Cognitive AI:

—Understanding, Reasoning, and Decision

Jie Tang

Computer Science

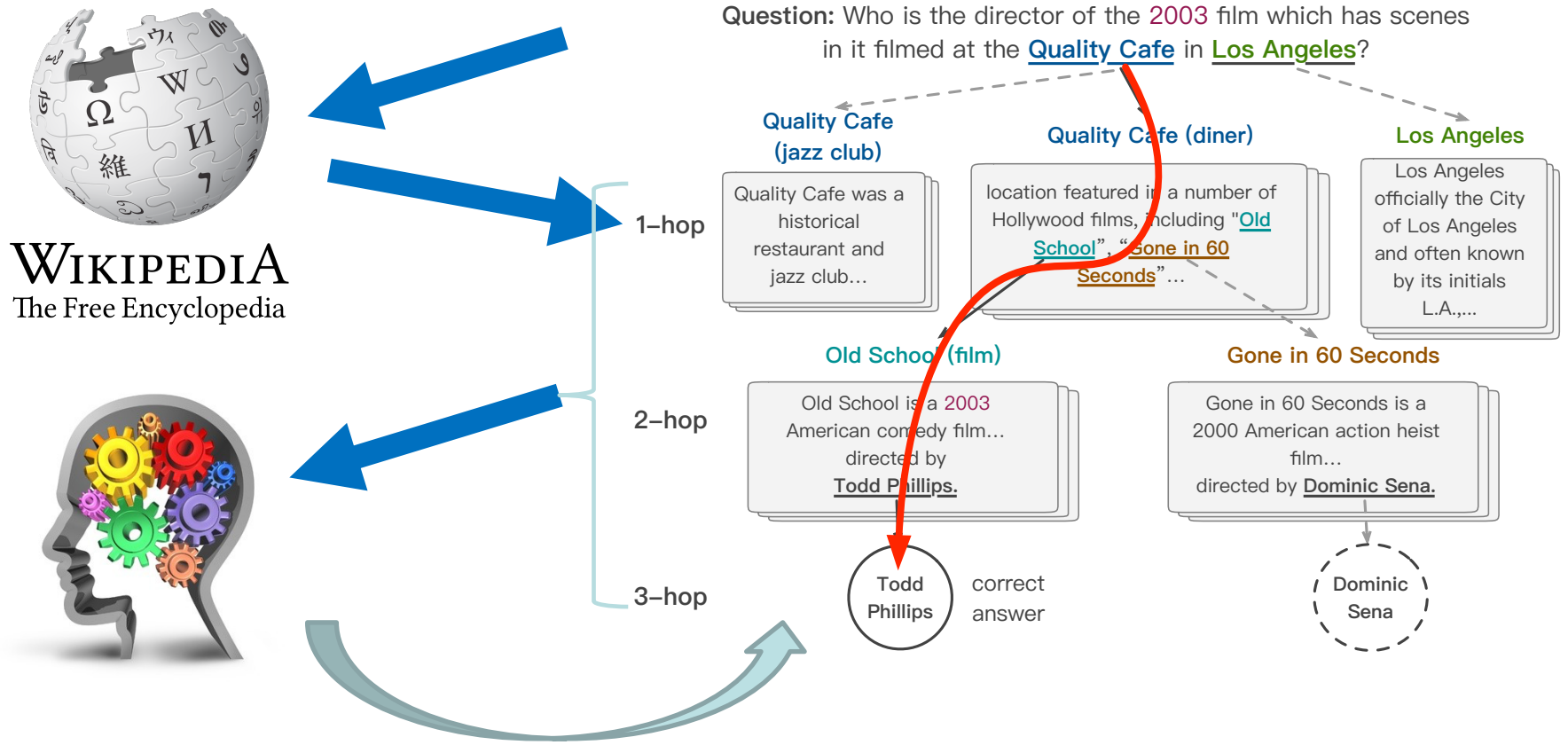
Tsinghua University

- Let us start with an example…

Reasoning

Question: Who is the director of the 2003 film which has scenes in it filmed at the Quality Cafe in Los Angeles?

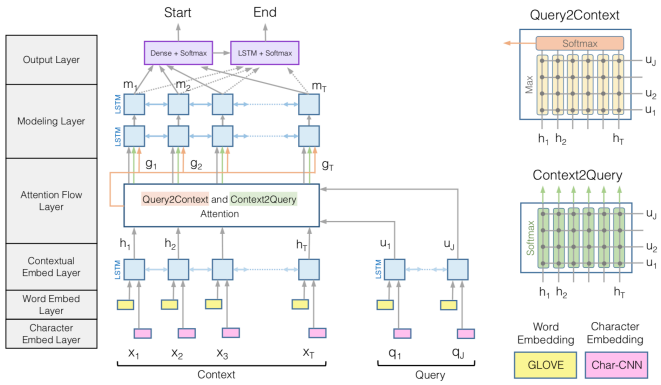
Cognitive Graph



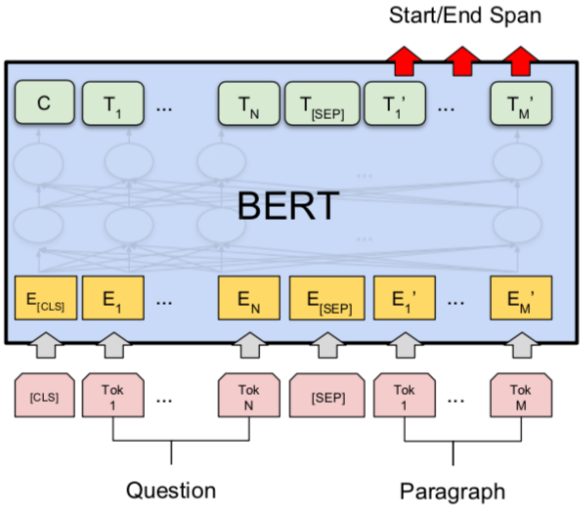
BIDAF, BERT, XLNet

- Modeling the whole document by pre-training
- However, lack of explainability

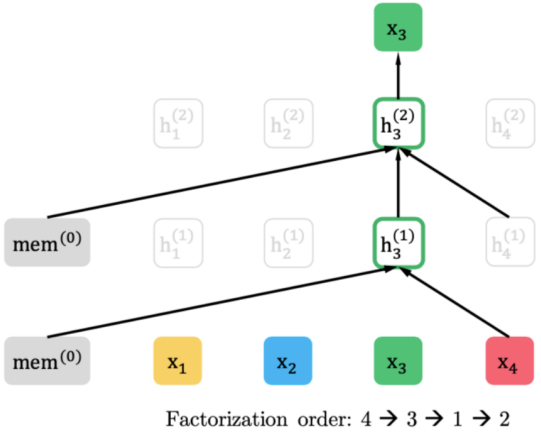
BiDAF



BERT



XLNet

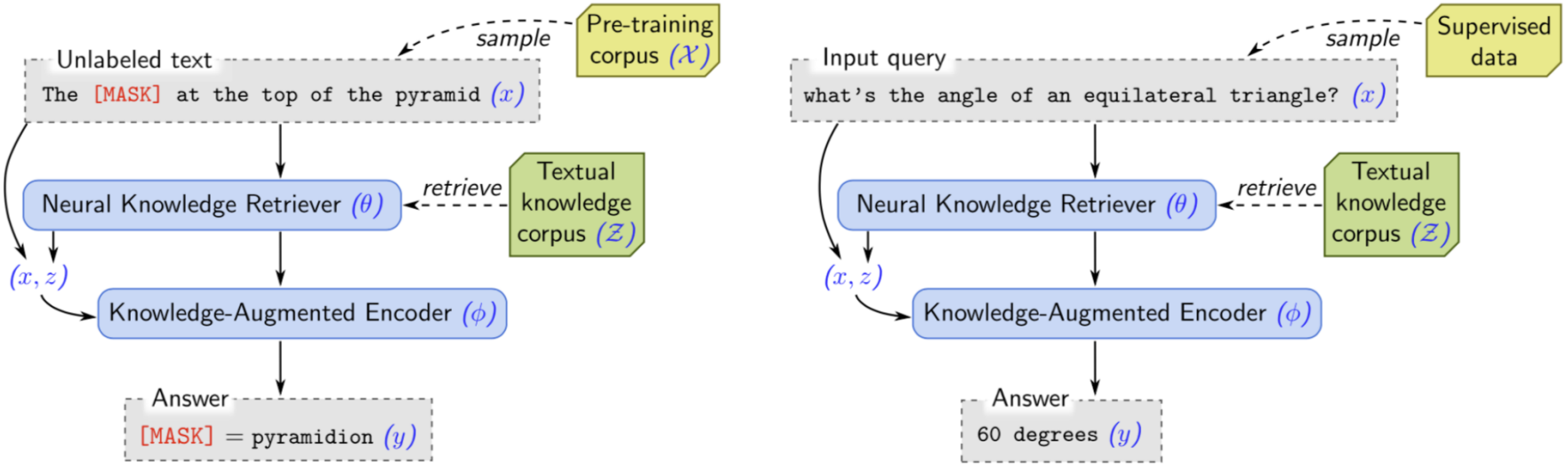


REALM: Retrieval-Augmented LM

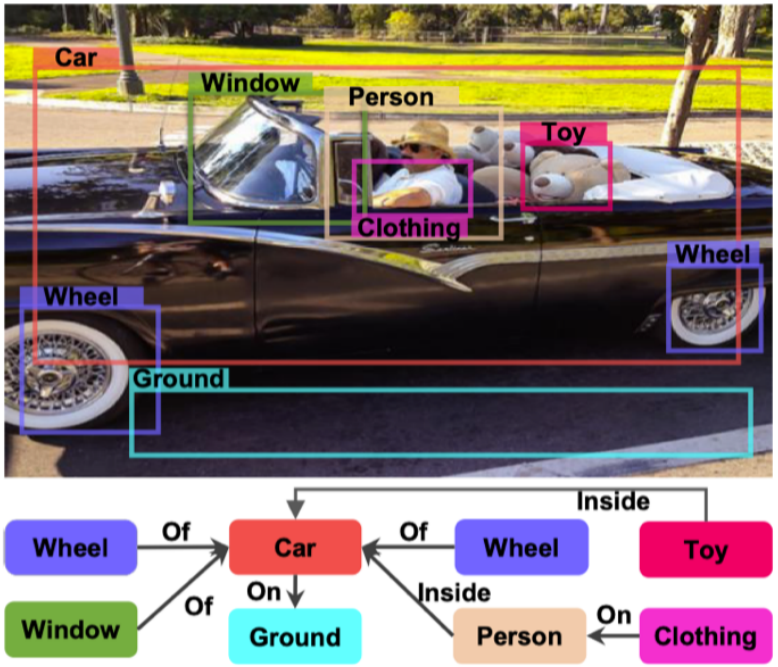
$$\text{BERT} : p(y|x)$$

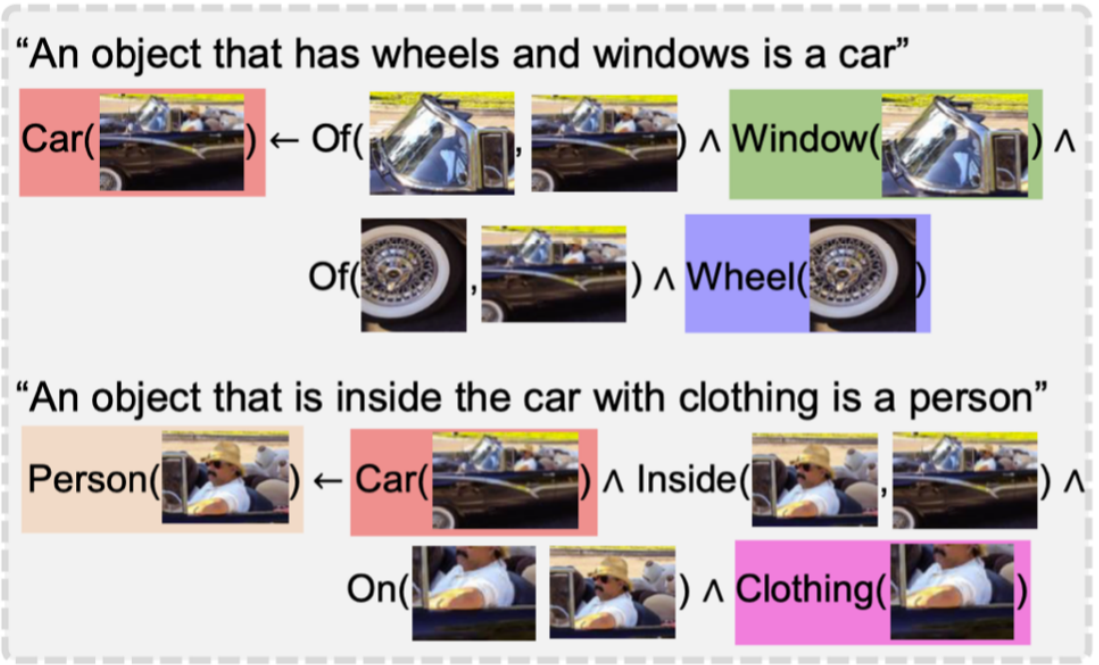
$$\text{REALM} : p(y|x) = \sum_{z \in \mathcal{Z}} p(y|z,x)p(z|x)$$

- where Z is the supporting set.



Inductive Logic Programming





They apply inductive logic learning on scene graphs generated by deep learning, to extract explainable rules of predicting the object class labels.

1. Yuan Yang and Le Song. Learn to Explain Efficiently via Neural Logic Inductive Learning. ICLR. 2020.

Challenge: only System 1 DL

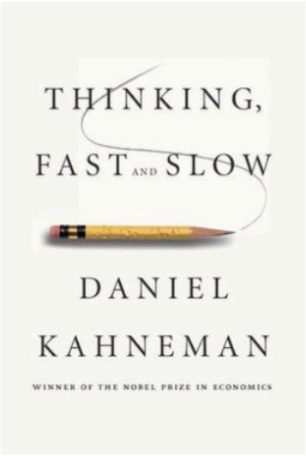
SYSTEM 1 VS. SYSTEM 2 COGNITION

2 systems (and categories of cognitive tasks):

Manipulates high-level / semantic concepts, which can be recombined combinatorially

System 1

- Intuitive, fast, **UNCONSCIOUS**, non-linguistic, habitual
- Current DL



System 2

- Slow, logical, sequential, **CONSCIOUS**, linguistic, algorithmic, planning, reasoning
- Future DL

Mila

3

1. From Bengio's NIPS'2019 Keynote

System 2 DL: Cognitive Graph

Question: Who is the director of the 2003 film which has scenes in it filmed at the Quality Cafe in Los Angeles?

Quality Café

The Quality Cafe is a now-defunct diner in Los Angeles, California. The restaurant has appeared as a location featured in a number of Hollywood films, including Old School, Gone in 60 Seconds, ...

Los Angeles

Los Angeles is the most populous city in California, the second most populous city in the United States, after New York City, and the third most populous city in North America.

Alessandro Moschitti

Alessandro Moschitti is a professor of the CS Department of the University of Trento, Italy. He is currently a Principal Research Scientist of the Qatar Computing Research Institute (QCRI)

WIKIPEDIA
The Free Encyclopedia

Old School

Old School is a 2003 American comedy film released by Dream Works Pictures and The Montecito Picture Company and directed by Todd Phillips.

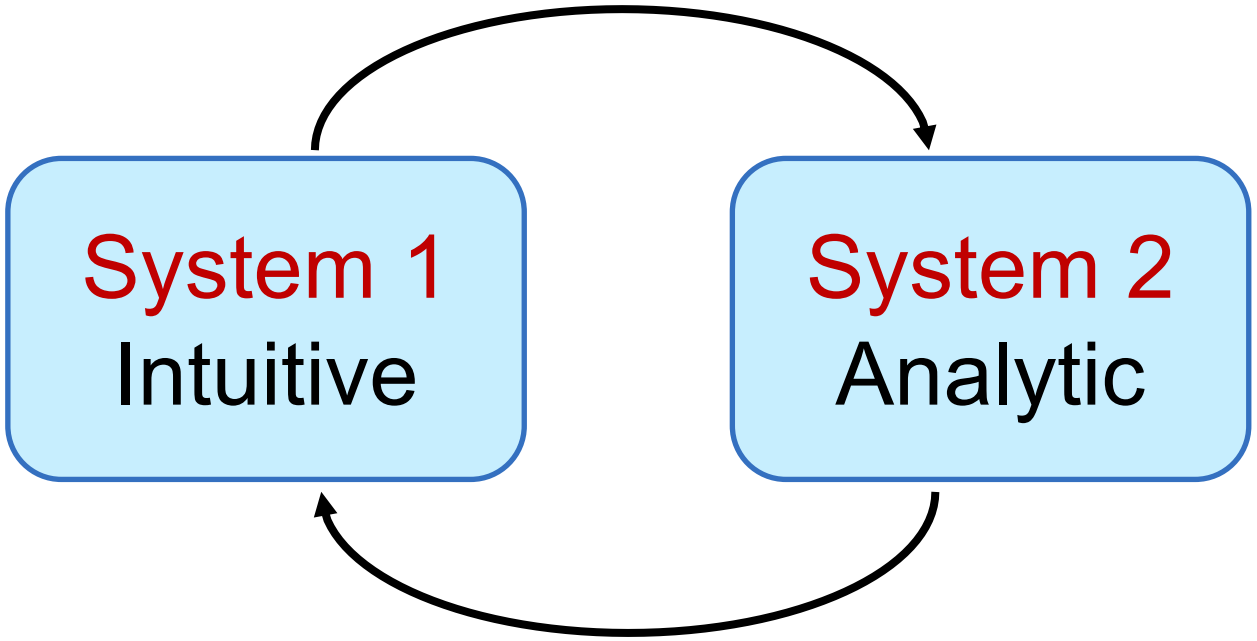
Todd Phillips

Todd Phillips is an American director, producer, screenwriter, and actor. He is best known for writing and directing films, including Road Trip (2000), Old School (2003), Starsky & Hutch (2004), and The Hangover Trilogy.

Tsinghua University

Tsinghua University is a major research university in Beijing and dedicated to academic excellence and global development. Tsinghua is perennially ranked as one of the top academic institutions in China, Asia, and worldwide...

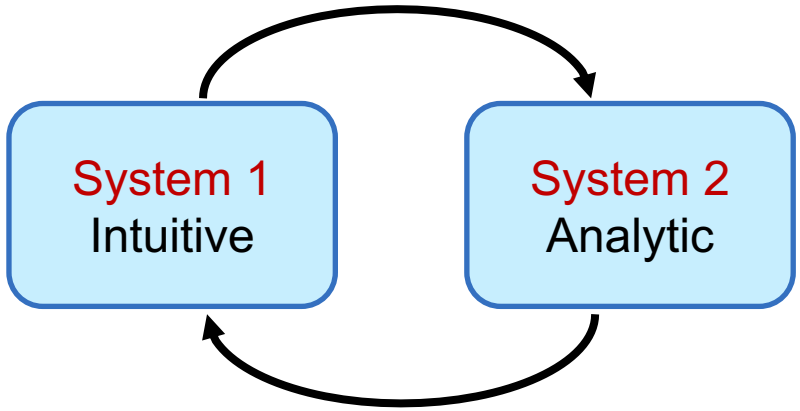
Cognitive Science



Dual Process Theory (Cognitive Science)

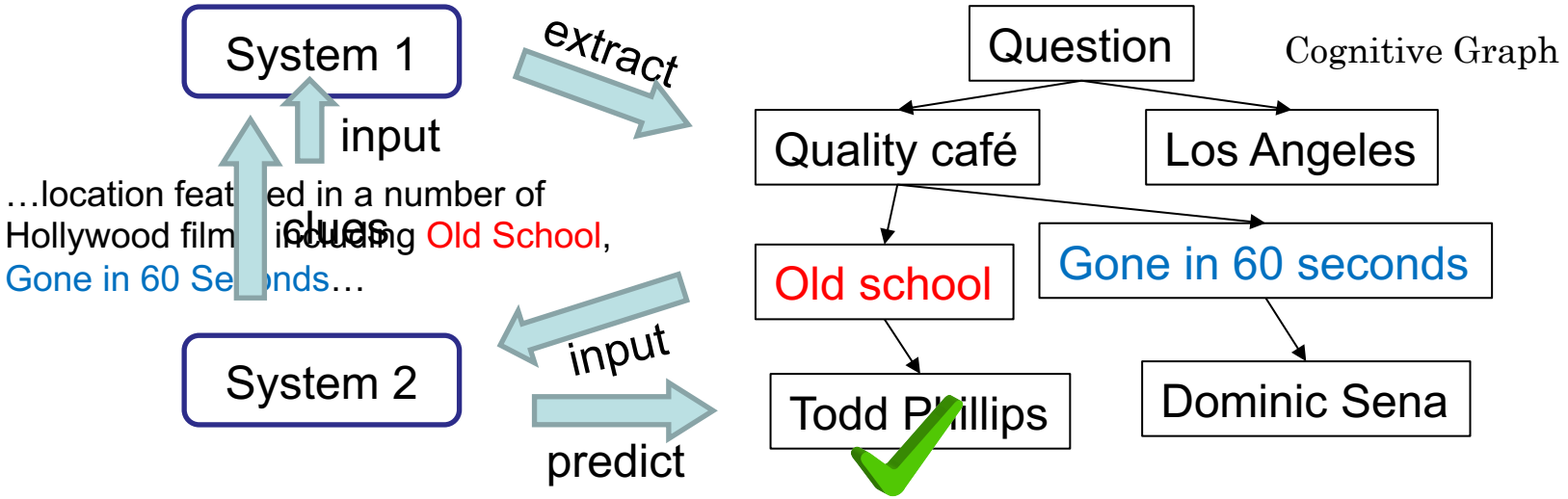
Reasoning w/ Cognitive Graph

- System 1:
  - Knowledge expansion by association in text when reading
- System 2:
  - Decision making w/ all the information



CogQA: Cognitive Graph for QA

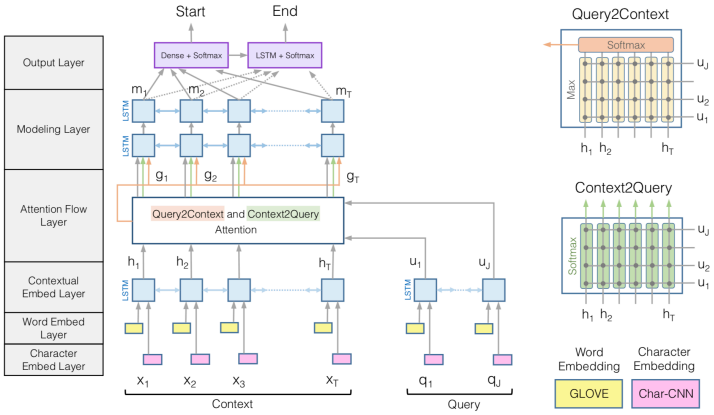
- An iterative framework corresponding to dual process theory
- System 1
  - extract entities to build the cognitive graph
  - generate semantic vectors for each node
- System 2
  - Do reasoning based on semantic vectors and graph
  - Feed clues to System 1 to extract next-hop entities



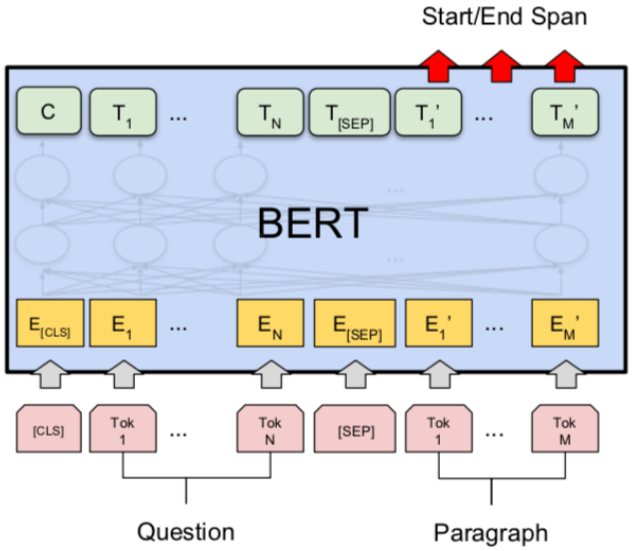
System 1: BIDAF, BERT

- reading comprehension: target at understanding the whole paragraph

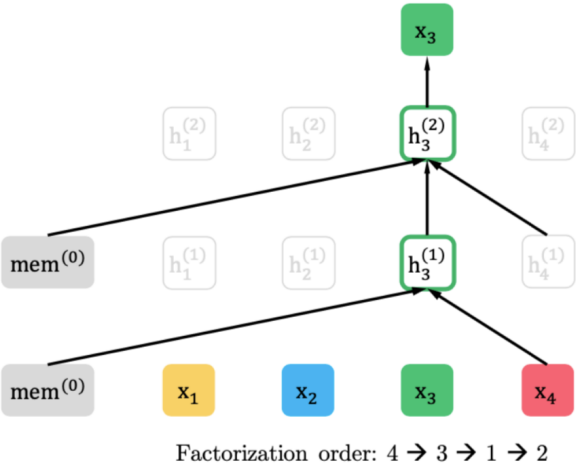
BiDAF



BERT

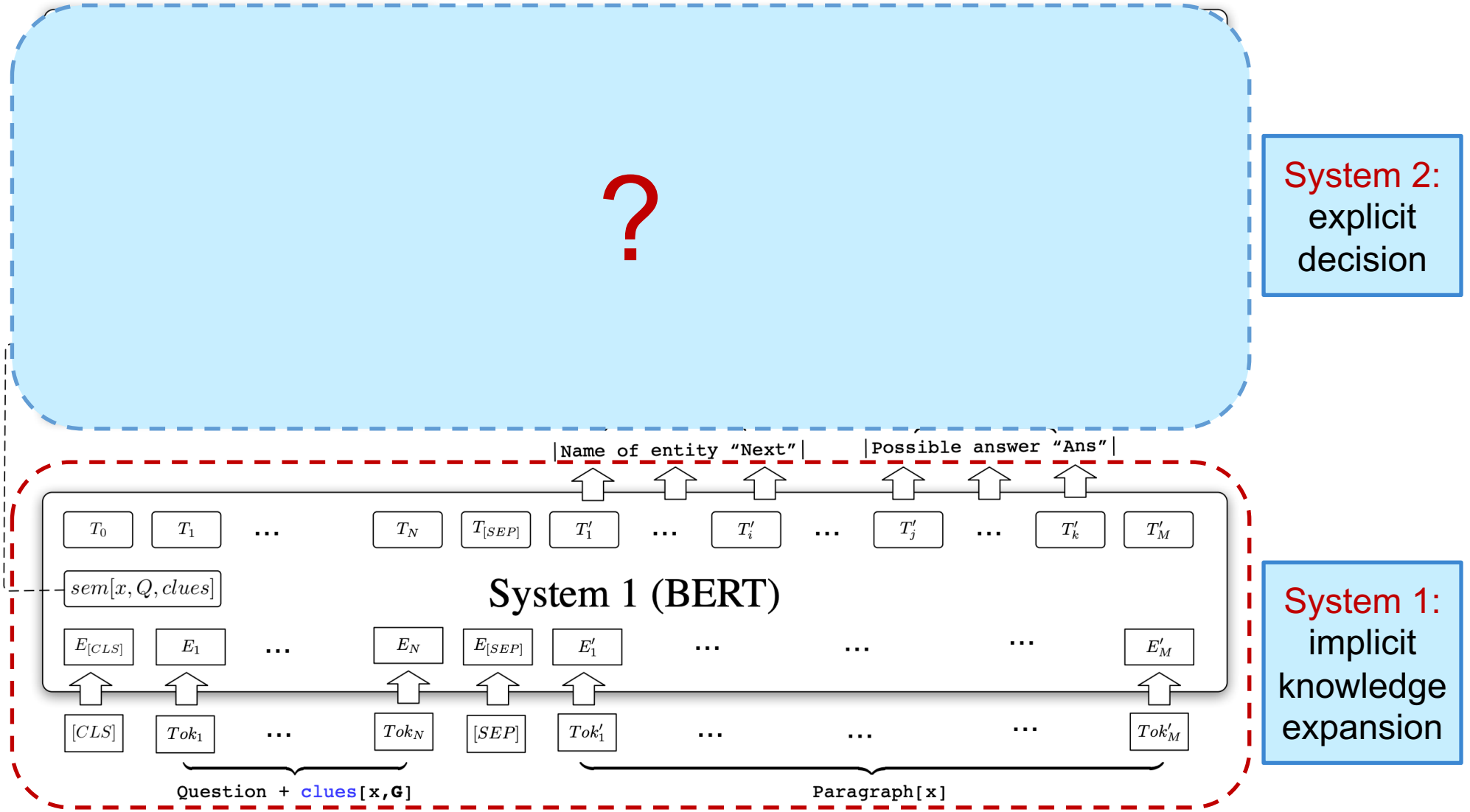


XLNet



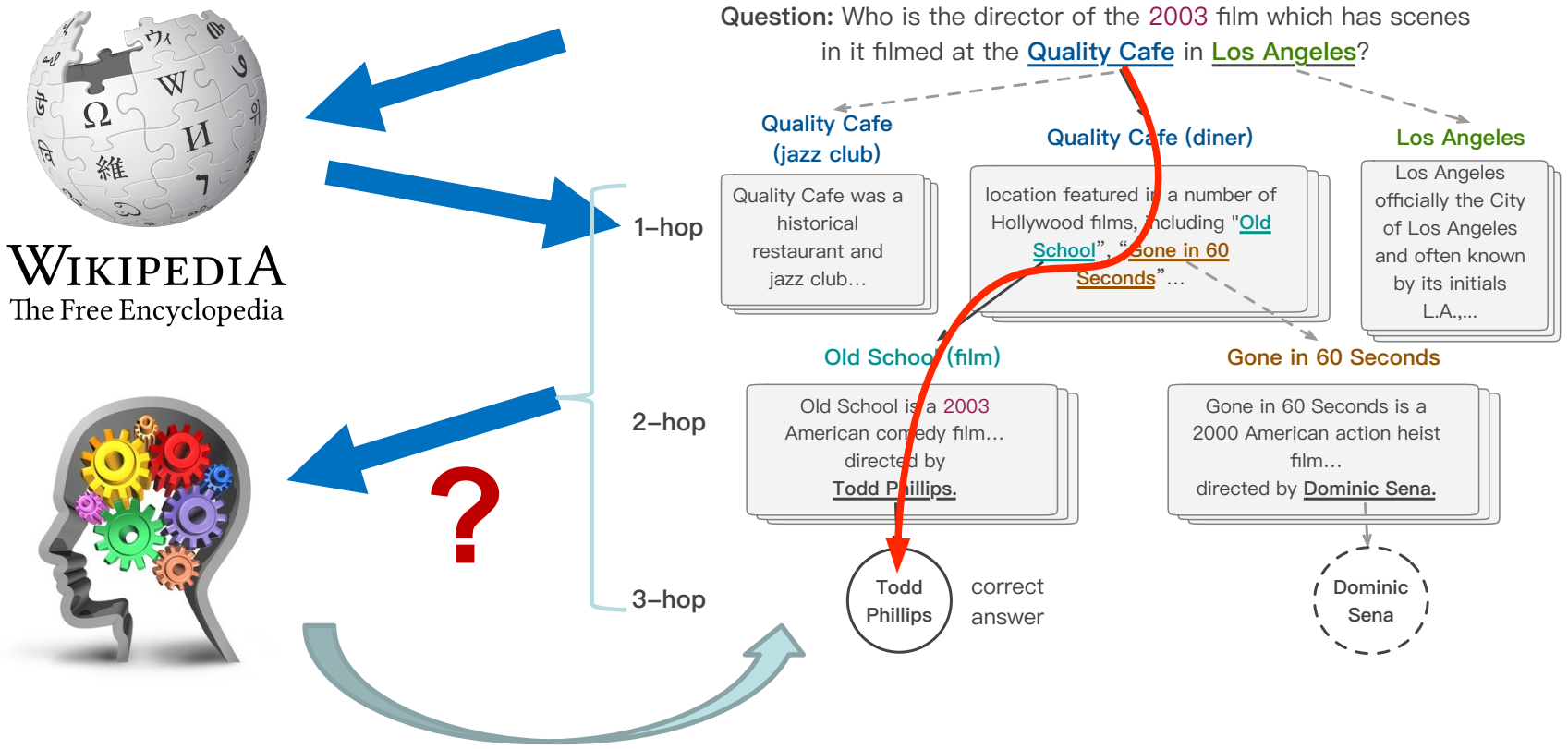
1. Rajpurkar, Pranav, et al. "SQuAD: 100,000+ Questions for Machine Comprehension of Text." EMNLP. 2016.

Cognitive Graph: DL + Dual Process Theory

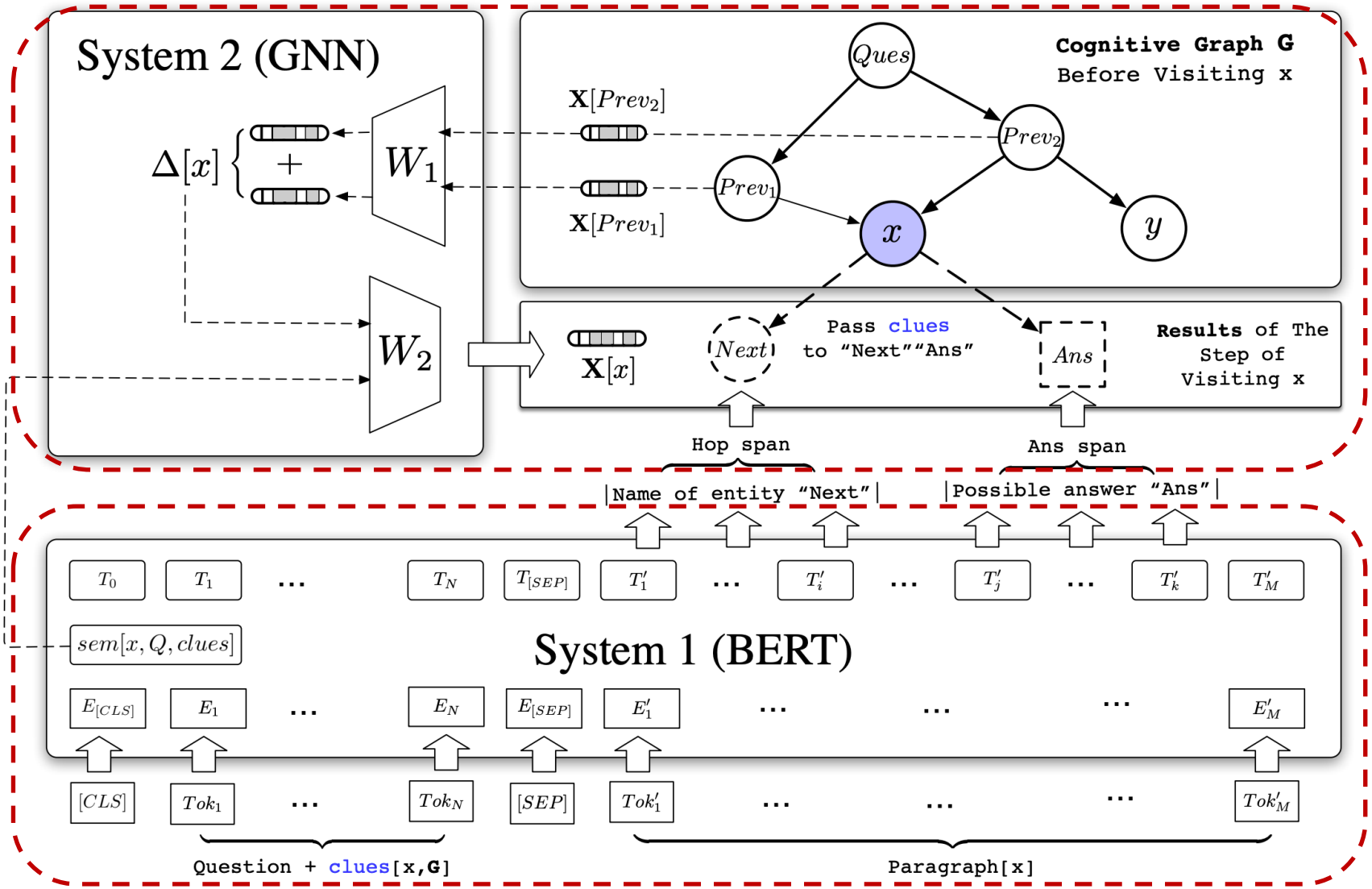


1. M. Ding, C. Zhou, Q. Chen, H. Yang, and J. Tang. Cognitive Graph for Multi-Hop Reading Comprehension at Scale. ACL'19.

System 2: Reasoning, and Decision

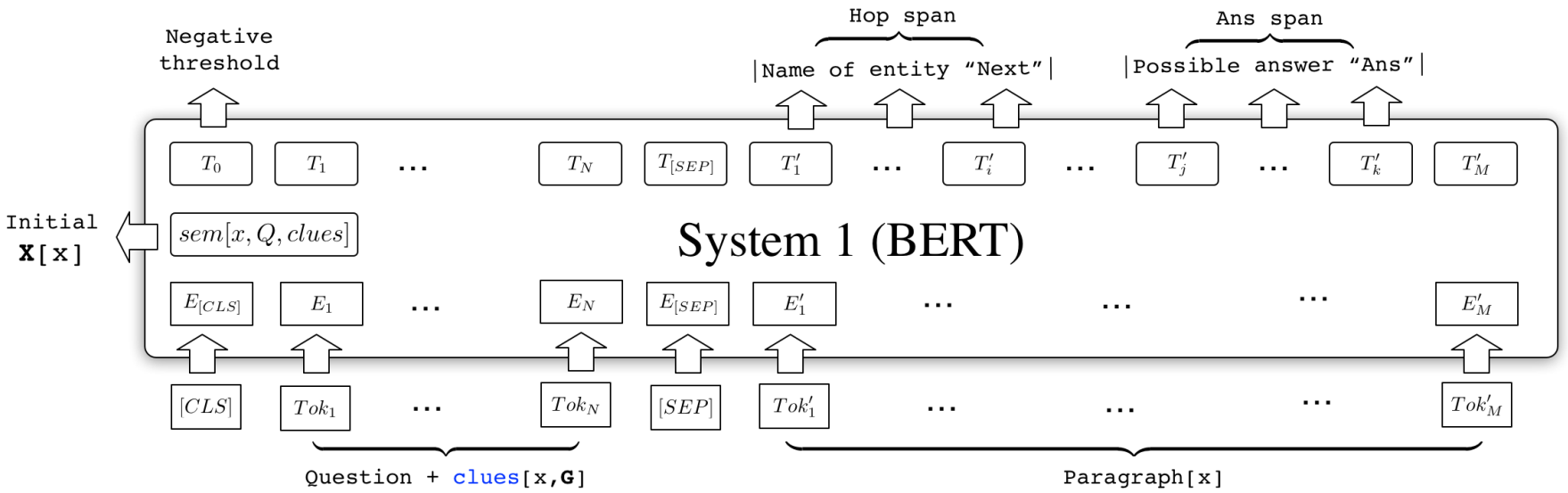


Cognitive Graph: DL + Dual Process Theory



1. M. Ding, C. Zhou, Q. Chen, H. Yang, and J. Tang. Cognitive Graph for Multi-Hop Reading Comprehension at Scale. ACL'19.

System 1: the BERT Implementation



$$[CLS]\ Question\ [SEP]\ clues[x, \mathcal{G}]\ [SEP] \quad Para[x]$$

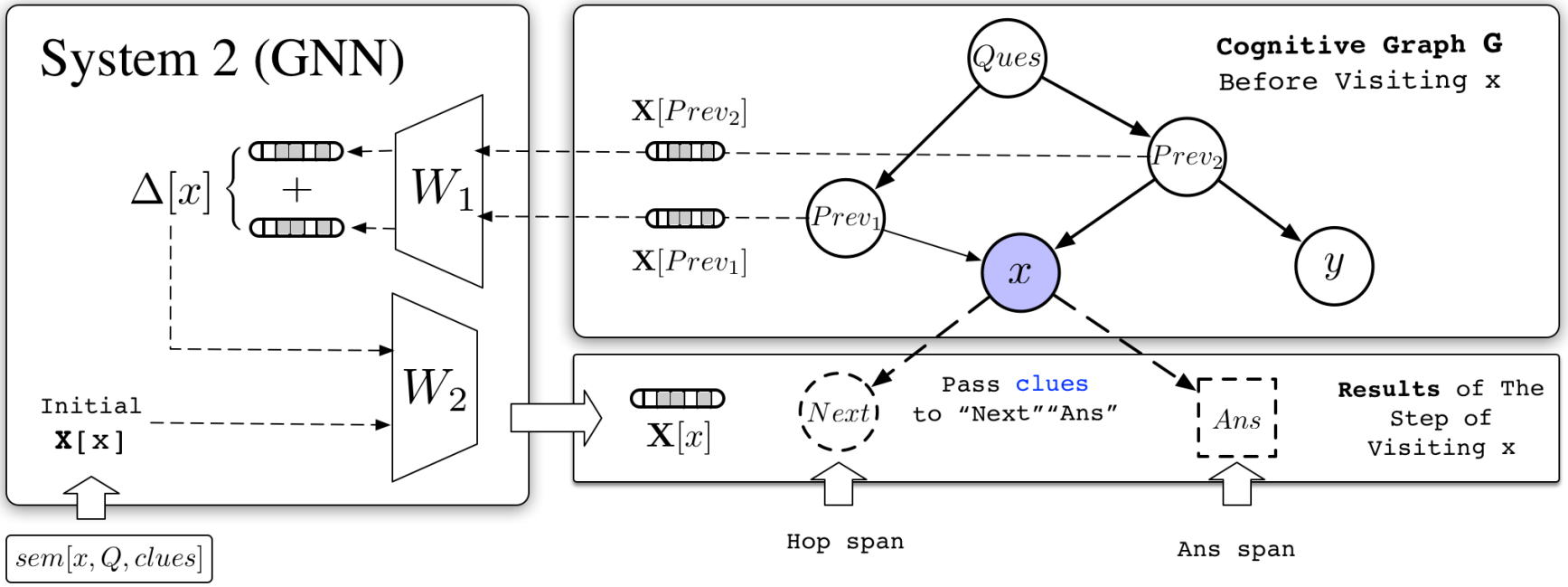
Sentence A Sentence B

$$P_{ans}^{start}[i] = \frac{e^{\mathbf{S}_{ans} \cdot \mathbf{T}_i}}{\sum_j e^{\mathbf{S}_{ans} \cdot \mathbf{T}_j}}$$

$$end_k = \underset{start_k \le j \le start_k + maxL}{\arg\max} P_{ans}^{end}[j]$$

- Extract top-k next-hop entities and answer candidates respectively
  - Predict the start and end probabilities of each position
- Generate semantic vectors for entities based on their documents
- Take the 0-th probability as negative threshold
  - Ignore the spans whose start probabilities are small than the negative threshold

System 2: the GNN implementation



At each step, hidden representations $\mathbf{X}$ for nodes are updated according to the propagation rules:

$$\Delta = \sigma((AD^{-1})^T \sigma(\mathbf{X}W_1))$$
$$\mathbf{X}' = \sigma(\mathbf{X}W_2 + \Delta)$$

Predictor $\mathcal{F}$ is a two-layer MLP, which predicts the final answer based on hidden representations $\mathbf{X}$:

$$answer = \underset{\text{answer node } x}{\arg\max}\ \mathcal{F}(\mathbf{X}[x])$$

Performance

- HotpotQA is a dataset with leaderboard similar to SQuAD
- CogQA ranked 1st from 21, Feb to 15, May (nearly 3 month)

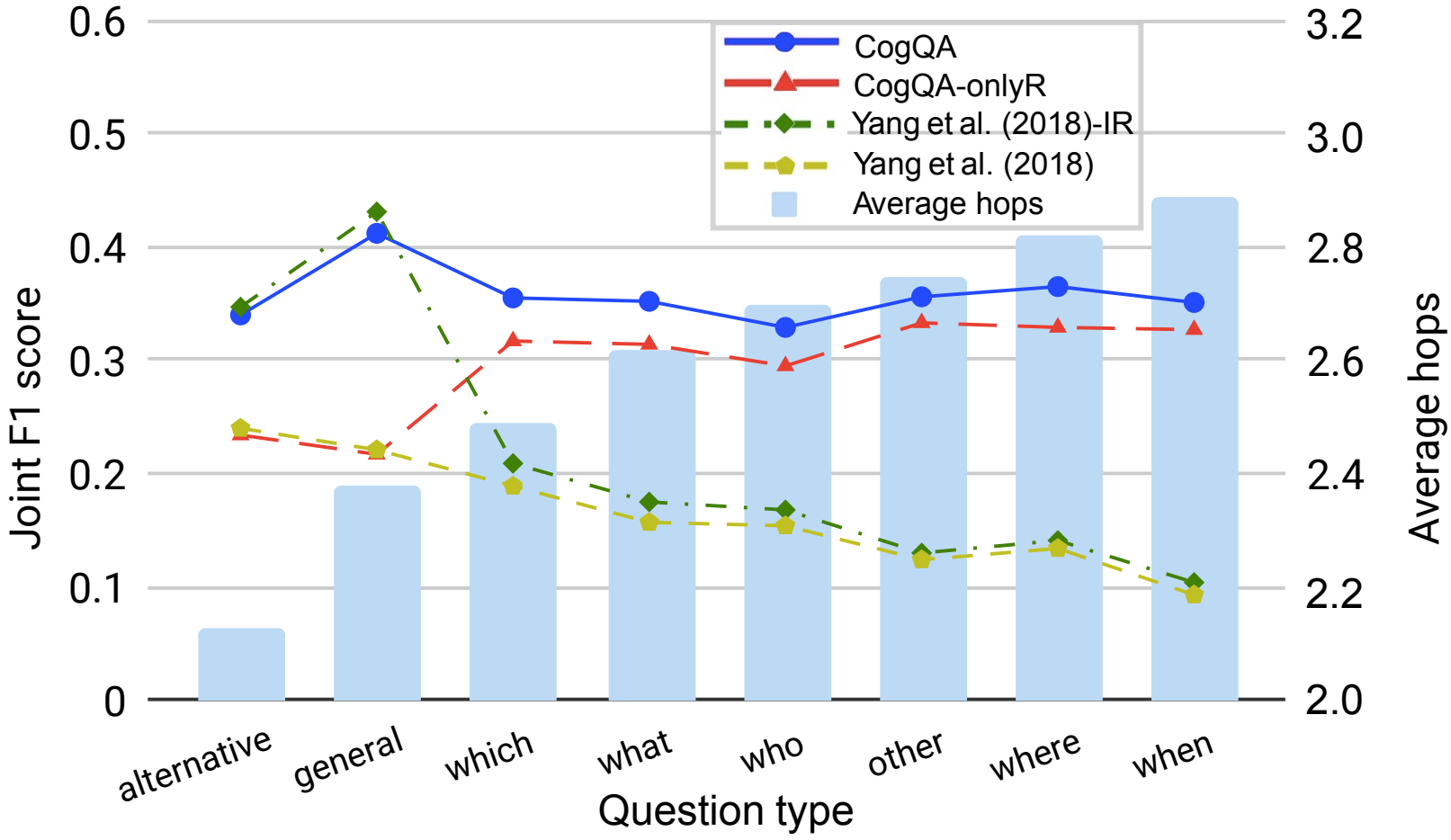
	Model	Ans				Sup				Joint			
		EM	F_1	Prec	Recall	EM	F_1	Prec	Recall	EM	F_1	Prec	Recall
Dev	Yang et al. (2018)	23.9	32.9	34.9	33.9	5.1	40.9	47.2	40.8	2.5	17.2	20.4	17.8
	Yang et al. (2018)-IR	24.6	34.0	35.7	34.8	10.9	49.3	52.5	52.1	5.2	21.1	22.7	23.2
	BERT	22.7	31.6	33.4	31.9	6.5	42.4	54.6	38.7	3.1	17.8	24.3	16.2
	CogQA-sys1	33.6	45.0	47.6	45.4	**23.7**	58.3	**67.3**	56.2	12.3	32.5	39.0	31.8
	CogQA-onlyR	34.6	46.2	48.8	46.7	14.7	48.2	56.4	47.7	8.3	29.9	36.2	30.1
	CogQA-onlyQ	30.7	40.4	42.9	40.7	23.4	49.9	56.5	48.5	**12.4**	30.1	35.2	29.9
	CogQA	**37.6**	**49.4**	**52.2**	**49.9**	23.1	**58.5**	64.3	**59.7**	12.2	**35.3**	**40.3**	**36.5**
Test	Yang et al. (2018)	24.0	32.9	-	-	3.86	37.7	-	-	1.9	16.2	-	-
	QFE	28.7	38.1	-	-	14.2	44.4	-	-	8.7	23.1	-	-
	DecompRC	30.0	40.7	-	-	N/A	N/A	-	-	N/A	N/A	-	-
	MultiQA	30.7	40.2	-	-	N/A	N/A	-	-	N/A	N/A	-	-
	GRN	27.3	36.5	-	-	12.2	48.8	-	-	7.4	23.6	-	-
	CogQA	**37.1**	**48.9**	-	-	**22.8**	**57.7**	-	-	**12.4**	**34.9**	-	-

Table 1: Results on HotpotQA (fullwiki setting). The test set is not public. The maintainer of HotpotQA only offers EM and F_1 for every submission. N/A means the model cannot find supporting facts.

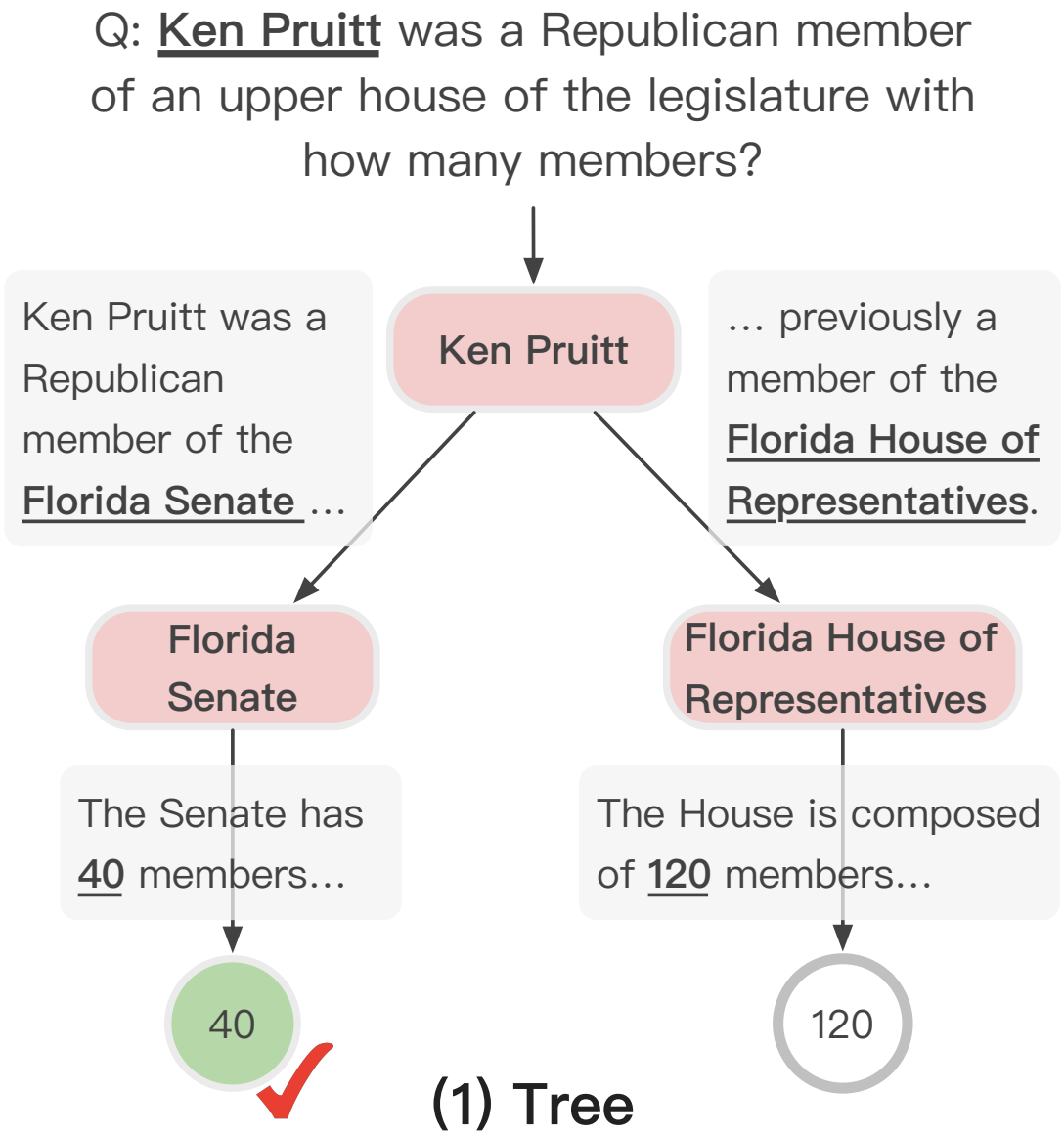
** Code available at https://github.com/THUDM/CogQA

Reasoning Power

CogQA Performs much better on question with more hops !

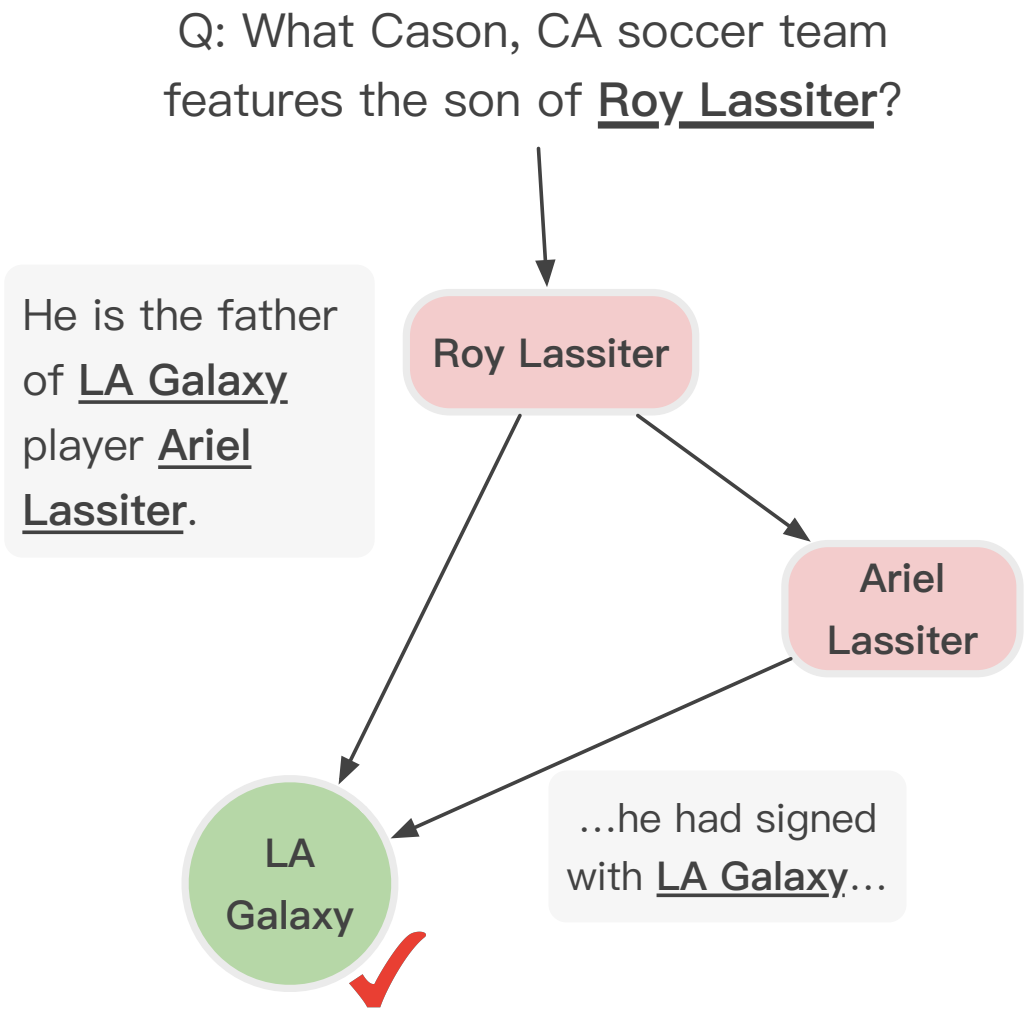


Case Study



- Tree-shape Cognitive Graph
- Users can verify the answer by comparing it with another possible reasoning chain.
- "*Upper House*" in the question is similar to "*Senate*" not "*House of Representative*"

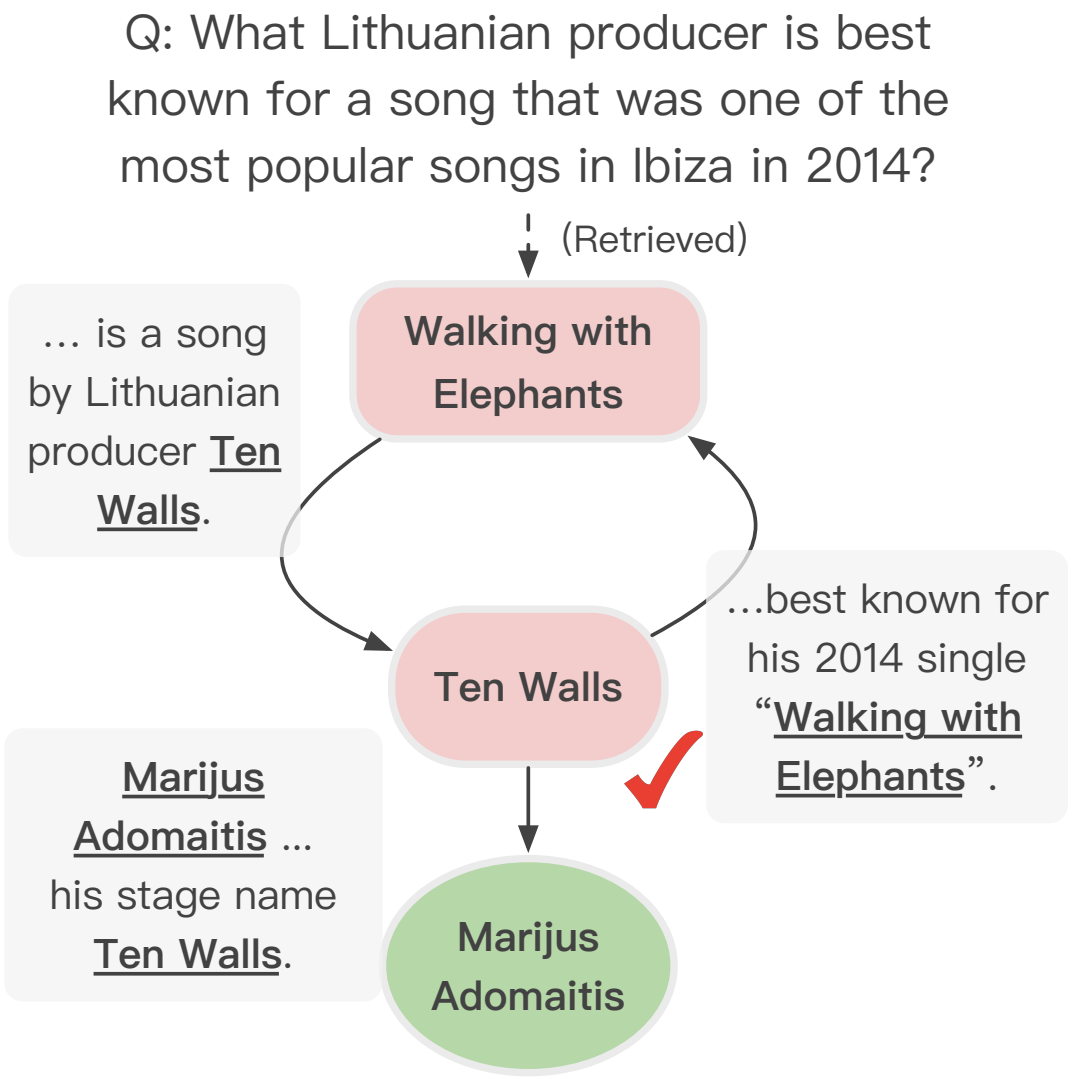
Case Study



(2) DAG

- DAG-shape Cognitive Graph
- Multiple supporting facts provides richer information, increasing the credibility of the answer.

Case Study



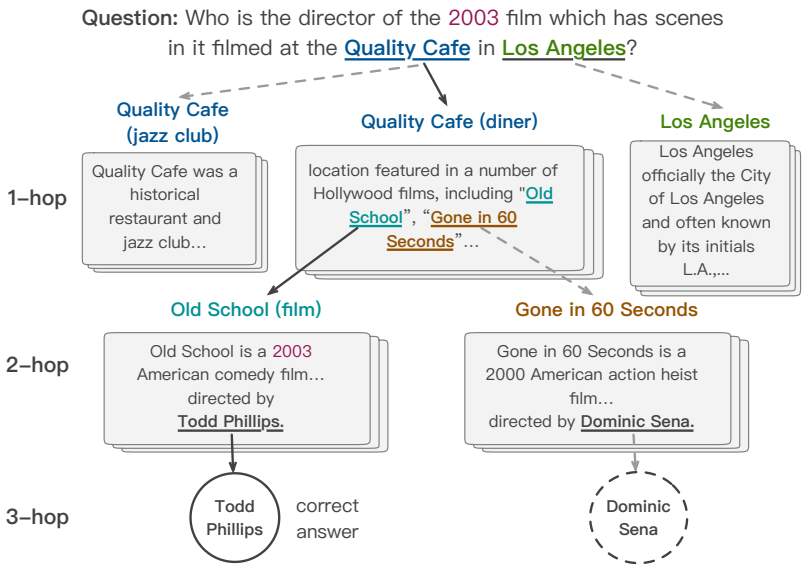
(3) Cyclic Graph

- CogQA gives the answer "Marijus Adomaitis" while the ground truth is "Ten Walls".
- By examining, Ten Walls is just the stage name of Marijus Adomaitis!
- Without cognitive graphs, black-box models cannot achieve it.

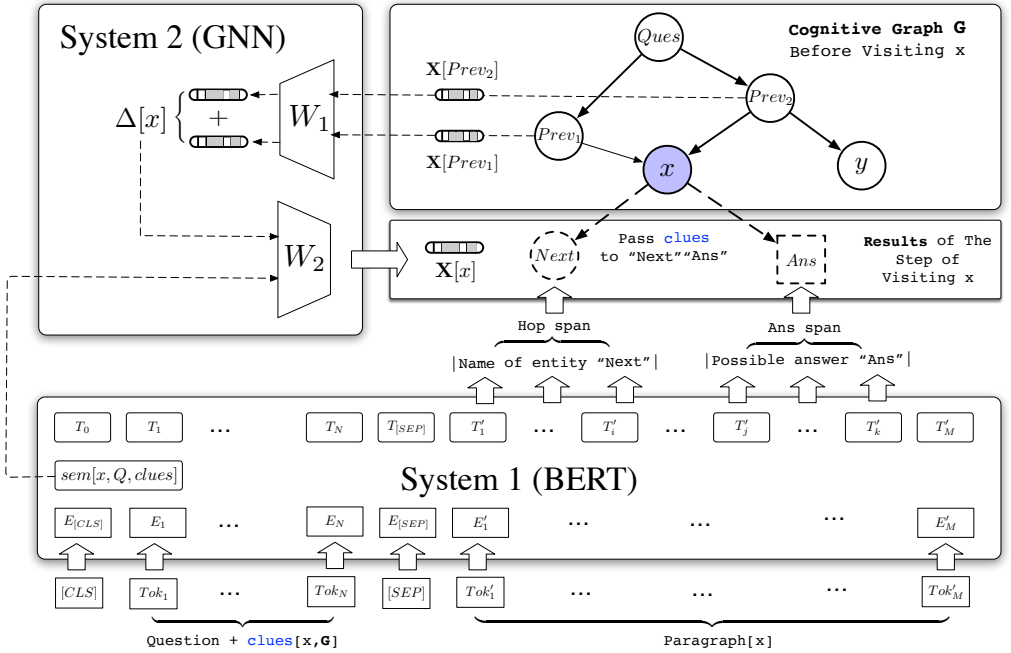
Summary

- Iterative Framework --> Myopic Retrieval
- Cognitive Graph --> Explainability
- Dual process theory --> System 2 Reasoning

Cognitive graph

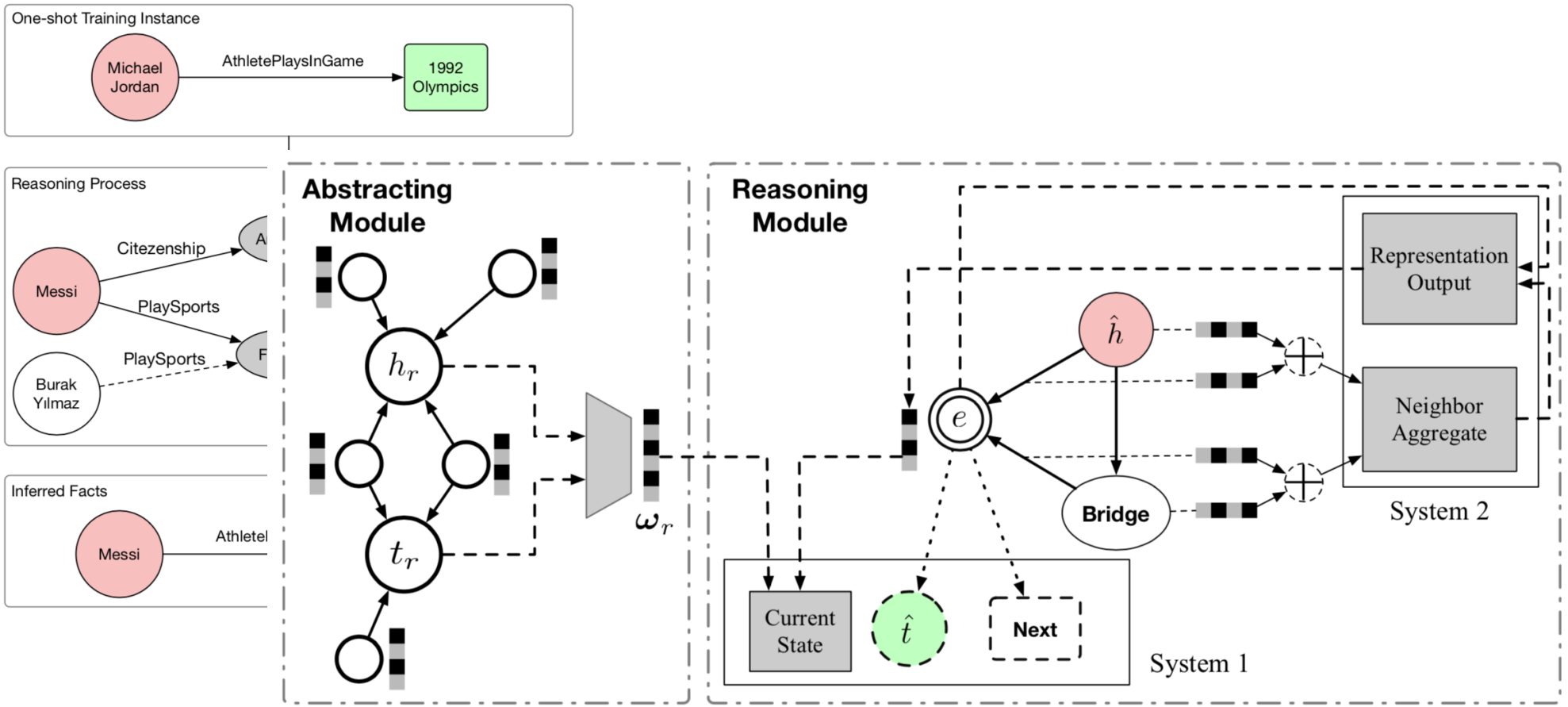


CogQA framework

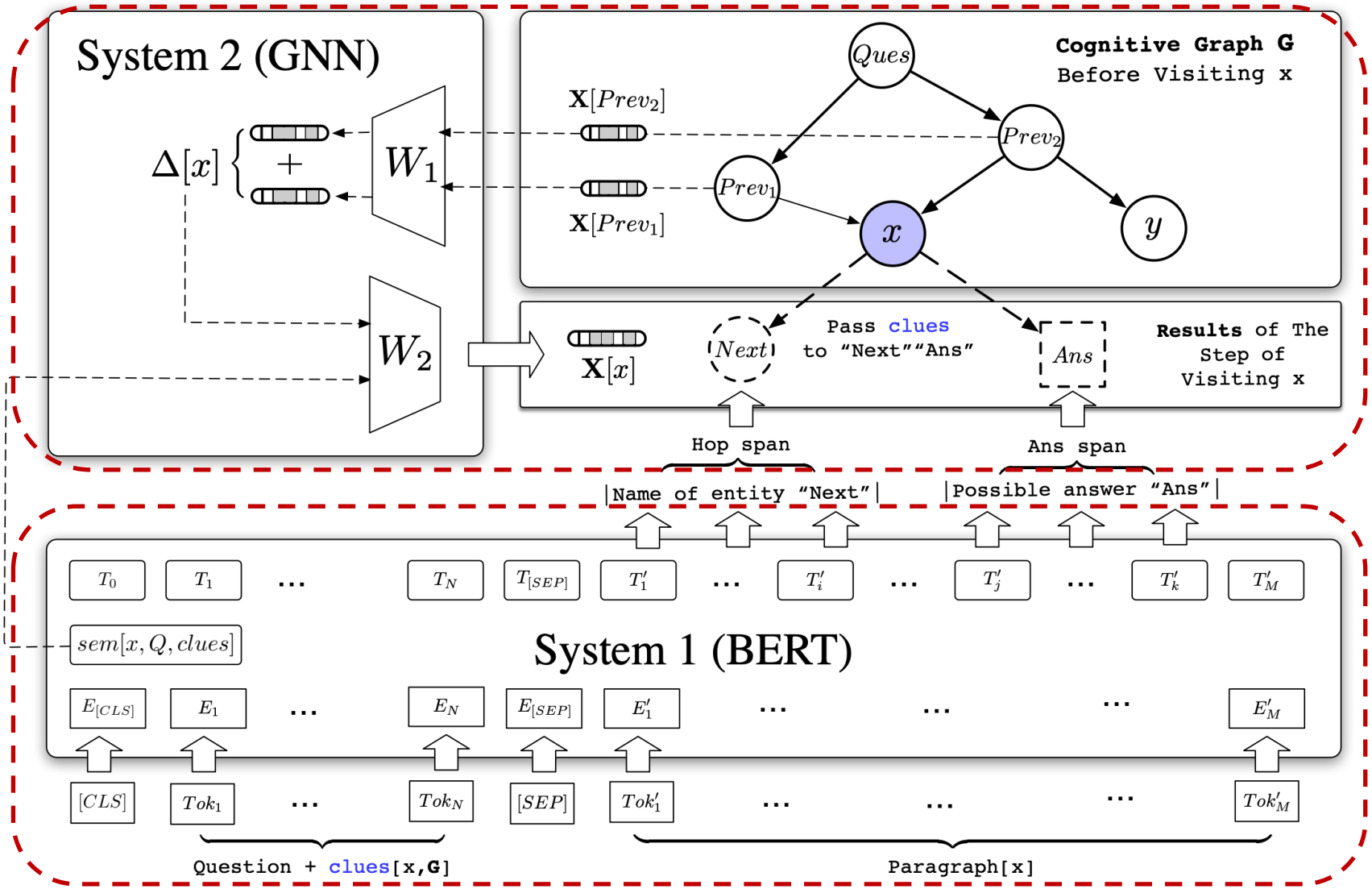


More Applications: KG completion

- Completing knowledge graph with cognitive graph

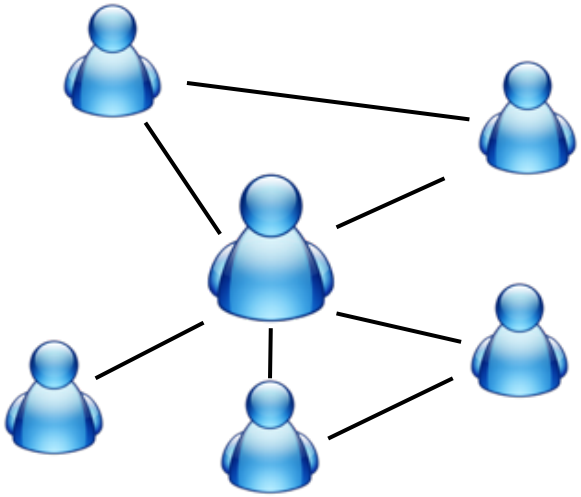


Revisiting Cognitive Graph

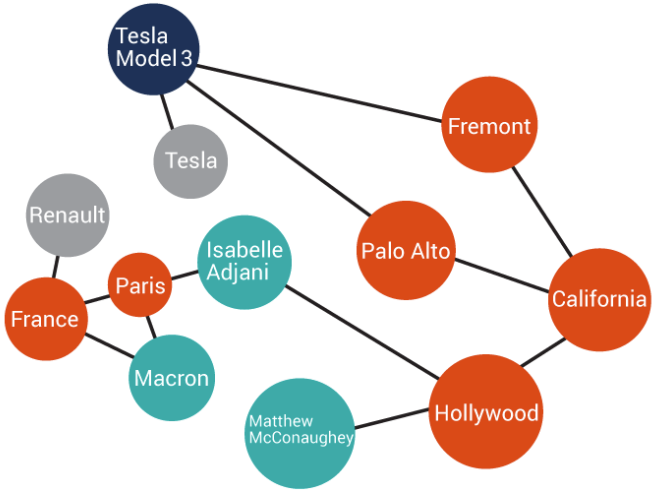


1. M. Ding, C. Zhou, Q. Chen, H. Yang, and J. Tang. Cognitive Graph for Multi-Hop Reading Comprehension at Scale. ACL'19.

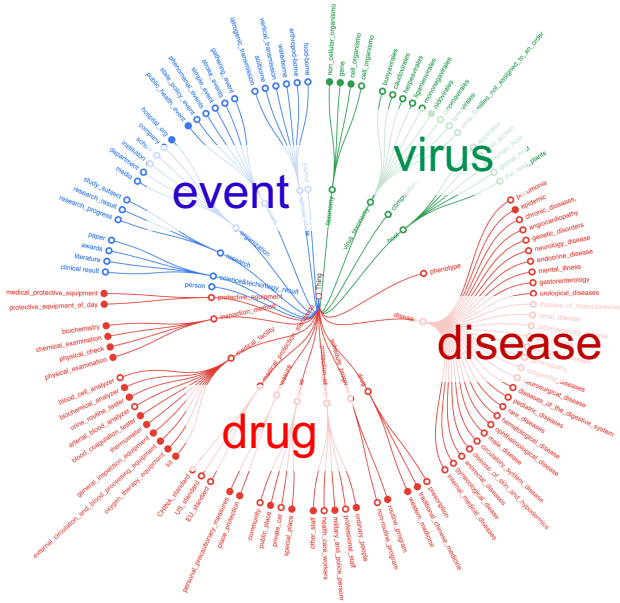
Networked Data



Social Network



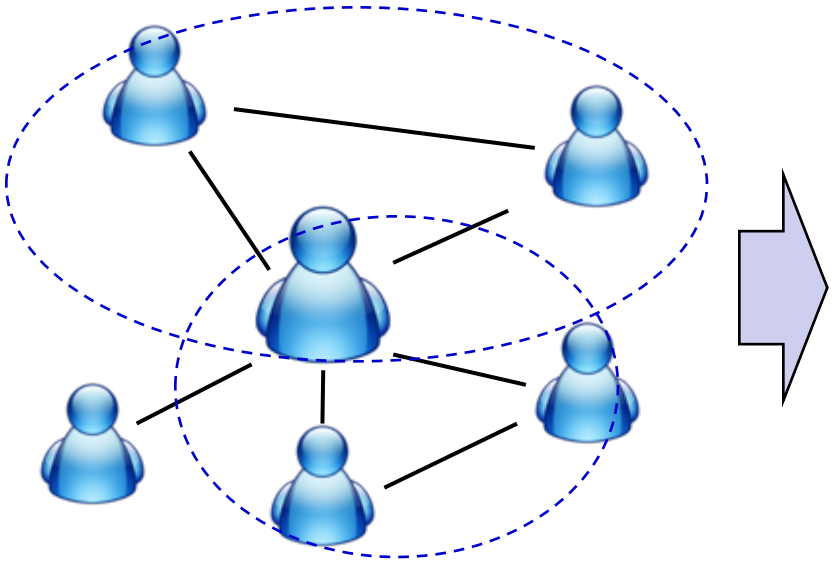
Knowledge Graph



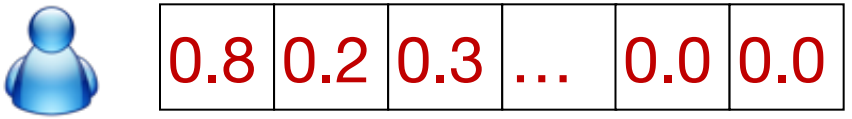
COVID Graph

Representation Learning on Networks

Representation Learning/
Graph Embedding

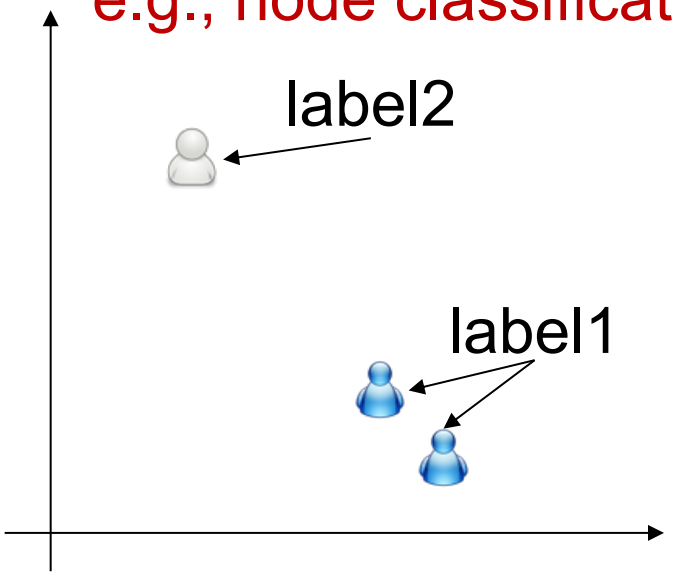


d-dimensional vector, $d << |V|$



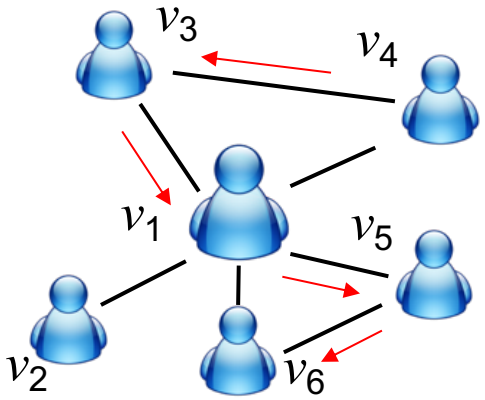
Users with the same label are located in closer

e.g., node classification



DeepWalk: Random Walk + Word2Vec

Random walk



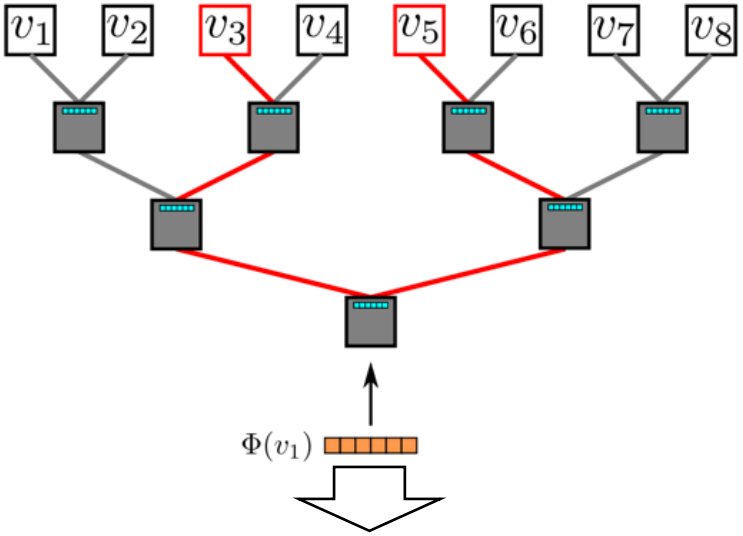
One example RW path

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

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

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

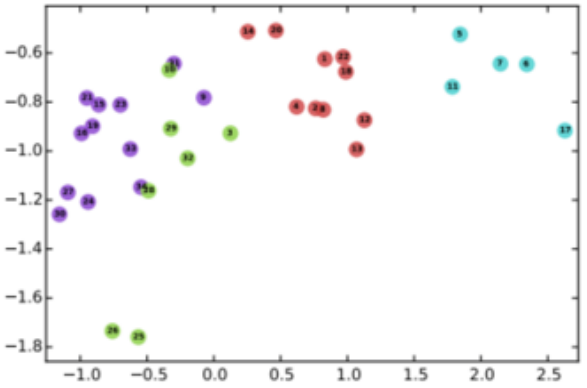
SkipGram with Hierarchical softmax



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

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.

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

$$\mathcal{L} = \sum_{v \in V} \sum_{c \in W_v} -\log(P(c|v))$$

$$p(c|v) = \frac{\exp(\boldsymbol{z}_v^\top \boldsymbol{z}_c)}{\sum_{u \in V} \exp(\boldsymbol{z}_v^\top \boldsymbol{z}_u)}$$

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 node classification
- DeepWalk (node 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

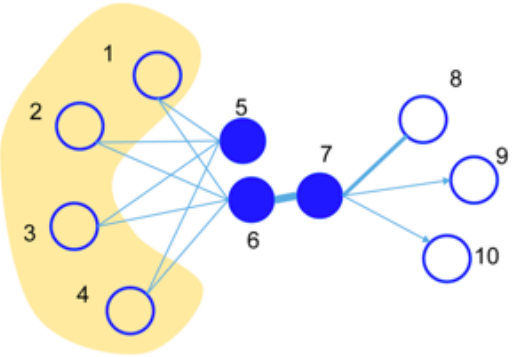
	% Labeled Nodes	1%	2%	3%	4%	5%	6%	7%	8%	9%	10%
Micro-F1(%)	DeepWalk	**37.95**	**39.28**	**40.08**	**40.78**	**41.32**	**41.72**	**42.12**	**42.48**	**42.78**	**43.05**
	SpectralClustering	—	—	—	—	—	—	—	—	—	—
	EdgeCluster	23.90	31.68	35.53	36.76	37.81	38.63	38.94	39.46	39.92	40.07
	Modularity	—	—	—	—	—	—	—	—	—	—
	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

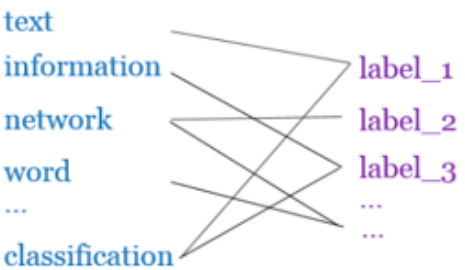
- Similar results on YouTube
- Spectral Clustering does not scale to large graphs

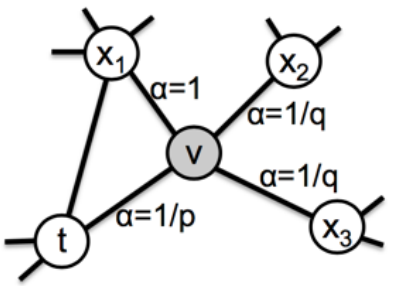
- DeepWalk utilizes fixed-length, unbiased random walks to generate context for each node, can we do better?

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.
- Metapath2vec[4]: meta-path-based random walks for heterogeneous networks.
- GATNE[5]: inductive learning for hetero-geneous networks.







1. J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei. 2015. Line: Large-scale information network embedding. *WWW'15*, 1067–1077.
2. J. Tang, M. Qu, and Q. Mei. 2015. Pte: Predictive text embedding through large-scale heterogeneous text networks. *KDD'15*, 1165–1174.
3. A. Grover and J. Leskovec. 2016. node2vec: Scalable feature learning for networks. *KDD'16*, 855–864.
4. Y. Dong, C. V. Nitesh and S. Ananthram. 2017. metapath2vec: Scalable Representation Learning for Heterogeneous Networks. KDD'14.
5. Y. Cen, X. Zou, J. Zhang, H. Yang, J. Zhou, and J. Tang. Representation Learning for Attributed Multiplex Heterogeneous Network. KDD'19.

LINE: First-order proximity

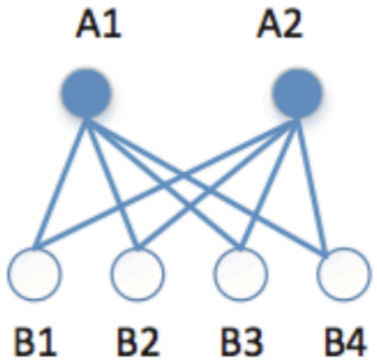
Definition

- The **first-order** proximity in a network is the **local** pairwise proximity between two vertices.
- For each pair of vertices linked by an edge (v_i, v_j), w_{ij} is the weight of the edge: indicates the first-order proximity between v_i and v_j.
- If no edge is observed between v_i and v_j, their **first-order** proximity is 0.

LINE: Second-order proximity

Definition

- The **second-order** proximity between a pair of vertices (v_i, v_j) in a network is the similarity between their neighborhood network structures.
- If no vertex is linked from/to both v_i and v_j, the **second-order** proximity between v_i and v_j is 0.



Model Optimization

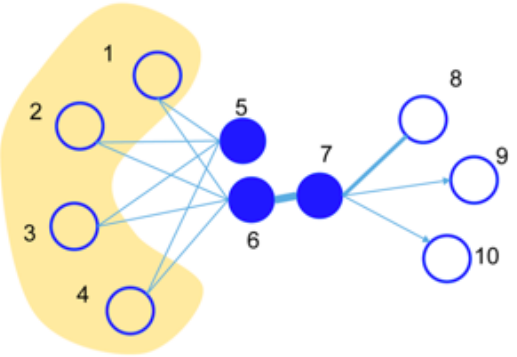
- Optimizing objective O_2 is computationally expensive.
- Adopt the approach of **negative sampling**. More specifically, it specifies the following objective function for each edge $e_{i,j}$,

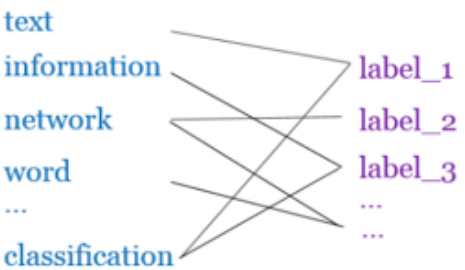
$$\underbrace{\log\sigma(\vec{u}_j'^T \cdot \vec{u}_i)}_{\text{the observed edges}} + \underbrace{\sum_{i=1}^{K} E_{v_n} \sim P_n(v)[\log\sigma(-\vec{u}_n'^T \cdot \vec{u}_i)]}_{\text{the negative edges drawn from the noise distribution}}$$

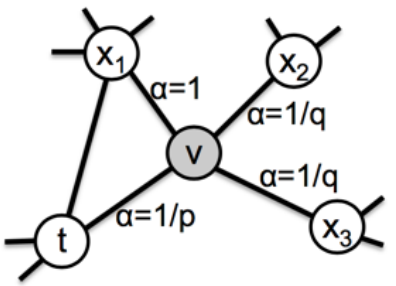
- We can use **asynchronous stochastic gradient algorithm (ASGD)** for optimizing above Eqn.

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.
- Metapath2vec[4]: meta-path-based random walks for heterogeneous networks.
- GATNE[5]: inductive learning for hetero-geneous networks.



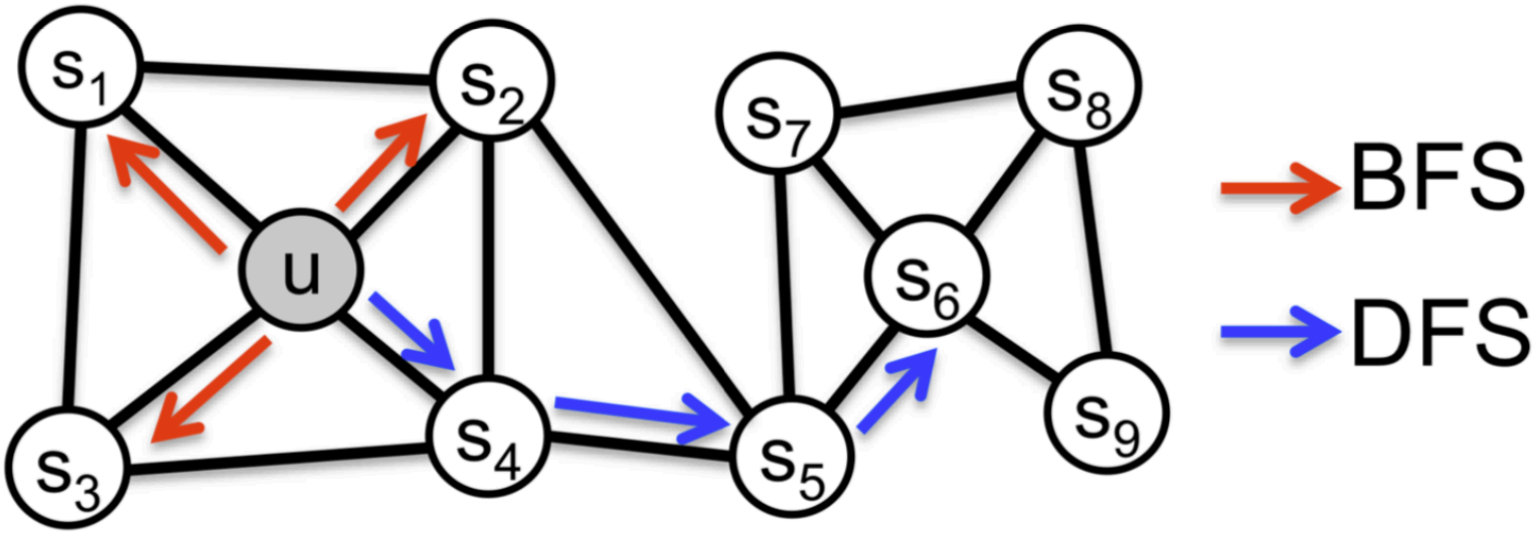




1. J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei. 2015. Line: Large-scale information network embedding. *WWW'15*, 1067–1077.
2. J. Tang, M. Qu, and Q. Mei. 2015. Pte: Predictive text embedding through large-scale heterogeneous text networks. *KDD'15*, 1165–1174.
3. A. Grover and J. Leskovec. 2016. node2vec: Scalable feature learning for networks. *KDD'16*, 855–864.
4. Y. Dong, C. V. Nitesh and S. Ananthram. 2017. metapath2vec: Scalable Representation Learning for Heterogeneous Networks. KDD'14.
5. Y. Cen, X. Zou, J. Zhang, H. Yang, J. Zhou, and J. Tang. Representation Learning for Attributed Multiplex Heterogeneous Network. KDD'19.

node2vec: Biased Walks

- Use biased random walks to trade off local and global views of the network

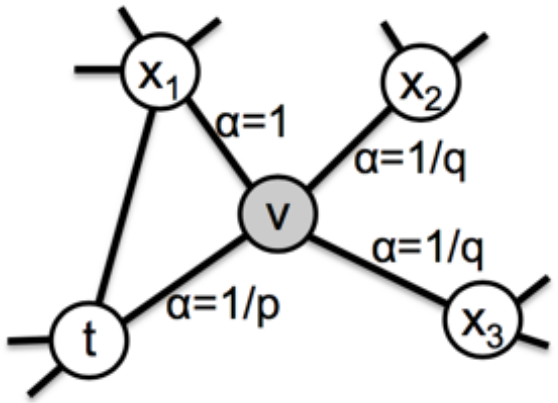


- Biased walks is a special case of random walk, thus node2vec is a special case of DeepWalk

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

node2vec

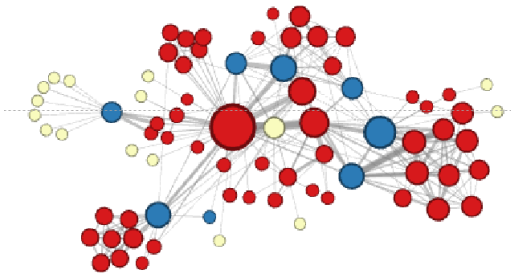
- Biased random walk R that given a node v generates random walk neighborhood $N_{rw}(v)$
- Return parameter p:
  - Return back to the previous node
- In-out parameter q:
  - Moving outwards (DFS) vs. inwards (BFS)



node2vec

- Biased random walk R that given a node v generates random walk neighborhood $N_{rw}(v)$
- Return parameter p:
  - Return back to the previous node
- In-out parameter q:
  - Moving outwards (DFS) vs. inwards (BFS)

Interactions of characters in a novel:



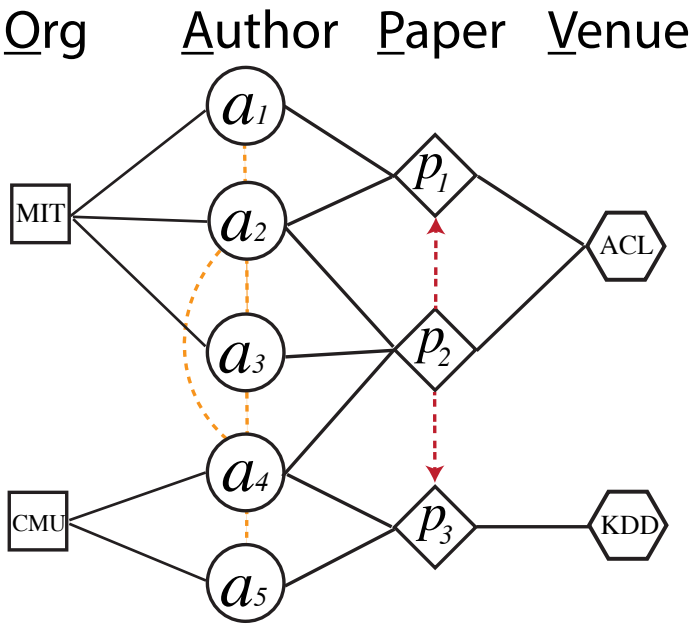
p=1, q=2
Microscopic view of the network neighbourhood

p=1, q=0.5
Macroscopic view of the network neighbourhood

Picture snipped from Leskovec

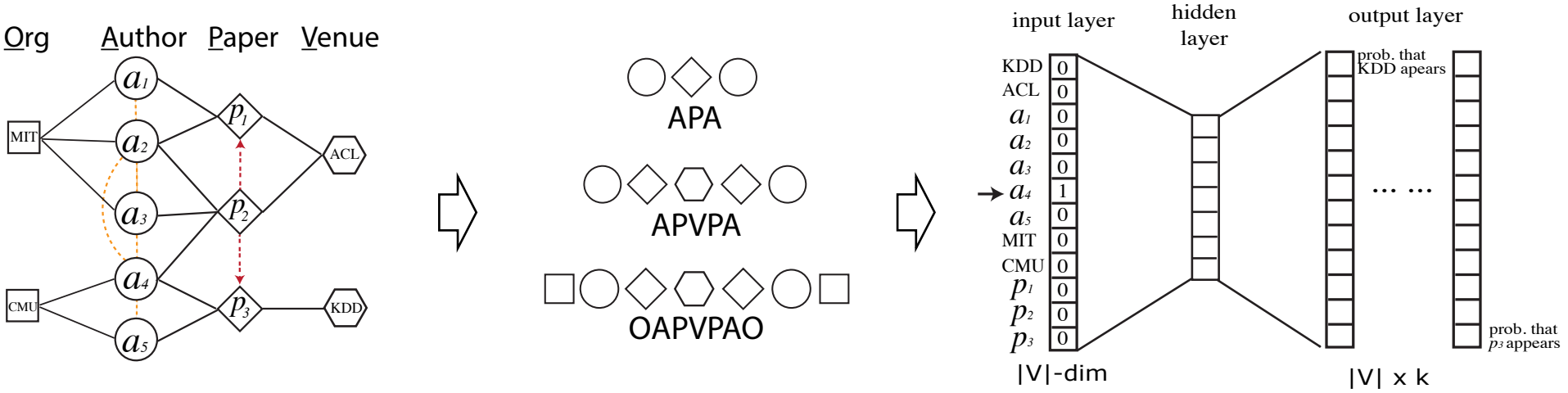
Metapath2vec: Heterogeneous Random Walk

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



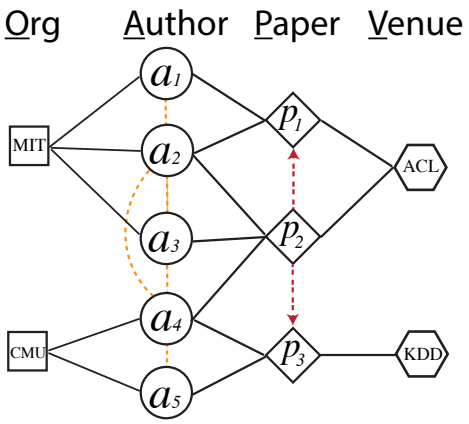
- How do we random walk over heterogeneous networks?
- How do we apply skip-gram over different types of nodes?

Metapath2vec: Heterogeneous Random Walk



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

Metapath2vec: Heterogeneous Random Walk



- Given a meta-path scheme

$$\mathcal{P}: V_1 \xrightarrow{R_1} V_2 \xrightarrow{R_2} \cdots V_t \xrightarrow{R_t} V_{t+1} \cdots \xrightarrow{R_{l-1}} V_l$$

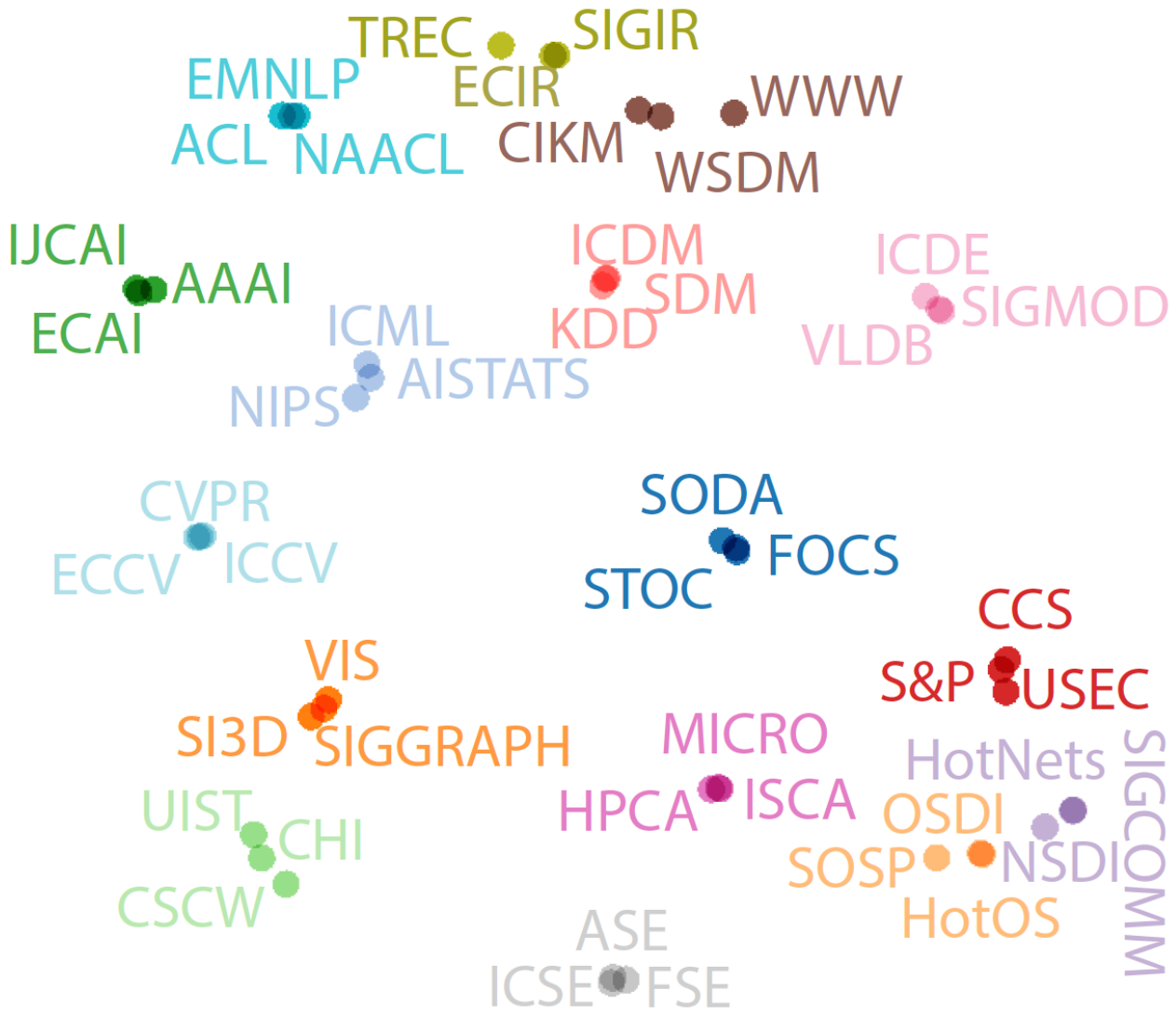
- The transition probability at step *i* is defined as

$$p(v^{i+1}|v_t^i, \mathcal{P}) = \begin{cases} \frac{1}{|N_{t+1}(v_t^i)|} & (v^{i+1}, v_t^i) \in E, \phi(v^{i+1}) = t+1 \\ 0 & (v^{i+1}, v_t^i) \in E, \phi(v^{i+1}) \neq t+1 \\ 0 & (v^{i+1}, v_t^i) \notin E \end{cases}$$

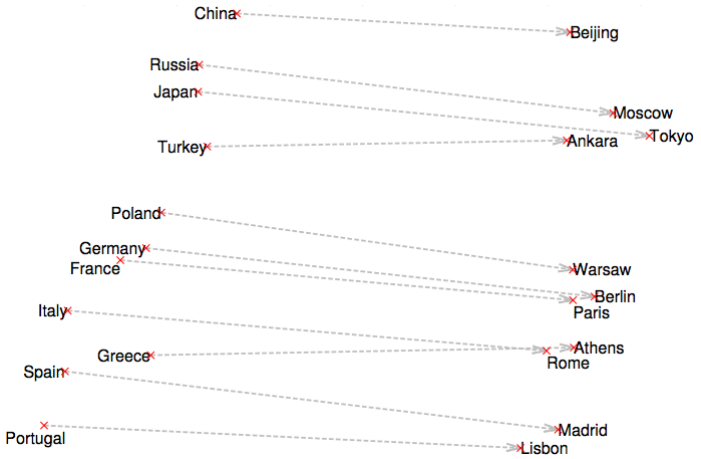
- Recursive guidance for random walkers, i.e.,

$$p(v^{i+1}|v_t^i) = p(v^{i+1}|v_1^i), \text{ if } t = l$$

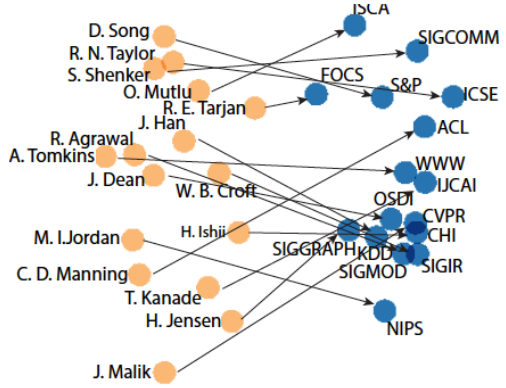
Application 2: Node Clustering



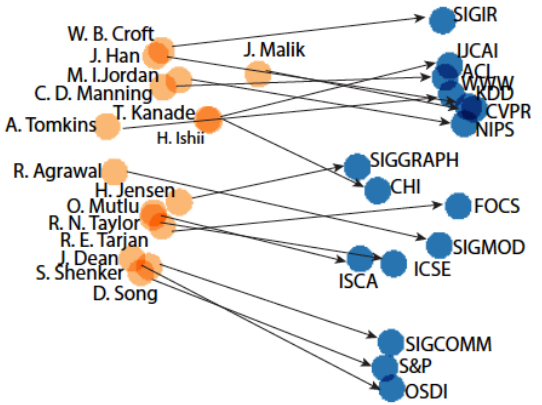
Visualization



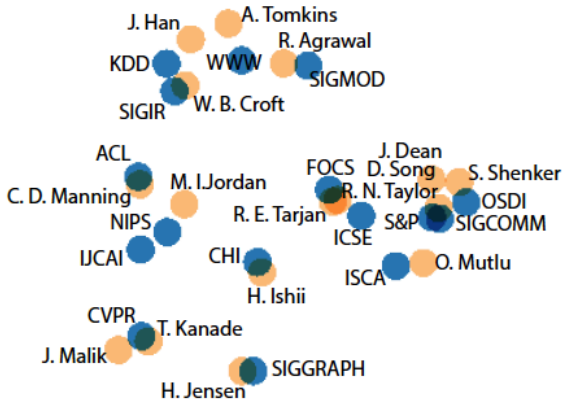
word2vec [Mikolov, 2013]



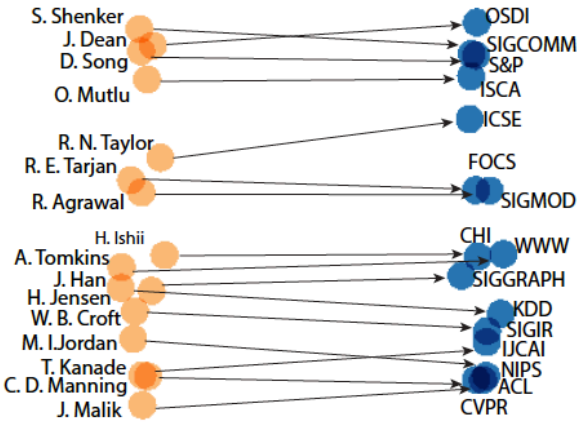
(a) DeepWalk/node2vec



(b) PTE



(c) *metapath2vec*

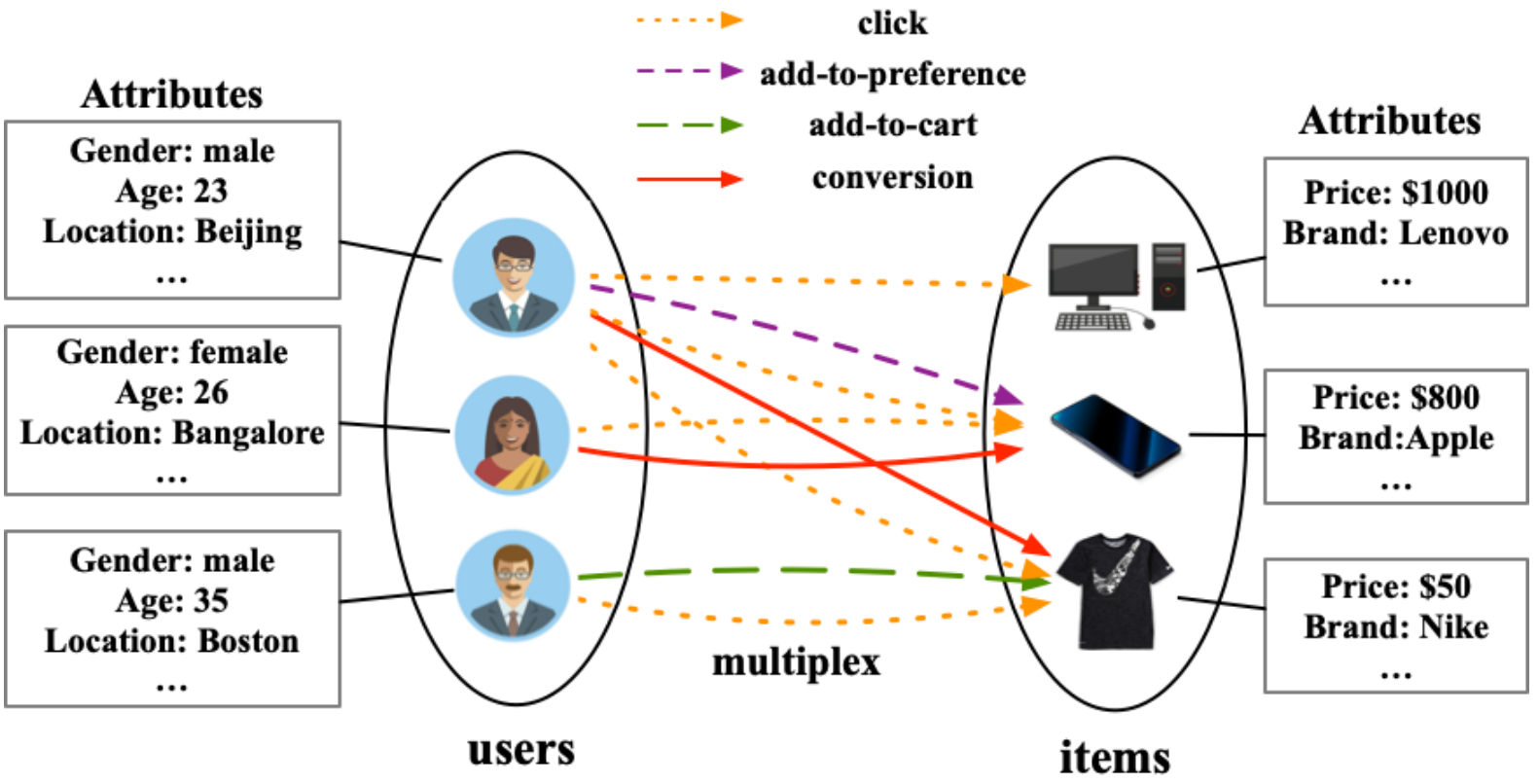


(d) *metapath2vec++*

http://projector.tensorflow.org/

GATNE: Attributed Multiplex Heterogeneous Network Embedding

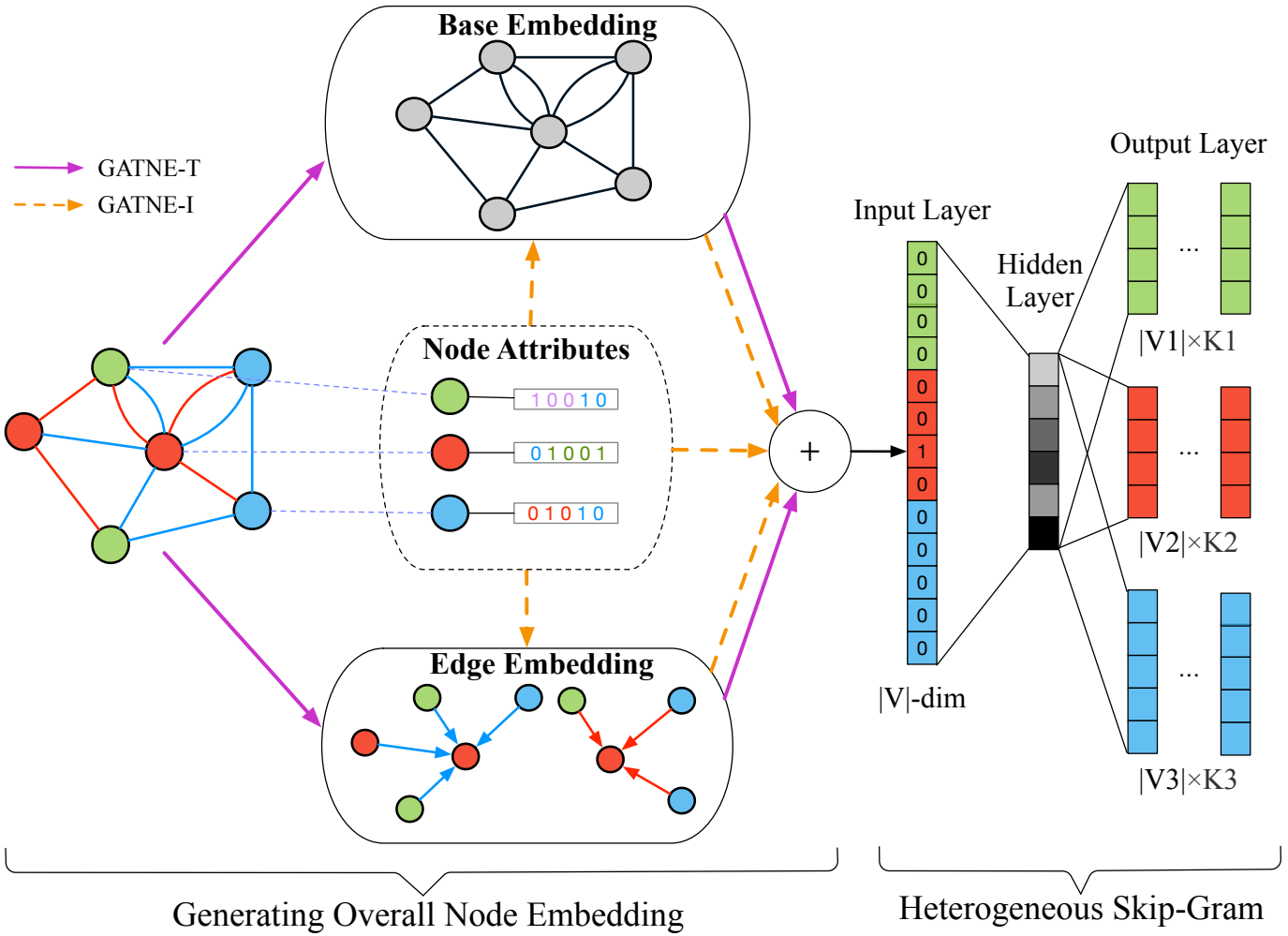
- Recommend items to users by considering *Attributed Multiplex Heterogeneous Networks* (AMHEN)



Different Types of Network Embedding

Network Type	Method	Heterogeneity		Attribute
		Node Type	Edge Type	
Homogeneous Network (HON)	DeepWalk [27] LINE [35] node2vec [10] NetMF [29] NetSMF [28]	Single	Single	/
Attributed Homogeneous Network (AHON)	TADW [41] LANE [16] AANE [15] SNE [20] DANE [9] ANRL [44]	Single	Single	Attributed
Heterogeneous Network (HEN)	PTE [34] metapath2vec [7] HERec [31]	Multi	Single	/
Attributed HEN (AHEN)	HNE [3]	Multi	Single	Attributed
Multiplex Heterogeneous Network (MHEN)	PMNE [22] MVE [30] MNE [43] mvn2vec [32]	Single	Multi	/
	GATNE-T	Multi	Multi	
Attributed MHEN (AMHEN)	*GATNE*-I	Multi	Multi	Attributed

Multiplex Heterogeneous Graph Embedding



1. Y. Cen, X. Zou, J. Zhang, H. Yang, J. Zhou and J. Tang. Representation Learning for Attributed Multiplex Heterogeneous Network. KDD'19.

Recommendation Results

- Data
  - Small: Amazon, YouTube, Twitter w/ 10*K* nodes
  - Large: Alibaba w/ 40M nodes and 0.5B edges

	Amazon			YouTube			Twitter			Alibaba-S		
	ROC-AUC	PR-AUC	F1	ROC-AUC	PR-AUC	F1	ROC-AUC	PR-AUC	F1	ROC-AUC	PR-AUC	F1
DeepWalk	94.20	94.03	87.38	71.11	70.04	65.52	69.42	72.58	62.68	59.39	60.62	56.10
node2vec	94.47	94.30	87.88	71.21	70.32	65.36	69.90	73.04	63.12	62.26	63.40	58.49
LINE	81.45	74.97	76.35	64.24	63.25	62.35	62.29	60.88	58.18	53.97	54.65	52.85
metapath2vec	94.15	94.01	87.48	70.98	70.02	65.34	69.35	72.61	62.70	60.94	61.40	58.25
ANRL	71.68	70.30	67.72	75.93	73.21	70.65	70.04	67.16	64.69	58.17	55.94	56.22
PMNE(n)	95.59	95.48	89.37	65.06	63.59	60.85	69.48	72.66	62.88	62.23	63.35	58.74
PMNE(r)	88.38	88.56	79.67	70.61	69.82	65.39	62.91	67.85	56.13	55.29	57.49	53.65
PMNE(c)	93.55	93.46	86.42	68.63	68.22	63.54	67.04	70.23	60.84	51.57	51.78	51.44
MVE	92.98	93.05	87.80	70.39	70.10	65.10	72.62	73.47	67.04	60.24	60.51	57.08
MNE	90.28	91.74	83.25	82.30	82.18	75.03	91.37	91.65	84.32	62.79	63.82	58.74
GATNE-T	**97.44**	**97.05**	**92.87**	**84.61**	81.93	**76.83**	**92.30**	91.77	**84.96**	66.71	67.55	62.48
GATNE-I	96.25	94.77	91.36	84.47	**82.32**	**76.83**	92.04	**91.95**	84.38	**70.87**	**71.65**	**65.54**

GATNE outperforms all sorts of baselines in the various datasets.

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

Alibaba Offline A/B Tests

- GATNE-I is deployed on Alibaba's distributed cloud platform for its recommendation system. The training dataset has about 100 million users and 10 million items with 10 billion interactions between them.

- Under the framework of A/B tests, an offline test is conducted on GATNE-I, MNE and DeepWalk. The experimental goal is to maximize Hit-Rate. The results demonstrate that GATNE-I improves Hit-Rate by **3.26% and 24.26%** compared to MNE and DeepWalk respectively.

Questions

- What are the fundamentals underlying the different models?

or

- Can we unify the different graph embedding approaches?

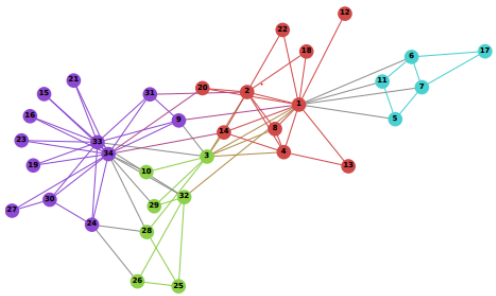
Unifying DeepWalk, LINE, PTE, and node2vec into Matrix Forms

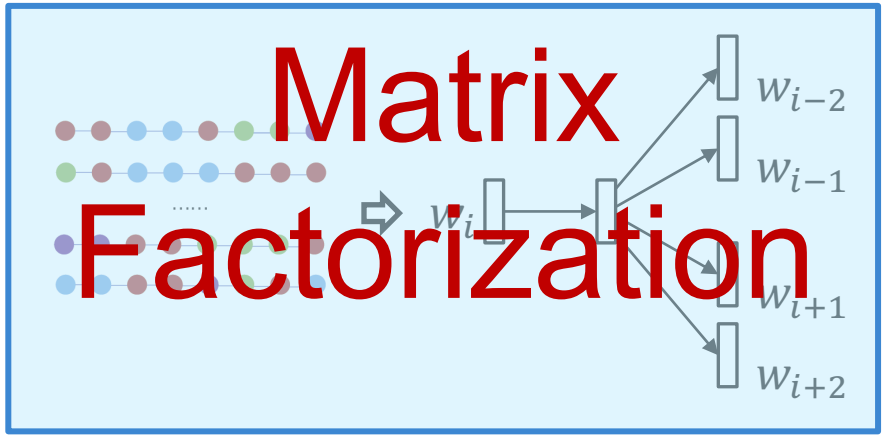
Algorithm	Closed Matrix Form
DeepWalk	$\log\left(\mathrm{vol}(G)\left(\frac{1}{T}\sum_{r=1}^{T}(\boldsymbol{D}^{-1}\boldsymbol{A})^r\right)\boldsymbol{D}^{-1}\right)-\log b$
LINE	$\log\left(\mathrm{vol}(G)\boldsymbol{D}^{-1}\boldsymbol{A}\boldsymbol{D}^{-1}\right)-\log b$
PTE	$\log\left(\begin{bmatrix}\alpha\,\mathrm{vol}(G_{\mathrm{ww}})(\boldsymbol{D}_{\mathrm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

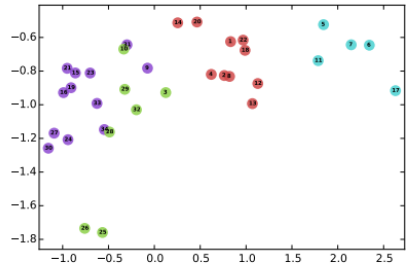
$\boldsymbol{A}$: $\boldsymbol{A}\in\mathbb{R}_+^{|V|\times|V|}$ is G's adjacency matrix with $\boldsymbol{A}_{i,j}$ as the edge weight between vertices i and j;
$\boldsymbol{D}_{\mathrm{col}}$: $\boldsymbol{D}_{\mathrm{col}}=\mathrm{diag}(\boldsymbol{A}^\top\boldsymbol{e})$ is the diagonal matrix with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

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

DeepWalk is factorizing a matrix







DeepWalk is asymptotically and implicitly factorizing

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

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

$\boldsymbol{A}$ Adjacency matrix

$\boldsymbol{D}$ Degree matrix

b: #negative samples

T: context window size

LINE

- Objective of LINE:

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

- Align it with the Objective of SGNS:

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

- LINE is actually factorizing

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

- Recall DeepWalk's matrix form:

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

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

PTE

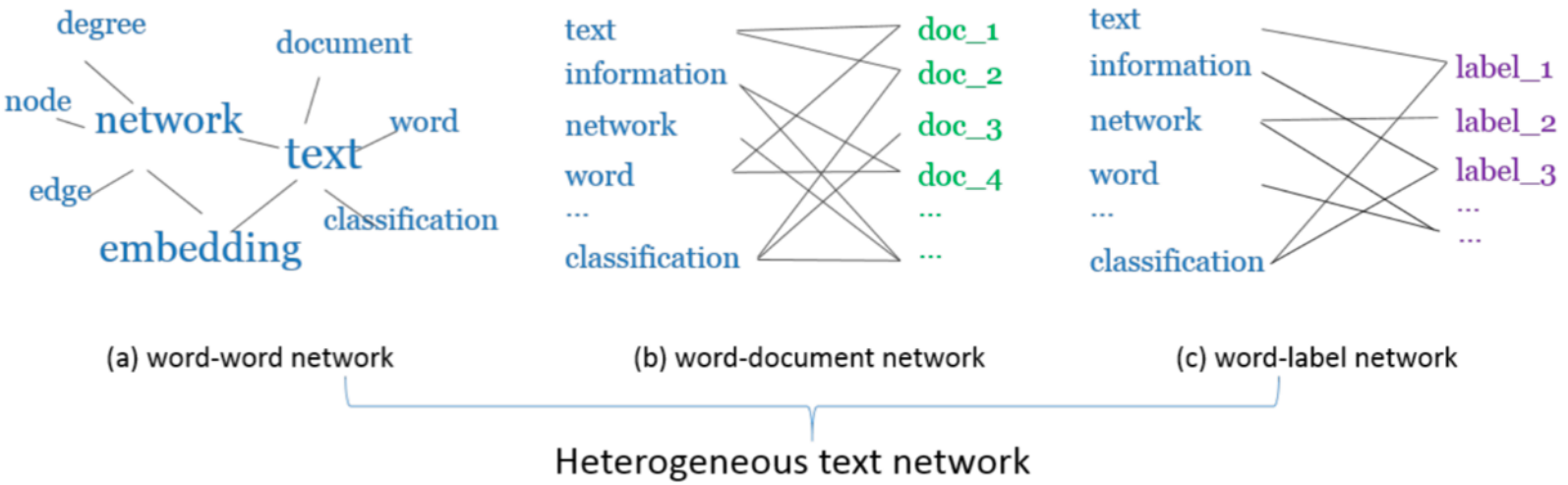


Figure 2: Heterogeneous Text Network.

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

Understanding node2vec

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

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

Stationary Distribution

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

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

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

Understanding node2vec

Theorem

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

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

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

Can we directly factorize the derived matrices for learning embeddings?

NetMF

- DeepWalk is implicitly factorizing

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

- NetMF is explicitly factorizing

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

NetMF

- DeepWalk is implicitly factorizing

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

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

- NetMF is explicitly factorizing

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

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

NetMF

1 Eigen-decomposition $\boldsymbol{D}^{-1/2}\boldsymbol{A}\boldsymbol{D}^{-1/2} \approx \boldsymbol{U}_h\boldsymbol{\Lambda}_h\boldsymbol{U}_h^\top$;

2 Approximate $\boldsymbol{M}$ with

$$\hat{\boldsymbol{M}} = \frac{\mathrm{vol}(G)}{b}\boldsymbol{D}^{-1/2}\boldsymbol{U}_h\left(\frac{1}{T}\sum_{r=1}^{T}\boldsymbol{\Lambda}_h^r\right)\boldsymbol{U}_h^\top\boldsymbol{D}^{-1/2};$$

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

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

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

Approximate $\boldsymbol{D}^{-1/2}\boldsymbol{A}\boldsymbol{D}^{-1/2}$ with its top-h eigenpairs $\boldsymbol{U}_h\boldsymbol{\Lambda}_h\boldsymbol{U}_h^\top$

The Arnoldi algorithm [1] for significant time reduction

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

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

- According to Frobenius norm's property

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

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

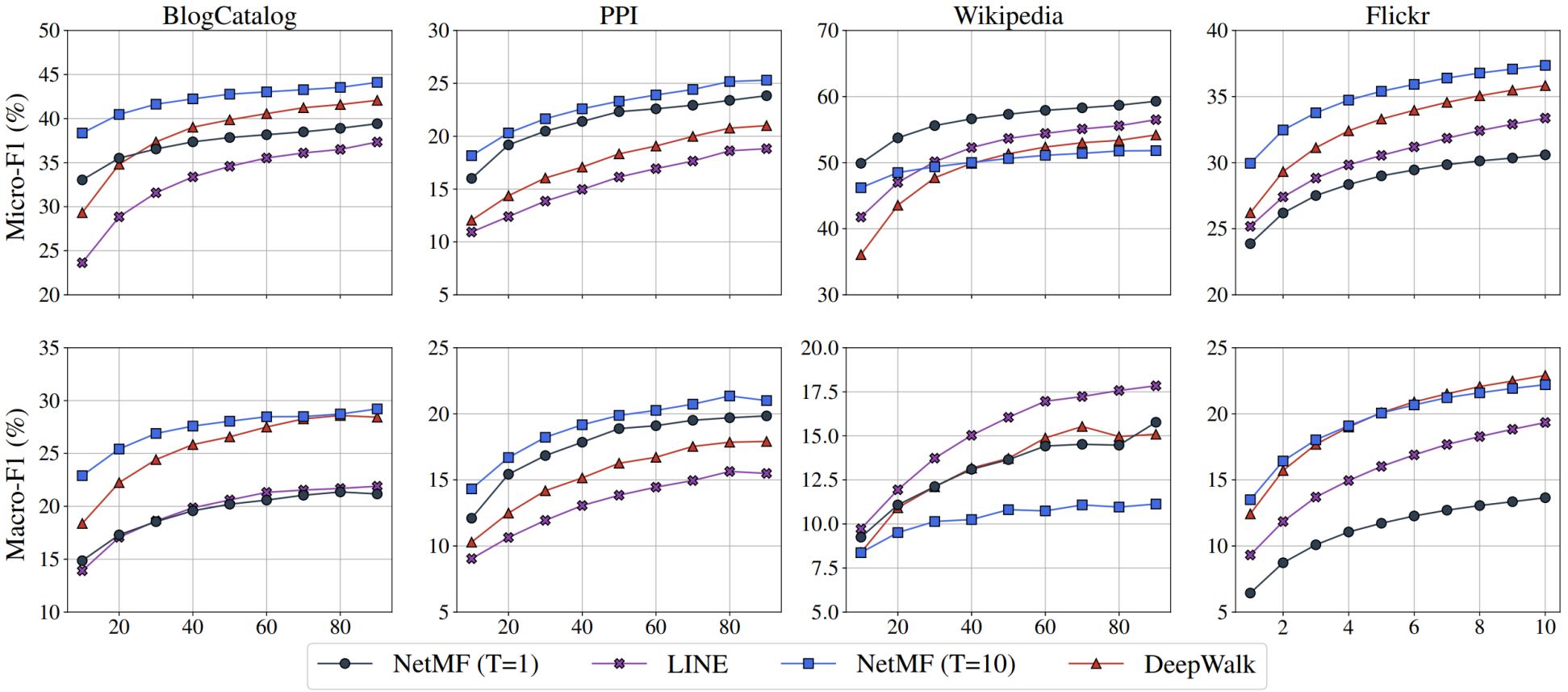
- and because $M'_{i,j}=\max(M_{i,j}, 1)>=1$, we have

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

- Also because the property of NGL,

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

Experimental Results



Predictive performance on varying the ratio of training data;

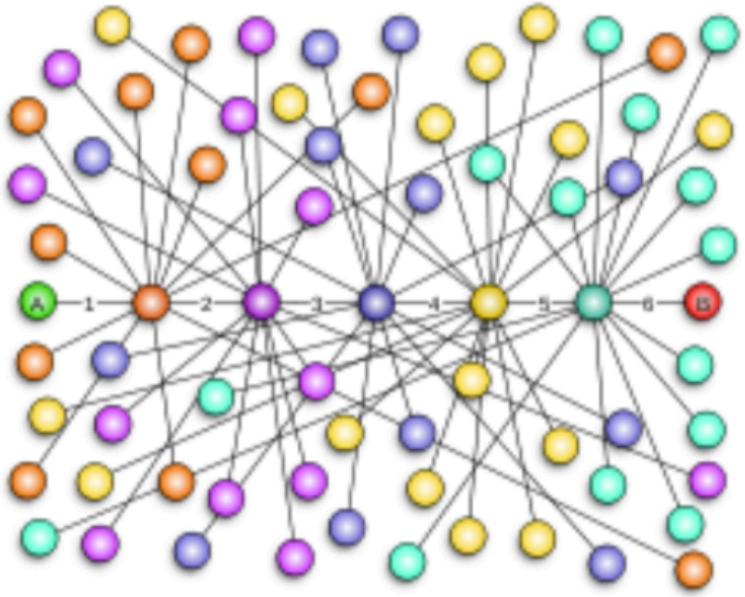
The *x*-axis represents the ratio of labeled data (%)

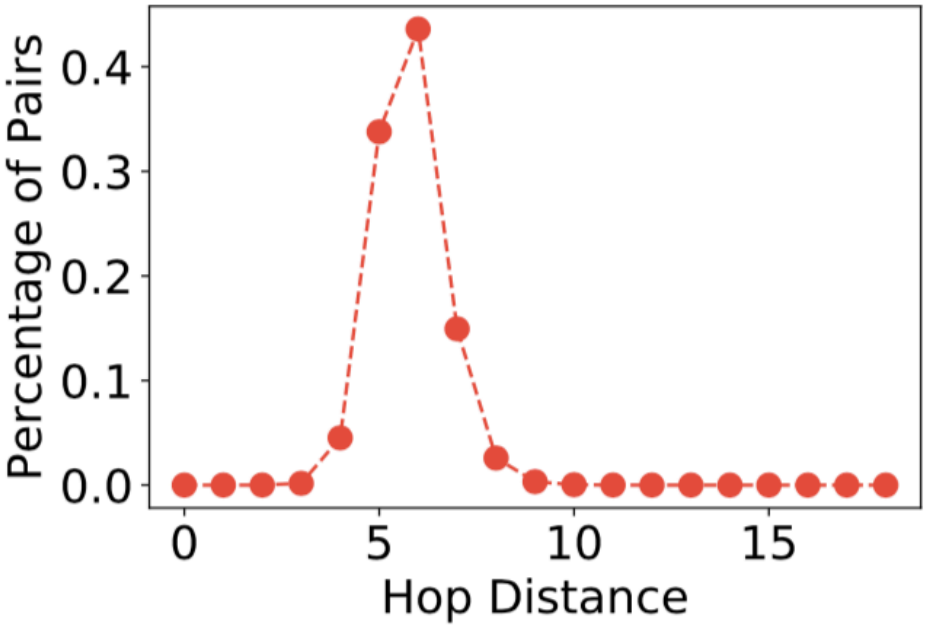
Network Embedding as Matrix Factorization

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

1. Qiu et al. Network embedding as matrix factorization: unifying deepwalk, line, pte, and node2vec. In *WSDM'18.* **The most cited paper in WSDM'18 as of Aug 2018**

Challenge in NetMF





Small world

Academic graph

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

Sparsify $\boldsymbol{S}$

For random-walk matrix polynomial $\boldsymbol{L} = \boldsymbol{D} - \sum_{r=1}^{T} \alpha_r \boldsymbol{D} \left(\boldsymbol{D}^{-1} \boldsymbol{A} \right)^r$

where $\sum_{r=1}^{T} \alpha_r = 1$ and α_r non-negative

One can construct a $(1+\epsilon)$-spectral sparsifier $\tilde{\boldsymbol{L}}$ with $O(n \log n \epsilon^{-2})$ non-zeros

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

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

1. D. Cheng, Y. Cheng, Y. Liu, R. Peng, and S.H. Teng, Efficient Sampling for Gaussian Graphical Models via Spectral Sparsification, COLT 2015.
2. D. Cheng, Y. Cheng, Y. Liu, R. Peng, and S.H. Teng. Spectral sparsification of random-walk matrix polynomials. arXiv:1502.03496.

Sparsify $\boldsymbol{S}$

For random-walk matrix polynomial $\boldsymbol{L} = \boldsymbol{D} - \sum_{r=1}^{T} \alpha_r \boldsymbol{D} \left(\boldsymbol{D}^{-1}\boldsymbol{A}\right)^r$

where $\sum_{r=1}^{T} \alpha_r = 1$ and α_r non-negative

One can construct a $(1+\epsilon)$-spectral sparsifier $\tilde{\boldsymbol{L}}$ with $O(n \log n \epsilon^{-2})$ non-zeros

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

Suppose $G = (V, E, \boldsymbol{A})$ and $\widetilde{G} = (V, \widetilde{E}, \widetilde{\boldsymbol{A}})$ are two weighted undirected networks. Let $\boldsymbol{L} = \boldsymbol{D}_G - \boldsymbol{A}$ and $\widetilde{\boldsymbol{L}} = \boldsymbol{D}_{\widetilde{G}} - \widetilde{\boldsymbol{A}}$ be their Laplacian matrices, respectively. We define G and $\widetilde{G}$ are $(1+\epsilon)$-spectrally similar if

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

1. D. Cheng, Y. Cheng, Y. Liu, R. Peng, and S.H. Teng, Efficient Sampling for Gaussian Graphical Models via Spectral Sparsification, COLT 2015.
2. D. Cheng, Y. Cheng, Y. Liu, R. Peng, and S.H. Teng. Spectral sparsification of random-walk matrix polynomials. arXiv:1502.03496.

NetSMF --- Sparse

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

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

- Factorize the constructed matrix

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

Results

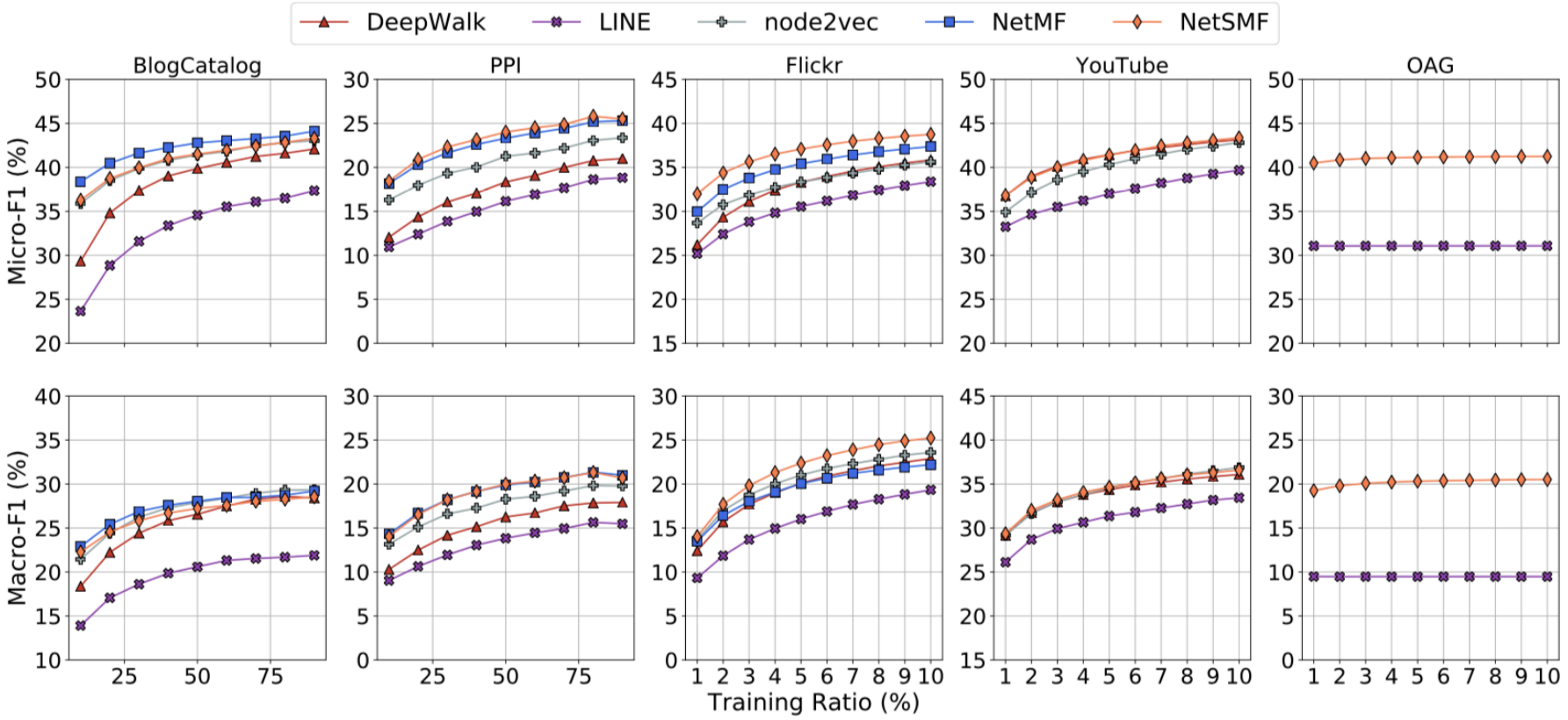


Figure 7: Predictive performance on varying the ratio of training data. The x-axis represents the ratio of labeled data (%), and the y-axis in the top and bottom rows denote the Micro-F1 and Macro-F1 scores

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

NE as Sparse Matrix Factorization

- node-context set $\mathcal{D} = E$ (sparsity)
- To avoid the trivial solution $(r_i = c_j, r_i^T c_j \to \infty, s.t. \hat{p} \to 1)$
- Local negative samples drawn from

$$P_{\mathcal{D},j} \propto \sum_{i:(i,j)\in\mathcal{D}} p_{i,j}$$

- Modify the loss (sum over the edge-->sparse)

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

NE as Sparse Matrix Factorization

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

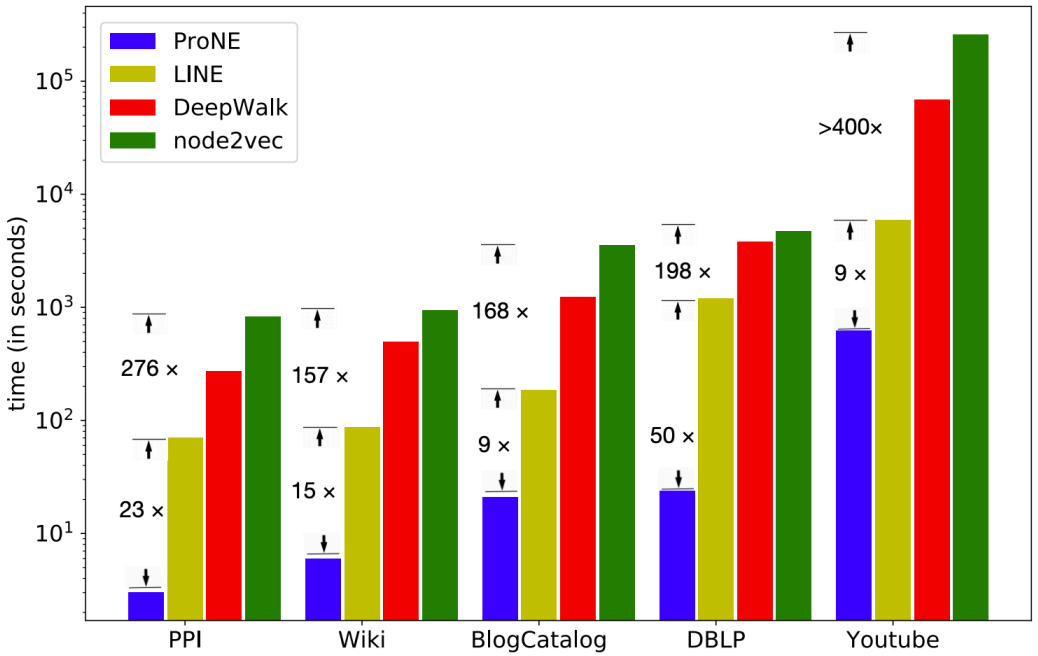
$$\log\left(\frac{\mathrm{vol}(G)}{b}\left(\frac{1}{T}\sum_{r=1}^{T}\left(\boldsymbol{D}^{-1}\boldsymbol{A}\right)^{r}\right)\boldsymbol{D}^{-1}\right) \quad \textbf{v.s.} \quad M_{i,j}=\begin{cases}\ln p_{i,j}-\ln(\lambda P_{D,j}) & ,(v_i,v_j)\in\mathcal{D}\\ 0 & ,(v_i,v_j)\notin\mathcal{D}\end{cases}$$

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

Complexity of ProNE

- Spectral propagation only involves sparse matrix multiplication! The complexity is linear!
- sparse matrix factorization + spectral propagation = $O(|V|d^2 + k|E|)$

Results



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

* 10 minutes on Youtube (~1M nodes)

Dataset	*DeepWalk*	*LINE*	*node2vec*	*ProNE*
PPI	272	70	828	**3**
Wiki	494	87	939	**6**
BlogCatalog	1,231	185	3,533	**21**
DBLP	3,825	1,204	4,749	**24**
Youtube	68,272	5,890	>5days	**627**

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

Effectiveness experiments

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

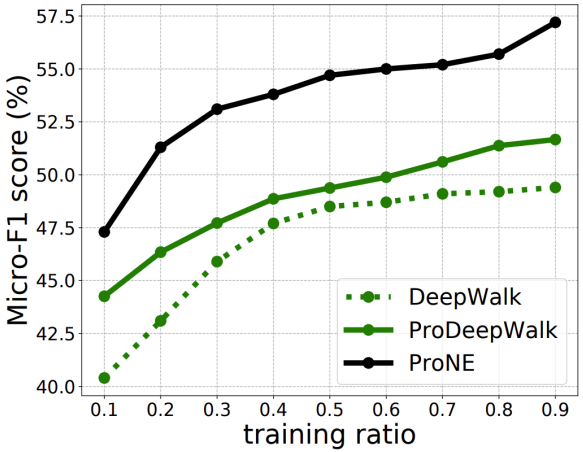
Dataset	training ratio	0.01	0.03	0.05	0.07	0.09
DBLP	DeepWalk	49.3	55.0	57.1	57.9	58.4
	LINE	48.7	52.6	53.5	54.1	54.5
	node2vec	48.9	55.1	57.0	58.0	58.4
	GraRep	50.5	52.6	53.2	53.5	53.8
	HOPE	**52.2**	55.0	55.9	56.3	56.6
	ProNE (SMF)	50.8	54.9	56.1	56.7	57.0
	ProNE	48.8	**56.2**	**58.0**	**58.8**	**59.2**
	($\pm\sigma$)	($\pm$1.0)	($\pm$0.5)	($\pm$0.2)	($\pm$0.2)	($\pm$0.1)
Youtube	DeepWalk	38.0	40.1	41.3	42.1	42.8
	LINE	33.2	35.5	37.0	38.2	39.3
	ProNE (SMF)	36.5	40.2	41.2	41.7	42.1
	ProNE	**38.2**	**41.4**	**42.3**	**42.9**	**43.3**
	($\pm\sigma$)	($\pm$0.8)	($\pm$0.3)	($\pm$0.2)	($\pm$0.2)	($\pm$0.2)

* ProNE (SMF) = ProNE w/

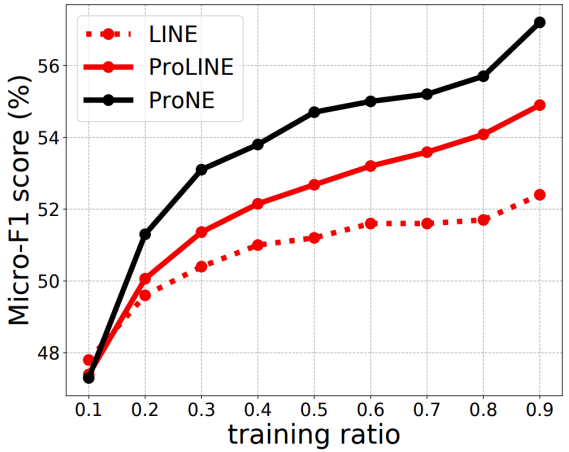
Embed 100,000,000 nodes by one thread:
29 hours with **performance superiority**

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

Spectral Propagation: *k*-way Cheeger

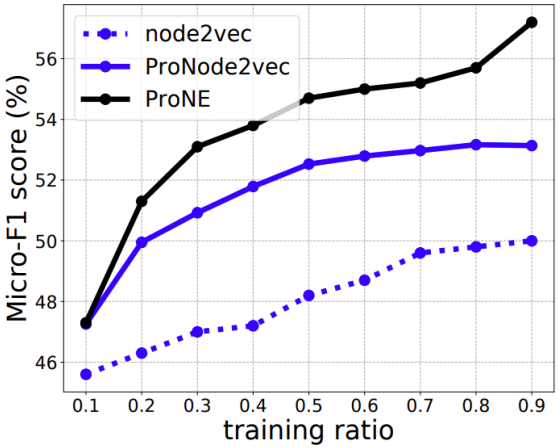


(a) ProDeepWalk

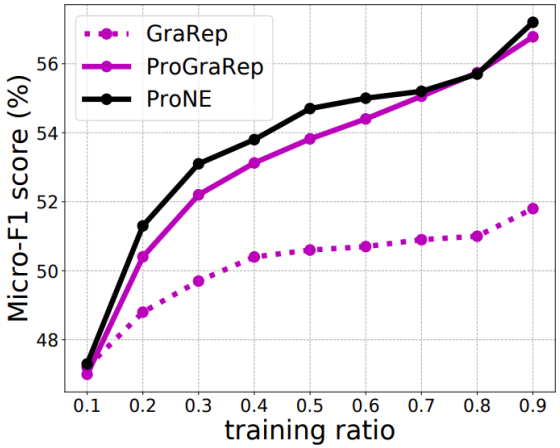


(b) ProLINE

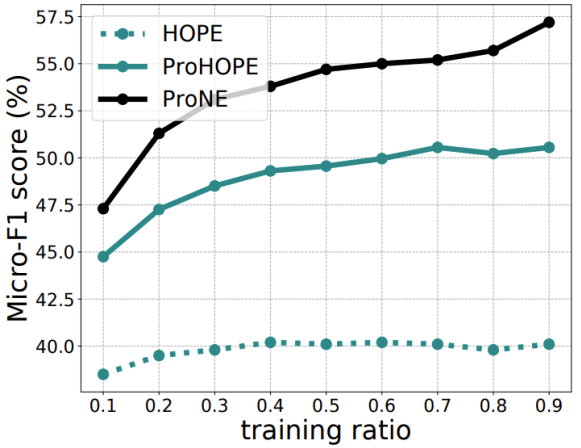
$$\frac{\lambda_k}{2} \leq \rho_G(k) \leq O(k^2)\sqrt{\lambda_k}$$



(c) ProNode2vec

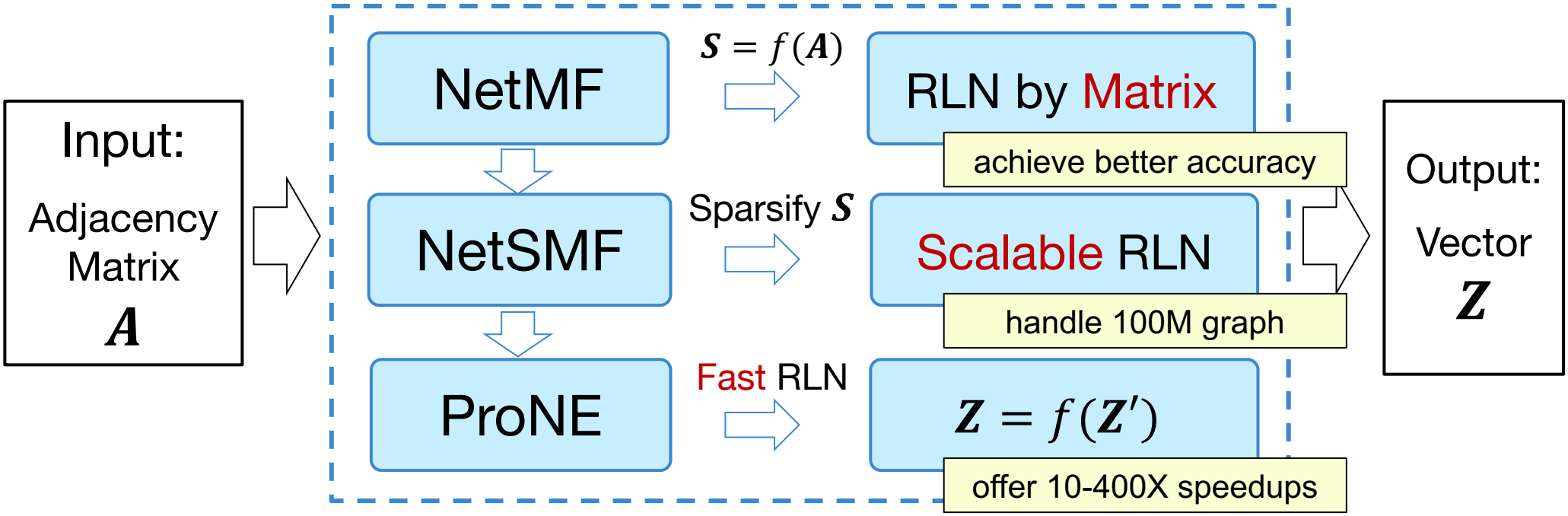


(d) ProGraRep



(e) ProHOPE

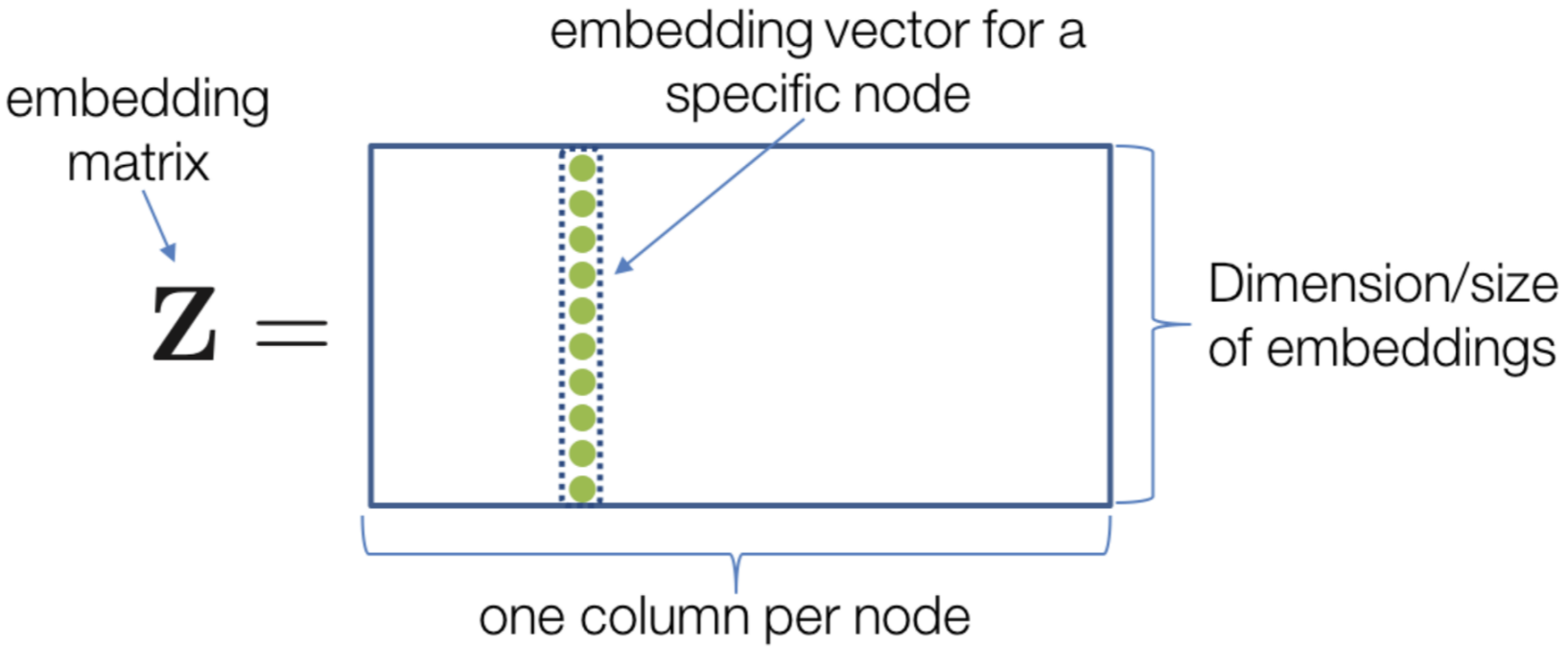
Representation Learning on Networks



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

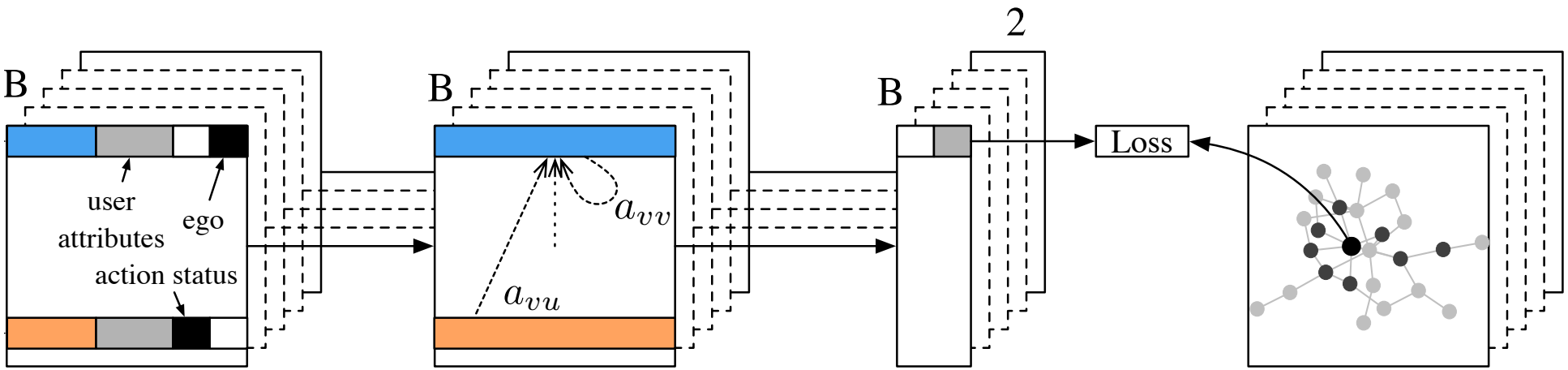
From Shallow to Deep

- So far we have focused on shallow encoders, i.e. embedding lookups:

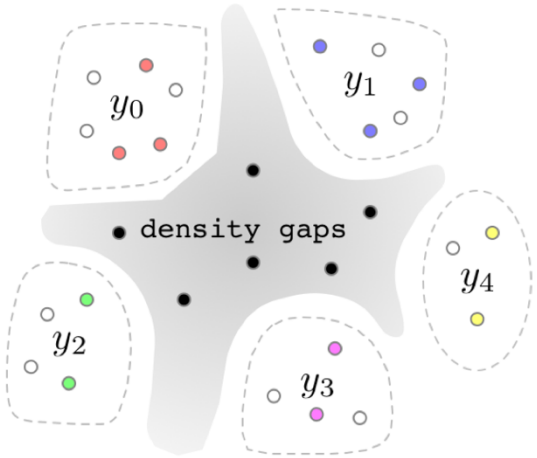


GCN

- GCNs can be considered as a simplification of the traditional graph spectral methods
- The common strategy is to model a node's neighborhood as the receptive field and then apply the *graph convolution* operation



Recent GCN Research



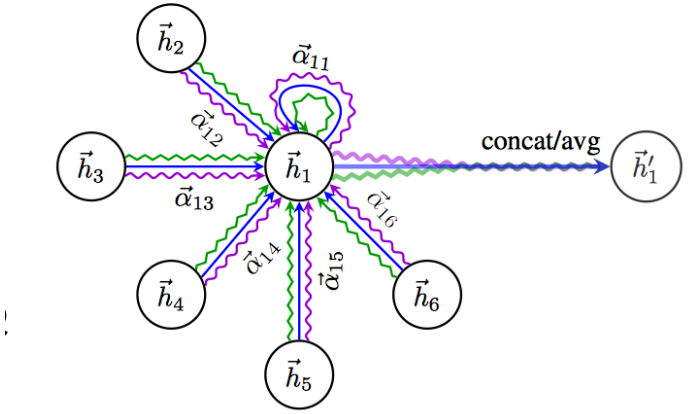
2018 GraphSGAN

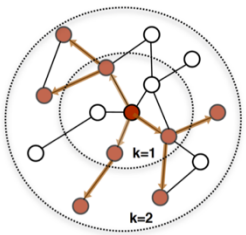
FastGCN 2018

2018 GAT

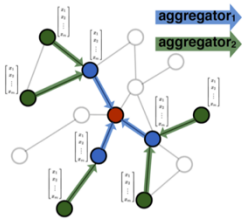
GraphSAGE 2017

2017 GCN

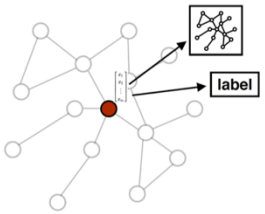




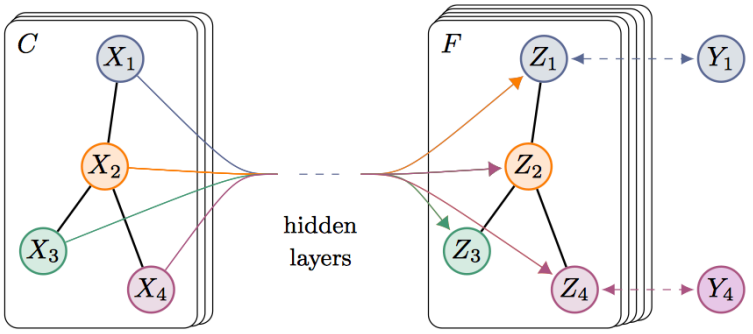
1. Sample neighborhood



2. Aggregate feature information from neighbors

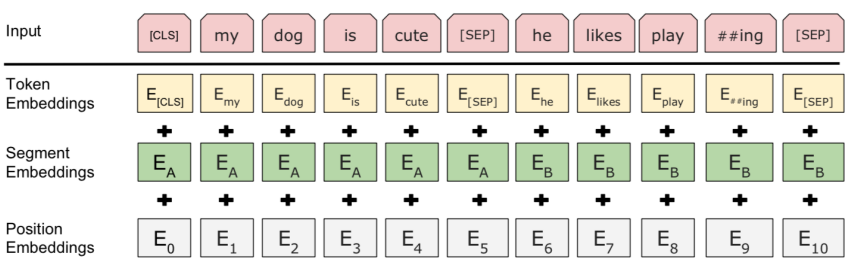


3. Predict graph context and label using aggregated information

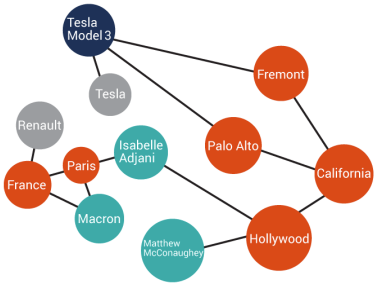


Challenges

- Graph Neural Network Pre-Training



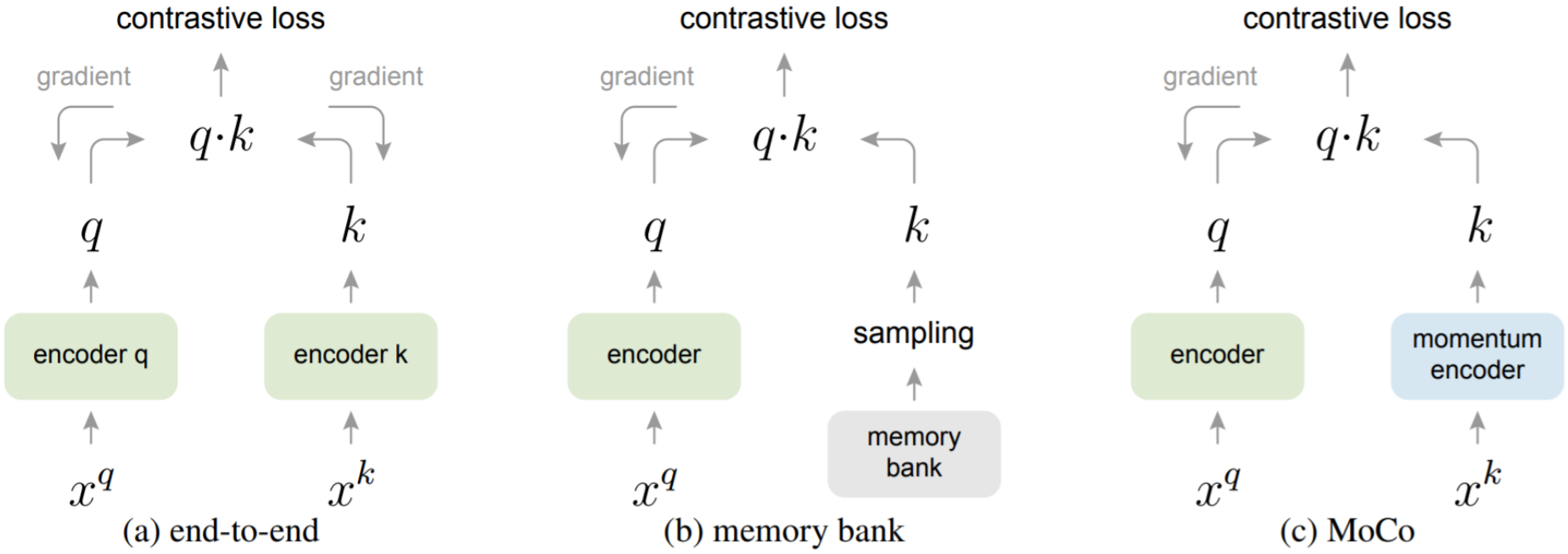
+



GCC: Graph Contrastive Coding for Graph Neural Network Pre-Training

1. J. Qiu, Q. Chen, Y. Dong, J. Zhang, H. Yang, M. Ding, K. Wang, and J. Tang. GCC: Graph Contrastive Coding for Graph Neural Network Pre-Training. *KDD'20*.

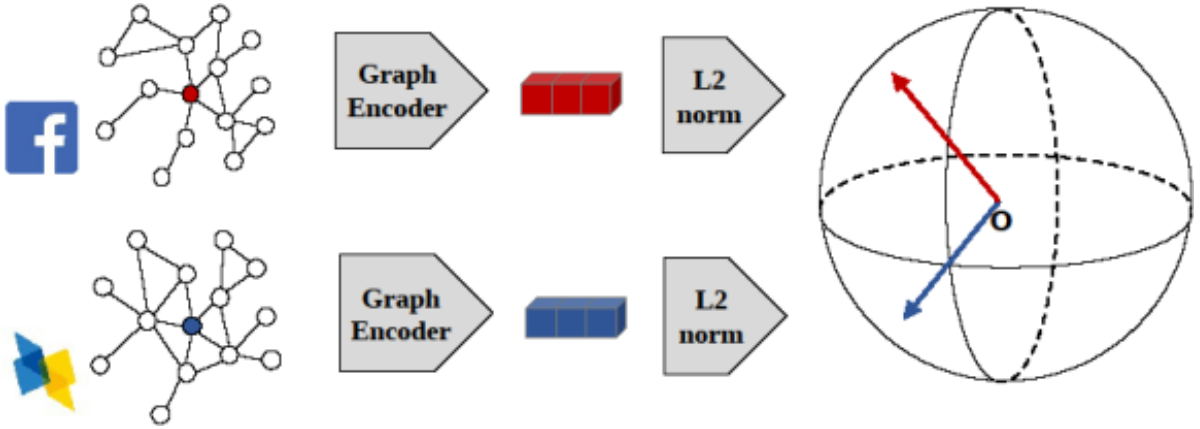
Revisit MoCo



- Momentum Contrast for Unsupervised Visual Representation Learning

Graph Contrastive Coding (GCC)

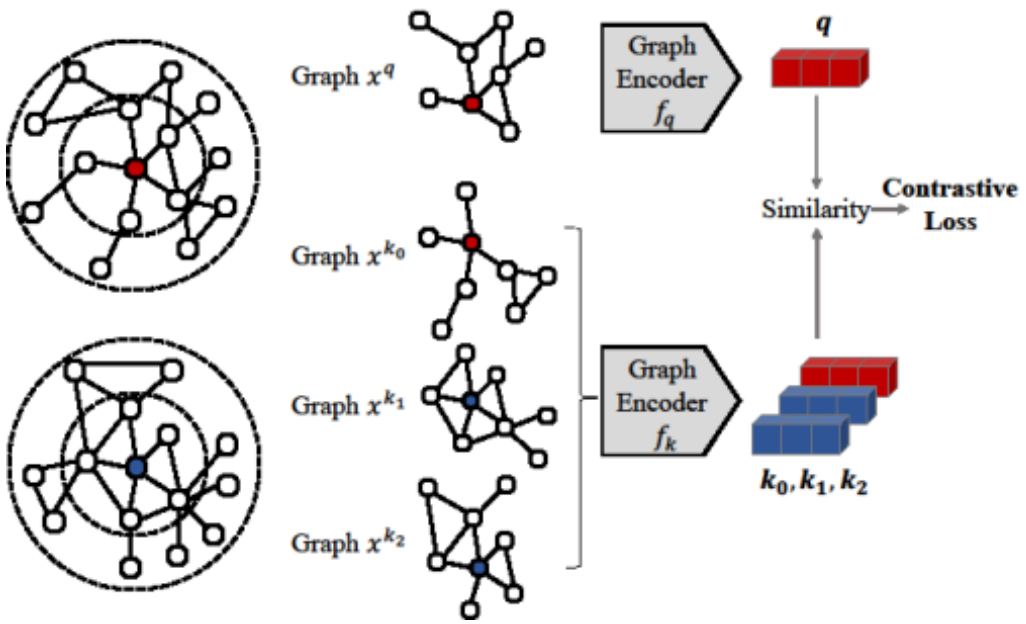
- Problem formulation:
  - Measuring vertex structural similarity



  - New wine in old bottles:
    - motif, centrality, clustering coefficient, structural diversity, edge density, etc

Graph Contrastive Coding (GCC)

- Define positive pairs by sub-graph sampling
  - Random walk with restart sub-graph sampling



1. J. Qiu, Q. Chen, Y. Dong, J. Zhang, H. Yang, M. Ding, K. Wang, and J. Tang. GCC: Graph Contrastive Coding for Graph Neural Network Pre-Training. *KDD'20*.

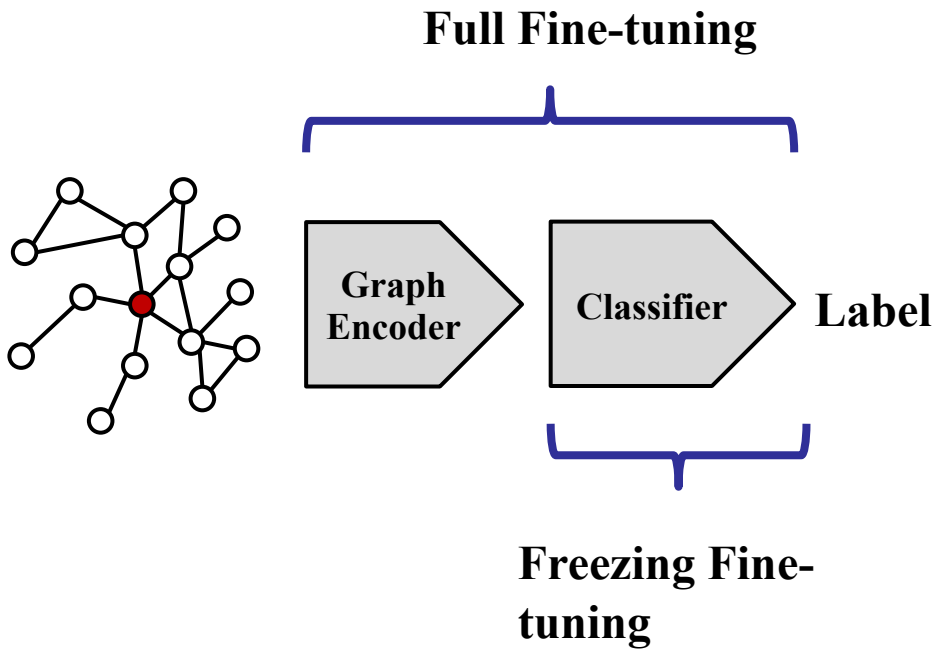
GCC Pre-Training / Fine-tuning

- Six real-world information networks for pre-training.

Table 1: Datasets for pre-training, sorted by number of vertices.

Dataset	Academia	DBLP (SNAP)	DBLP (NetRep)	IMDB	Facebook	LiveJournal
$\|V\|$	137,969	317,080	540,486	896,305	3,097,165	4,843,953
$\|E\|$	739,384	2,099,732	30,491,458	7,564,894	47,334,788	85,691,368

- Fine-tuning Tasks:
  - Node classification
  - Graph classification
  - Similarity search
- Two fine-tuning settings.



Node Classification

Datasets	US-Airport	H-index
$\|V\|$	1,190	5,000
$\|E\|$	13,599	44,020
ProNE	62.3	69.1
GraphWave	60.2	70.3
Struc2vec	**66.2**	> 1 Day
GCC (E2E, freeze)	64.8	**78.3**
GCC (MoCo, freeze)	65.6	75.2
GCC (rand, full)	64.2	76.9
GCC (E2E, full)	**68.3**	80.5
GCC (MoCo, full)	67.2	**80.6**

Graph Classification

Datasets	IMDB-B	IMDB-M	COLLAB	RDT-B	RDT-M
# graphs	1,000	1,500	5,000	2,000	5,000
# classes	2	3	3	2	5
Avg. # nodes	19.8	13.0	74.5	429.6	508.5
DGK	67.0	44.6	73.1	78.0	41.3
graph2vec	71.1	50.4	–	75.8	47.9
InfoGraph	**73.0**	**49.7**	–	82.5	53.5
GCC (E2E, freeze)	71.7	49.3	74.7	87.5	52.6
GCC (MoCo, freeze)	72.0	49.4	**78.9**	**89.8**	**53.7**
DGCNN	70.0	47.8	73.7	–	–
GIN	**75.6**	**51.5**	80.2	**89.4**	**54.5**
GCC (rand, full)	**75.6**	50.9	79.4	87.8	52.1
GCC (E2E, full)	70.8	48.5	79.0	86.4	47.4
GCC (MoCo, full)	73.8	50.3	**81.1**	87.6	53.0

Similarity Search

	KDD-ICDM		SIGIR-CIKM		SIGMOD-ICDE	
$\|V\|$	2,867	2,607	2,851	3,548	2,616	2,559
$\|E\|$	7,637	4,774	6,354	7,076	8,304	6,668
# groud truth		697		874		898
k	20	40	20	40	20	40
Random	0.0198	0.0566	0.0223	0.0447	0.0221	0.0521
RolX	0.0779	0.1288	0.0548	0.0984	0.0776	0.1309
Panther++	0.0892	0.1558	**0.0782**	0.1185	0.0921	0.1320
GraphWave	0.0846	**0.1693**	0.0549	0.0995	**0.0947**	**0.1470**
GCC (E2E)	**0.1047**	0.1564	0.0549	**0.1247**	0.0835	0.1336
GCC (MoCo)	0.0904	0.1521	0.0652	0.1178	0.0846	0.1425

Next 10

Edward Feigenbaum
Turing Award Winner

Cognitive Reasoning

—Trillion-scale common-sense knowledge graph

Tim Berners Lee
Turing Award Winner

Big Data ⇨ Knowledge ⇨ Intelligence

* AI = Knowledge + Intelligence

1. J. Tang, J. Zhang, L. Yao, J. Li, L. Zhang, and Z. Su. ArnetMiner: Extraction and Mining of Academic Social Networks. KDD'08. pp.990-998.

挑战与未来(Next 30)

意识

—让计算机具有自我意识

认知推理 ⇨ 记忆 ⇨ 自我意识

- Next AI = Reasoning + Memory + Consciousness

1. J. Tang, J. Zhang, L. Yao, J. Li, L. Zhang, and Z. Su. ArnetMiner: Extraction and Mining of Academic Social Networks. KDD'08. pp.990-998.

意识：全局工作理论—GWT

所谓的有意识学习，就是有目标的机器学习，给定数据，训练一个模型，通过模型进行分类（决策），所以有意识学习输出是一个短期记忆模型，但这个短期记忆模型比较简单：数据有限、无背景知识。所以无意识就对应着长期记忆模型，长期记忆模型有点类似半监督或者无监督学习模型，或者当下比较流行的预训练和自监督学习；无意识处理对应多种长期记忆，所以无意识可以考虑多个不同的处理器，这些处理器之间可以有链接，也可以没有，很多时候是并行处理，但针对特定任务，比如有意识思考某个问题的时候，形成特定连接，包括无意识处理器（无监督模型）和有意识处理器（有监督模型）之间的连接，这里可以考虑成fine-tune。当然连接权重可以通过外界反馈强化学习来实现。无意识处理器之间的连接以及和有意识处理器之间的连接可以类比为注意力机制。最后值得注意的是长期记忆的构造和实现，人脑记忆保存的是模型图，而不是概念图。每个长期记忆都可能是一个模型，可以生成样本，具体学习方法，可以想象一下是一个层次聚类。通过这样就可以用有监督、无监督、强化、注意力、fine-tune来实现GWT模型。

Related Publications

For more, check http://keg.cs.tsinghua.edu.cn/jietang

- Jiezhong Qiu, Qibin Chen, Yuxiao Dong, Jing Zhang, Hongxia Yang, Ming Ding, Kuansan Wang, and Jie Tang. GCC: Graph Contrastive Coding for Structural Graph Representation Pre-Training. KDD'20.
- Zhen Yang, Ming Ding, Chang Zhou, Hongxia Yang, Jingren Zhou, and Jie Tang. Understanding Negative Sampling in Graph Representation Learning. KDD'20.
- Yukuo Cen, Jianwei Zhang, Xu Zou, Chang Zhou, Hongxia Yang, and Jie Tang. Controllable Multi-Interest Framework for Recommendation. KDD'20.
- Yuxiao Dong, Ziniu Hu, Kuansan Wang, Yizhou Sun and Jie Tang. Heterogeneous Network Representation Learning. IJCAI'20.
- Ming Ding, Chang Zhou, Qibin Chen, Hongxia Yang, and Jie Tang. Cognitive Graph for Multi-Hop Reading Comprehension at Scale. ACL'19.
- Jie Zhang, Yuxiao Dong, Yan Wang, Jie Tang, and Ming Ding. ProNE: Fast and Scalable Network Representation Learning. IJCAI'19.
- Yukuo Cen, Xu Zou, Jianwei Zhang, Hongxia Yang, Jingren Zhou and Jie Tang. Representation Learning for Attributed Multiplex Heterogeneous Network. KDD'19.
- Fanjin Zhang, Xiao Liu, Jie Tang, Yuxiao Dong, Peiran Yao, Jie Zhang, Xiaotao Gu, Yan Wang, Bin Shao, Rui Li, and Kuansan Wang. OAG: Toward Linking Large-scale Heterogeneous Entity Graphs. KDD'19.
- Qibin Chen, Junyang Lin, Yichang Zhang, Hongxia Yang, Jingren Zhou and Jie Tang. Towards Knowledge-Based Personalized Product Description Generation in E-commerce. KDD'19.
- Yifeng Zhao, Xiangwei Wang, Hongxia Yang, Le Song, and Jie Tang. Large Scale Evolving Graphs with Burst Detection. IJCAI'19.
- Yu Han, Jie Tang, and Qian Chen. Network Embedding under Partial Monitoring for Evolving Networks. IJCAI'19.
- Yifeng Zhao, Xiangwei Wang, Hongxia Yang, Le Song, and Jie Tang. Large Scale Evolving Graphs with Burst Detection. IJCAI'19.
- Jiezhong Qiu, Yuxiao Dong, Hao Ma, Jian Li, Chi Wang, Kuansan Wang, and Jie Tang. NetSMF: Large-Scale Network Embedding as Sparse Matrix Factorization. WWW'19.
- Jiezhong Qiu, Jian Tang, Hao Ma, Yuxiao Dong, Kuansan Wang, and Jie Tang. DeepInf: Modeling Influence Locality in Large Social Networks. KDD'18.
- Jiezhong Qiu, Yuxiao Dong, Hao Ma, Jian Li, Kuansan Wang, and Jie Tang. Network Embedding as Matrix Factorization: Unifying DeepWalk, LINE, PTE, and node2vec. WSDM'18.
- Jie Tang, Jing Zhang, Limin Yao, Juanzi Li, Li Zhang, and Zhong Su. ArnetMiner: Extraction and Mining of Academic Social Networks. KDD'08.

Thank you !

Collaborators:

Jie Zhang, Ming Ding, Jiezhong Qiu, Qibin Chen, Yifeng Zhao, Yukuo Cen, Yu Han, Fanjin Zhang, Xu Zou, Yan Wang, et al. (**THU**)

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

Yuxiao Dong, Chi Wang, Hao Ma, Kuansan Wang (**Microsoft**)

Jie Tang, KEG, Tsinghua U

Download all data & Codes

http://keg.cs.tsinghua.edu.cn/jietang

https://keg.cs.tsinghua.edu.cn/cogdl/

https://github.com/THUDM