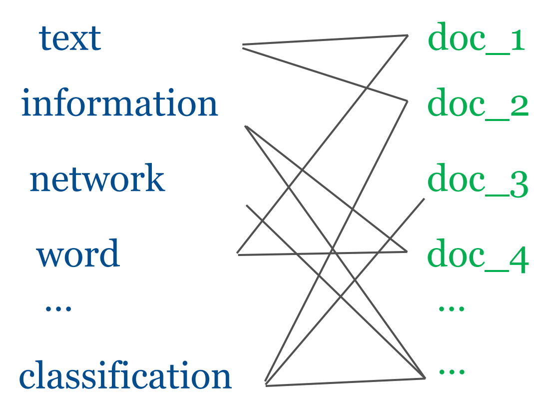
The Skip-gram model is quite effective and efficient …

Information networks encode the relationships between the data objects …

Unstructured text



Word co-occurrence network



Word-document network

Word Analogy

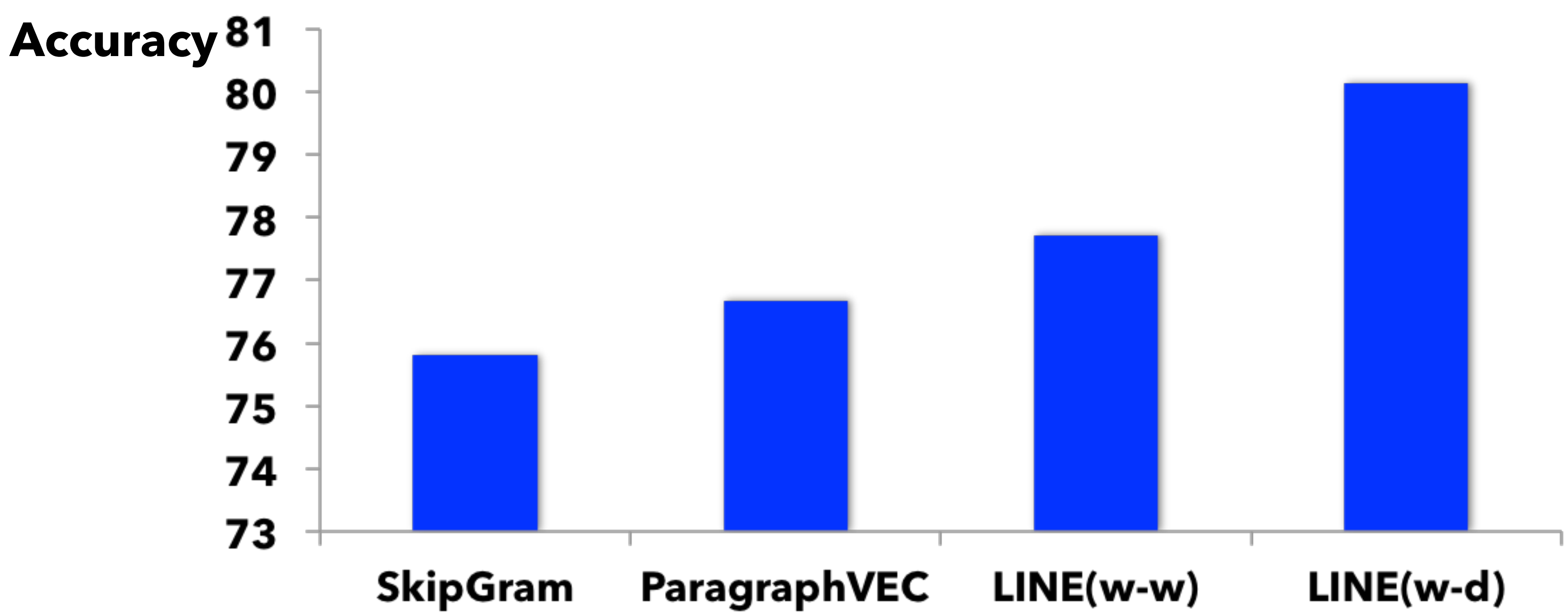
- Entire Wikipedia articles => word co-occurrence network (~2M words, 1B edges)
- Size of word co-occurrence networks does not grow linearly with data size
  - Only the weights of edges change

Algorithm	Semantic(%)	Syntactic(%)	Overall
GF	61.38	44.08	51.93
SkipGram	69.14	57.94	63.02
LINE(1^{st})	58.08	49.42	53.35
LINE(2^{nd})	73.79	59.72	66.10

LINE(2^{nd}) > LINE(1^{st})
LINE(2^{nd}) > SkipGram

Text Classification (on Long Documents)

- Word co-occurrence network (w-w) , word-document network (w-d) to learn the word embedding
- Document embedding as average of word embeddings in the document

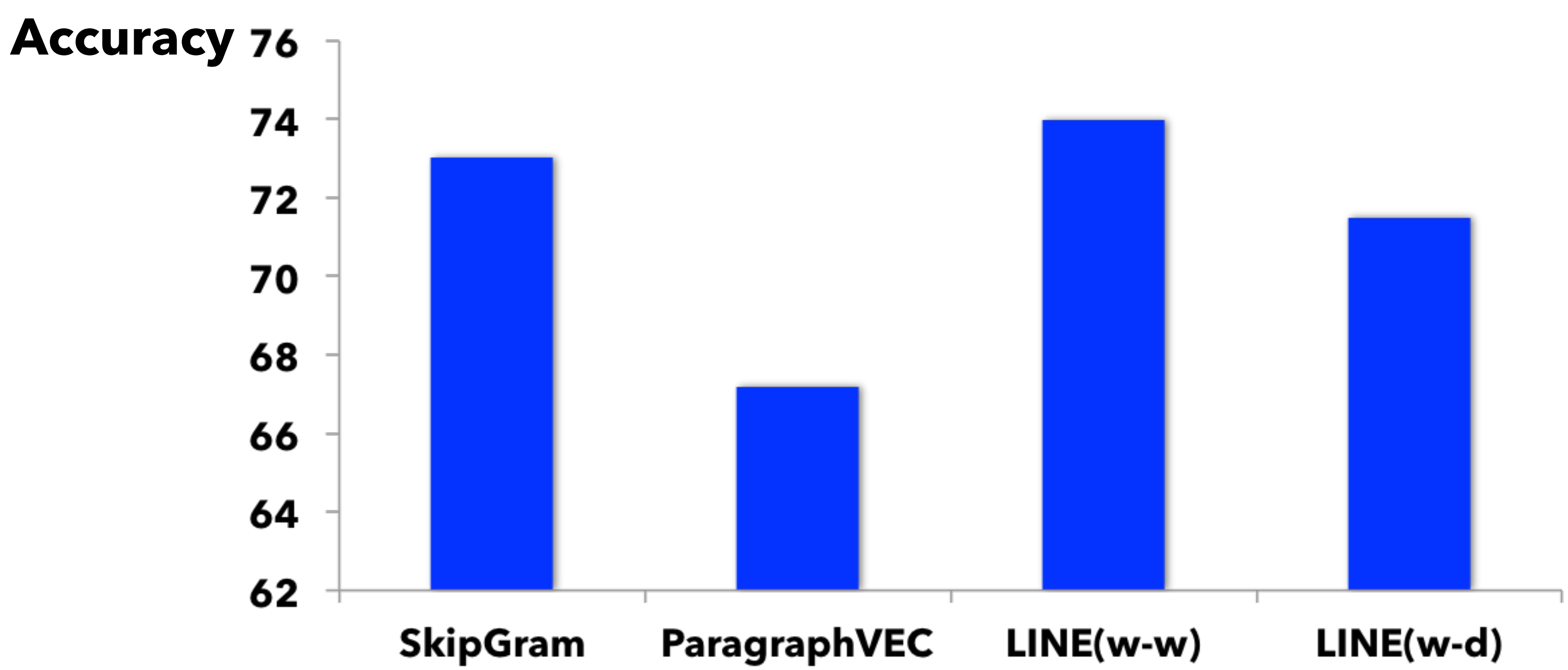


LINE(w-w) > SkipGram LINE(w-d) > ParagraphVEC

LINE(w-d) > LINE(w-w)

Text Classification (on Short Documents)

- Word co-occurrence network (w-w) , word-document network (w-d) to learn the word embedding
- Document embedding as average of word embeddings in the document



LINE(w-w) > SkipGram LINE(w-d) > ParagraphVEC

LINE(w-w) > LINE(w-d)

Extensions

- Other variants
- Multi-view networks
- Networks with node attributes
- Heterogeneous networks
- Task-specific network embedding

Extensions

- Other variants
- Multi-view networks
- Networks with node attributes
- Heterogeneous networks
- Task-specific network embedding

Other Variants

- Leverage global structural information (Cao et al. 2015)
- Non-linear methods based on autoencoders (Wang et al. 2016)
- Directed network embedding (Ou et al. 2016)
- Signed network embedding (Wang et al. 2017)

- Shaosheng Cao, Wei Lu, and Qiongkai Xu. GraRep: Learning graph representations with global structural information. CIKM' 2015.
- Mingdong Ou, Peng Cui, Jian Pei, Wenwu Zhu. Asymmetric transitivity preserving graph embedding. KDD, 2016.
- Daixing Wang, Peng Cui, Wenwu Zhu. Structural deep network embedding. KDD, 2016.
- Suhang Wang, Jiliang Tang, Charu Aggarwal, Yi Chang, Huan Liu. Signed network embedding in social media. SDM 2017.

Extensions

- Other variants
- Multi-view networks
- Networks with node attributes
- Heterogeneous networks
- Task-specific network embedding

Multi-view Network Embedding (Qu and Tang et al. 2017)

- Multiple types of relationships between nodes exist in real-world networks
  - E.g., following, retweeting relationships between users in Twitter
  - Each type of relationship => a view of the network
  - Multiple types of relationships => multi-view networks
- Infer **robust** node representations with multiple views
  - Complementary information in different views

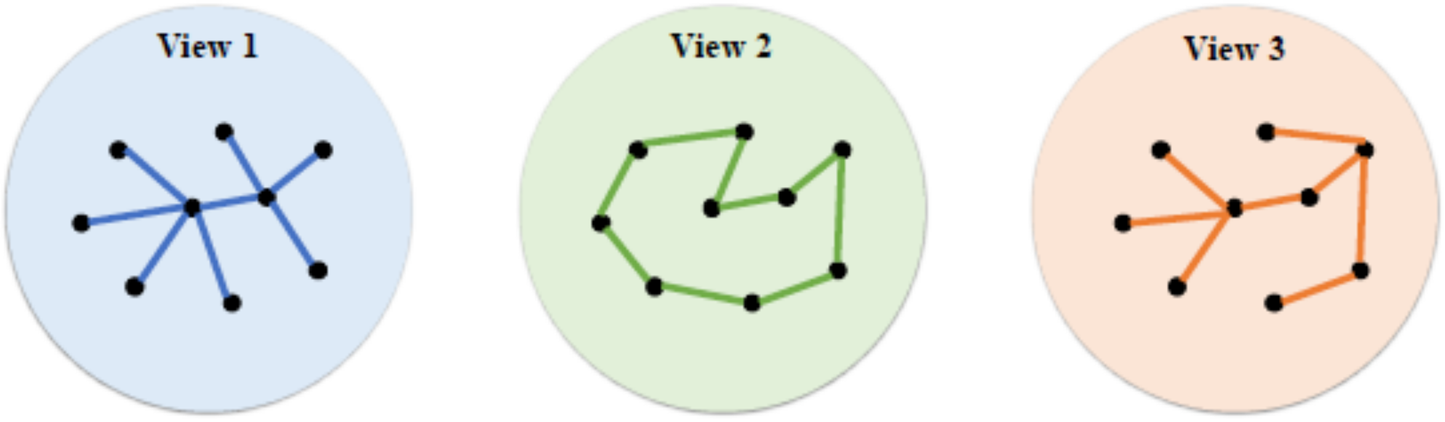
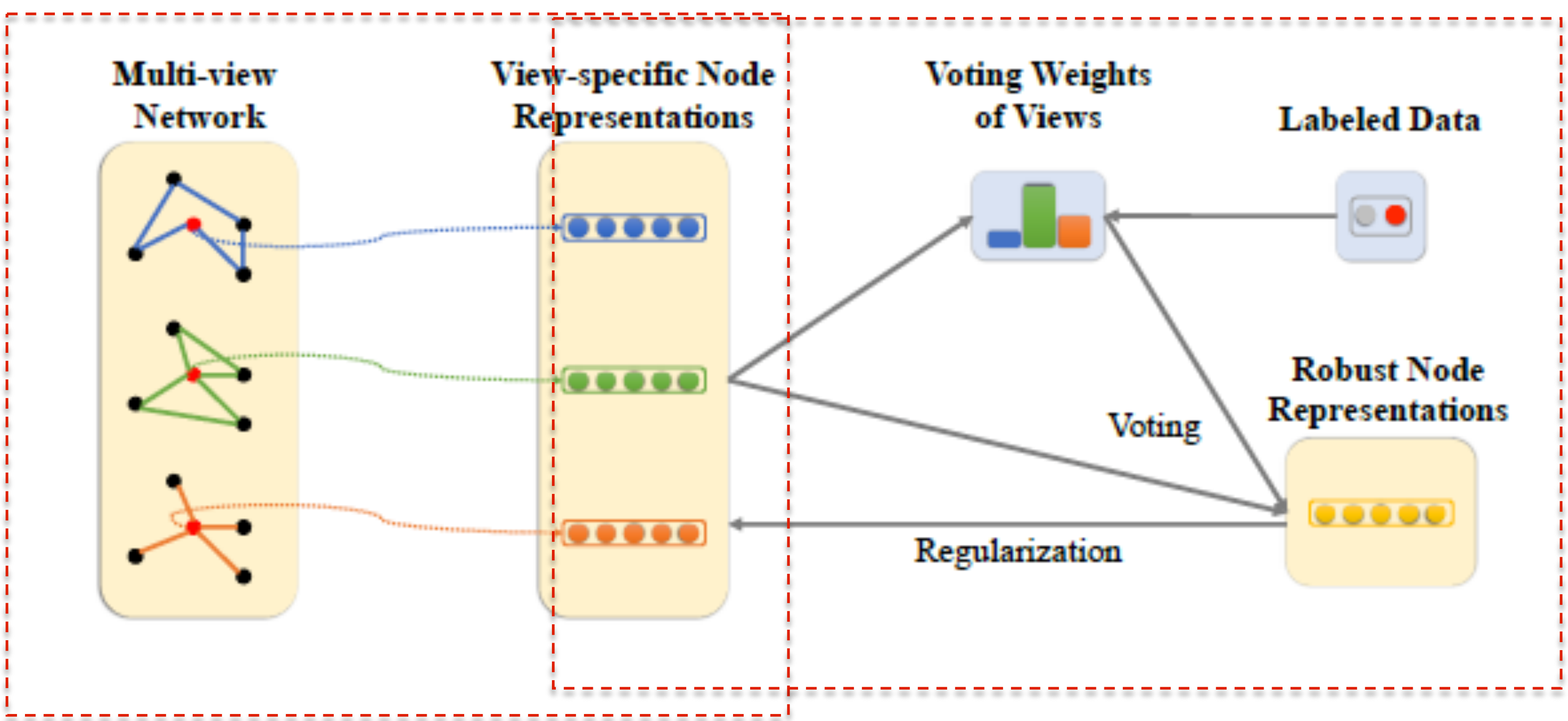


Figure: Networks with multiple views

Meng Qu, **Jian Tang**, Jingbo Shang, Xiang Ren, Ming Zhang, and Jiawei Han. **Learning Distributed Node Representations for Networks with Multiple Views.** To appear in CIKM 2017.

A Co-Regularization Approach

- Each node has a robust representation and multiple view-specific representations
- Preserve the structure of different views through view-specific representations
- Promote the collaboration of different views to vote for robust representations
  - Regularize the view-specific representations



A Co-Regularization Approach

- Objective

$$O_{collab} = \sum_{k=1}^{K} O_k + \eta R,$$

$$O_k = - \sum_{(i,j)\in E_k} w_{ij}^{(k)} \log p_k(v_j|v_i).$$

View-specific objective

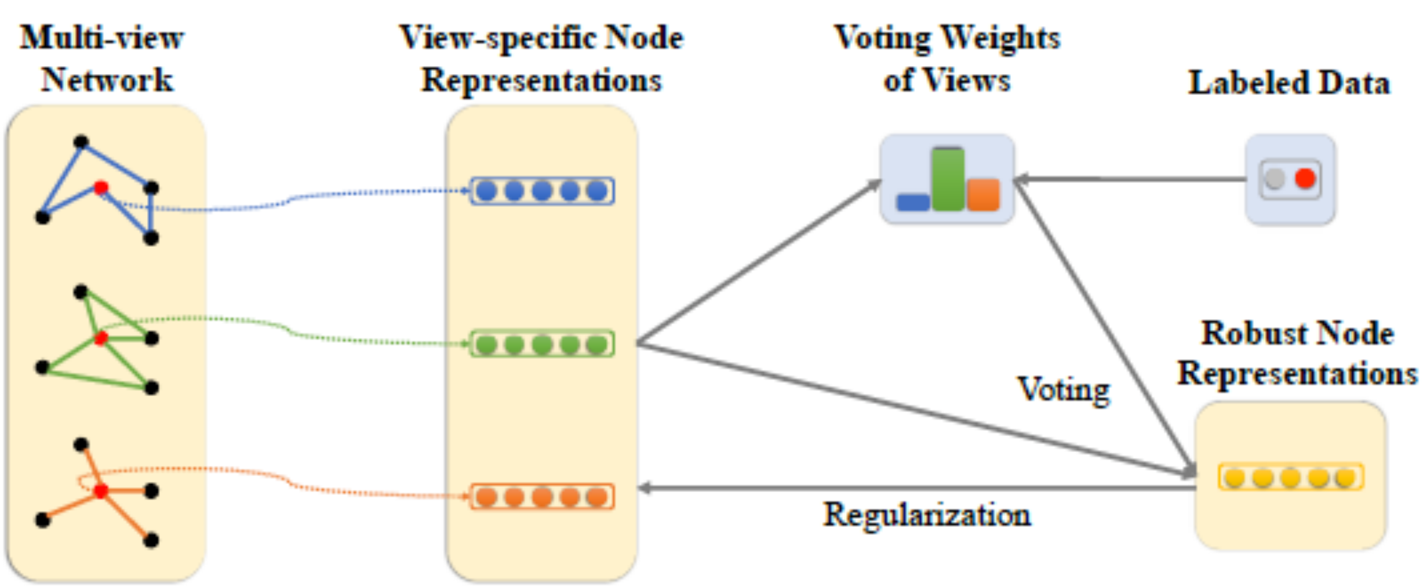
$$R = \sum_{i=1}^{|V|} \sum_{k=1}^{K} \lambda_i^k ||\mathbf{x}_i^k - \mathbf{x}_i||_2^2,$$

Regularization objective

$\mathbf{x}_i$:robust embedding of node i

$\mathbf{x}_i^k$:view-specific node embedding of node i

λ_i^k :weights of views of node i



Learning the Weights of the Views via Neural Attention

- According to the regularization term:

$$R = \sum_{i=1}^{|V|} \sum_{k=1}^{K} \lambda_i^k ||\mathbf{x}_i^k - \mathbf{x}_i||_2^2, \quad \Rightarrow \quad \mathbf{x}_i = \sum_{k=1}^{K} \lambda_i^k \mathbf{x}_i^k .$$

- Learning the weights with supervised data, e.g., node classification

$$O_{attn} = \sum_{v_i \in S} L(\mathbf{x}_i, y_i),$$

- Define the attention weight of views for each node:

$$\lambda_i^k = \frac{\exp(\mathbf{z}_k^T \mathbf{x}_i^C)}{\sum_{k'=1}^{K} \exp(\mathbf{z}_{k'}^T \mathbf{x}_i^C)},$$

$\mathbf{x}_i^C$: concatenation of view-specific embeddings of node i

$\mathbf{z}_k^T$: embedding of view k

Results of Multi-label Node Classification

Category	Algorithm	DBLP		Flickr		PPI	
		Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1
Single View	LINE	70.29	70.77	34.49	54.99	20.69	24.70
	node2vec	71.52	72.22	34.43	54.82	21.20	25.04
Multi View	node2vec-merge	72.05	72.62	29.15	52.08	21.00	24.60
	node2vec-concat	70.98	71.34	32.21	53.67	21.12	25.28
	CMSC	-	-	-	-	8.97	13.10
	MultiNMF	51.26	59.97	18.16	51.18	5.19	9.84
	MultiSPPMI	54.34	55.65	32.56	53.80	20.21	23.34
	MVE-NoCollab	71.85	72.40	28.03	54.62	18.23	22.40
	MVE-NoAttn	73.36	73.77	32.41	54.18	22.24	25.41
	MVE	**74.51**	**74.85**	**34.74**	**58.95**	**23.39**	**26.96**

MVE > MVE-NoAttn > LINE/node2vec

Without learning the weights of views

Single best view

Extensions

- Other variants
- Multi-view networks
- Networks with node attributes
- Heterogeneous networks
- Task-specific network embedding

Networks with Node Attributes (Yang et al. 2015, N.Kipf et al. 2016, Liao et al. 2017)

- Networks with text information (Yang et al. 2015)
- Networks with attributes (Liao et al. 2017)
  - Gender, location, text, …
- Variational graph autoencoders (N.Kipf et al. 2016)
  - Encode the node with neighborhood structures and attributes
  - Decode the neighborhood structures

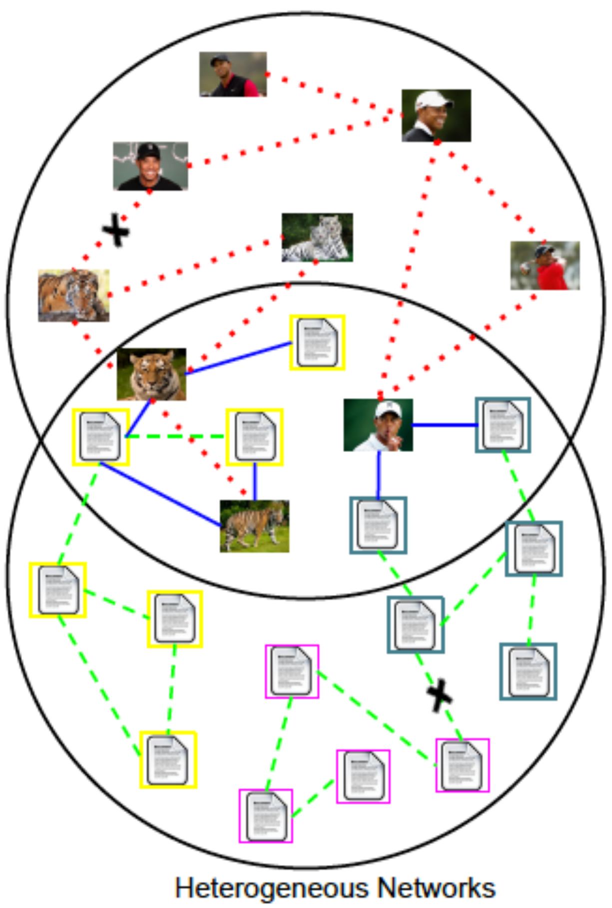
- Cheng Yang, Zhiyuan Liu, Deli Zhao, Maosong Sun, Edward Y. Chang. **Network representation learning with rich text information.** IJCAI 2015.
- Thomas N.Kipf and Max Welling. **Variational Graph Auto-encoders.** NIPS Workshop 2016.
- Lizi Liao, Xiangnan He, Hanwang Zhang, and Tat-Seng Chua. **Attributed Social Network Embedding**. arXiv, 2017.

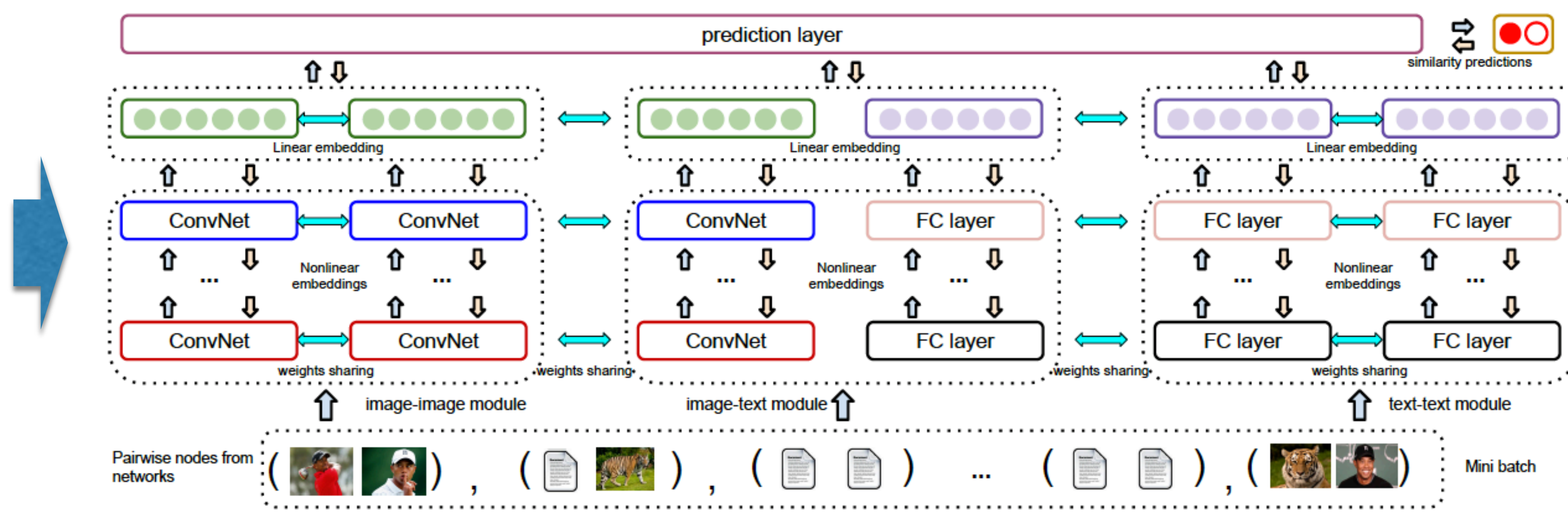
Extensions

- Other variants
- Multi-view networks
- Networks with node attributes
- Heterogeneous networks
- Task-specific network embedding

Heterogeneous Network Embedding via Deep Architectures (Chang et al. 2015)

- Heterogeneous networks of images and text
- Make the embeddings of linked objects close to each other
  - image-image, image-text, text-text

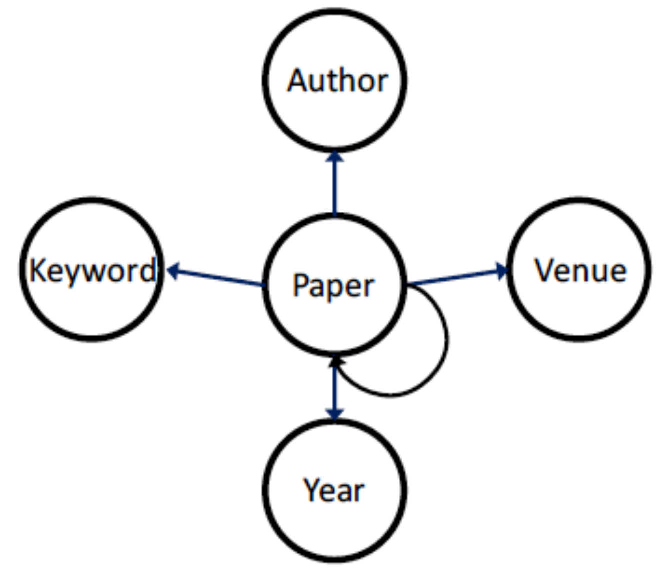


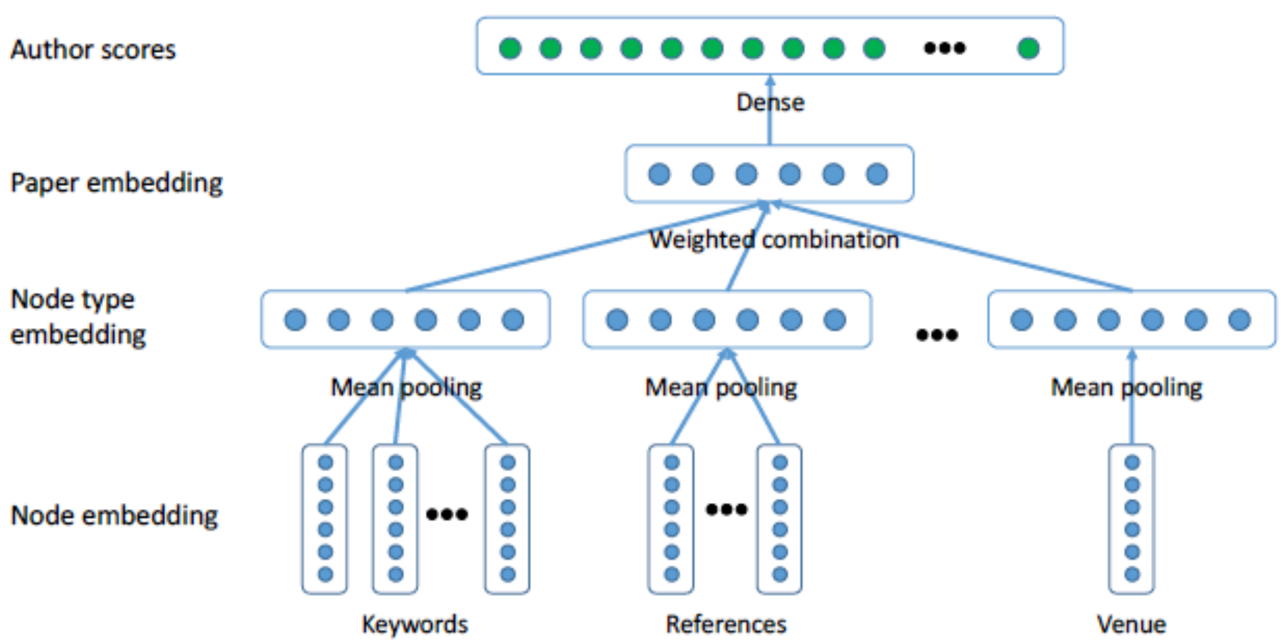


Siyu Chang, Wei Han, Jiliang Tang, Guo-Jun Qi, Charu C. Aggarwal, Thomas S. Huang. Heterogeneous network embedding via Deep Architectures. KDD'15

Heterogeneous Star Network Embedding (Chen et al. 2017)

- Heterogeneous Star Networks
  - Paper, keywords, authors, venues
- Aims to embed the center objects
  - paper





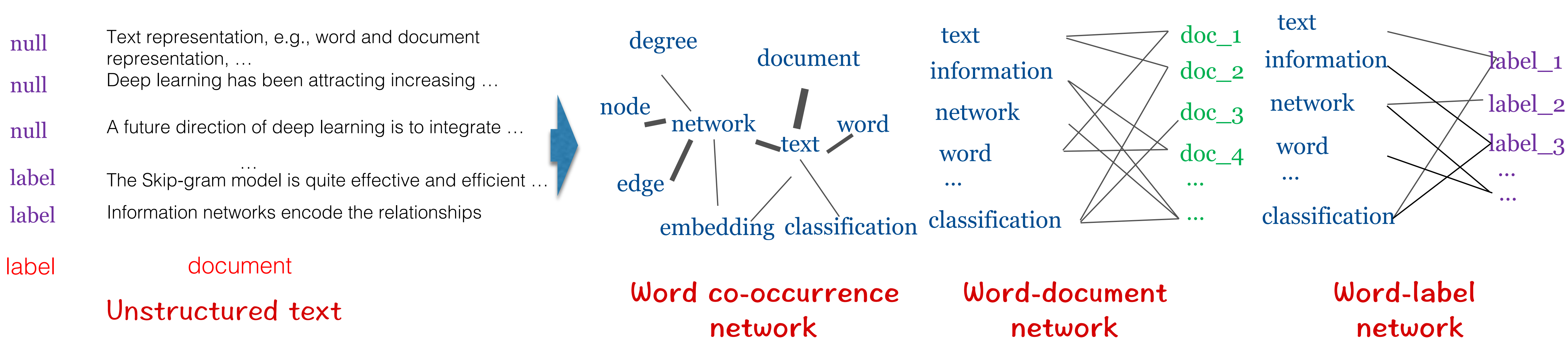
Ting Chen and Yizhou Sun, "Task-Guided and Path-Augmented Heterogeneous Network Embedding for Author Identification. WSDM'17.

Extensions

- Other variants
- Multi-view networks
- Networks with node attributes
- Heterogeneous networks
- Task-specific network embedding

Semi-supervised Text Representation (Tang et al. 2015b)

- Heterogeneous text network
    - Word-word, word-document, and word-label networks
    - Different levels of word co-occurrences: **local context-level, document-level, label-level**
- Learning word embeddings through jointly training the heterogeneous networks
- Document embeddings as the average of word embeddings



Jian Tang, Meng Qu, and Qiaozhu Mei. **PTE: Predictive Text Embedding through Large-scale Heterogeneous Text Networks.** KDD'15.

Results on Text Classification of Long Documents

		20newsgroup		Wikipedia		IMDB	
Type	Algorithm	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1
Unsupervised	LINE(G_wd)	79.73	78.40	80.14	80.13	89.14	89.14
Predictive embedding	CNN	80.15	79.43	79.25	79.32	89.00	89.00
	PTE(G_wl)	82.70	81.97	79.00	79.02	85.98	85.98
	PTE(G_ww+G_wl)	83.90	83.11	81.65	81.62	89.14	89.14
	PTE(G_wd+G_wl)	84.39	83.64	82.29	82.27	89.76	89.76
	PTE(joint)	84.20	83.39	82.51	82.49	89.80	89.80

PTE > CNN

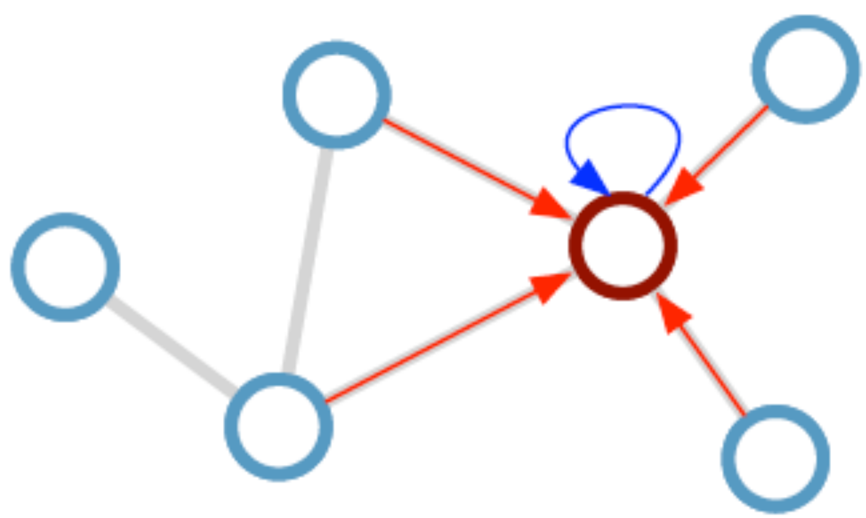
Results on Text Classification of Short Documents

		20newsgroup		Wikipedia		IMDB	
Type	Algorithm	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1
Unsupervised	LINE(G_ww)	74.22	70.12	71.13	71.12	73.84	73.84
Predictive embedding	CNN	76.16	73.08	72.71	72.69	75.97	75.96
	PTE(G_wl)	76.45	72.74	73.44	73.42	73.92	73.91
	PTE(G_ww+G_wl)	76.80	73.28	72.93	72.92	74.93	74.92
	PTE(G_wd+G_wl)	77.46	74.03	73.13	73.11	75.61	75.61
	PTE(joint)	77.15	73.61	73.58	73.57	75.21	75.21

PTE $\approx$ CNN

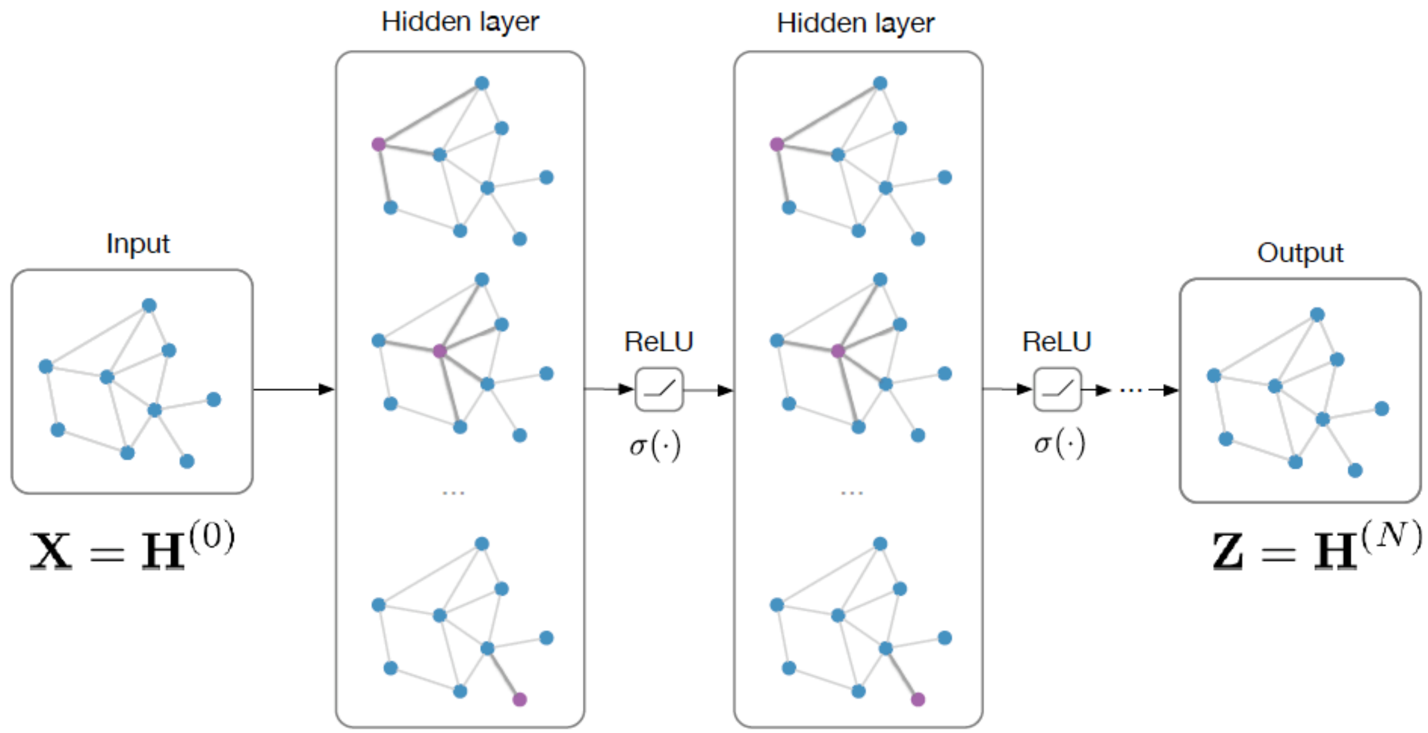
Semi-supervised Classification with Graph Convolutional Networks (Kipf et al. 2017)

- Task: Given a graph G = (V, E), and the features of nodes $X \in R^{N \times D}$, and the labels of a subset of nodes are given.
- Learning the node representations through **graph convolutional networks**
  - Combining node representations (self-link) and representations of neighbors



Thomas N.Kipf and Max Welling. **Semi-Supervised Classification with Graph Convolutional Networks**. ICLR'17.

Multi-layer Graph Convolution Neural Networks



$$\mathbf{H}^{(l+1)} = \sigma\left(\hat{\mathbf{A}}\mathbf{H}^{(l)}\mathbf{W}^{(l)}\right)$$

- Final objective:

$$\mathcal{L} = -\sum_{l \in \mathcal{Y}_L} \sum_{f=1}^{F} Y_{lf} \ln Z_{lf}$$

- Starting from the node features $H^{(0)} = X$
- Define the propagation rule

$$\tilde{A} = A + I_N$$ Add the self-links

$$\hat{\mathbf{A}} = \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$$ Normalize the matrix

$$\mathbf{H}^{(l+1)} = \sigma\left(\hat{\mathbf{A}}\mathbf{H}^{(l)}\mathbf{W}^{(l)}\right)$$ Nonlinear propagation

Experimental Results (Kipf & ICLR 2017)

Table 2: Summary of results in terms of classification accuracy (in percent).

Method	Citeseer	Cora	Pubmed	NELL
ManiReg [3]	60.1	59.5	70.7	21.8
SemiEmb [28]	59.6	59.0	71.1	26.7
LP [32]	45.3	68.0	63.0	26.5
DeepWalk [22]	43.2	67.2	65.3	58.1
ICA [18]	69.1	75.1	73.9	23.1
Planetoid* [29]	64.7 (26s)	75.7 (13s)	77.2 (25s)	61.9 (185s)
GCN (this paper)	**70.3** (7s)	**81.5** (4s)	**79.0** (38s)	**66.0** (48s)
GCN (rand. splits)	67.9 ± 0.5	80.1 ± 0.5	78.9 ± 0.7	58.4 ± 1.7

GCN > Label Propagation

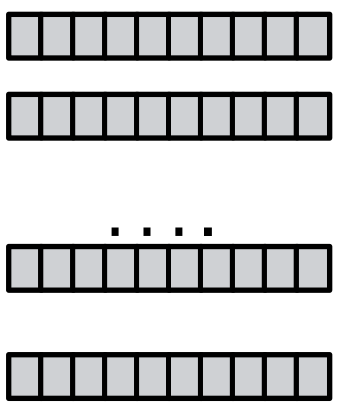
Outline

- **Part I: Learning Node Representations of Networks**
  - Related Work: Laplacian Eigenmap, Word2Vec
  - LINE, DeepWalk, and Node2Vec
  - Extensions
- **Part II: Visualizing Networks and High-Dimensional Data**
  - t-SNE
  - LargeVis
- **Pat III: Learning Representations of Entire Networks**
  - Graph kernels
  - End-to-end methods
- **Part IV: Summary, Challenges & Future Work**

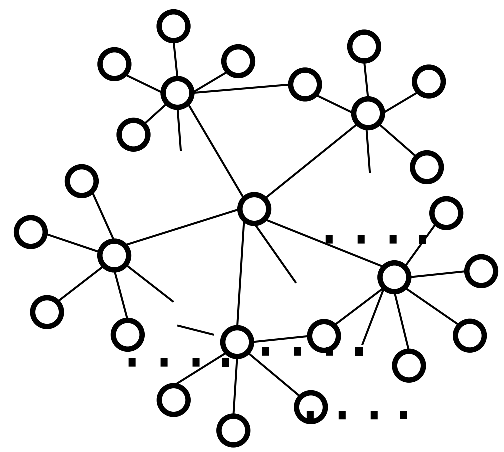
Extremely Low-dimensional Representations: 2D/3D for Visualizing Networks

K-Nearest Neighbor Graph (KNN-G) Construction

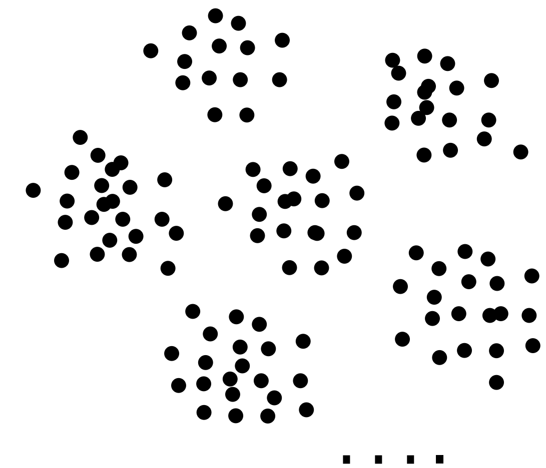
Graph Layout



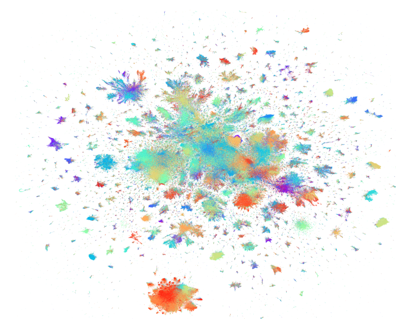
High-dimensional Data



Networks

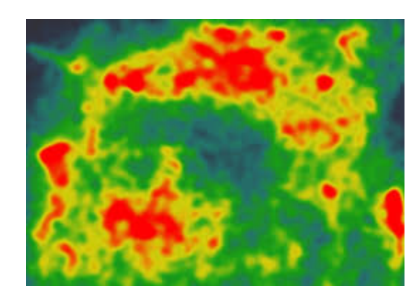


2D/3D Layout



Scatter Plots

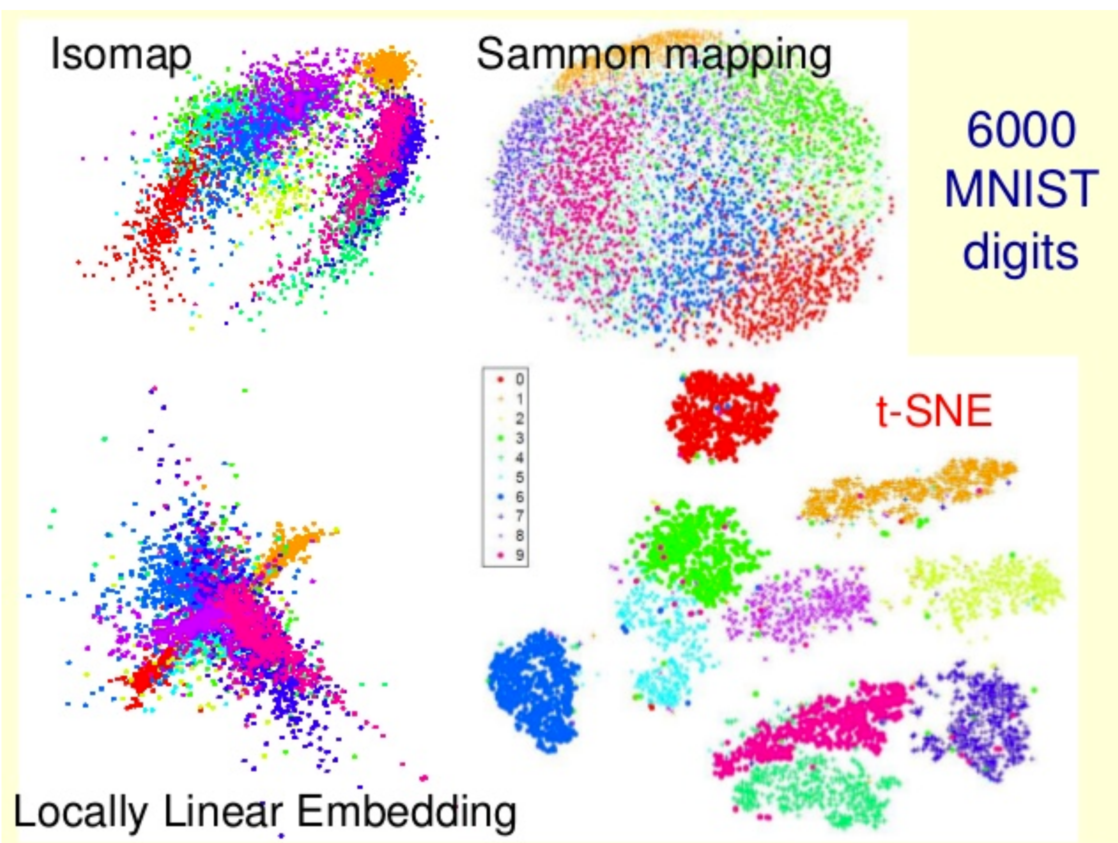
Network Diagrams



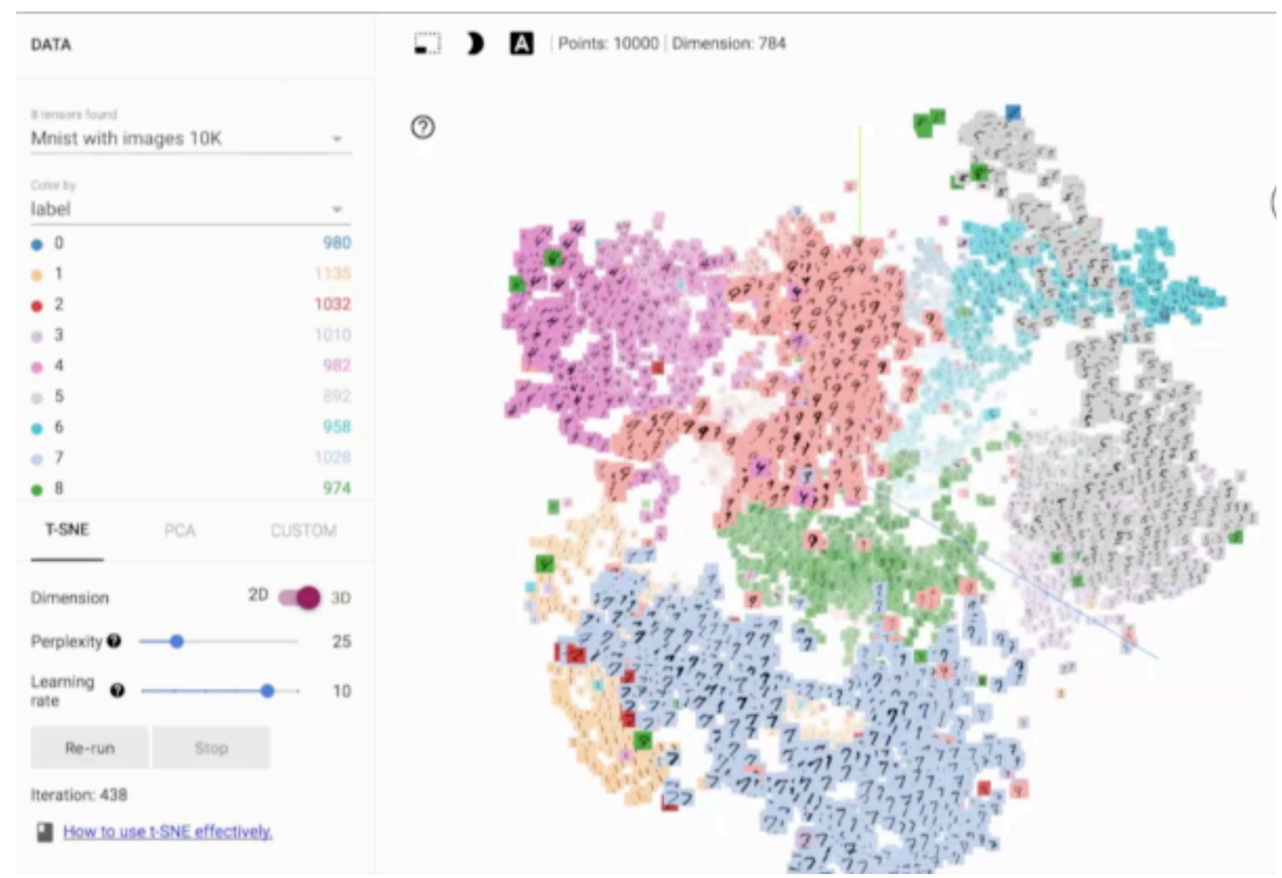
Heatmaps

t-SNE (Maarten and Hinton, 2008, 2014)

- State-of-the-art algorithms for high-dimensional data visualization
- Deployed in Tensorbord for visualizing the representations learned by deep neural networks.



Visualizations of MNIST Data



TensorBoard Visualizations by t-SNE

L.J.P. van der Maaten and G.E. Hinton. **Visualizing High-Dimensional Data Using t-SNE.** JMLR, 2008.
L.J.P. van der Maaten. **Accelerating t-SNE using Tree-Based Algorithms.** JMLR, 2014.

Constructing the K-nearest Neighbor Graph

- Finding the nearest neighbors for all the data points
  - Vantage-point tree
- Calculating the weights of the edges between the data points

$$p_{j|i} = \begin{cases} \frac{\exp(-d(\mathbf{x}_i, \mathbf{x}_j)^2 / 2\sigma_i^2)}{\sum_{k \in \mathcal{N}_i} \exp(-d(\mathbf{x}_i, \mathbf{x}_k)^2 / 2\sigma_i^2)}, & \text{if } j \in \mathcal{N}_i \\ 0, & \text{otherwise} \end{cases}$$

$\mathcal{N}_i$: nearest neighbors of node i

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2N}.$$

- Complexity: O(NlogN) w.r.t. the number of data points N

K-nearest Neighbor Graph Visualization

- Similarity between two data points *i* and *j* in low-dimensional space is defined as:

$$q_{ij} = \frac{(1 + \|\mathbf{y}_i - \mathbf{y}_j\|^2)^{-1}}{\sum_{k \neq l}(1 + \|\mathbf{y}_k - \mathbf{y}_l\|^2)^{-1}},$$

$$q_{ii} = 0$$

$\mathbf{y}_i$: low-dimensional representations (coordinates) of node i

- Objective: minimize the similarities defined in the high-dimensional spaces and low-dimensional spaces

$$C(\mathcal{E}) = KL(P||Q) = \sum_{i \neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}}.$$

- The complexity: O(NLogN) (Maarten, 2014).

Limitations of t-SNE

- K-NNG construction: complexity grows O(NlogN) to the number of data poitns N
- Graph layout: complexity is O(NlogN)
- Very sensitive parameters

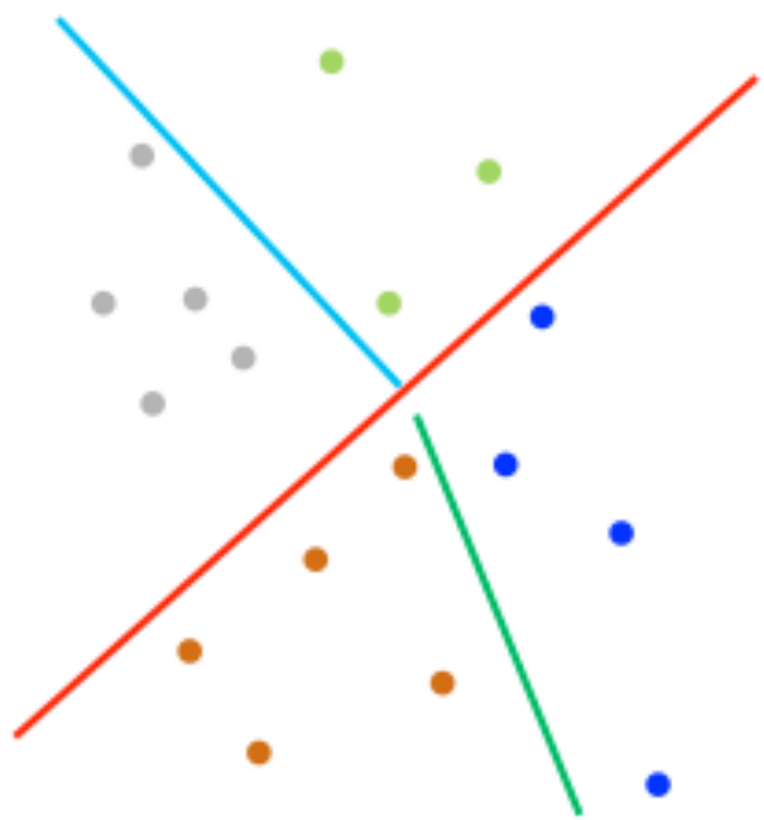
LargeVis (Tang et al., Best Paper Nomination at WWW 2016)

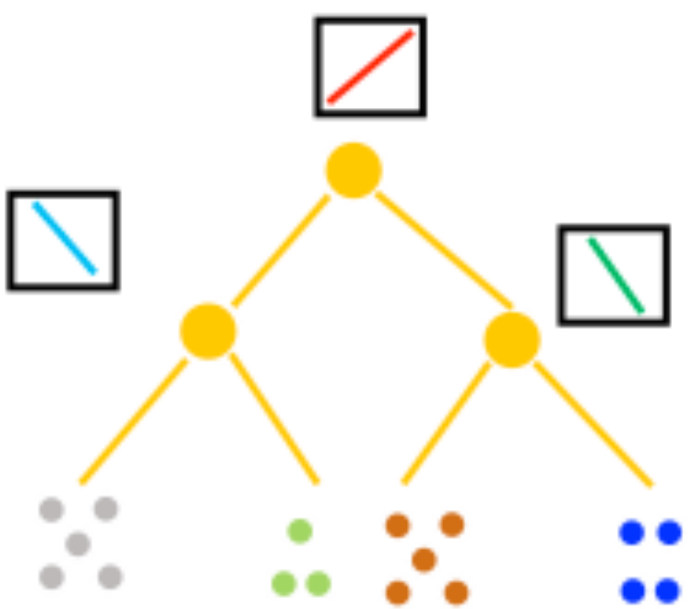
- Efficient approximation of K-NNG construction
  - 30 times faster than t-SNE (3 million data points)
  - Better time-accuracy tradeoff
- Efficient probabilistic model for graph layout
  - O(NlogN) -> O(N)
  - 7 times faster than t-SNE (3 million data points)
  - Better visualization layouts
  - Stable parameters across different data sets

Jian Tang, Jingzhou Liu, Ming Zhang, and Qiaozhu Mei. **Visualizing Large-scale and High-dimensional Data.** WWW'16

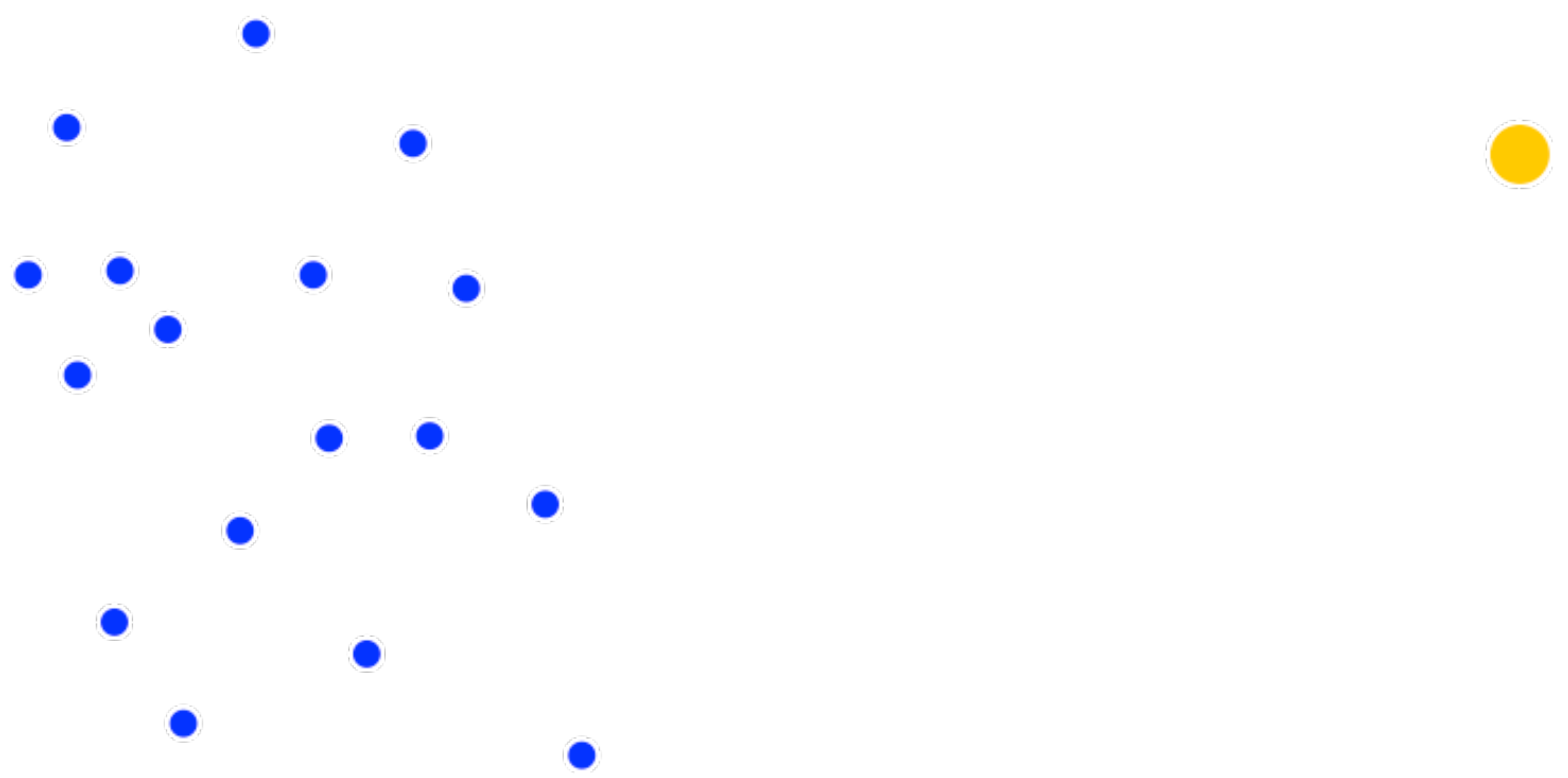
Random Projection Trees

- Partition the whole space into different regions with multiple hyperplanes

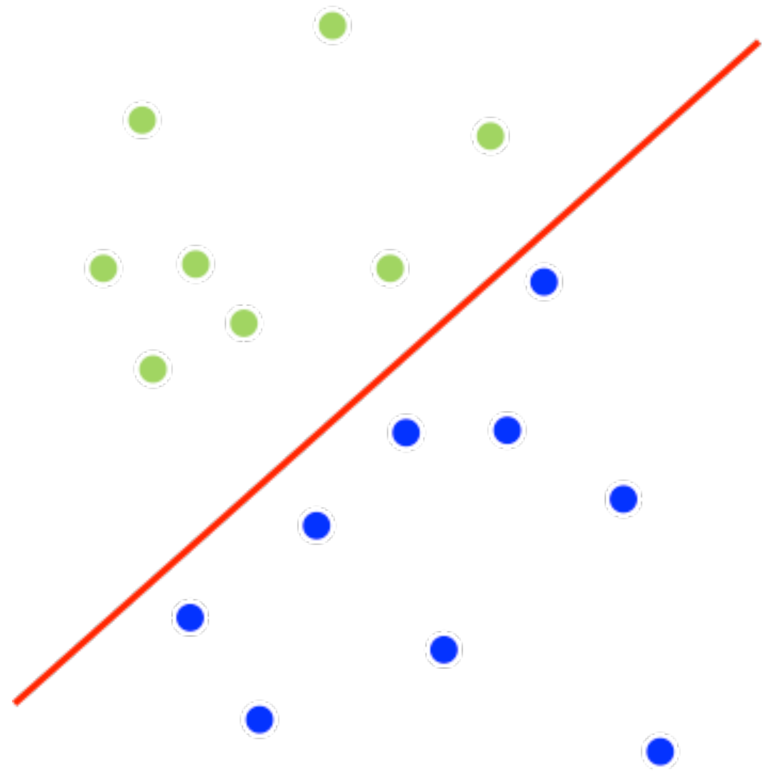


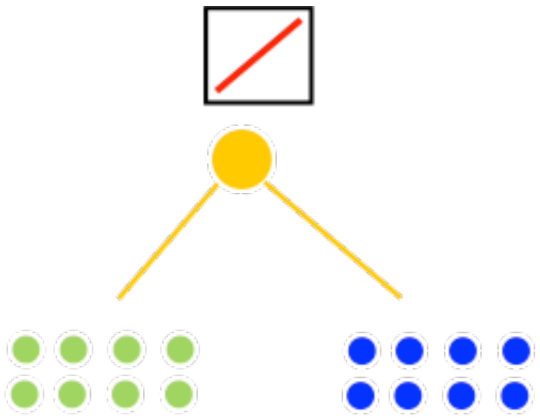


Random Projection Trees

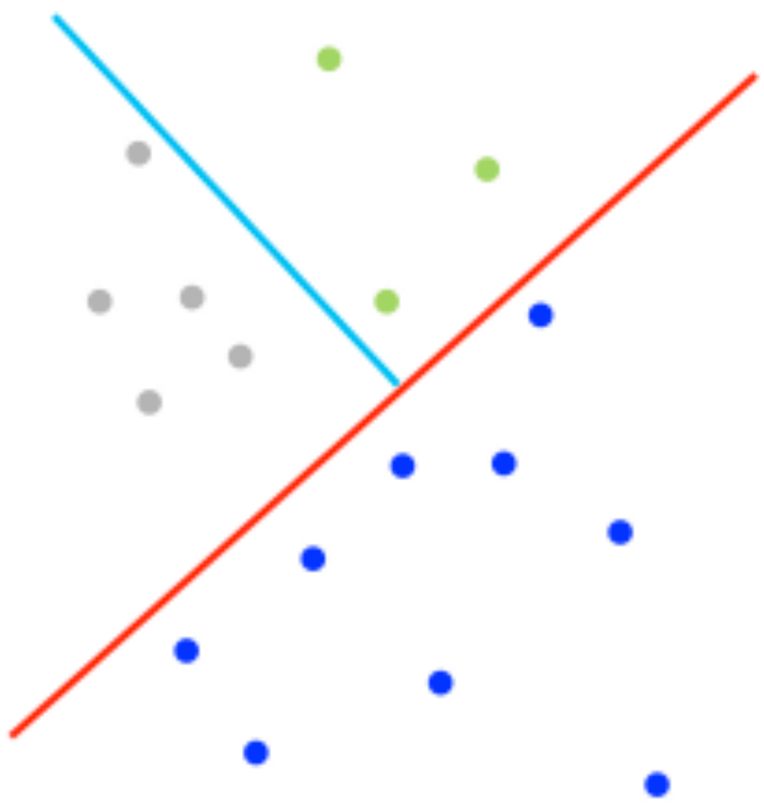


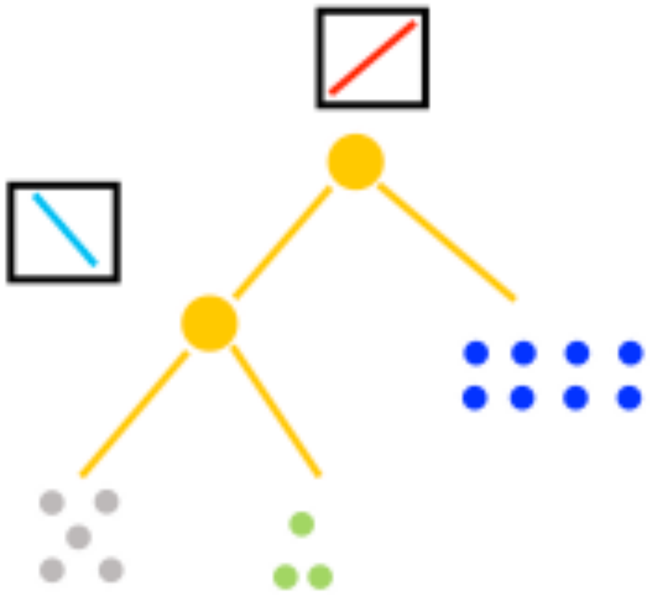
Random Projection Trees



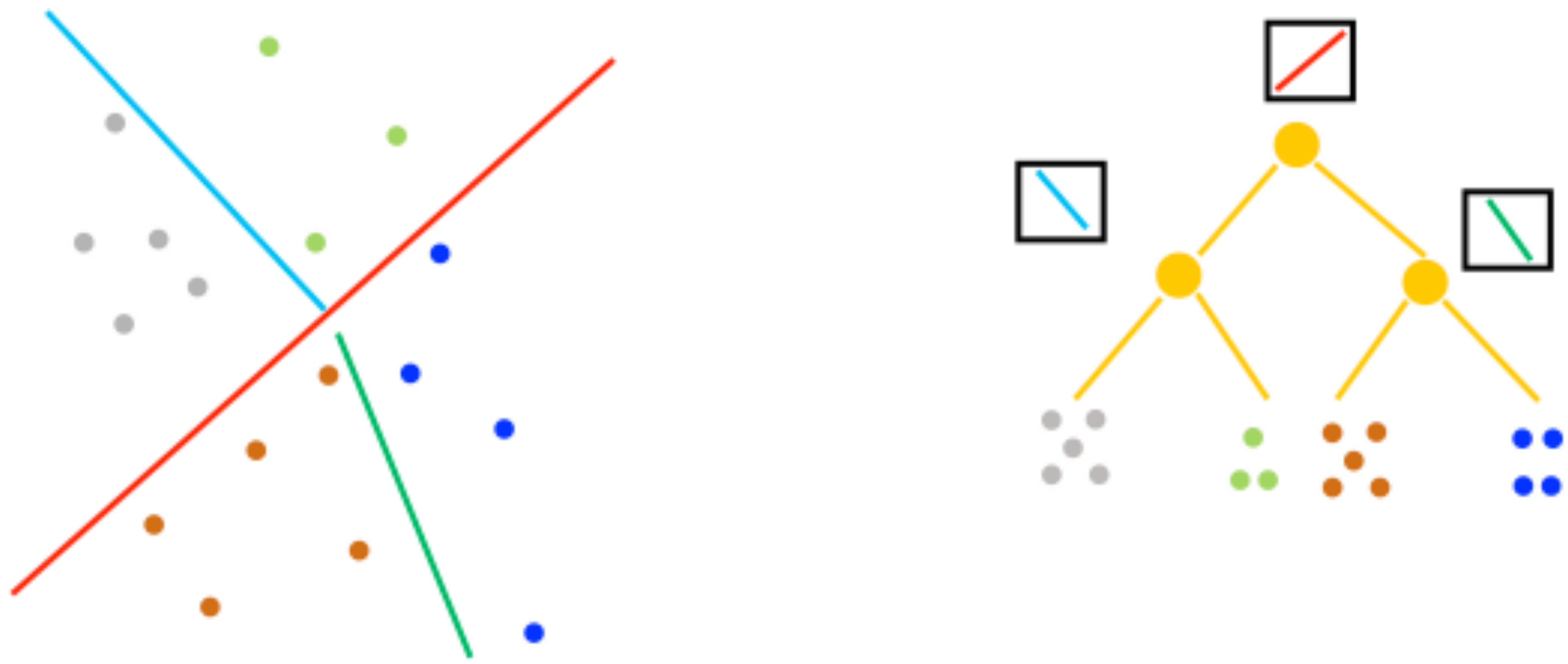


Random Projection Trees



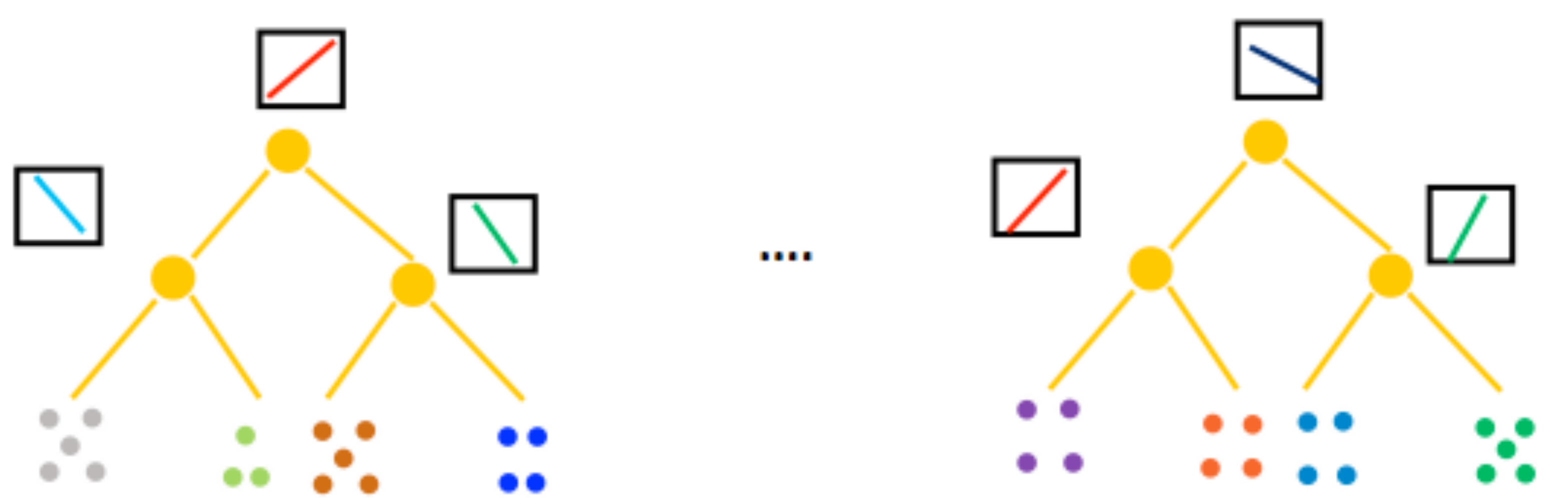


Random Projection Trees



K-NNG Construction

- Search nearest neighbors through traversing trees
  - Only data points in the leaf are considered as nearest neighbors
- Multiple trees are usually used to improve the accuracy
  - e.g., hundreds

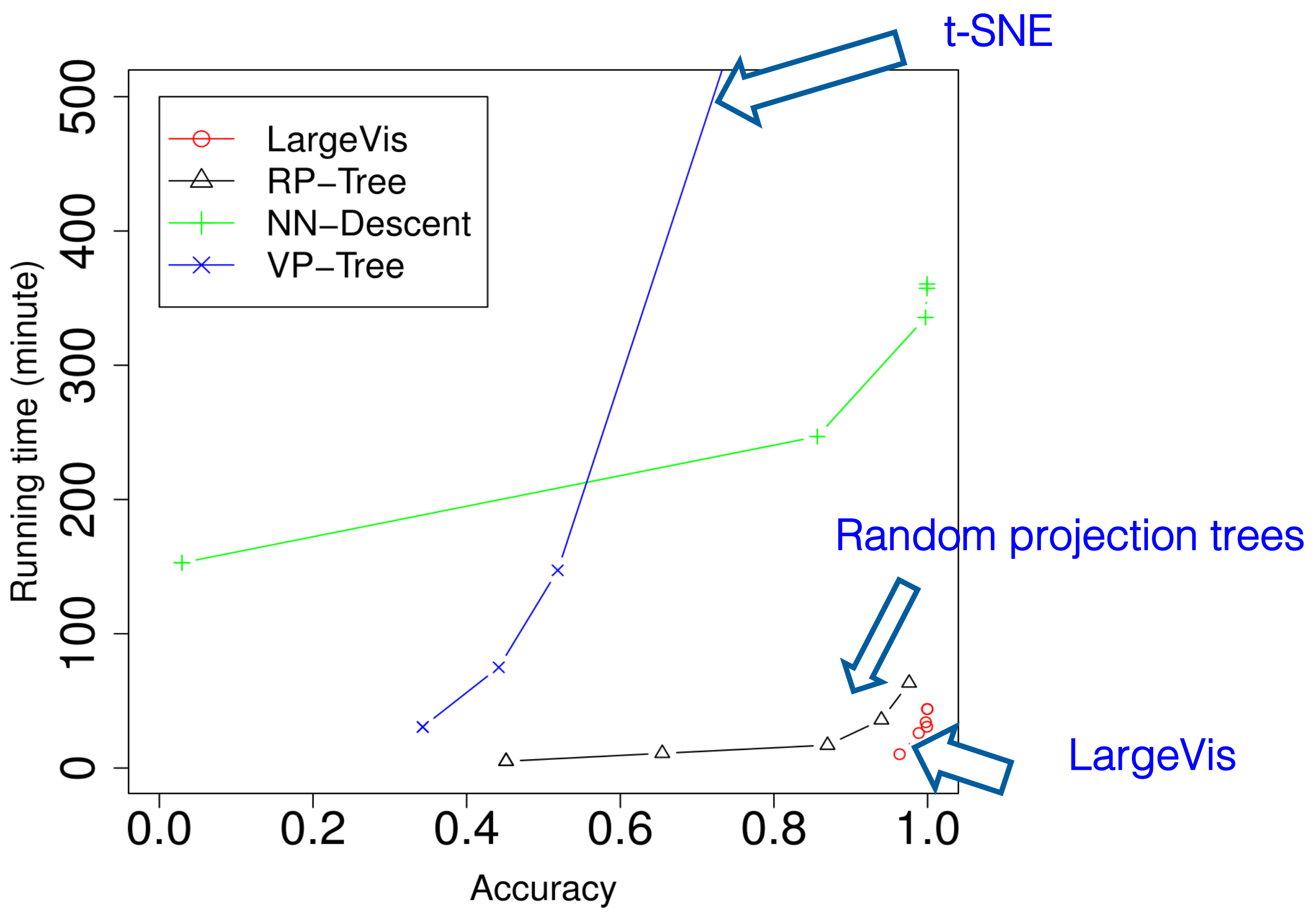


Reduce the Number of Trees

- Construct a less accurate K-NNG with a few trees
- Iteratively refine the K-NNG through "neighbor exploring"
  - "A neighbor of my neighbor is also likely to be my neighbor"
  - Second-order neighbors are also treated as candidates of first-order neighbors

It Works!

- X axis: accuracy of K-NNG
- Y axis: running time (minutes)
- tSNE: 16 hours (95% accuracy)
- LargeVis: 25 minutes
  - >30 times faster than t-SNE



Learning the Layout of KNN Graph

- Preserve the similarities of the nodes in 2D/3D space
  - Represent each node with a 2D/3D vector
  - Keep similar data close while dissimilar data far apart
- Probability of observing a binary edge between nodes (i,j):

$$p(e_{ij}=1)=\frac{1}{1+\|\vec{y}_i-\vec{y}_j\|^2}$$

- Likelihood of observing a weighted edge between nodes (i,j):

$$p(e_{ij}=w_{ij})=p(e_{ij}=1)^{w_{ij}}$$

A Probabilistic Model for Graph Layout

- Objective:

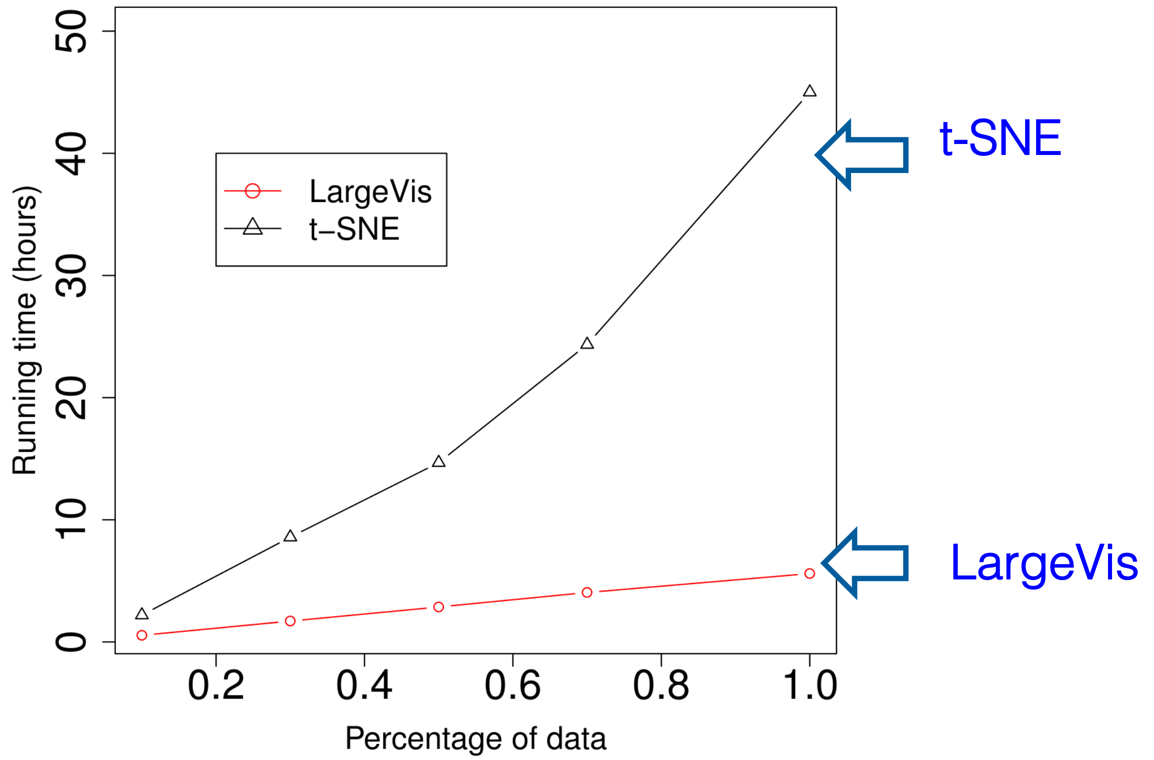
$$O = \prod_{(i,j)\in E} p(e_{ij} = w_{ij}) \prod_{(i,j)\in \bar{E}} (1 - p(e_{ij} = w_{ij}))^{\gamma}$$

γ: an unified weight assigned to negative edge

- Randomly sample some negative edges
- Optimized through asynchronous stochastic gradient descent
- Time complexity: linear to the number of data points

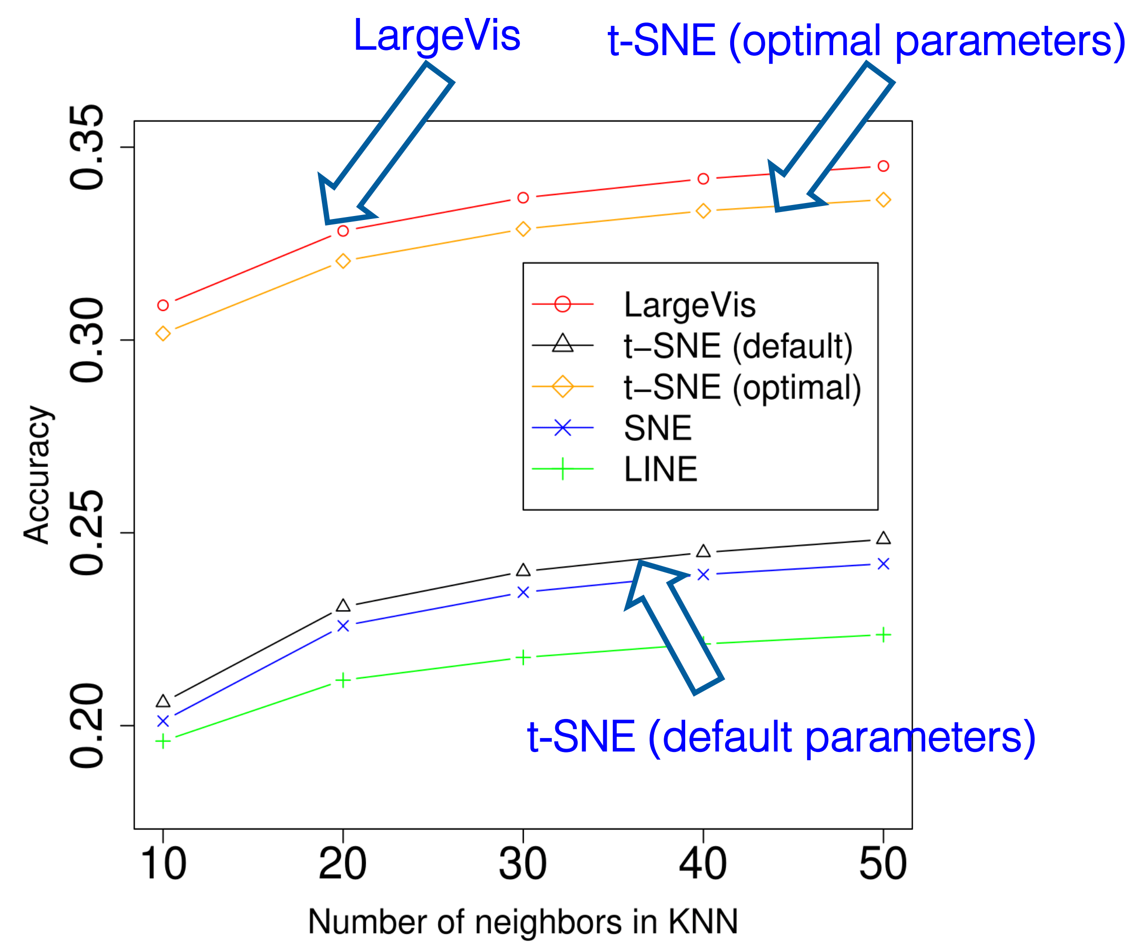
It Works Too!

- Time complexity
  - t-SNE: O(NlogN)
  - LargeVis: O(N)
- On 3 million data points
  - t-SNE: 45 hours
  - LargeVis: 5.6 hours
  - Seven times faster



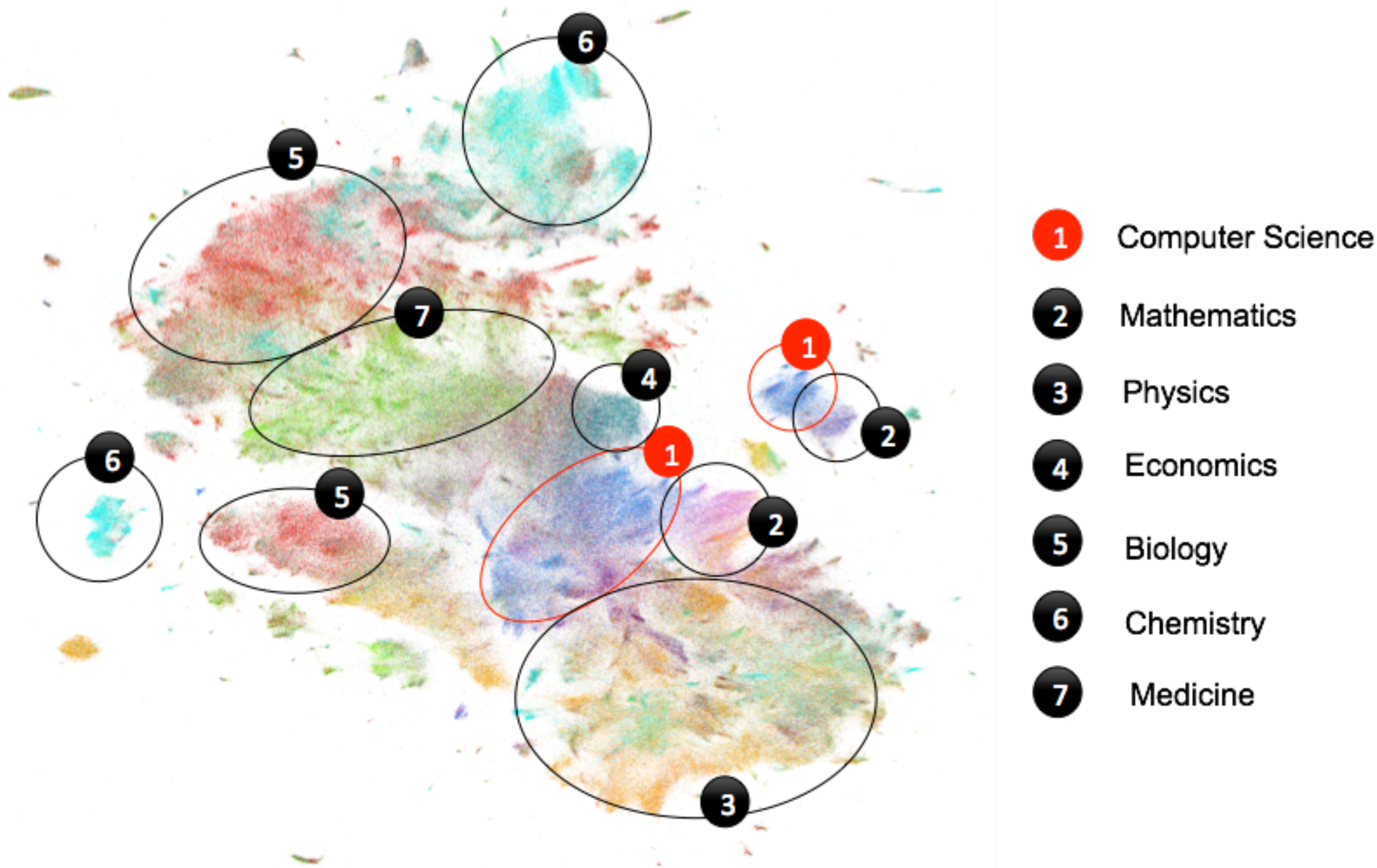
Visualization Quality

- Metric: *classification accuracy* with KNN on 2D space
- Configuration:
  - LargeVis with default parameters
  - t-SNE with default and optimal parameters (tuned per data set)
- LargeVis ≈ t-SNE with optimal parameters
- LargeVis >> t-SNE with default parameters
- Parameters of LargeVis are very stable

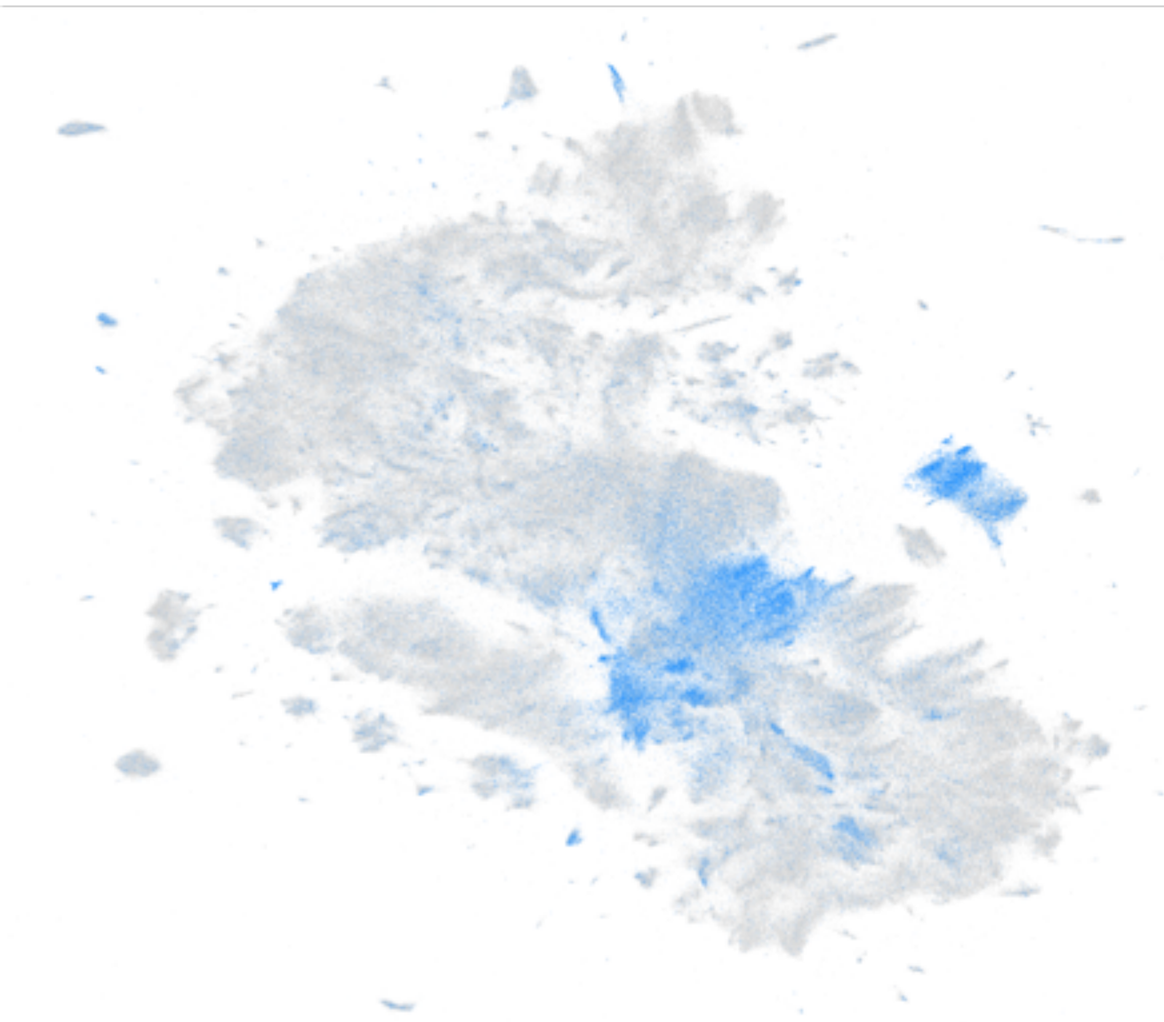


10M Scientific Papers on One Slide

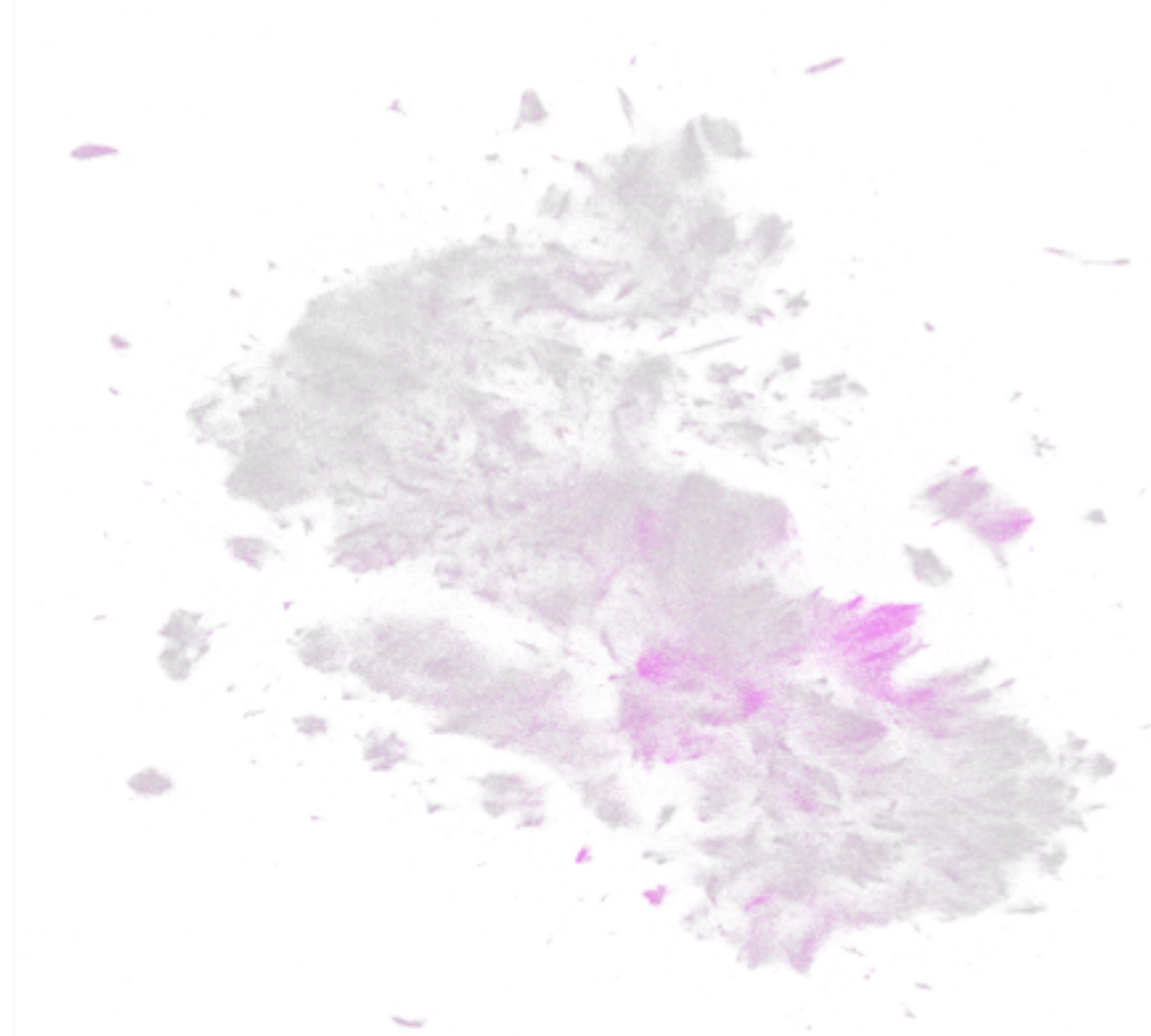
10M Scientific Papers on One Slide



Computer Science

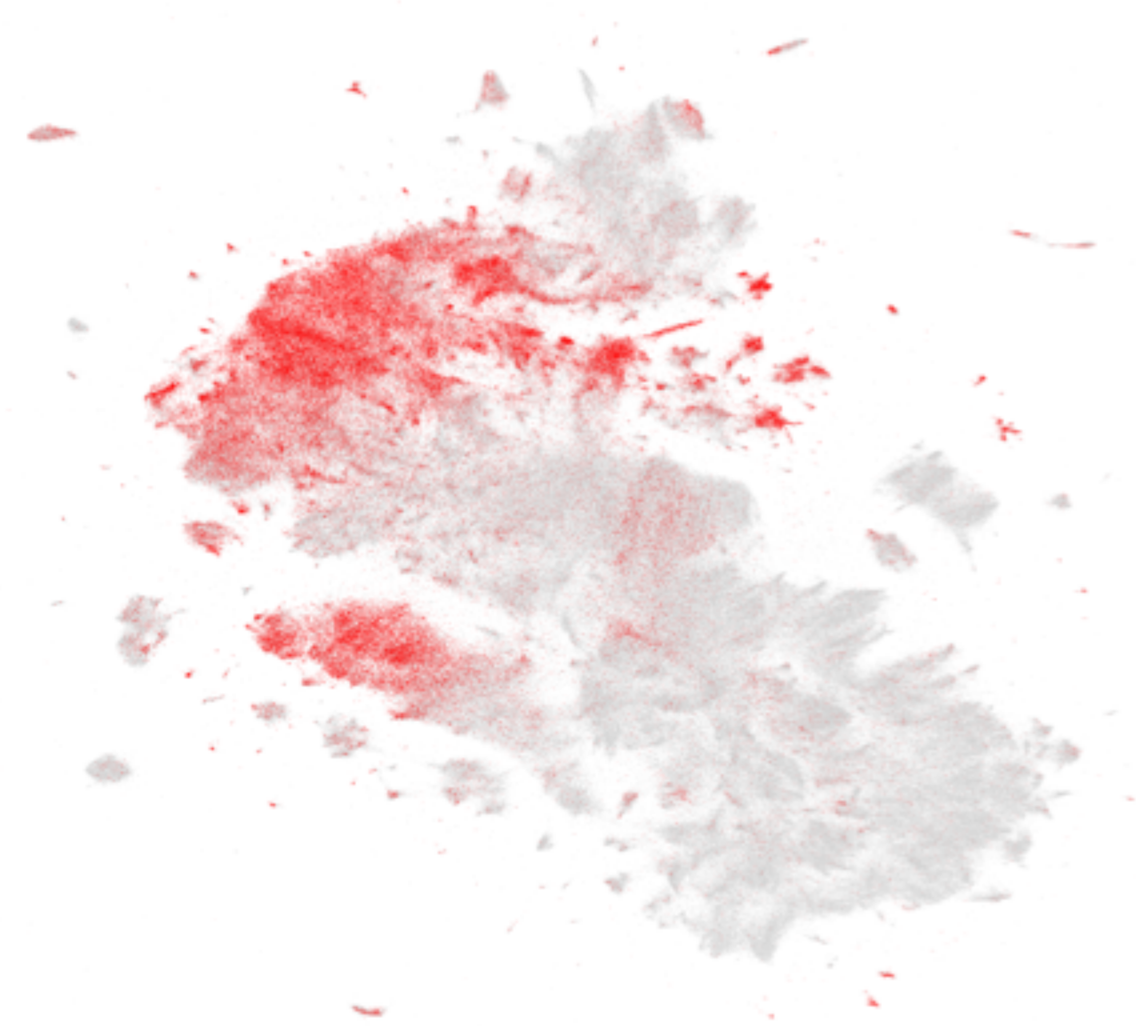


Mathematics

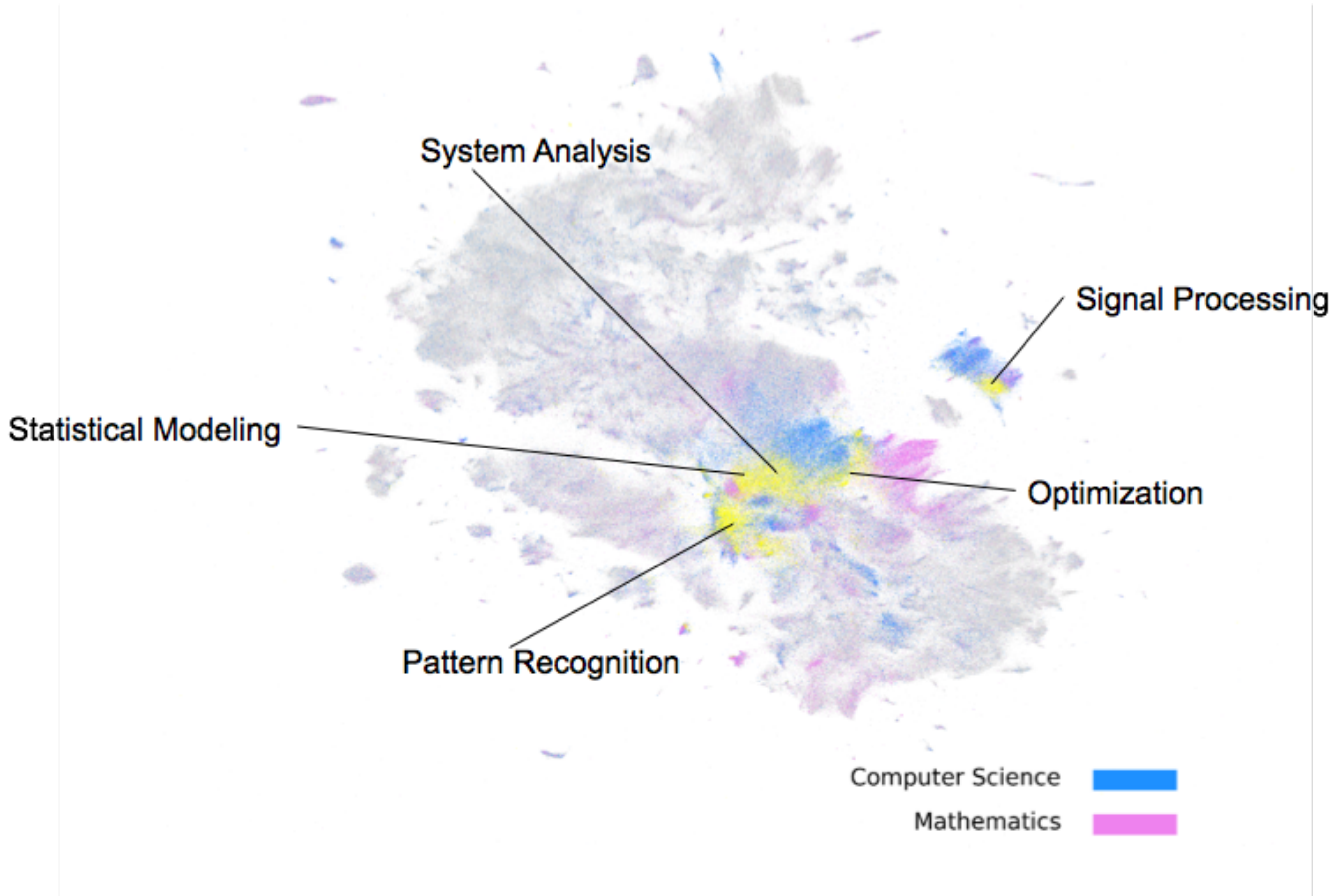


Physics

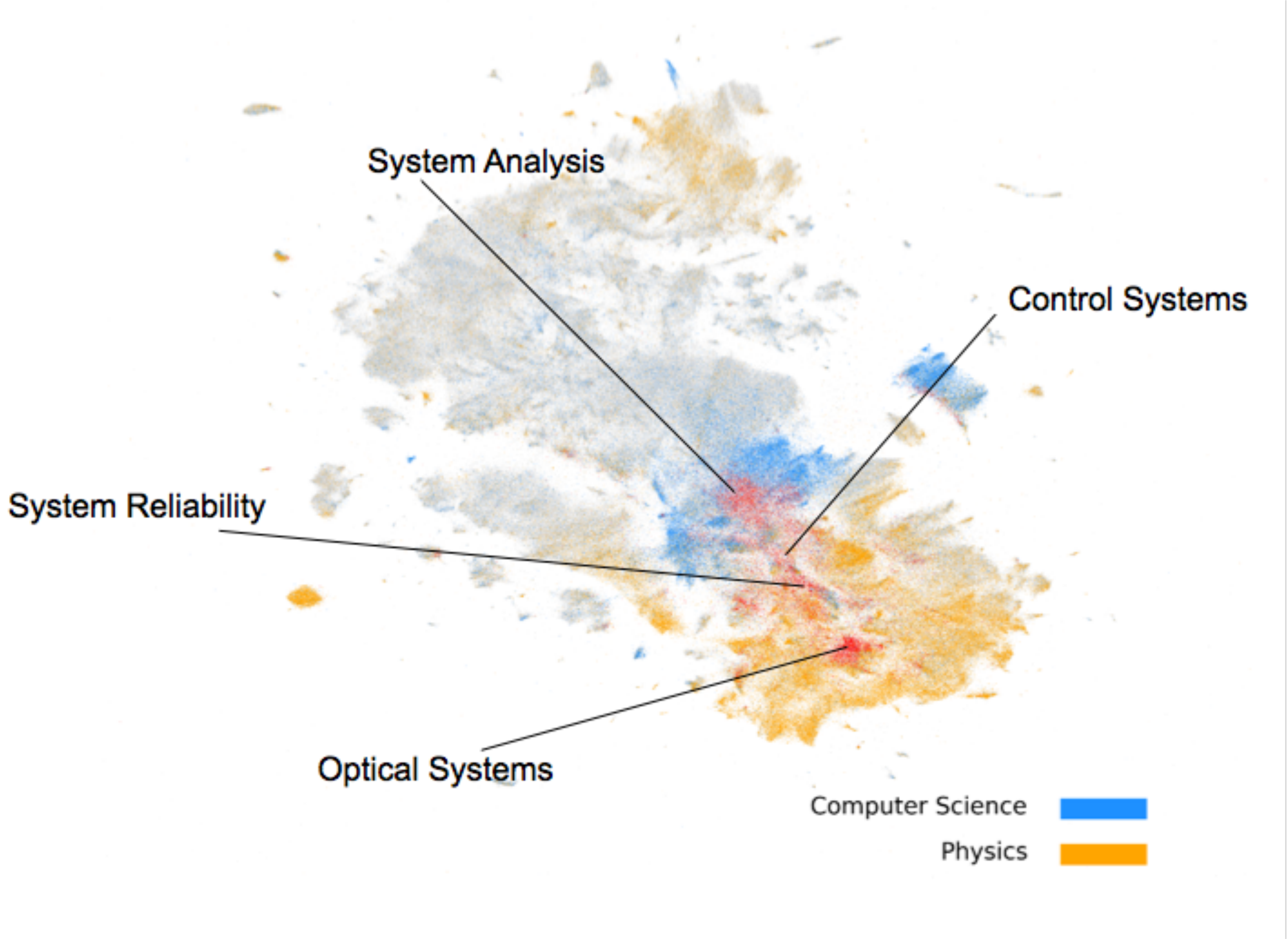
Biology

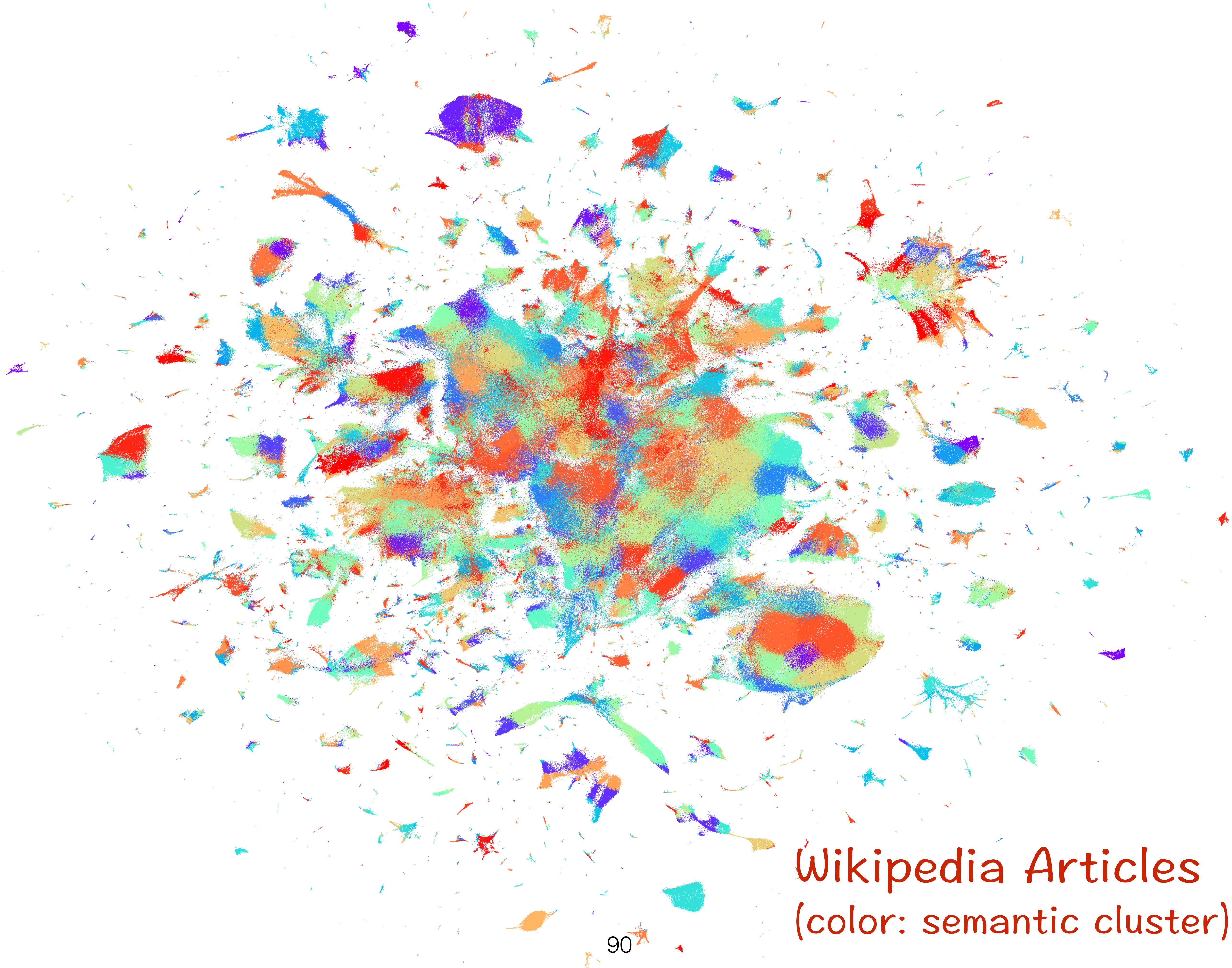


Computer Science vs. Mathematics

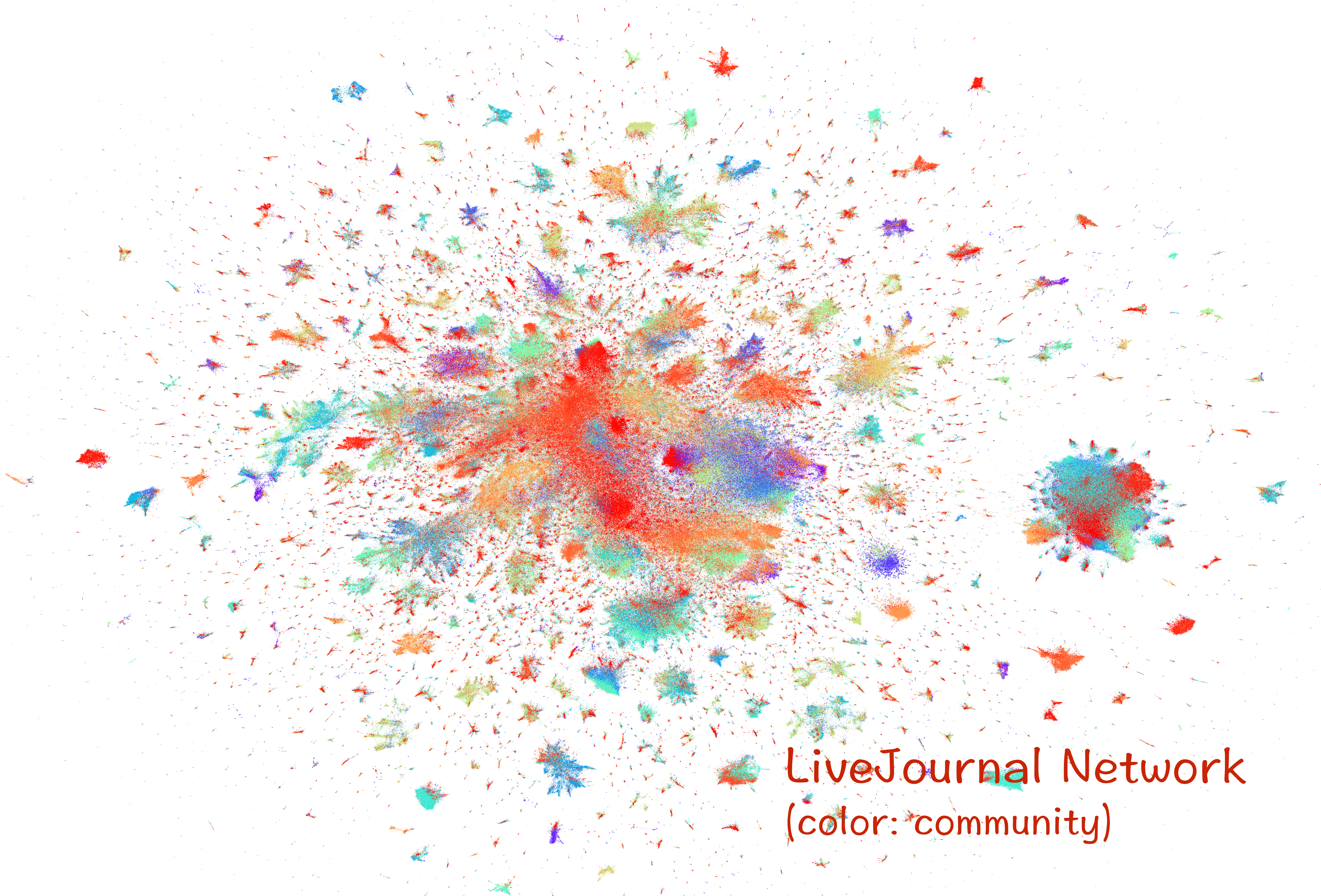


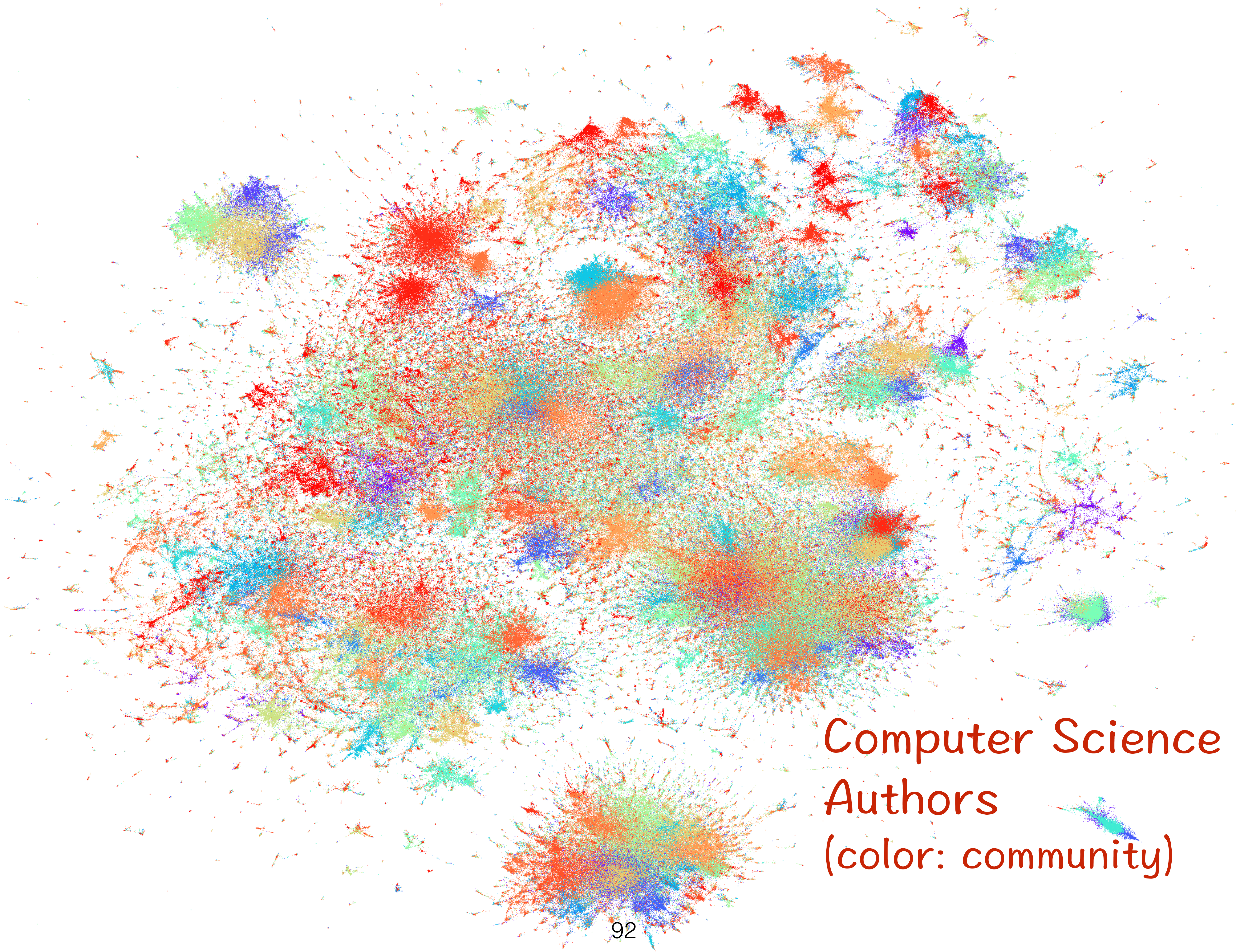
Computer Science vs. Physics





Wikipedia Articles
(color: semantic cluster)


LiveJournal Network
(color: community)



Computer Science Authors
(color: community)

Summary

- LargeVis: a new technique for visualizing networks and high-dimensional data
- A better tool than t-SNE.
  - >7 times faster than t-SNE on three million data points

Impact

Our release:

LINE: (C++) — https://github.com/tangjianpku/LINE
(271 stars, released since 2015.3)

LargeVis : (C++&Python) — https://github.com/lferry007/LargeVis
(289 stars, released since 2016.7)

Other tools based on our implementation:

R version in CRAN: https://github.com/elbamos/largeVis

LargeVis Tutorial: https://jlorince.github.io/viz-tutorial/

Interactive Visualization: https://github.com/NLeSC/DiVE

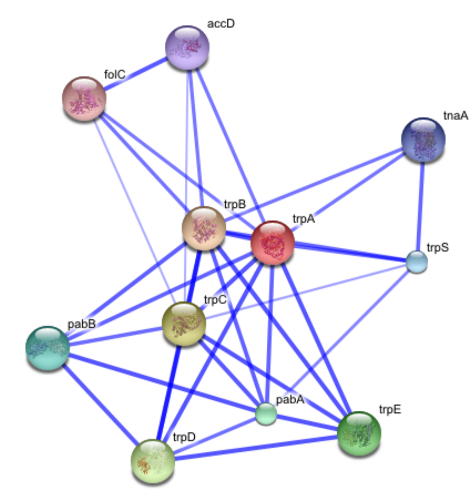
...

Outline

- **Part I: Learning Node Representations of Networks**
    - Laplacian Eigenmap
    - Word2Vec
    - LINE, DeepWalk, and Node2Vec
- **Part II: Visualizing Networks and High-Dimensional Data**
    - t-SNE
    - LargeVis
- **Pat III: Learning Representations of Entire Networks**
    - Graph kernels
    - End-to-end methods
- **Part IV: Summary, Challenges & Future Work**

Beyond node representations

- Node representations are good for
  - Node classification
  - Recommendation
  - Link prediction
  - …
- How about …
  - Information cascades
  - Community detection
  - Protein function prediction
- We want to learn **graph** representations



Road map

- Non end-to-end method
    - Graph kernels
        - Manually designed kernel matrix
        - Kernel matrix is later used for down-stream tasks
- End-to-end methods
    - Matrix-based
    - Sequence-based
    - Graphical model based

Road map

- Non end-to-end method
    - Graph kernels
        - Manually designed kernel matrix
        - Kernel matrix is later used for down-stream tasks
- End-to-end methods
    - Matrix-based
    - Sequence-based
    - Graphical model based

Kernels

- Quantify similarity of two objects
  - $K(X, X') = \langle \Phi(X), \Phi(X') \rangle$
  - $\Phi(\cdot)$ maps objects to embedding space

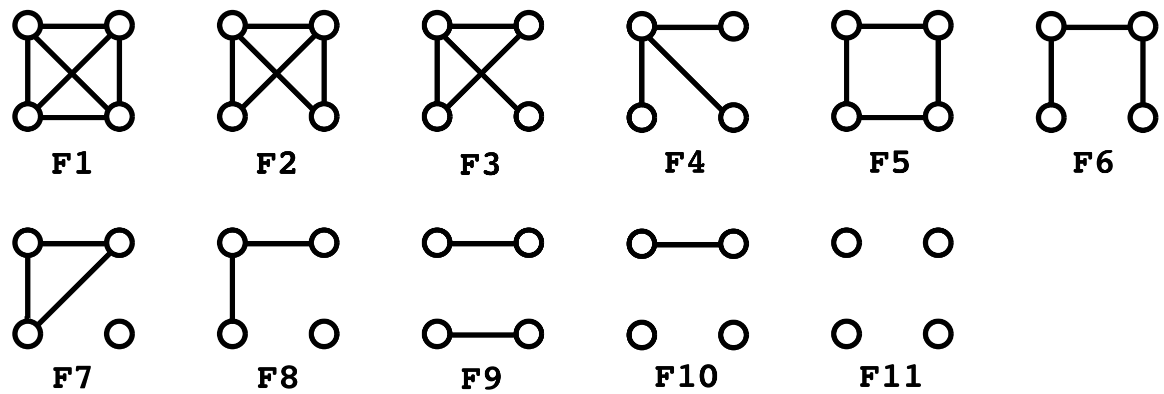
Graph kernels

- Intuition
  - Design graph substructures
  - Compare them to find similarity K(G, G')
  - Embedding of a graph is its similarity to all other graphs

- Many graph kernels
  - Shortest Path Kernel [Borgwardt+ '05]
  - Graphlet Kernel [Shervashidze+ '09]
  - Weisfeiler-Lehman Kernel [Shervashidze+ '11]
  - ...

Graphlet kernel

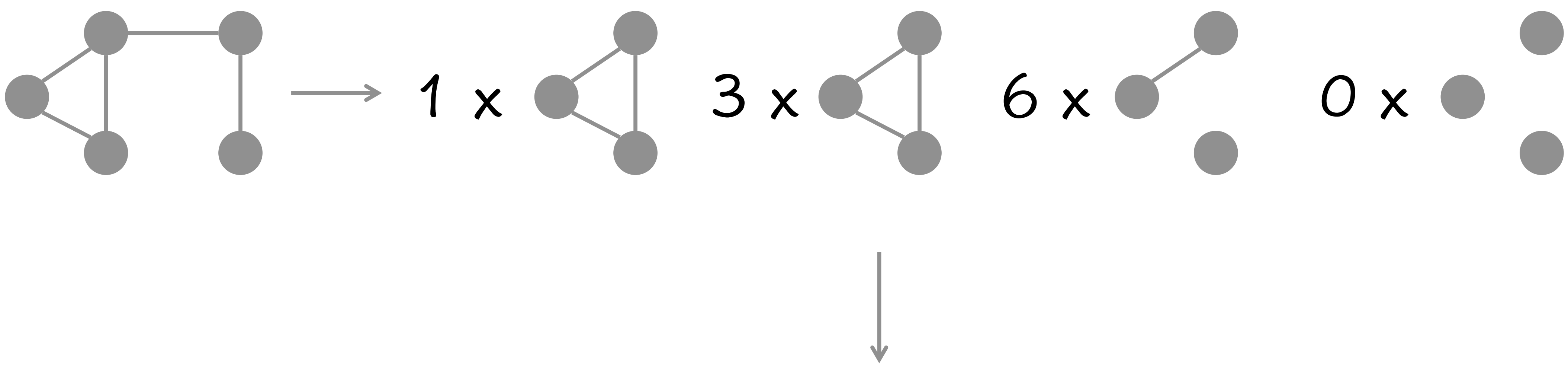
- Count #graphlets



Graphlets of size 4

- $v_G = (\#F_1, \#F_2, \cdots, \#F_{11})$ defines feature vector
    - $\#F_i$ is the number of graphlet F_i in G
- Isomorphic graphs have identical graphlet distribution.
- Graphlet kernel $K(G, G') = v_G^T v_{G'}$

Example of graphlet kernel



$$v_G = (1, 3, 6, 0)$$

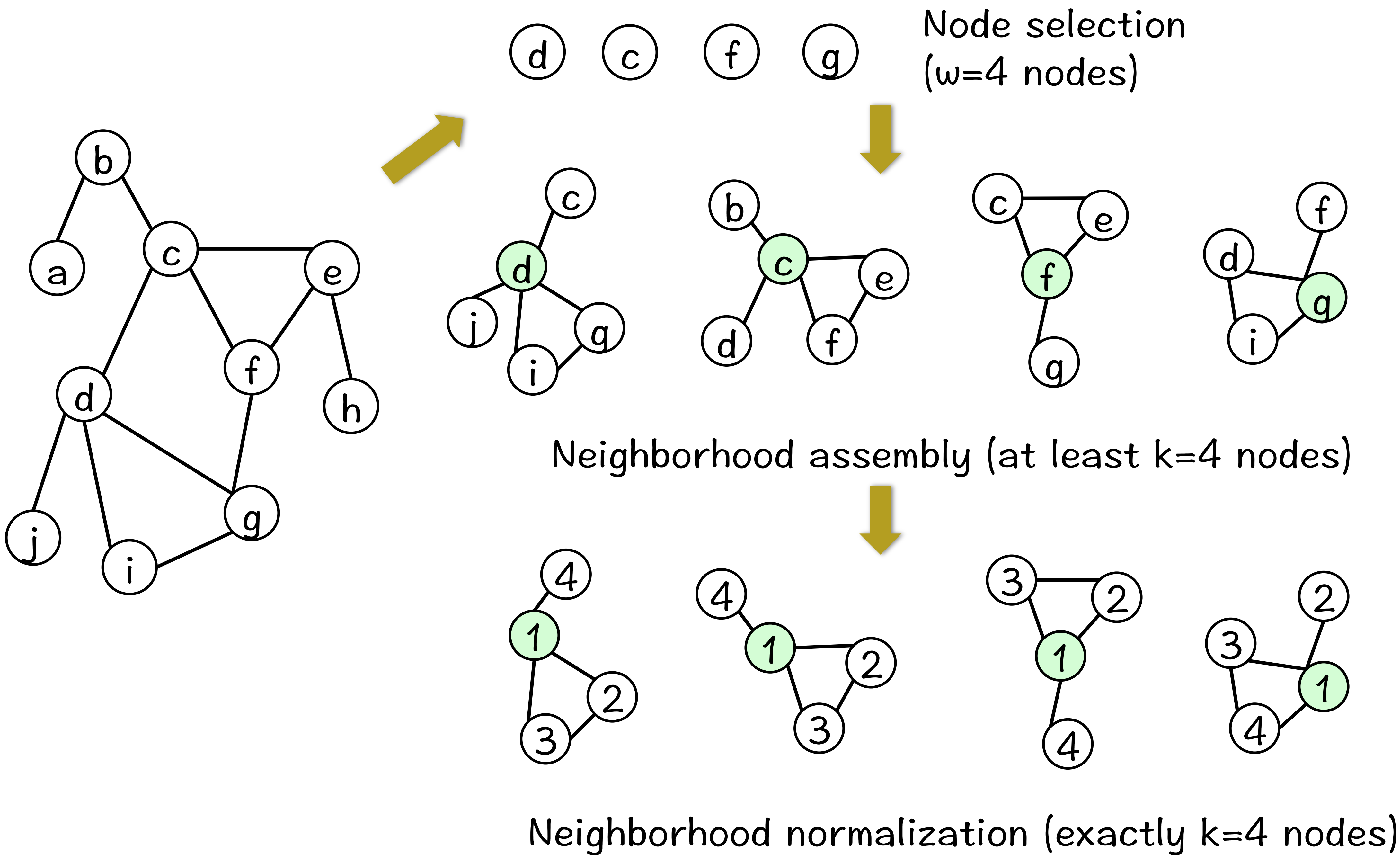
Road map

- Non end-to-end method
  - Graph kernels
    - Manually designed kernel matrix
    - Kernel matrix is later used for down-stream tasks
- End-to-end methods
  - Matrix-based
  - Sequence-based
  - Graphical model based

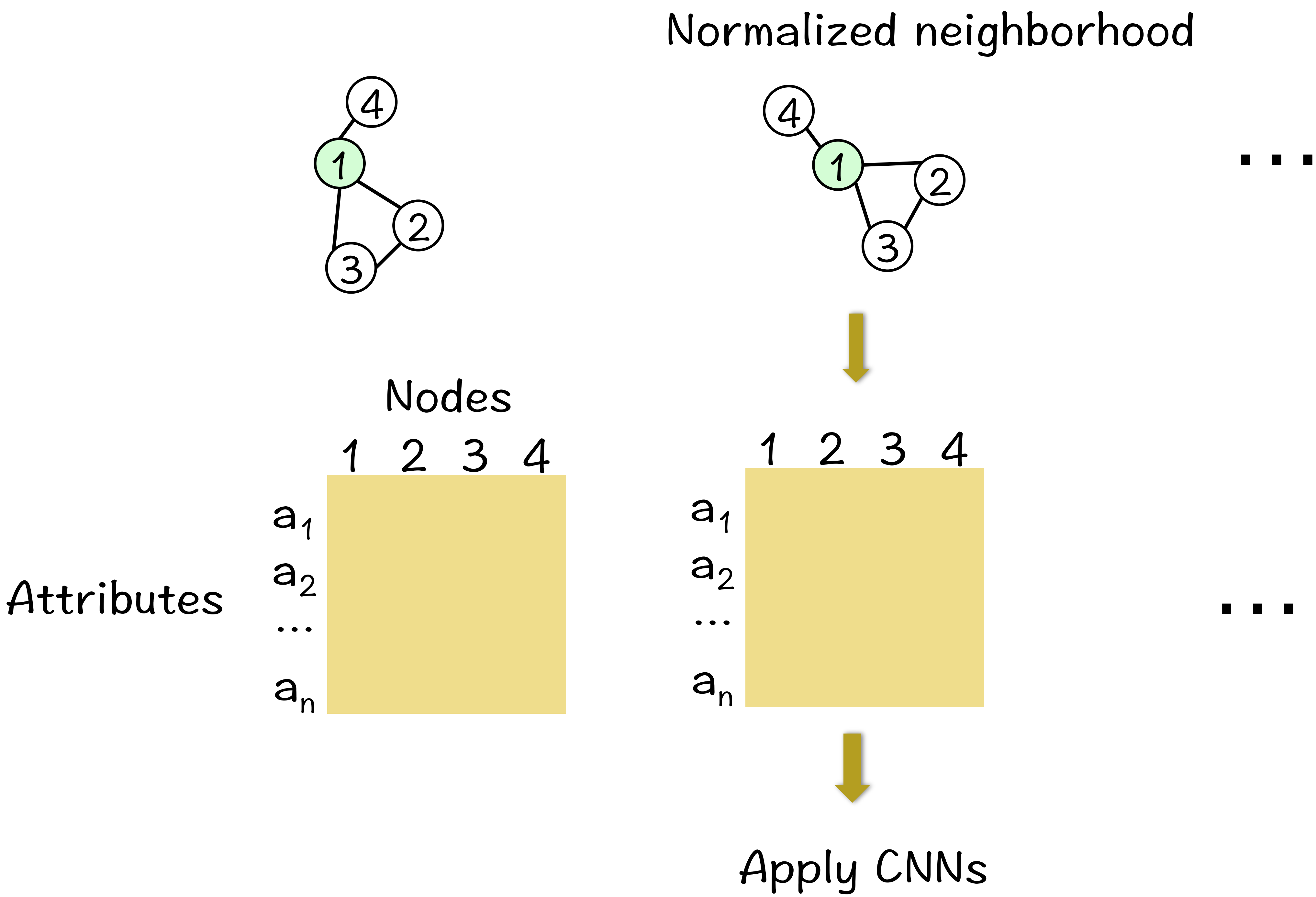
Matrix-based methods

- Represent graphs as matrices
  - Similar to images
  - Convolutional neural networks (CNNs) can be applied
- A simple way -- affinity matrix
  - Sensitive to node order permutations
  - Isomorphic graphs can be mapped to different matrices
  - Problem: how to find a good intermediate matrix?

PATCHY-SAN [Niepert+ '16]

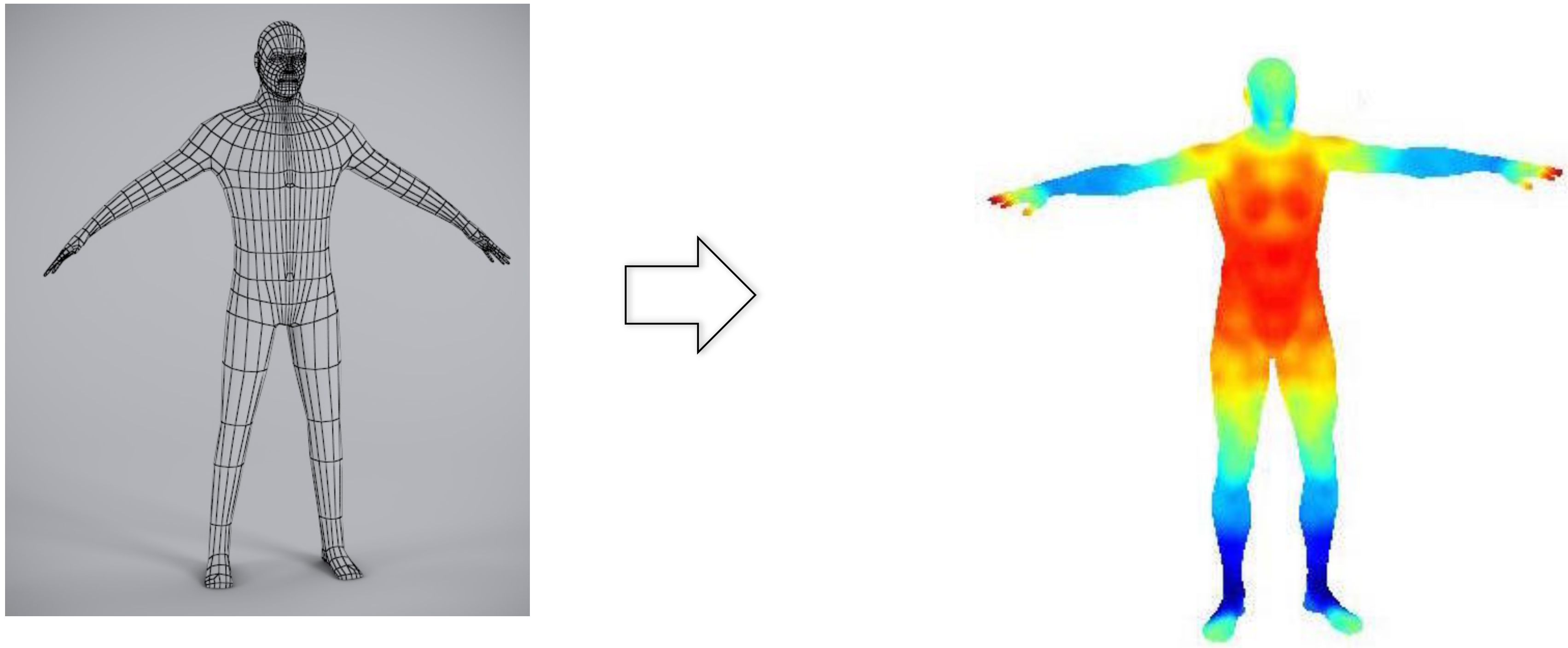


PATCHY-SAN [Niepert+ '16]



DeepGraph [Li+ '17a]

- Heat Kernel Signature (HKS)
  - Proposed in computer vision [Sun+, '09]
  - Represent the surface of 3D objects
  - Model the amount of heat flow on nodes overtime
    - Simulated on the snapshot of a graph



Heat kernel

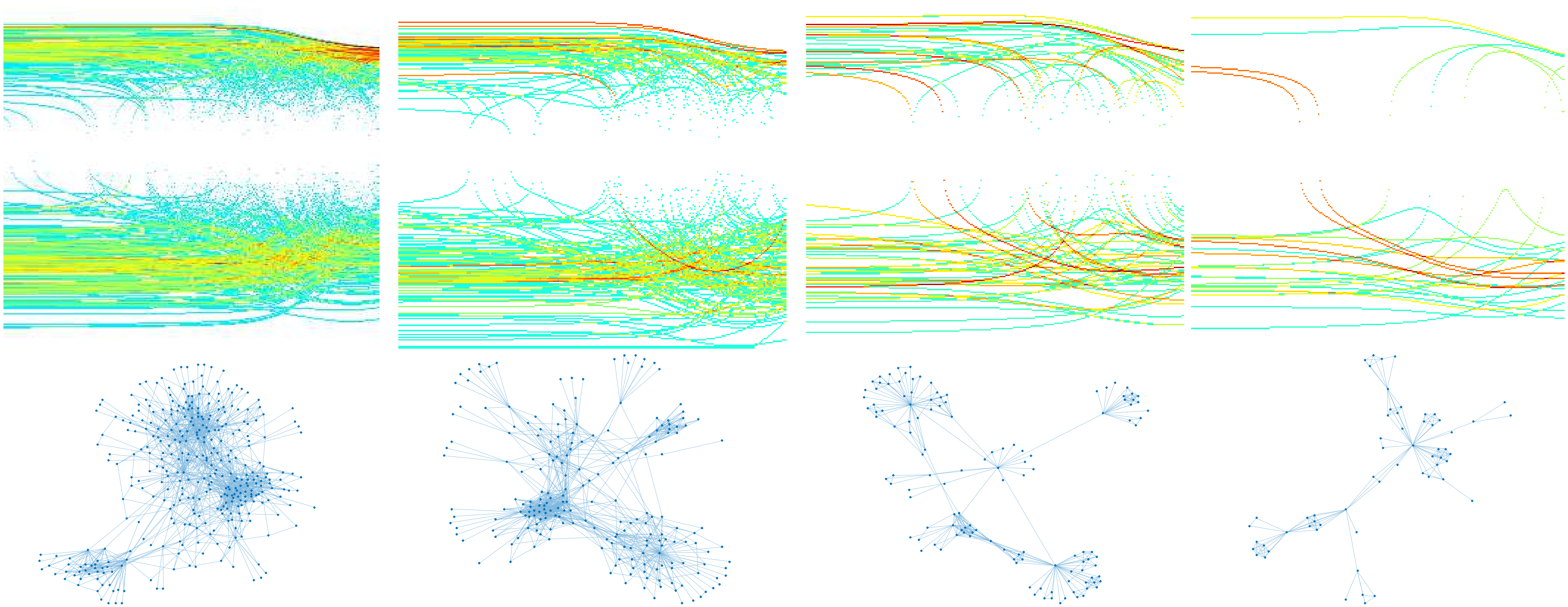
- There is a unit amount of heat on each node
- Heat starts to flow at time t = 0
- $h_t(i,j)$ is the amount of heat flow
  - Among node i and j after time t
  - Through all edges between i and j
- Calculate $h_t(i,j)$
  - f(t, i, j, eigenvalue, eigenvector of g(adjacency matrix))

HKS Graph Descriptor

- Heat kernel signature (HKS) H
  - $H_{i,t} = h_t(i,i)$
  - i-th node, t-th sampled time point
- HKS Graph descriptor S
  - Independent of #nodes
  - Compute histograms for each column $H_{\cdot,t}$
  - $S_{k,t}$ -- #nodes in k-th bin at time t
  - Row -- heat density dynamics over diffusion steps
  - Column -- static heat density patterns at t

Visualizing the graph descriptor

Convolutional architecture can be applied



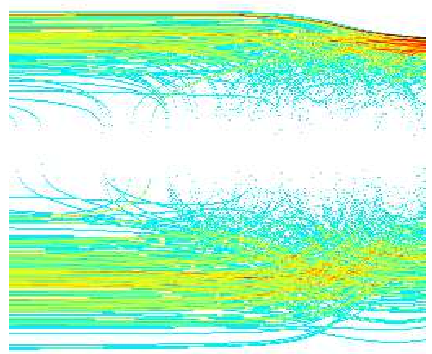
Friendship network from Facebook

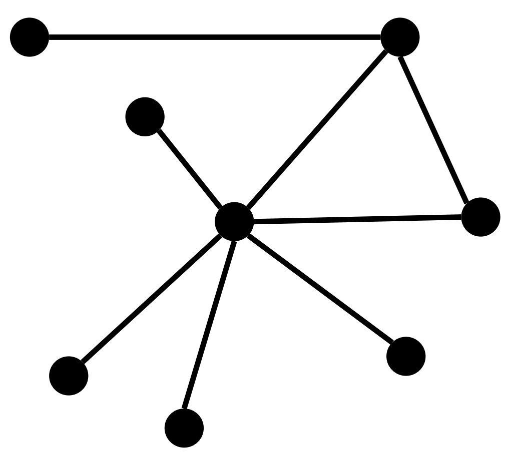
Author's Collaboration network from ACL

Pipeline

Low-level representation of network

HKS Graph descriptor

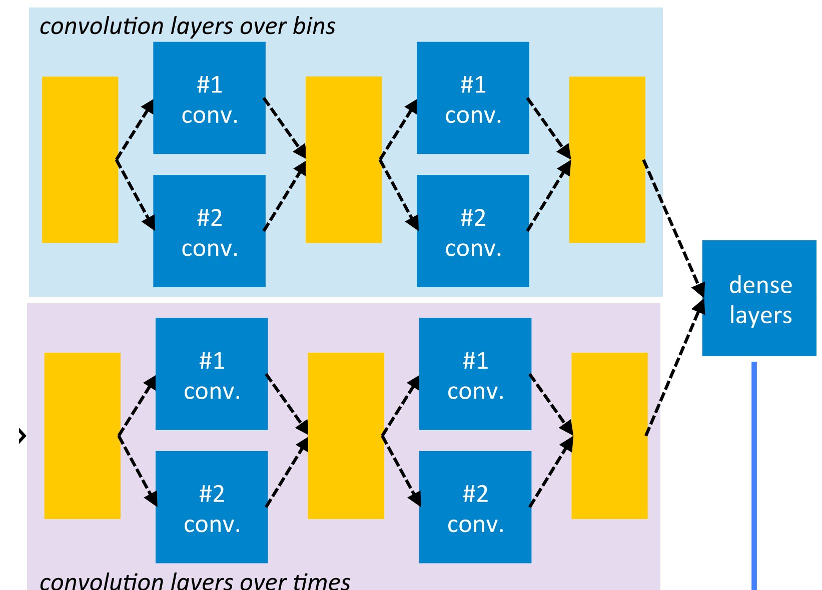




Graph structure

High-level representation of network

Multi-column, multi-resolution convolutional neural network



Output unit

Prediction of the network growth

Road map

- Non end-to-end method
  - Graph kernels
    - Manually designed kernel matrix
    - Kernel matrix is later used for down-stream tasks
- End-to-end methods
  - Matrix-based
  - Sequence-based
  - Graphical model based

Current node embedding methods

- DeepWalk [Perozzi+ '14] and node2vec [Grover+ '16]
  - Sample random walk sequences
  - Sequence ⇔ sentence
  - Node ⇔ word
  - Word2vec can be used [Mikolov+ '13]
- DeepWalk, LINE [Tang+ '15] and node2vec
  - Obtain graph embedding
  - Average node embeddings
  - Lead to significant loss of information

DeepCas [Li+ '17b]

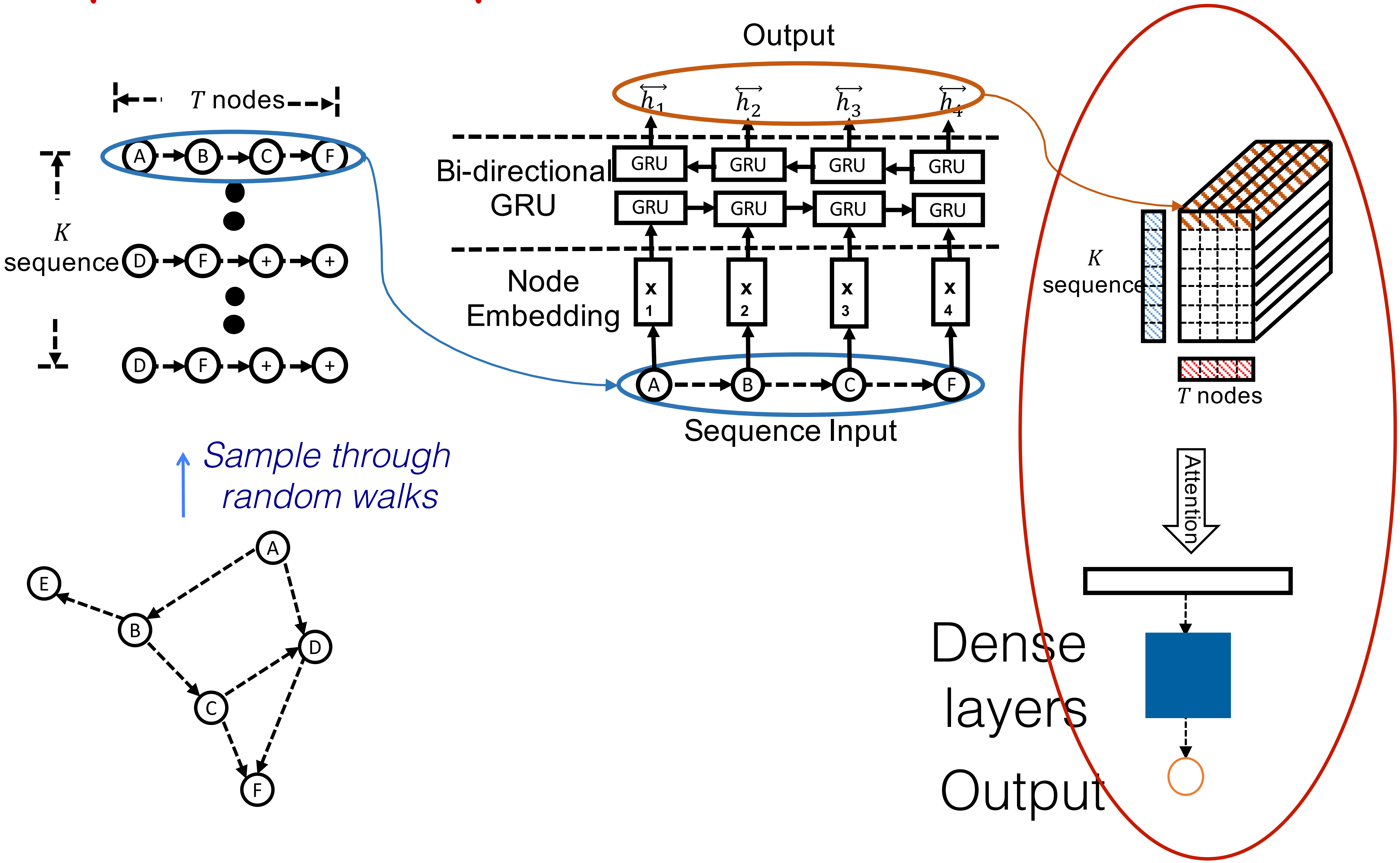
- Inspired by DeepWalk [Perozzi+ '14]
  - Make an analogy
  - Node ⇔ word
  - Sampled random path ⇔ sentence

We can adapt deep learning methods developed for text

How to assemble by end-to-end learning?

  - Graph ⇔ document
  - A set of graphs ⇔ document collection

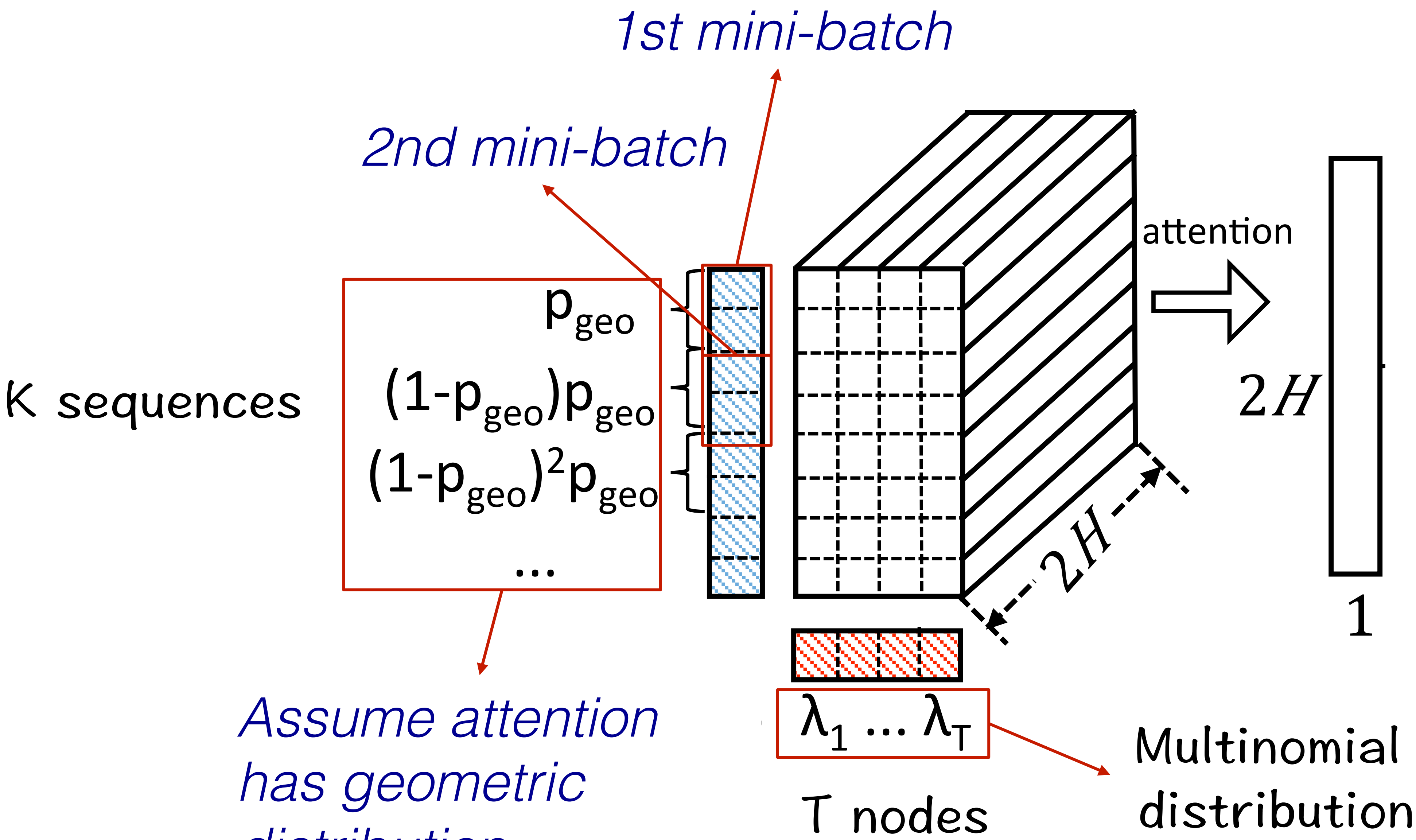
Pipeline of DeepCas



From sequence to graph representation

- Random walk has a terminating probability
  - Decides the expected #sequences
  - Learn it by examining
    - Represent the graph well → good prediction
  - Intuition
    - We sample enough sequences
    - Partition the sequences into "mini-batches"
    - Read in more until enough → stop random walk
  - Implement the intuition
    - A geometric distribution of attentions over mini-batches

Assemble sequences to a document (graph)



Assume attention has geometric distribution.

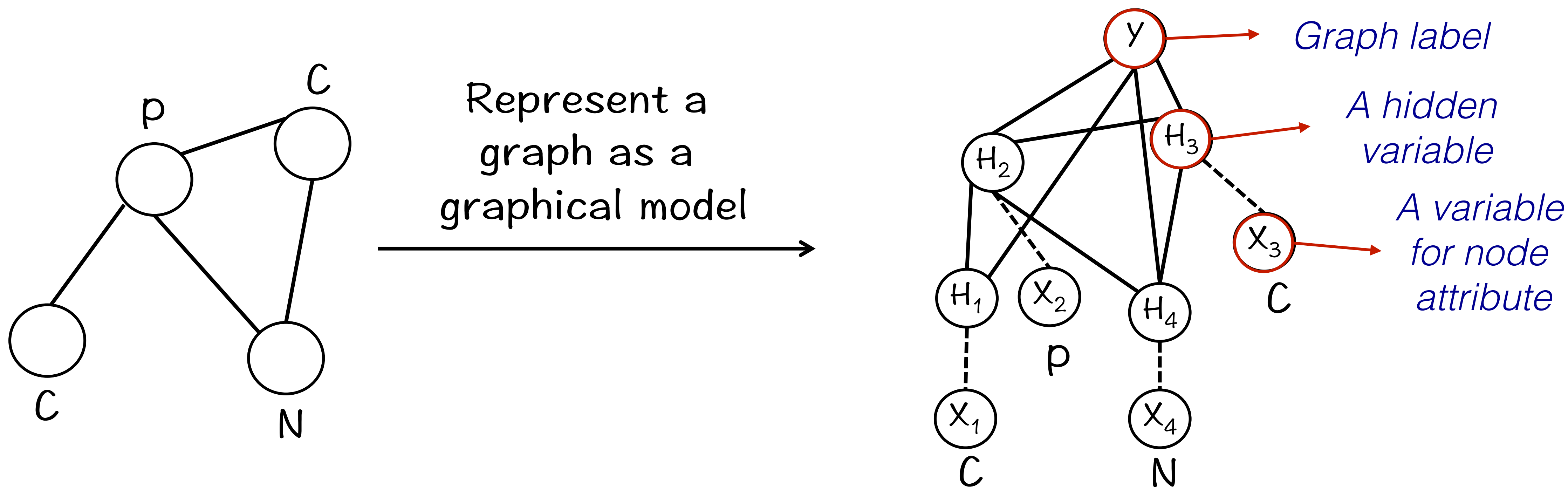
We learn p_{geo} to learn the actual #seq

Road map

- Non end-to-end method
  - Graph kernels
    - Manually designed kernel matrix
    - Kernel matrix is later used for down-stream tasks
- End-to-end methods
  - Matrix-based
  - Sequence-based
  - Graphical model based

Structure2vec [Dai+ '16]

- Construct graphical models for graphs



- A Markov random field $p(\{H_i\},\{X_i\}) \propto \prod_{i \in V} \Phi(H_i, X_i) \prod_{(i,j) \in E} \Psi(H_i, H_j)$
  - Φ: node potentials
  - Ψ: edge potentials

Embedding latent variable models

- Standard maximum likelihood estimation is difficult
- Embed the posterior marginal $p(H_i \mid \{x_i\})$ to u_i
  - $u_i = \int_{\mathcal{H}} \phi(h_i) p(h_i | \{x_i\}) dh_i$
  - $\phi(h_i)$ is a feature map to be learned
  - u_i is an embedding vector for node i

Embedding latent variable models

- Standard maximum likelihood estimation is difficult
- Embed the posterior marginal $p(H_i \mid \{x_i\})$ to u_i
- u_i can be computed by approximate inference
    - Parameterize it as a neural network
    - $\tilde{u}_i = \sigma\left(W_1 x_i + W_2 \Sigma_{j \in N(i)} \tilde{u}_j + W_3 \Sigma_{j \in N(i)} x_j\right)$
    - $\{W_1, W_2, W_3\}$ are parameters
    - $N(i)$ are neighbors of i
    - σ is an activation function

Discriminative training

- We have embedding vectors $\{u_i\}$
  - $\tilde{u}_i = \sigma\ (W_1 x_i + W_2 \Sigma_{j \in N(i)}\ \tilde{u}_j + W_3 \Sigma_{j \in N(i)} x_j)$
- Represent a graph by $\Sigma_i\ \tilde{u}_i$
- Minimize the empirical square loss
  - $(y - \theta^T \sigma\ (\Sigma_i\ \tilde{u}_i))^2$
  - y is the graph label
  - θ is a parameter

Conclusion

- Learning Representations of Entire Networks
  - End-to-end methods usually work better
    - When there are particular tasks at hand
  - No general consensus on which methods consistently work better

Outline

- **Part I: Learning Node Representations of Networks**
  - Laplacian Eigenmap
  - Word2Vec
  - LINE, DeepWalk, and Node2Vec
- **Part II: Visualizing Networks and High-Dimensional Data**
  - t-SNE
  - LargeVis
- **Pat III: Learning Representations of Entire Networks**
  - Graph kernels
  - End-to-end methods
- **Part IV: Summary, Challenges & Future Work**

Summary

- Network representation is a new methodology for analyzing and mining networks
- State-of-the-art approaches for node representation learning
  - LINE, DeepWalk, and Node2Vec
  - Moving towards to task-specific node representations (e.g., PTE and GraphConv)
- Visualizing large-scale networks and high-dimensional data
  - LargeVis
  - Sales up to tens of millions of nodes or data points
- Learning representations of network substructures
  - DeepCas, Stru2Vec

Challenges & Future Work

- Scalability
  - How to scale up to networks with billions of nodes
- Hierarchical representations
  - How to learn hierarchical representations of networks
- Dynamic
- Heterogeneous networks
  - Multiple types of nodes, multiple types of edges
- Learning isomorphism-invariance representations of entire networks
- ...

References

Node Embeddings

[Belkin et al. 2003] Mikhail Belkin and Partha Niyogi. **Laplacian Eigenmaps for Dimensionality Reduction and Data Representation**. Neural Computation, 2003.

[Mikolov et al. 2014] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, Jeffrey Dean. **Distributed Representations of Words and Phrases and their Compositionality.** NIPS 2014.

[Tang et al. 2015a] **Jian Tang**, Meng Qu, Mingzhe Wang, Jun Yan, and Qiaozhu Mei. **LINE: Large-scale Information Network Embedding**. WWW'15

[Perozzi et al. 2014] Bryan Perozzi, Rami Al-Rfou, Steven Skiena. DeepWalk: Online Learning of Social Representations. KDD'14

[Grover et al. 2016] Aditya Grover and Jure Leskovec. node2vec: Scalable Feature Learning for Networks. KDD'16

[Cao et al. 2015] Shaosheng Cao, Wei Lu, and Qiongkai Xu. GraRep: learning graph representations with global structural information. CIKM'15.

[Qu et al. 2017] Meng Qu, **Jian Tang**, Jingbo Shang, Xiang Ren, Ming Zhang, and Jiawei Han. **Learning Distributed Node Representations for Networks with Multiple Views.**

[Yang et al. 2015] Cheng Yang, Zhiyuan Liu, Deli Zhao, Maosong Sun, Edward Y. Chang. **Network representation learning with rich text information.** IJCAI 2015.

[Kipf et al. 2016] Thomas N.Kipf and Max Welling. Variational Graph Auto-encoders. NIPS Workshop 2016.

[Liao et al. 2017]Lizi Liao, Xiangnan He, Hanwang Zhang, and Tat-Seng Chua. Attributed Social Network Embedding. arXiv, 2017.

[Tang et al. 2015b] **Jian Tang**, Meng Qu, and Qiaozhu Mei. **PTE: Predictive Text Embedding through Large-scale Heterogeneous Text Networks**. KDD'15.

[Kipf et al. 2017]Thomas N.Kipf and Max Welling. **Semi-Supervised Classification with Graph Convolutional Networks**. ICLR'17.

[Chang et al. 2017] Siyu Chang, Wei Han, Jiliang Tang, Guo-Jun Qi, Charu C. Aggarwal, Thomas S. Huang. Heterogeneous network embedding via Deep Architectures. KDD'15

[Chen et al. 2017] Ting Chen and Yizhou Sun, "Task-Guided and Path-Augmented Heterogeneous Network Embedding for Author Identification. WSDM'17.

References

[Wang et al. 2017] Daixin Wang, Peng Cui, Wenwu Zhu. Structural deep network embedding. KDD, 2016.

Node Visualizations

[Maaten et al. 2008] L.J.P. van der Maaten and G.E. Hinton. Visualizing High-Dimensional Data Using t-SNE. JMLR, 2008.

[Maaten et al. 2014] L.J.P. van der Maaten. Accelerating t-SNE using Tree-Based Algorithms. JMLR, 2014.

[Tang et al. 2016] Jian Tang, Jingzhou Liu, Ming Zhang, and Qiaozhu Mei. **Visualizing Large-scale and High-dimensional Data.** WWW'16

Graph Embeddings

[Li+ '17a] Cheng Li, Xiaoxiao Guo, and Qiaozhu Mei. 2016. DeepGraph: Graph Structure Predicts Network Growth. arXiv preprint arXiv:1610.06251 (2016).

[Niepert+ '16] Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. 2016. Learning convolutional neural networks for graphs. In Proceedings of the 33rd annual international conference on machine learning. ACM.

[Borgwardt+ '05] Borgwardt, Karsten M., and Hans-Peter Kriegel. "Shortest-path kernels on graphs." Data Mining, Fifth IEEE International Conference on. IEEE, 2005.

[Shervashidze+ '09] Shervashidze, Nino, et al. "Efficient graphlet kernels for large graph comparison." Artificial Intelligence and Statistics. 2009.

[Shervashidze+ '11] Shervashidze, Nino, et al. "Weisfeiler-lehman graph kernels." Journal of Machine Learning Research 12.Sep (2011): 2539-2561.

[Li+ '17b] Cheng Li, Jiaqi Ma, Xiaoxiao Guo, and Qiaozhu Mei. 2017. DeepCas: an End-to-end Predictor of Information Cascades. In Proceedings of the 26th international conference on World wide web.

[Dai+ '16] Dai, Hanjun, Bo Dai, and Le Song. "Discriminative embeddings of latent variable models for structured data." International Conference on Machine Learning. 2016.

tangjianpku@gmail.com

Optimization

- The gradient w.r.t. the embedding $\mathbf{y}_i$

$$\frac{\partial C}{\partial \mathbf{y}_i} = 4 \sum_{j \neq i} (p_{ij} - q_{ij}) q_{ij} Z (\mathbf{y}_i - \mathbf{y}_j),$$

- Z is the partition function:

$$Z = \sum_{k \neq l} (1 + \|\mathbf{y}_k - \mathbf{y}_l\|^2)^{-1}$$

- The complexity w.r.t. the number of data points *N* is O(N^2)
- Too expensive!

Barnes-Hut Approximation

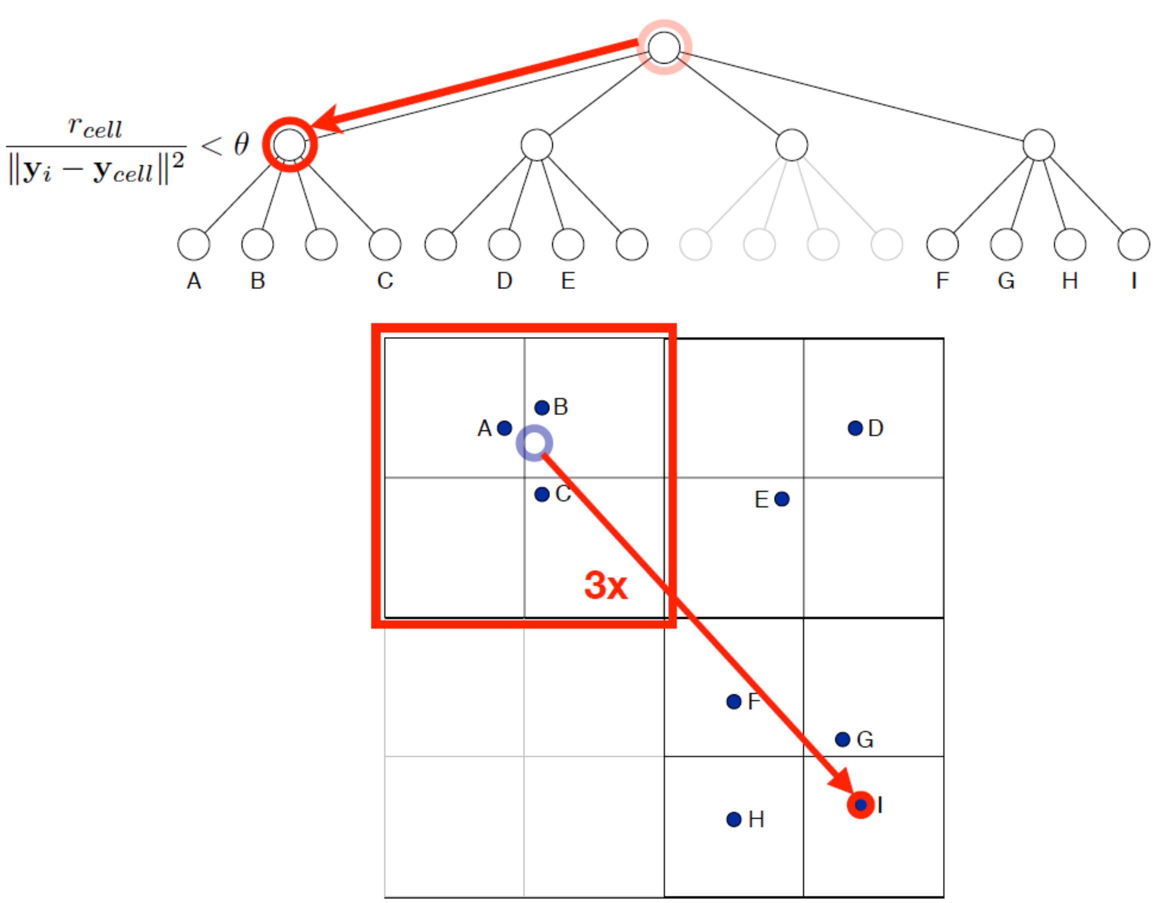
- Rewriting the gradient as:

$$\frac{\partial C}{\partial \mathbf{y}_i} = 4\left(\sum_{j\neq i} p_{ij}q_{ij}Z(\mathbf{y}_i - \mathbf{y}_j) - \sum_{j\neq i} q_{ij}^2 Z(\mathbf{y}_i - \mathbf{y}_j)\right),$$

Attractive forces
Complexity: linear to the number of edges

Repulsive forces
Complexity: O(N^2)

- Constructing a quadtree of the nodes according to the current low-dimensional representations



From O(N^2) to O(NLogN)!

Sum of node i and nodes in a cell: $-q_{ij}^2 Z(\mathbf{y}_i - \mathbf{y}_j)$

$$-N_{cell}q_{i,cell}^2 Z(\mathbf{y}_i - \mathbf{y}_{cell})$$