Cognitive Graph for Understanding, Reasoning, and Decision

Jie Tang

Computer Science

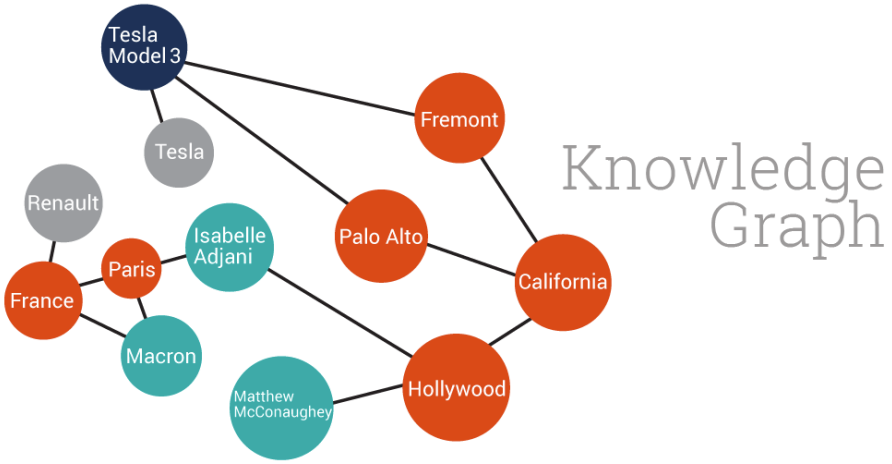
Tsinghua University

The slides can be downloaded at http://keg.cs.tsinghua.edu.cn/jietang

Knowledge Graph

- “Knowledge graph” was used by Google in 2012
- Knowledge engineering, expert system
- CYC: the world's longest-lived AI project (1985)

Edward Feigenbaum
Father of KB
Turing Award



Tim Berners Lee
Father of WWW
Turing Award

Networked World

facebook

- 2 billion MAU
- 26.4 billion minutes/day

Alibaba Group 阿里巴巴集团

- >777 million trans. (alipay)
- 200 billion on 11/11

twitter

- 320 million MAU
- Peak: 143K tweets/s

新浪微博 weibo.com

- 462 million users
- influencing our daily life

Instagram

- 700 million MAU
- 95 million pics/day

头条 今日头条

- ~1.5 billion MAU
- 70 minutes/user/day

snapchat

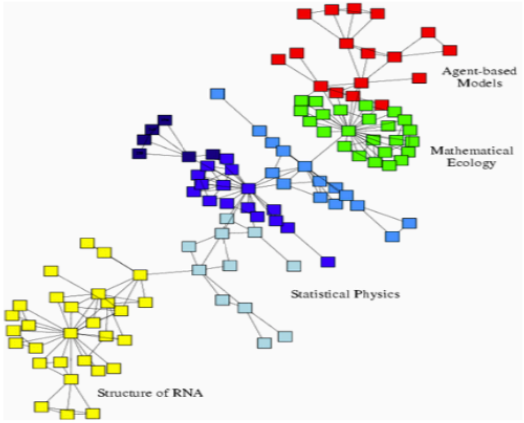
- 300 million MAU
- 30 minutes/user/day

WeChat

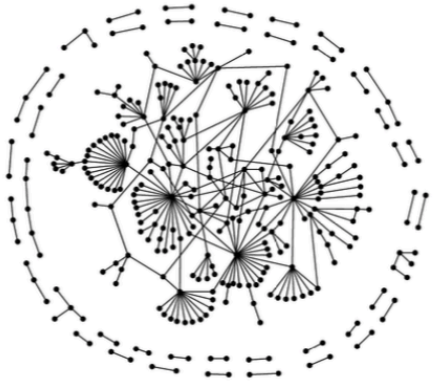
- QQ: 860 million MAU
- WeChat: 1.1 billion MAU

Many Data are Networks[1]

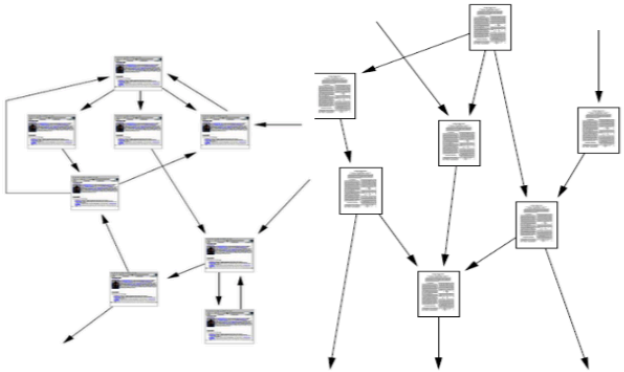
Social networks



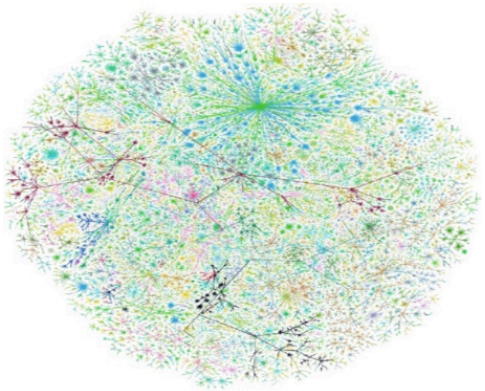
Economic networks



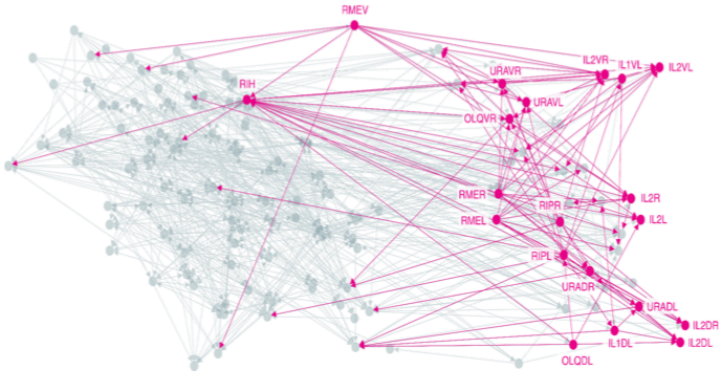
Biomedical networks



Information networks:
Web & citations



Internet



Networks of neurons

1. Example from Jure's slides

Machine Learning with Networks

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

Let us start with a challenging question

—Multi-hop Question Answering (QA)

Multi-hop QA

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...

Challenge 1: Myopic Retrieval Problem

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

Document A

Old School is a 2003 American comedy film released by DreamWorks Pictures and The Montecito Picture Company…

Document B

Many directors shoot scenes in Hollywood, Los Angeles, which is notable as the home of the U.S. film industry.

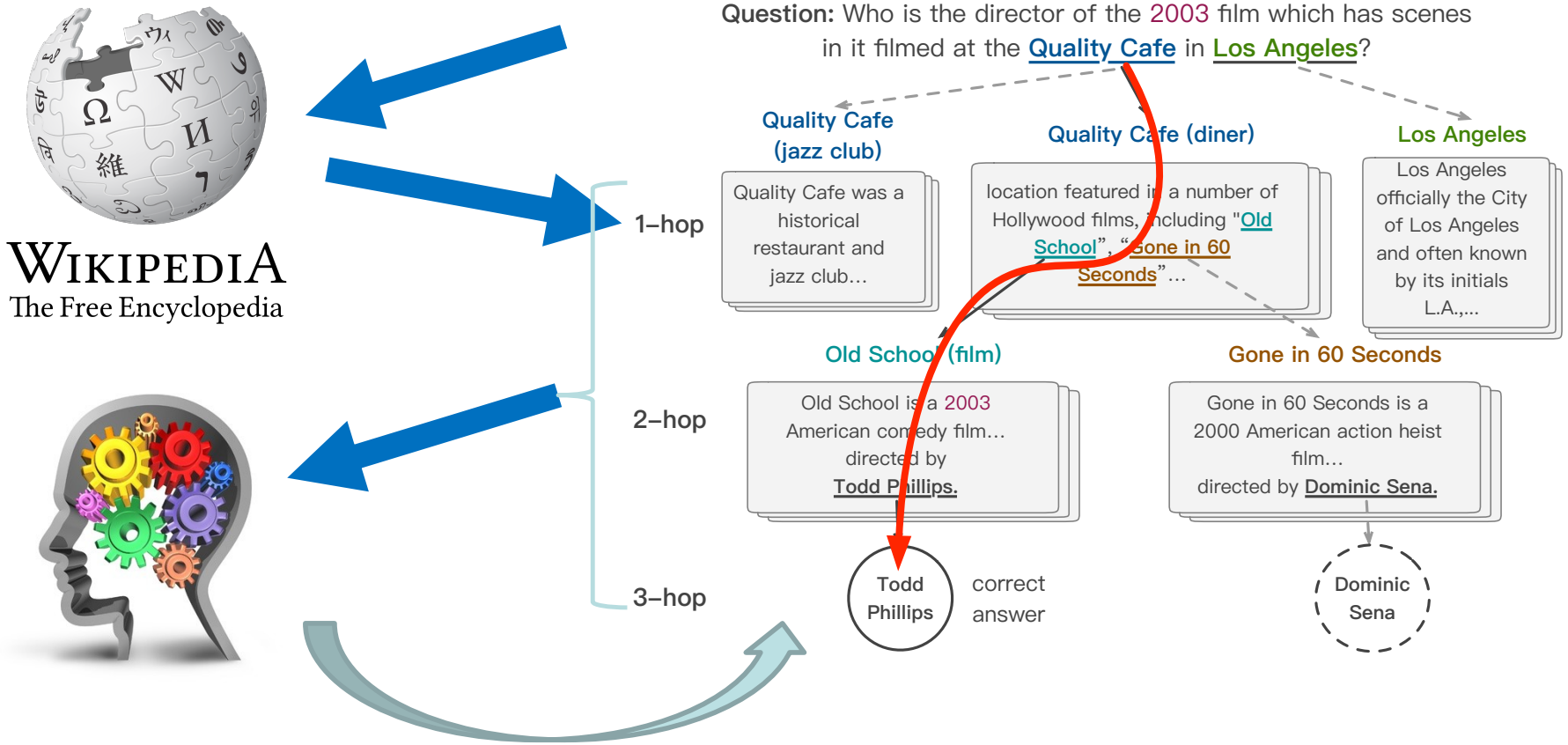
However in multi-hop QA, paragraphs multiple hops away from the question could share few common words with the question, leading to a failed retrieval.

Challenge 2: Explainability

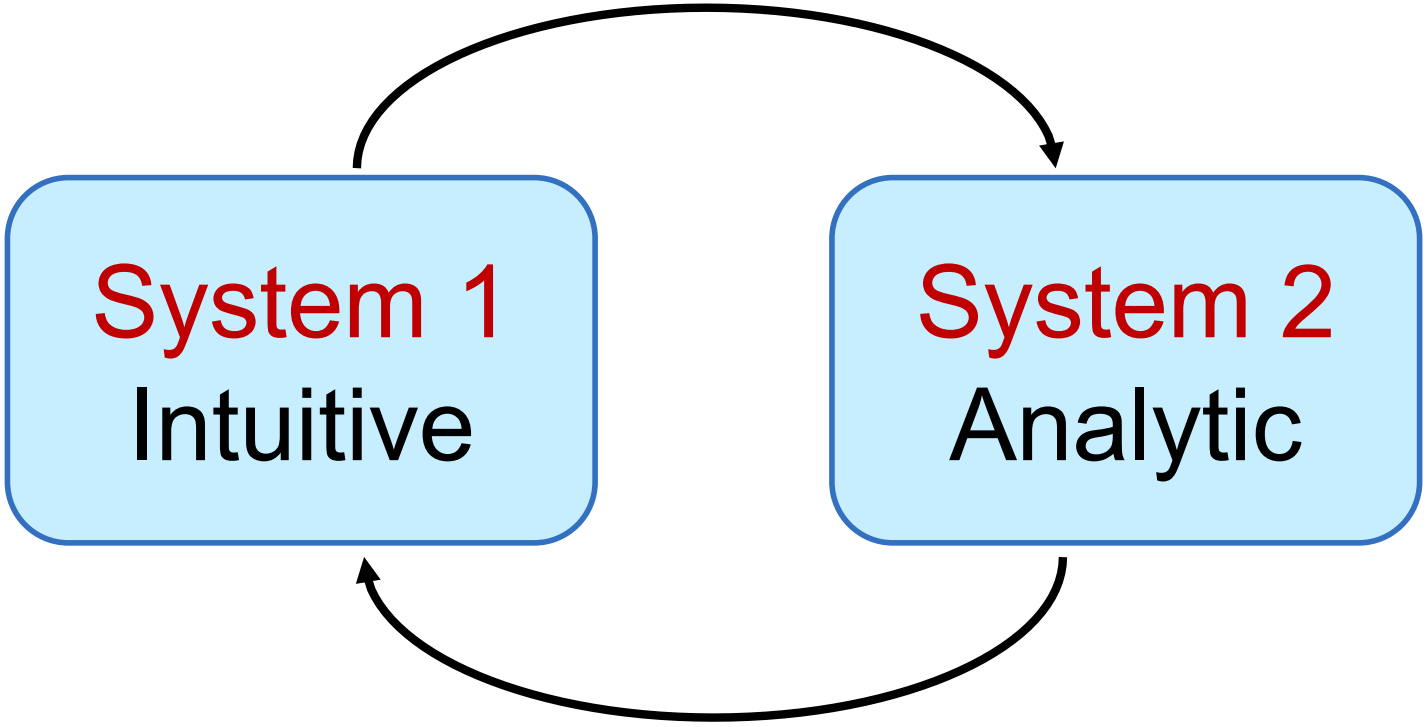
- Most RC models can only be seen as black-boxes:
  - Input: The question and a paragraph
  - Output: The answer span (start and end positions)

- To enable a verification by the user, we need:
  - The reasoning path (graph)
  - The supporting facts (sentences) at each hop
  - Other possible answers and their reasoning paths for comparison

Cognitive Graph: Representation, Reasoning, and Decision



Connecting to Cognitive Science

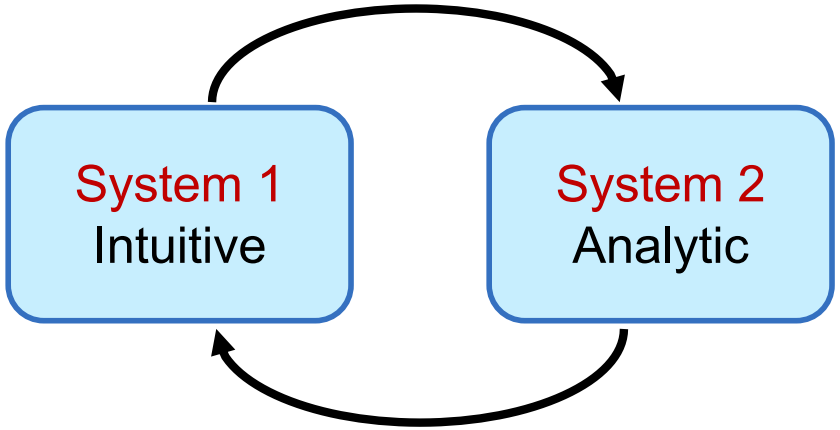


Dual Process Theory

1. Jonathan St BT Evans. 1984. Heuristic and analytic processes in reasoning. British Journal of Psychology, 75(4):451–468.

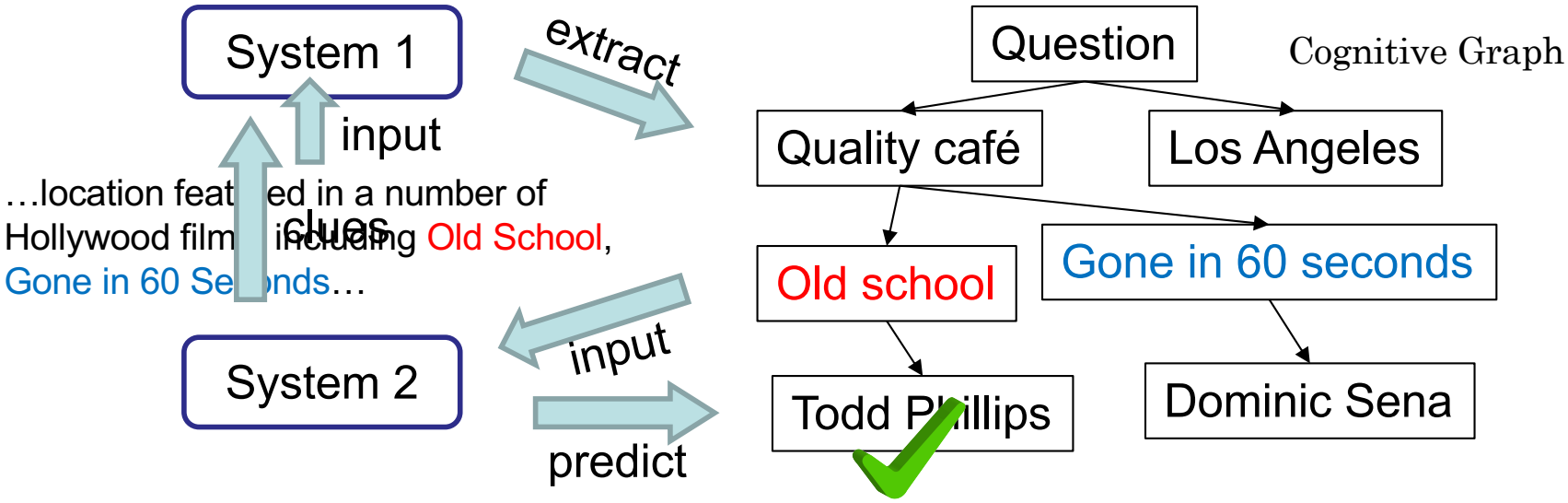
Reasoning w/ Cognitive Graph

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

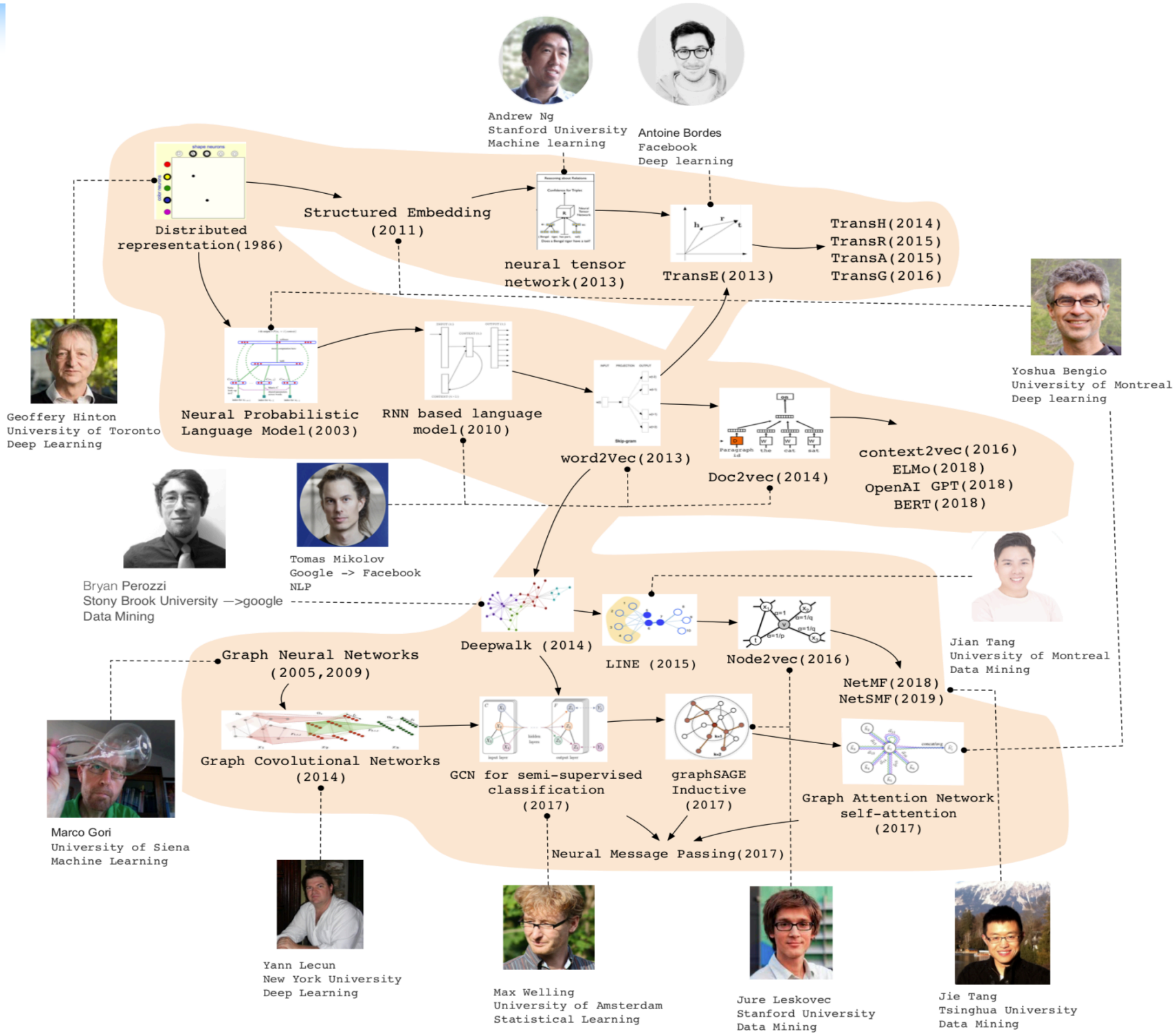


CogQA: Cognitive Graph for QA

- 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



- Q1: How to represent a concept in a graph?


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

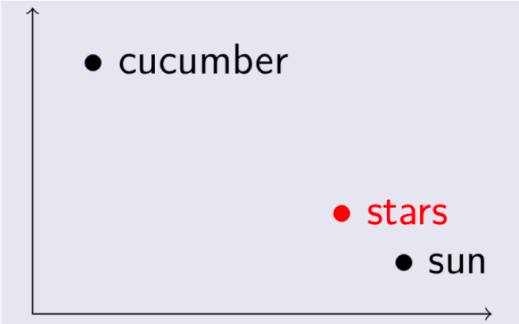
Recall word2vec for NLP

- Learning a representation for words:

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

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

	shining	bright	trees	dark	look
stars	38	45	2	27	12

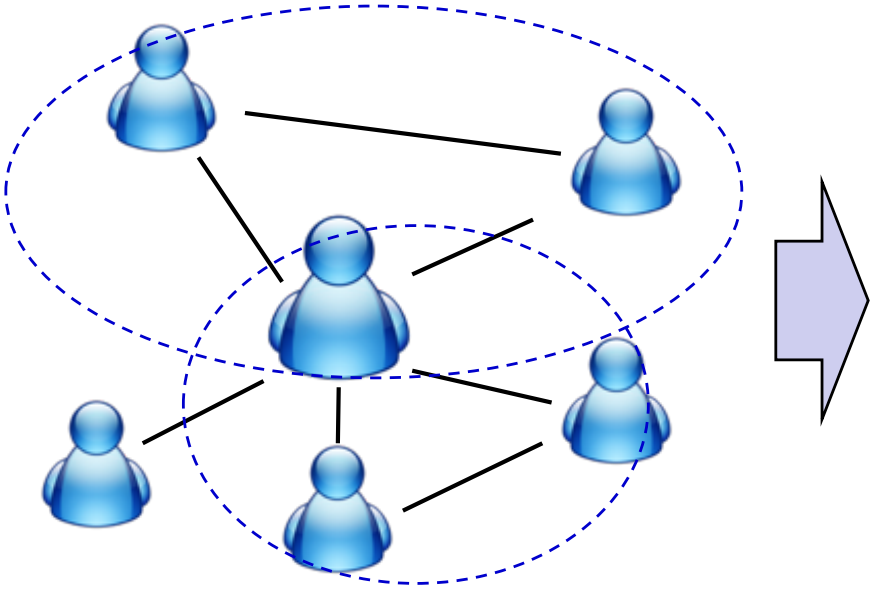


- Ideally, we should have

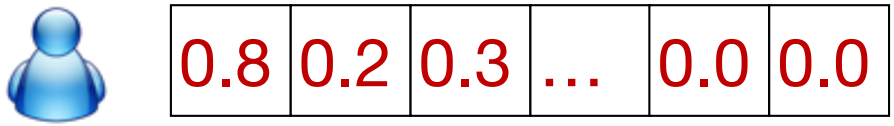
$$\|\phi(\text{sun}) - \phi(\text{stars})\| < \|\phi(\text{cucumber}) - \phi(\text{stars})\|$$

Review Representation Learning for Graphs

Representation Learning/
Graph Embedding

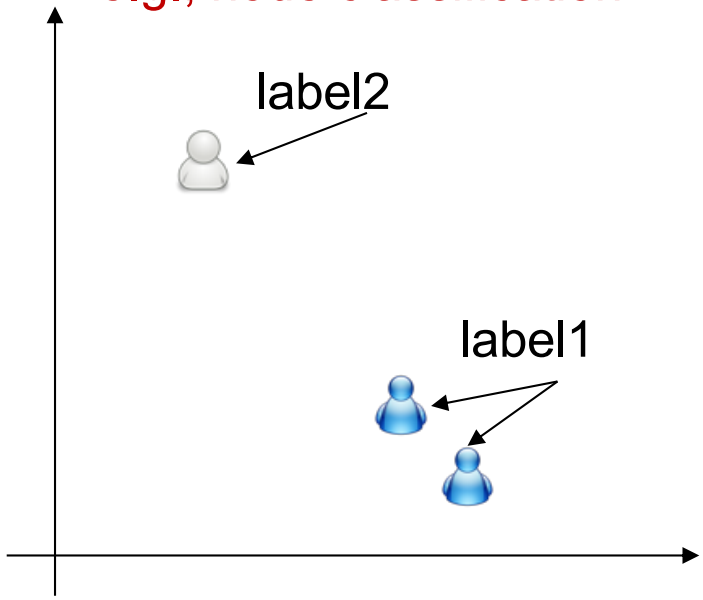


d-dimensional vector, $d << |V|$



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

e.g., node classification

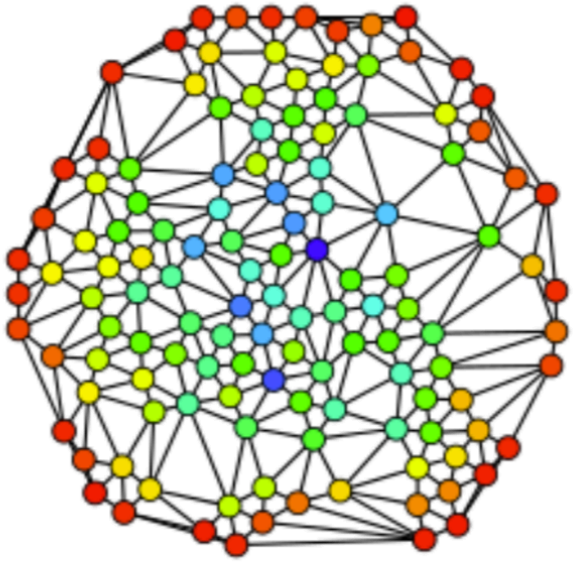


Why is it hard?

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

Node Embeddings

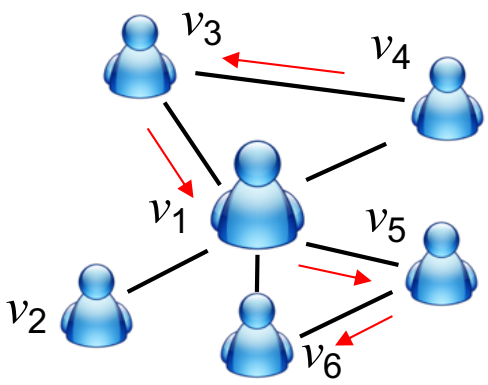
- How to learn a representation for each node of networks?



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

DeepWalk

Random walk



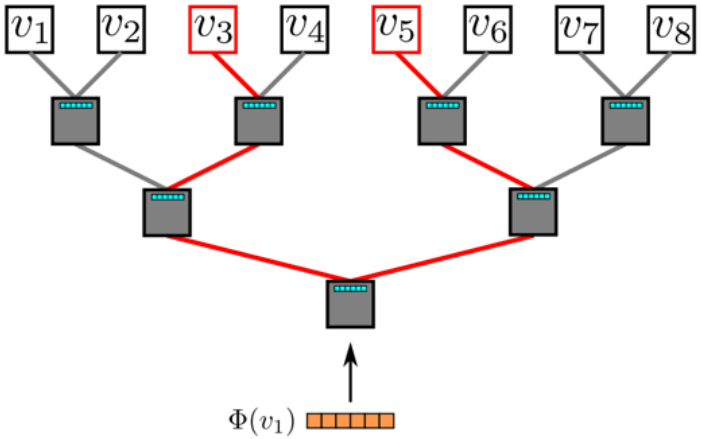
One example RW path

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

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

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

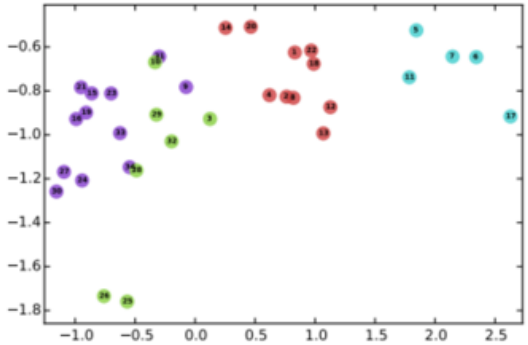
SkipGram with Hierarchical softmax



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

Hierarchical softmax

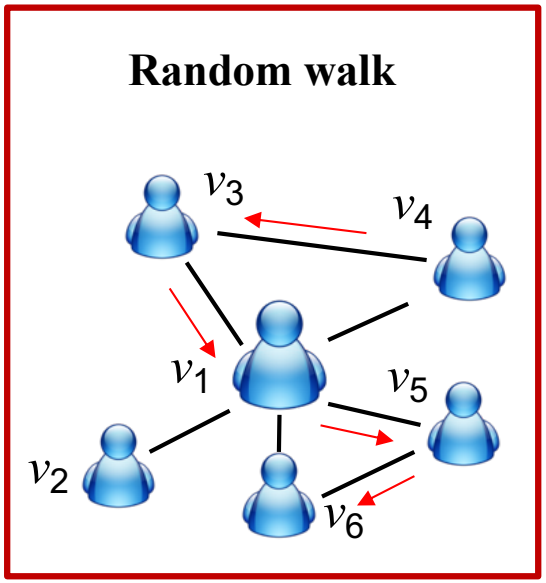
$$P(v_j | \Phi(v_i)) = \prod_{l=1}^{\log |V|} P(b_l | \Phi(v_i)) = \prod_{l=1}^{\log |V|} 1/(1 + e^{-\Phi(v_i) \cdot \Psi(b_l)})$$



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

DeepWalk

Random walk



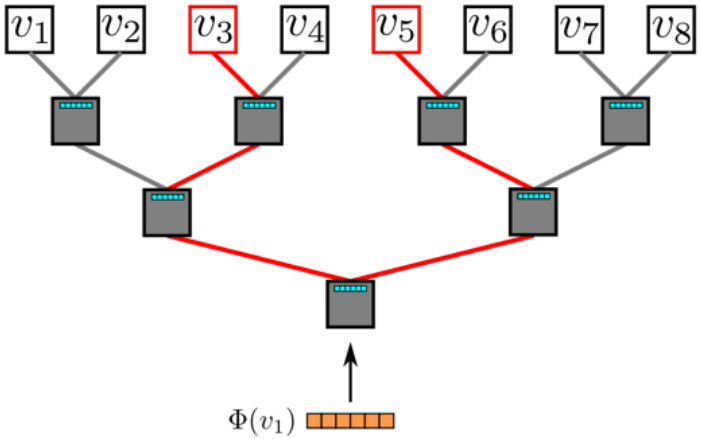
One example RW path

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

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

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

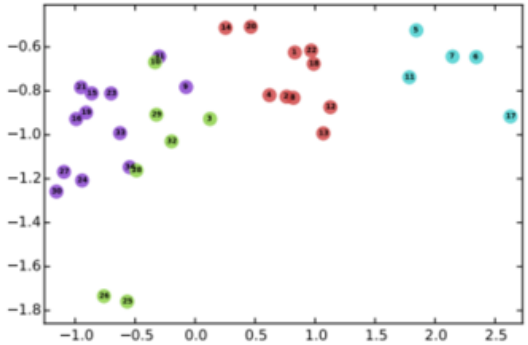
SkipGram with Hierarchical softmax



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

Hierarchical softmax

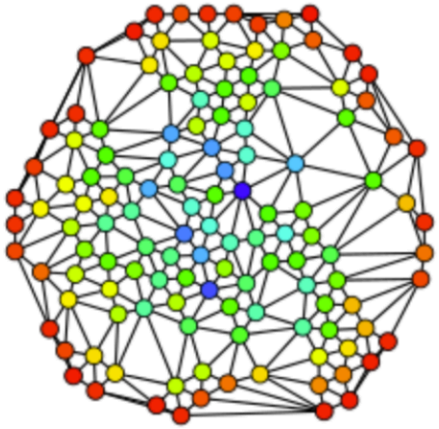
$$P(v_j | \Phi(v_i)) = \prod_{l=1}^{\log |V|} P(b_l | \Phi(v_i)) = \prod_{l=1}^{\log |V|} 1/(1 + e^{-\Phi(v_i) \cdot \Psi(b_l)})$$



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

Random Walk

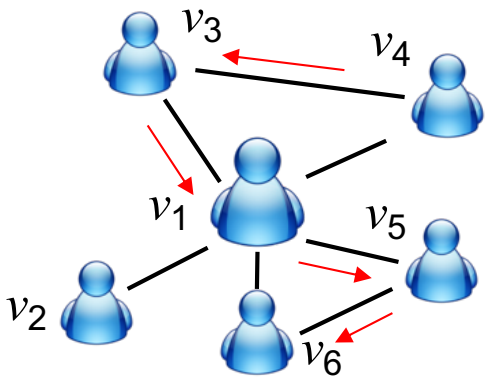
- Generate γ random walks for each vertex
- Each random walk has length t
  - in each random walk step, jump to the next vertex uniformly.
- Example: $v_{46} \rightarrow v_{45} \rightarrow v_{71} \rightarrow v_{24} \rightarrow v_5 \rightarrow v_1 \rightarrow v_{17}$
- Finally, for vertex v_1 in a network,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



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

DeepWalk

Random walk



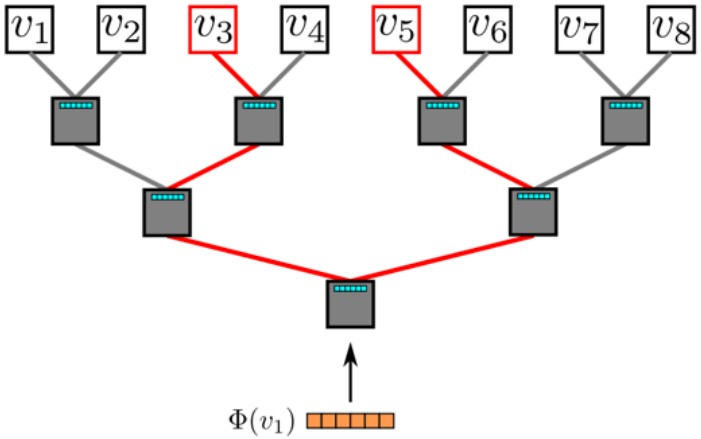
One example RW path

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

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

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

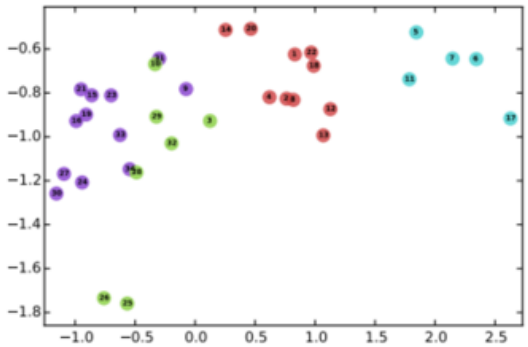
SkipGram with Hierarchical softmax



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

Hierarchical softmax

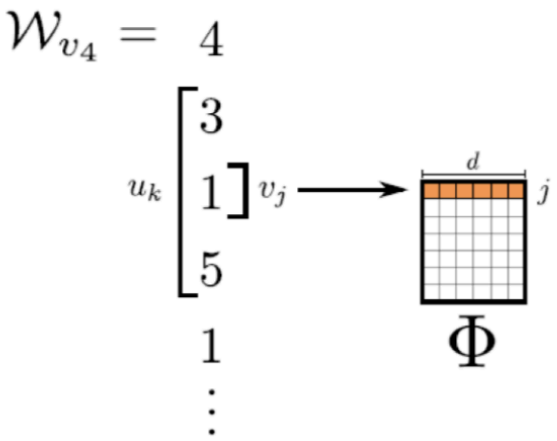
$$P(v_j | \Phi(v_i)) = \prod_{l=1}^{\log |V|} P(b_l | \Phi(v_i)) = \prod_{l=1}^{\log |V|} 1/(1 + e^{-\Phi(v_i) \cdot \Psi(b_l)})$$



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

Representation Mapping

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

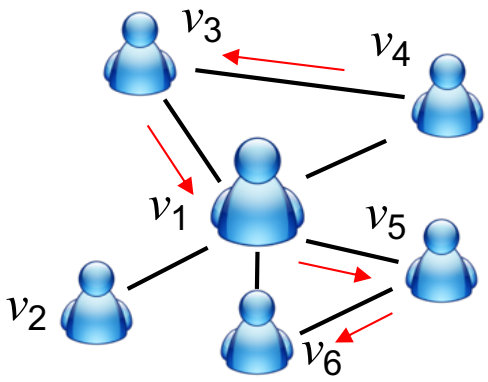


- Define a window size **w**: if $\mathbf{w} = 1$ then $\mathbf{v} = v_1$
- Map the current vertex v_1 to its representation in R^d
- Maximize (skip-gram)

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

DeepWalk

Random walk



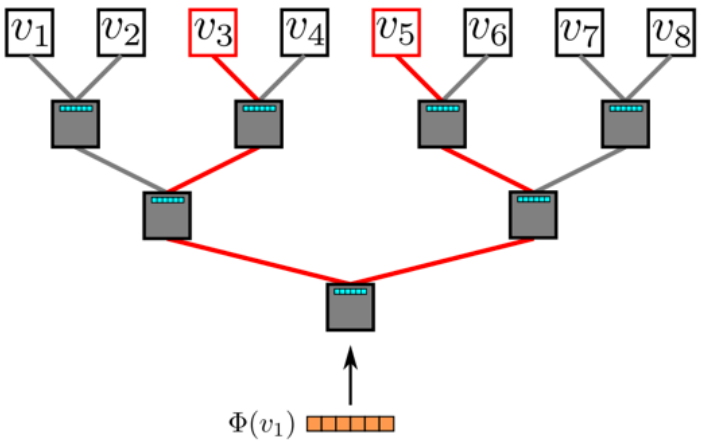
One example RW path

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

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

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

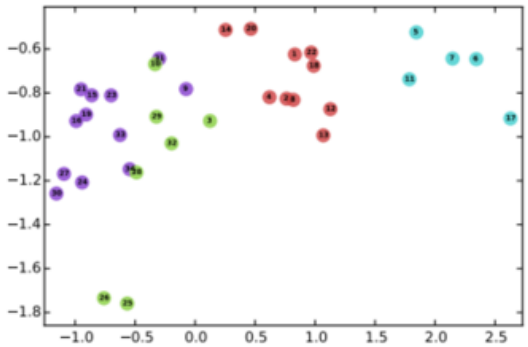
SkipGram with Hierarchical softmax



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

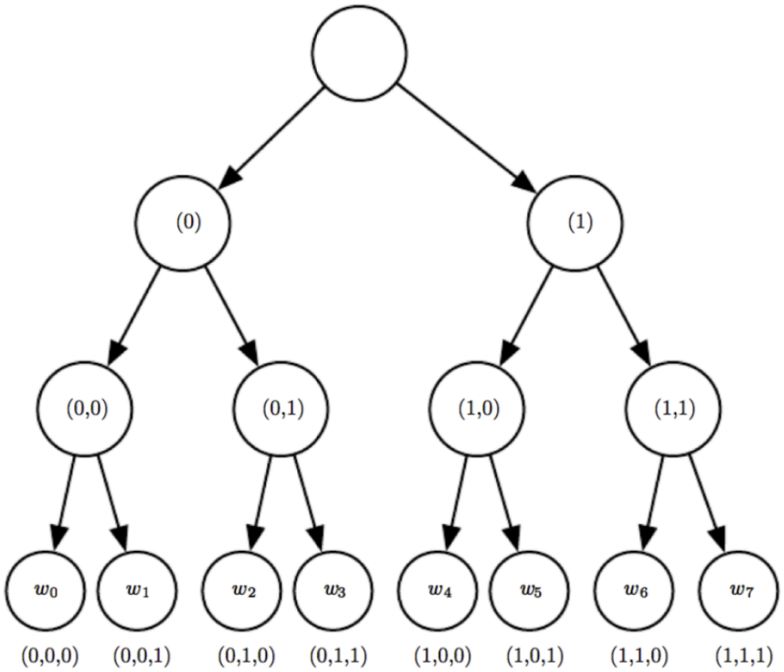
Hierarchical softmax

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



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

Hierarchical Softmax



- Then

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

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

Parameter Learning

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

Results: BlogCatalog

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

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

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

Results: YouTube

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

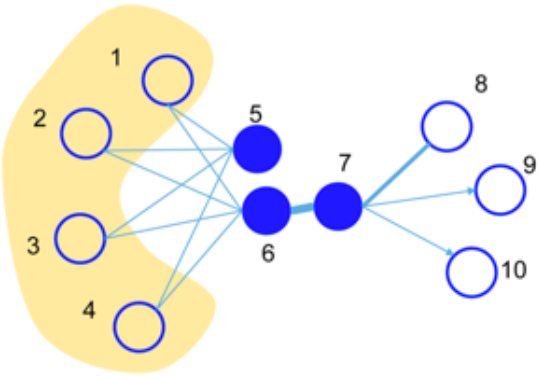
	% Labeled Nodes	1%	2%	3%	4%	5%	6%	7%	8%	9%	10%
	DeepWalk	**37.95**	**39.28**	**40.08**	**40.78**	**41.32**	**41.72**	**42.12**	**42.48**	**42.78**	**43.05**
Micro-F1(%)	SpectralClustering	—	—	—	—	—	—	—	—	—	—
	EdgeCluster	23.90	31.68	35.53	36.76	37.81	38.63	38.94	39.46	39.92	40.07
	Modularity	—	—	—	—	—	—	—	—	—	—
	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
	DeepWalk	**29.22**	**31.83**	**33.06**	**33.90**	**34.35**	**34.66**	**34.96**	**35.22**	**35.42**	**35.67**
Macro-F1(%)	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

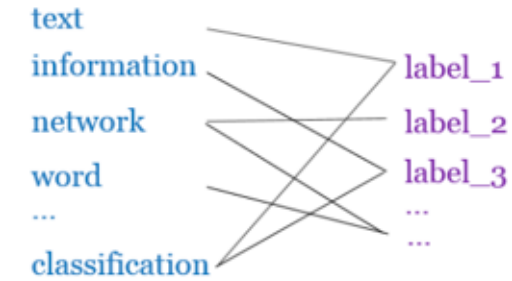
- Similar results on YouTube
- Spectral Clustering does not scale to large graphs

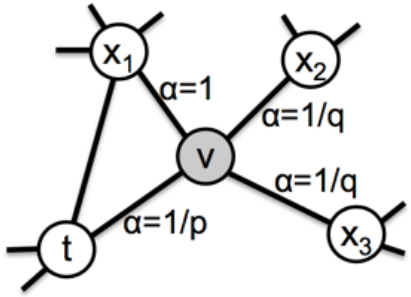
- DeepWalk utilizes fixed-length, unbiased random walks to generate context for each node, can we do better?

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.







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

Questions

- What are the fundamentals underlying the different models?

or

- Can we unify the different graph embedding approaches?

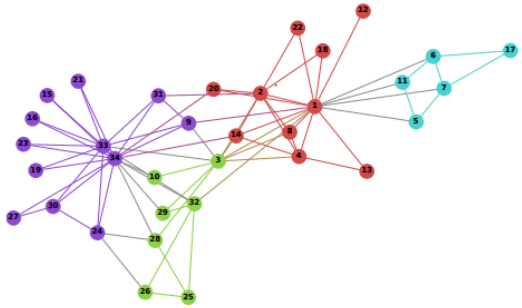
Unifying DeepWalk, LINE, PTE, and node2vec into Matrix Forms

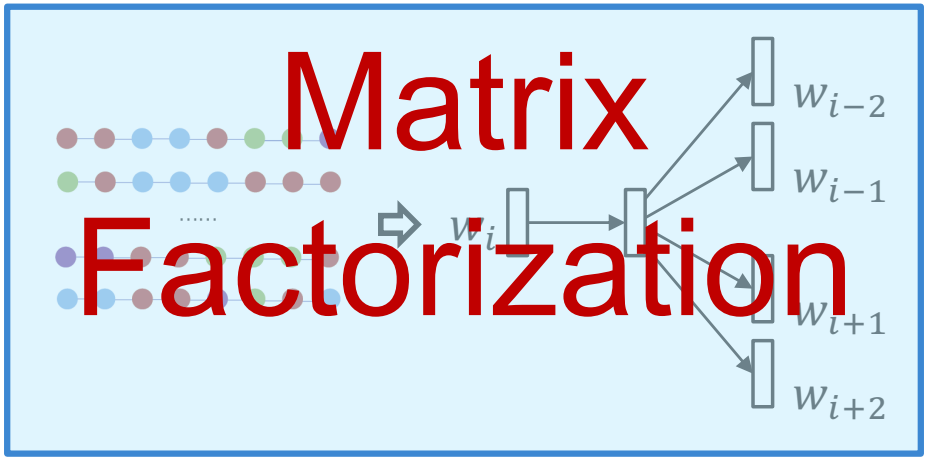
Algorithm	Closed Matrix Form
DeepWalk	$\log\left(\mathrm{vol}(G)\left(\frac{1}{T}\sum_{r=1}^{T}(\boldsymbol{D}^{-1}\boldsymbol{A})^r\right)\boldsymbol{D}^{-1}\right)-\log b$
LINE	$\log\left(\mathrm{vol}(G)\boldsymbol{D}^{-1}\boldsymbol{A}\boldsymbol{D}^{-1}\right)-\log b$
PTE	$\log\left(\begin{bmatrix}\alpha\,\mathrm{vol}(G_{\mathrm{ww}})(\boldsymbol{D}_{\mathrm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

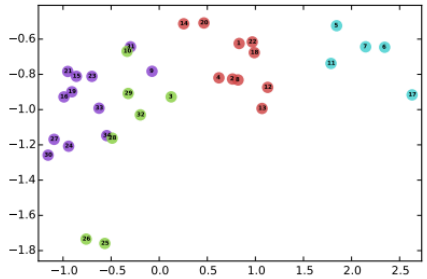
$\boldsymbol{A}$: $\boldsymbol{A}\in\mathbb{R}_{+}^{|V|\times|V|}$ is G's adjacency matrix with $\boldsymbol{A}_{i,j}$ as the edge weight between vertices i and j;
$\boldsymbol{D}_{\mathrm{col}}$: $\boldsymbol{D}_{\mathrm{col}}=\mathrm{diag}(\boldsymbol{A}^{\top}\boldsymbol{e})$ is the diagonal matrix 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

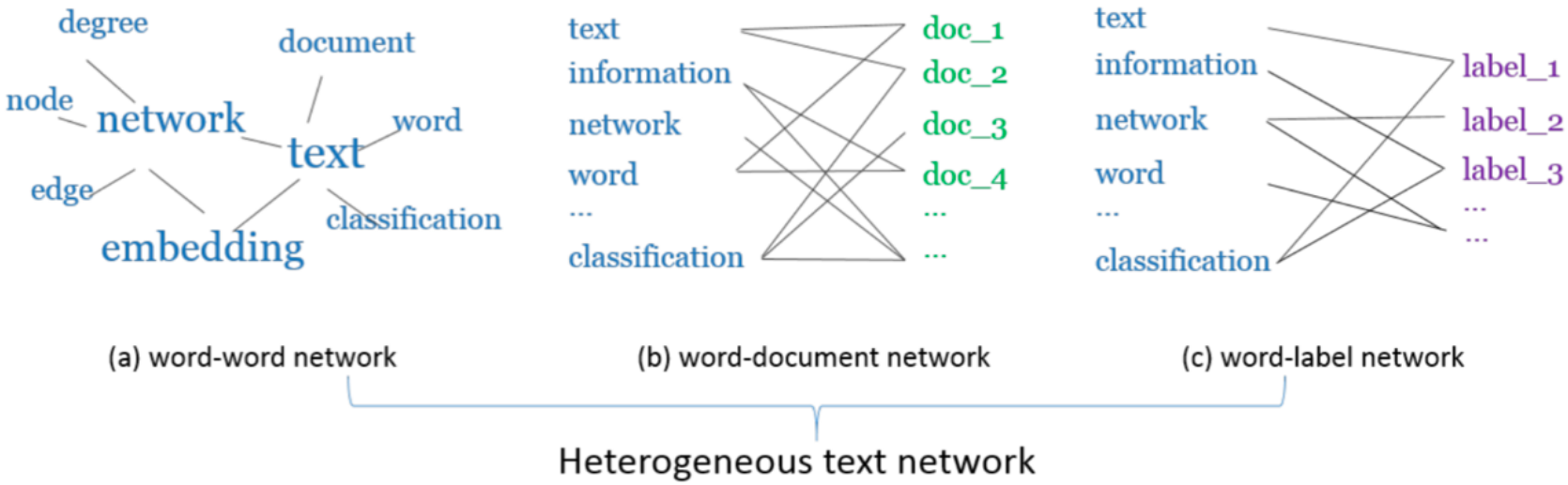
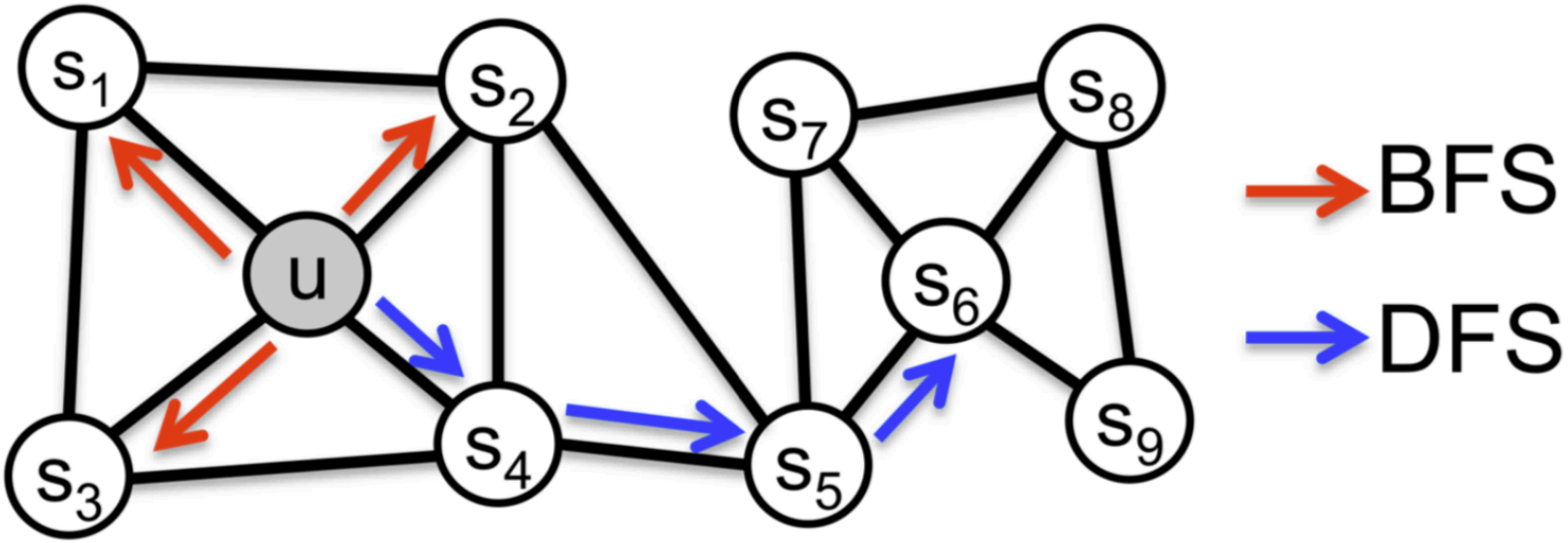


Figure 2: Heterogeneous Text Network.

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

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.

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

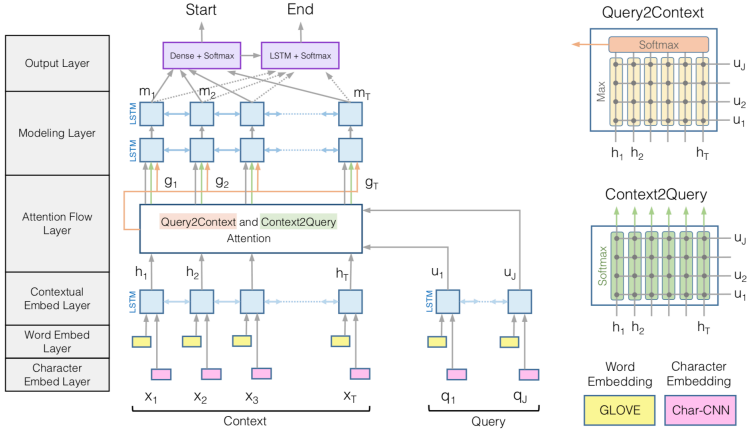
Algorithm	Closed Matrix Form
DeepWalk	$\log\left(\mathrm{vol}(G)\left(\frac{1}{T}\sum_{r=1}^{T}(\boldsymbol{D}^{-1}\boldsymbol{A})^r\right)\boldsymbol{D}^{-1}\right)-\log b$
LINE	$\log\left(\mathrm{vol}(G)\boldsymbol{D}^{-1}\boldsymbol{A}\boldsymbol{D}^{-1}\right)-\log b$
PTE	$\log\left(\begin{bmatrix}\alpha\,\mathrm{vol}(G_{\mathrm{ww}})(\boldsymbol{D}_{\mathrm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

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**

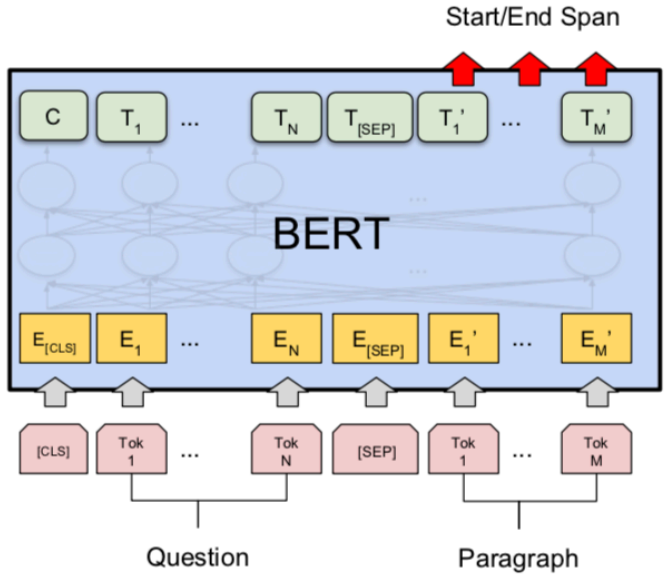
Recent Progress: BIDAF, BERT

- reading comprehension: target at understanding the whole paragraph

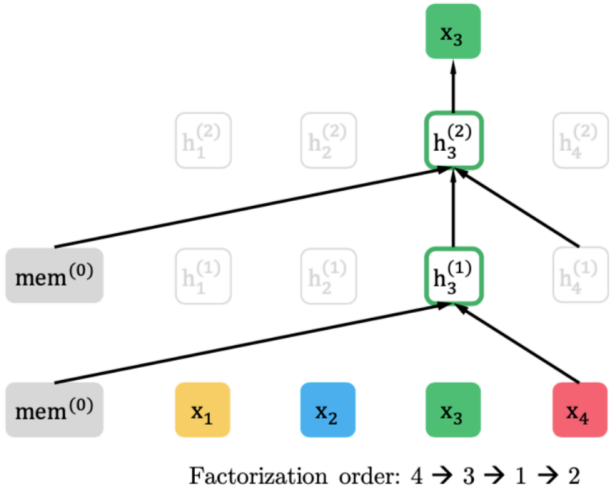
BiDAF



BERT

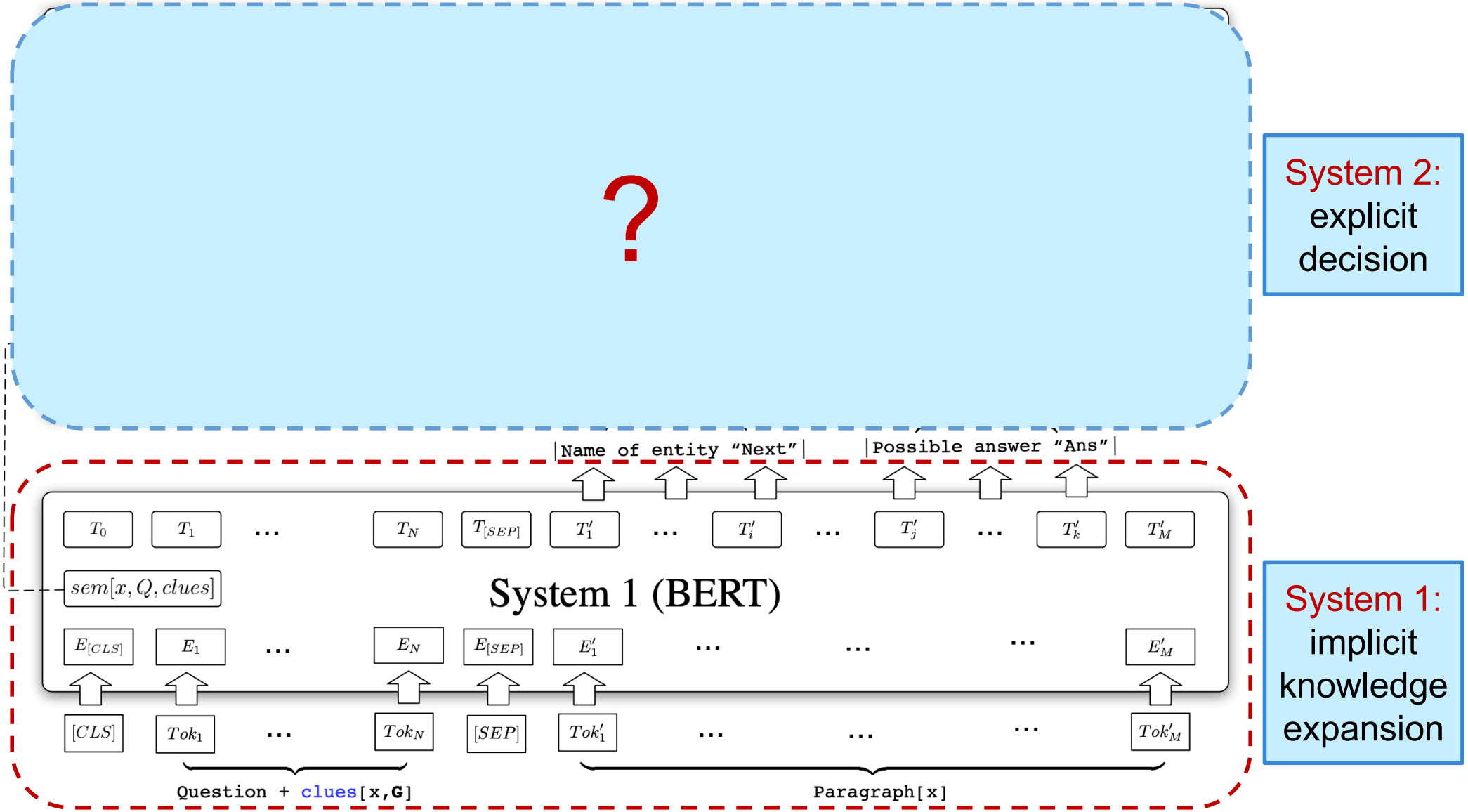


XLNet



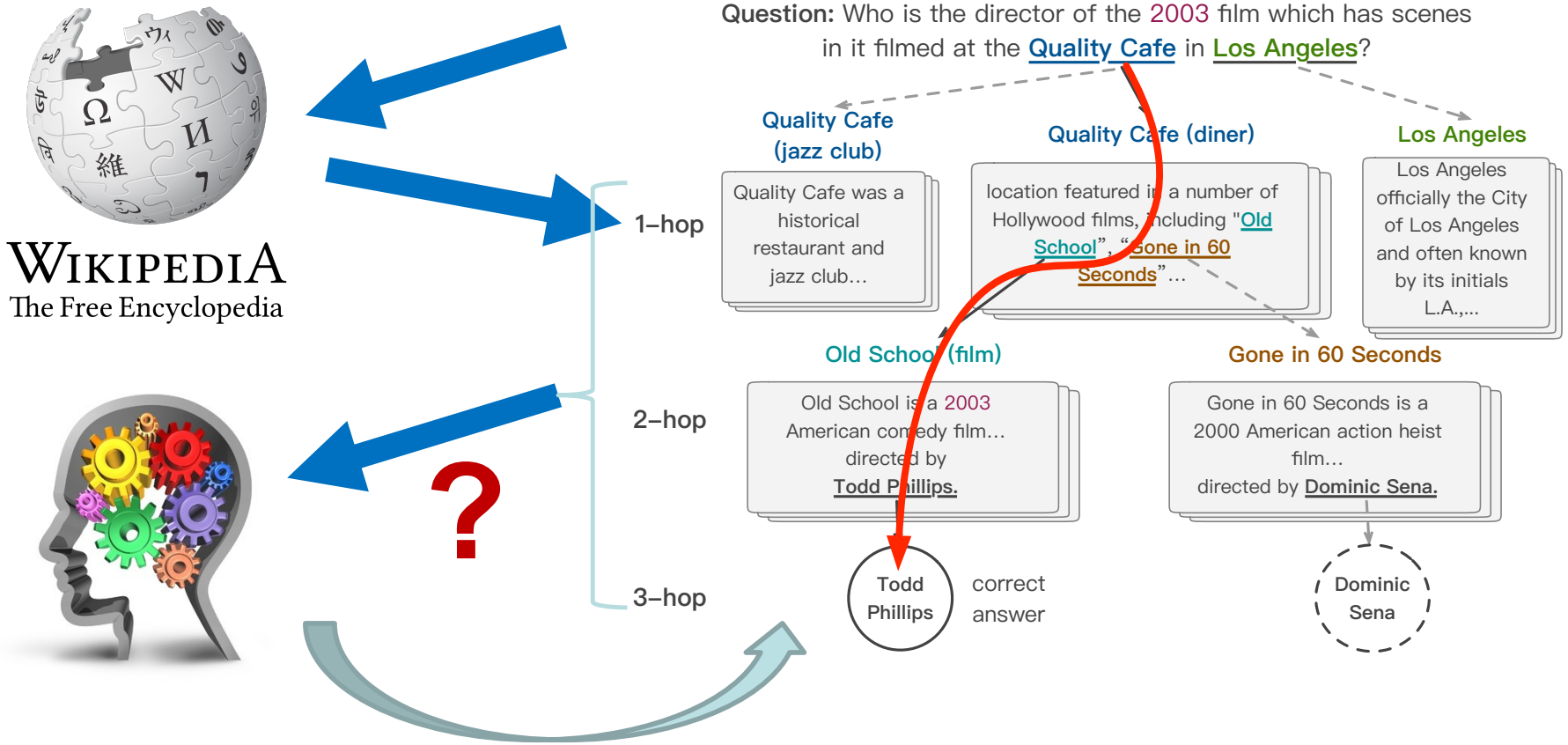
1. Rajpurkar, Pranav, et al. "SQuAD: 100,000+ Questions for Machine Comprehension of Text." EMNLP. 2016.

Cognitive Graph: DL + Dual Process Theory



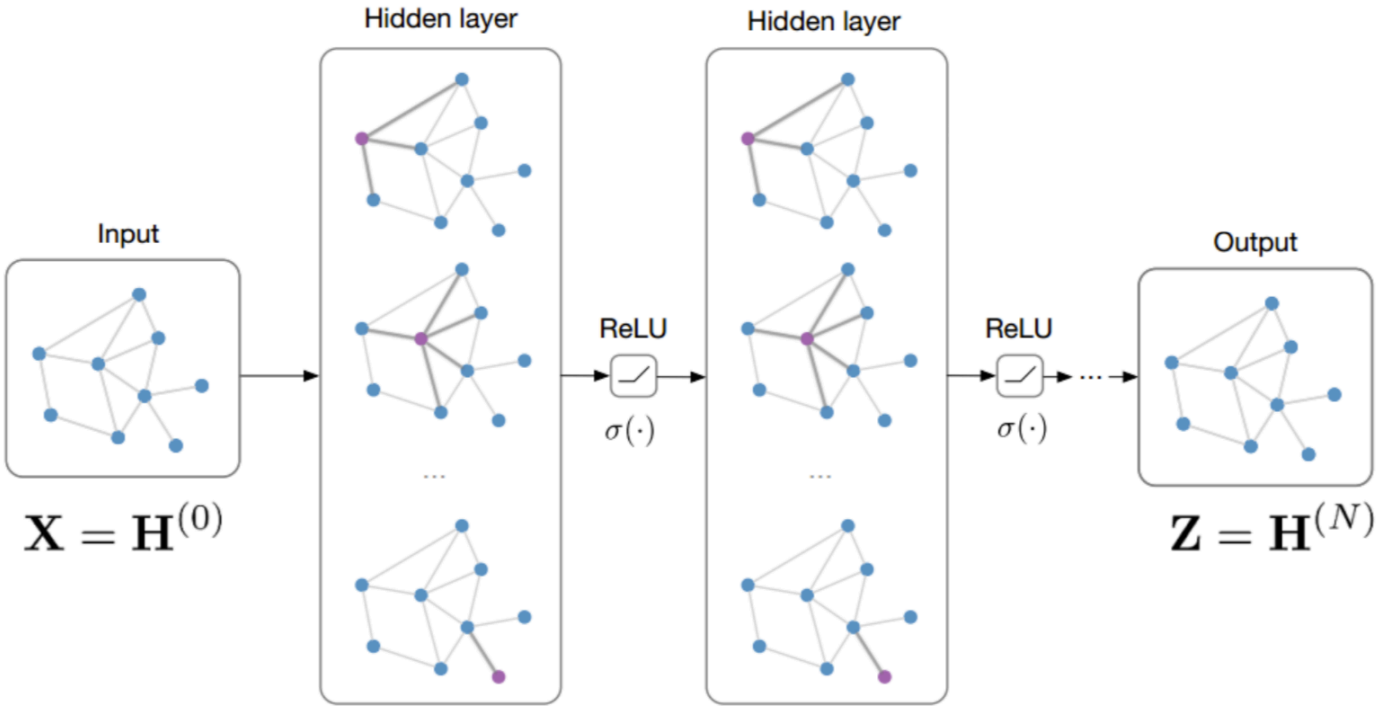
1. M. Ding, C. Zhou, Q. Chen, H. Yang, and J. Tang. Cognitive Graph for Multi-Hop Reading Comprehension at Scale. ACL'19.

Cognitive Graph: Representation, Reasoning, and Decision



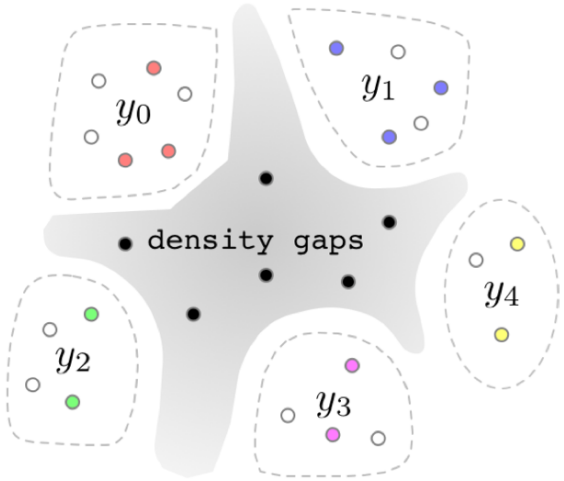
GNN/GCN Model Architecture

- **Input**: preprocess adjacency matrix $\hat{\mathbf{A}}$ and feature matrix $\mathbf{X} \in R^{N \times E}$



- where $\hat{\mathbf{A}} = \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$, $\tilde{A} = A + I_N$ is the adjacency matrix of graph G with added self-connections and $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$

Recent GCN Research



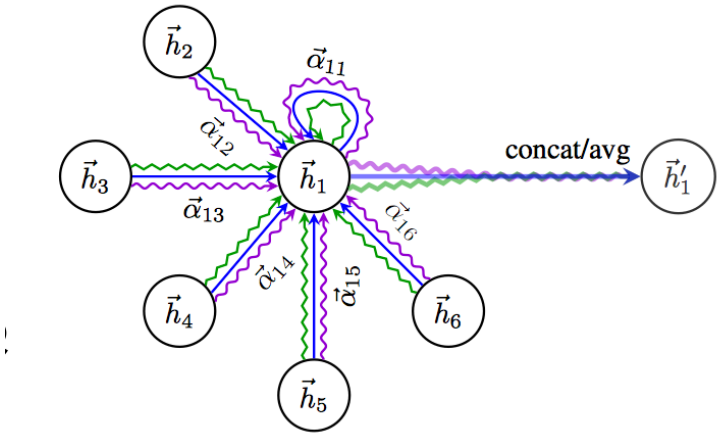
2018 GraphSGAN

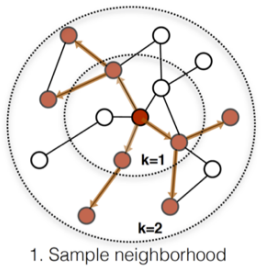
FastGCN 2018

2018 GAT

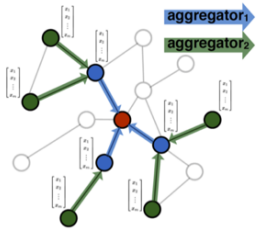
GraphSAGE 2017

2017 GCN

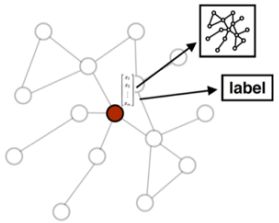




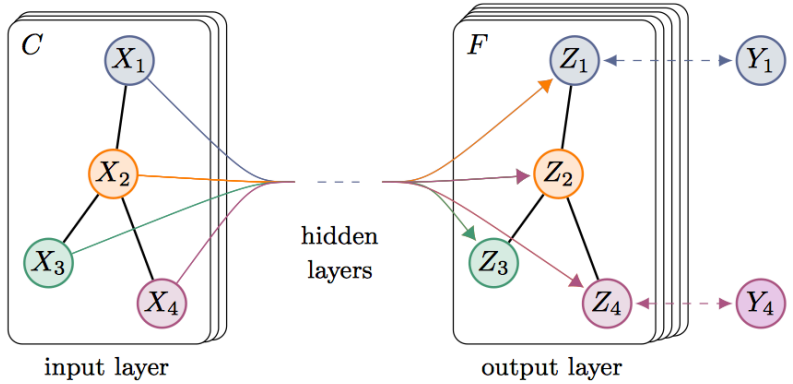
1. Sample neighborhood



2. Aggregate feature information from neighbors



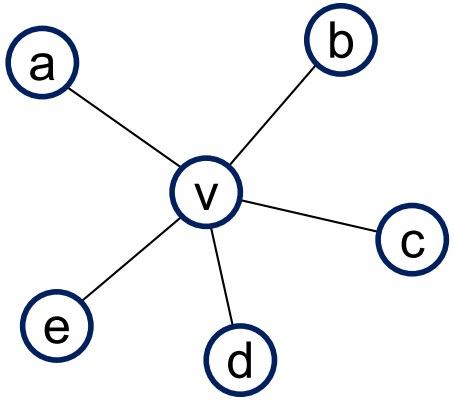
3. Predict graph context and label using aggregated information



input layer

output layer

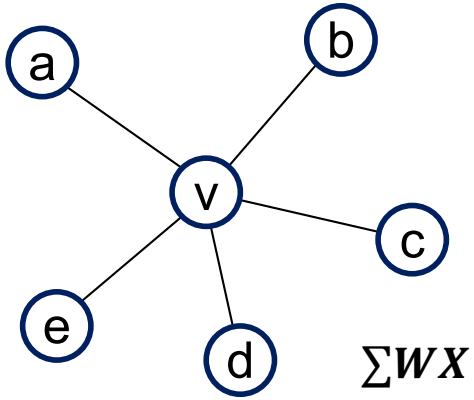
The Core of Graph Neural Networks



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

- **Neighborhood Aggregation:**
  - Aggregate neighbor information and pass into a neural network

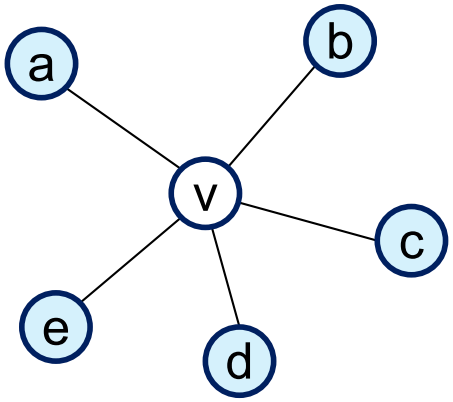
Graph Neural Networks



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

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

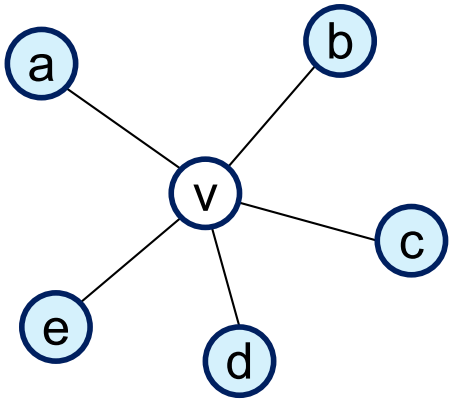
GNN: Graph Convolutional Networks



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

1. Kipf et al. Semi-supervised Classification with Graph Convolutional Networks. ICLR 2017

GNN: Graph Convolutional Networks



parameters in layer k

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

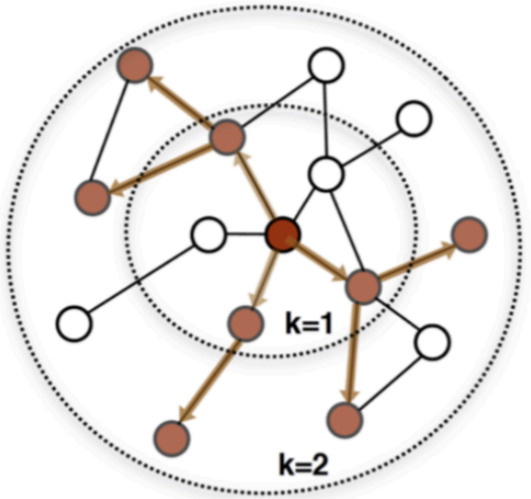
$$\boldsymbol{h}_v^k = \sigma\left(\boldsymbol{W}_k \sum_{u \in \mathrm{N}(v) \cup v} \frac{h_u^{k-1}}{\sqrt{|N(u)||N(v)|}}\right)$$

node v's embedding at layer k

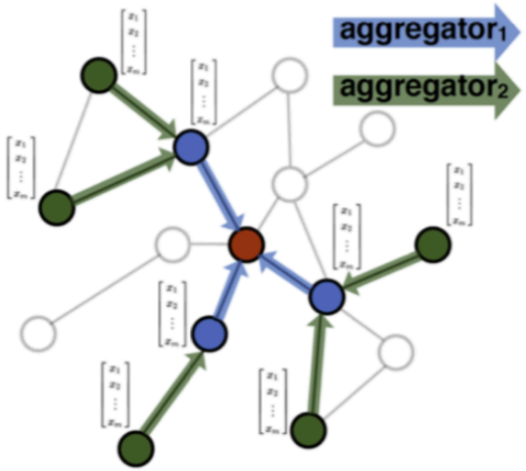
the neighbors of node v

GraphSAGE Model Arthitecture

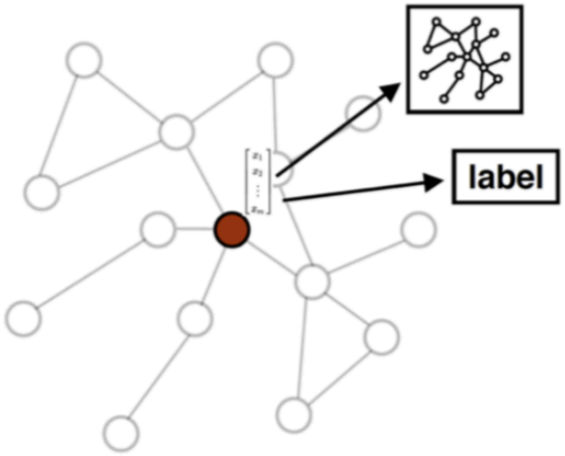
- Idea: Node's neighborhood defines a computation graph.
- Learn how to propagate information across the graph to compute node features



1. Sample neighborhood



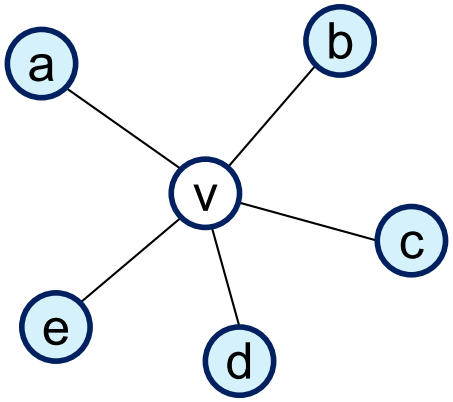
2. Aggregate feature information from neighbors



3. Predict graph context and label using aggregated information

1. Hamilton et al. Inductive Representation Learning on Large Graphs. NIPS 2017

GraphSage



GCN

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

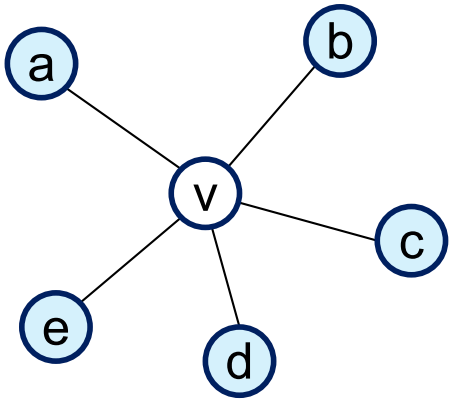
GraphSage

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

Generalized aggregation: any differentiable function that maps set of vectors to a single vector

1. Hamilton et al. Inductive Representation Learning on Large Graphs. NIPS 2017
2. Slide snipping from "Hamiltion & Tang, AAAI 2019 Tutorial on Graph Representation Learning"

GraphSage



GCN

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

GraphSage

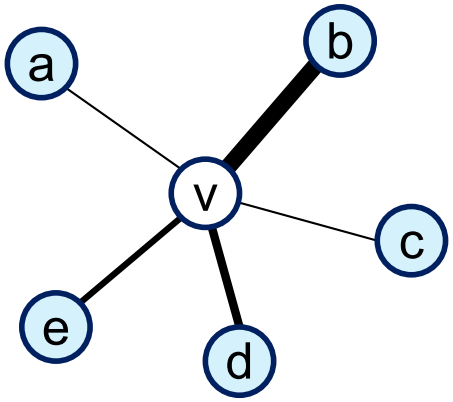
Instead of summation, it concatenate neighbor & self embeddings

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

Generalized aggregation: any differentiable function that maps set of vectors to a single vector

1. Hamilton et al. Inductive Representation Learning on Large Graphs. NIPS 2017
2. Slide snipping from "Hamiltion & Tang, AAAI 2019 Tutorial on Graph Representation Learning"

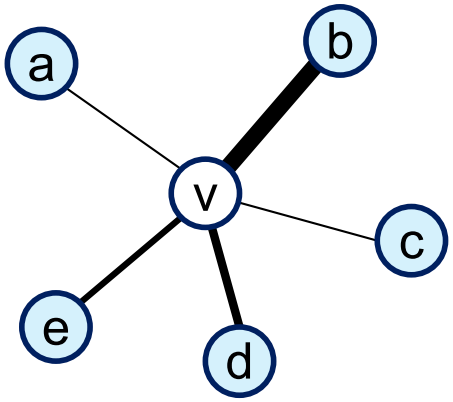
Graph Attention Networks



Realistically, neighbors play different influences

1. Velickovic et al. Graph Attention Networks. ICLR 2018

Graph Attention Networks



GCN

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

Graph Attention

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

Learned attention weights

1. Velickovic et al. Graph Attention Networks. ICLR 2018

GAT Results

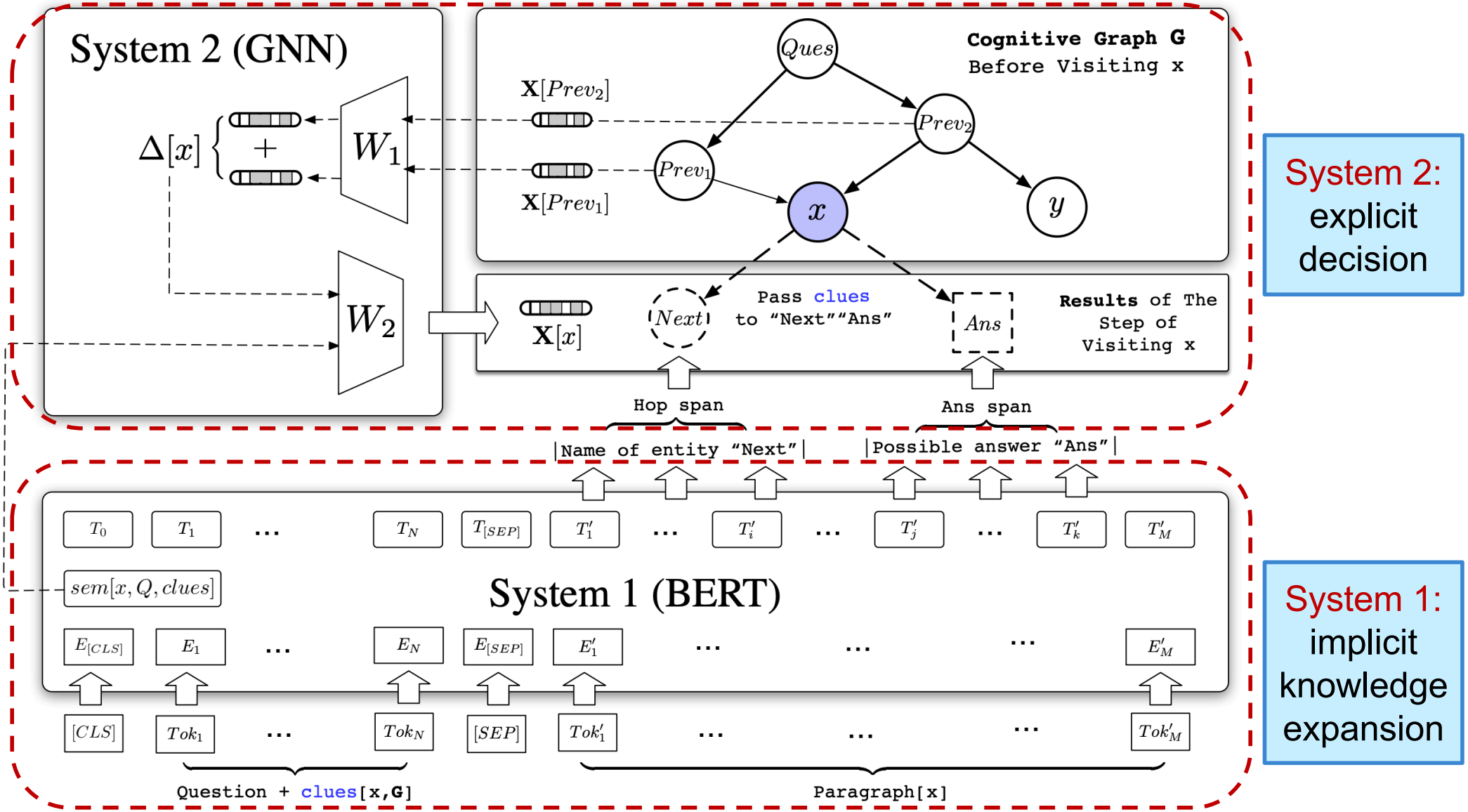
Transductive

Method	Cora	Citeseer	Pubmed
MLP	55.1%	46.5%	71.4%
ManiReg (Belkin et al., 2006)	59.5%	60.1%	70.7%
SemiEmb (Weston et al., 2012)	59.0%	59.6%	71.7%
LP (Zhu et al., 2003)	68.0%	45.3%	63.0%
DeepWalk (Perozzi et al., 2014)	67.2%	43.2%	65.3%
ICA (Lu & Getoor, 2003)	75.1%	69.1%	73.9%
Planetoid (Yang et al., 2016)	75.7%	64.7%	77.2%
Chebyshev (Defferrard et al., 2016)	81.2%	69.8%	74.4%
GCN (Kipf & Welling, 2017)	81.5%	70.3%	**79.0%**
MoNet (Monti et al., 2016)	81.7 ± 0.5%	—	78.8 ± 0.3%
GCN-64*	81.4 ± 0.5%	70.9 ± 0.5%	**79.0** ± 0.3%
GAT (ours)	**83.0** ± 0.7%	**72.5** ± 0.7%	**79.0** ± 0.3%

Inductive

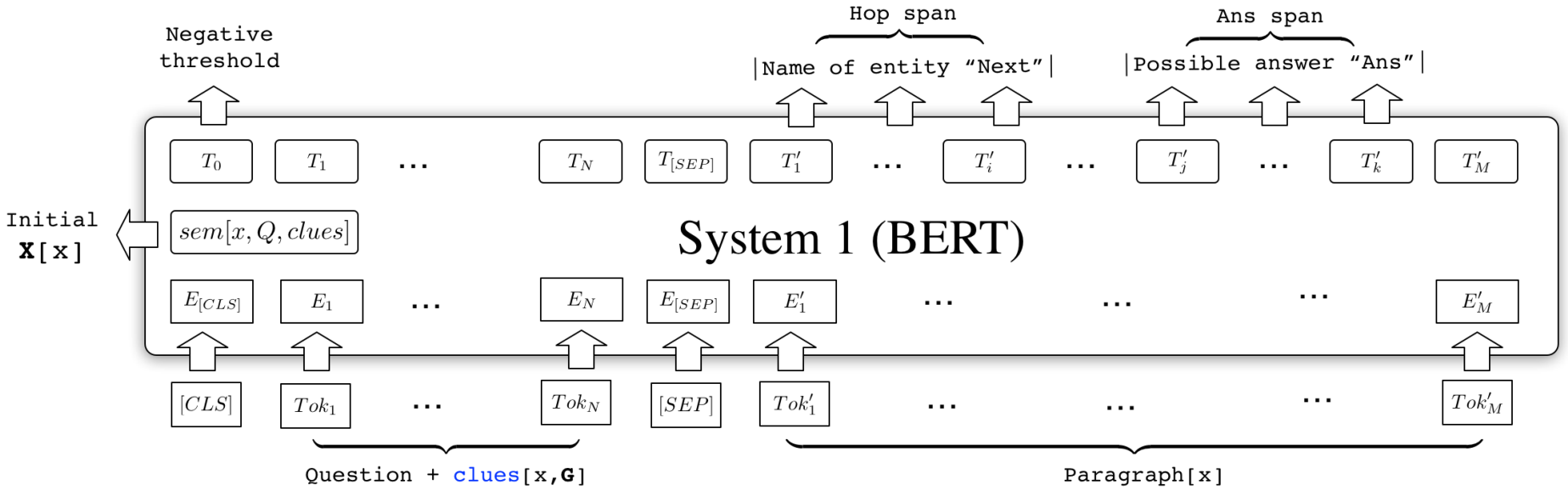
Method	PPI
Random	0.396
MLP	0.422
GraphSAGE-GCN (Hamilton et al., 2017)	0.500
GraphSAGE-mean (Hamilton et al., 2017)	0.598
GraphSAGE-LSTM (Hamilton et al., 2017)	0.612
GraphSAGE-pool (Hamilton et al., 2017)	0.600
GraphSAGE*	0.768
Const-GAT (ours)	0.934 ± 0.006
GAT (ours)	**0.973** ± 0.002

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



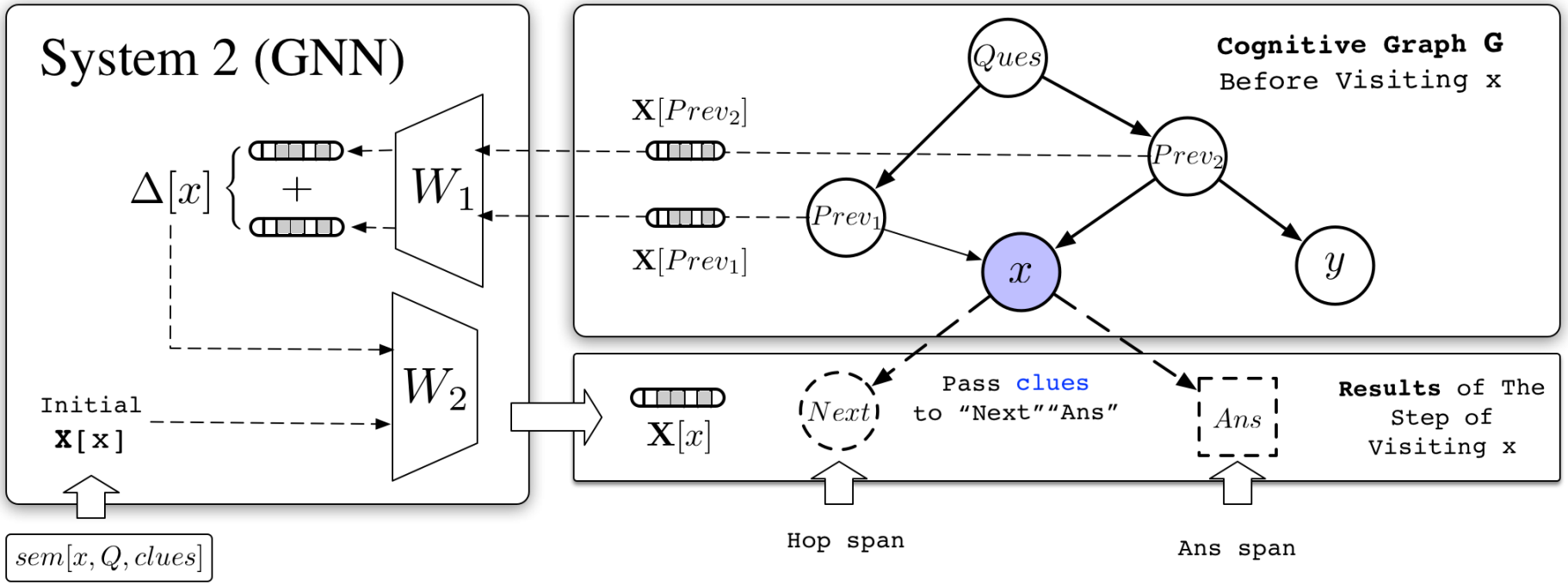
$$\underbrace{[CLS]\ Question\ [SEP]\ clues[x,\mathcal{G}]\ \ [SEP]}_{Sentence\ A}\quad\underbrace{Para[x]}_{Sentence\ B}$$

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

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

- 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 $\mathbf{X}$:

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

CogQA Algorithm and Training

Algorithm 1: Cognitive Graph QA

Input:
System 1 model $\mathcal{S}_1$, System 2 model $\mathcal{S}_2$,
Question Q, Predictor $\mathcal{F}$,Wiki Database $\mathcal{W}$

Initialize cognitive graph $\mathcal{G}$ with entities mentioned in Q and mark them *frontier nodes*
repeat
  pop a node x from frontier nodes
  collect $clues[x, \mathcal{G}]$ from predecessor nodes of x
    // eg. $clues$ can be sentences where x is mentioned
  fetch $para[x]$ in $\mathcal{W}$ if any
  generate $sem[x, Q, clues]$ with $\mathcal{S}_1$ // initial $\mathbf{X}[x]$
  if *x is a hop node* **then**
    find hop and answer spans in $para[x]$ with $\mathcal{S}_1$
    for *y in hop spans* **do**
      if *$y \notin \mathcal{G}$ and $y \in \mathcal{W}$* **then**
        create a new hop node for y
      if *$y \in \mathcal{G}$ and $edge(x, y) \notin \mathcal{G}$* **then**
        add edge (x, y) to $\mathcal{G}$
        mark node y as a frontier node
    end
    for *y in answer spans* **do**
      add new answer node y and edge (x, y) to $\mathcal{G}$
    end
  end
  update hidden representation $\mathbf{X}$ with $\mathcal{S}_2$
until *there is no frontier node in $\mathcal{G}$ or $\mathcal{G}$ is large enough*;
Return $\arg\max_{\text{answer node } x} \mathcal{F}(\mathbf{X}[x])$

- Preprocess
  - Mark the spans of next-hop entities and answer by fuzzy matching
  - Golden cognitive graph and negative (hop and span) nodes
- Train Task #1 (System 1):
  - Next-hop spans and answer spans extraction
- Train Task #2 (System 1 & 2):
  - Prediction with GNN

Performance

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

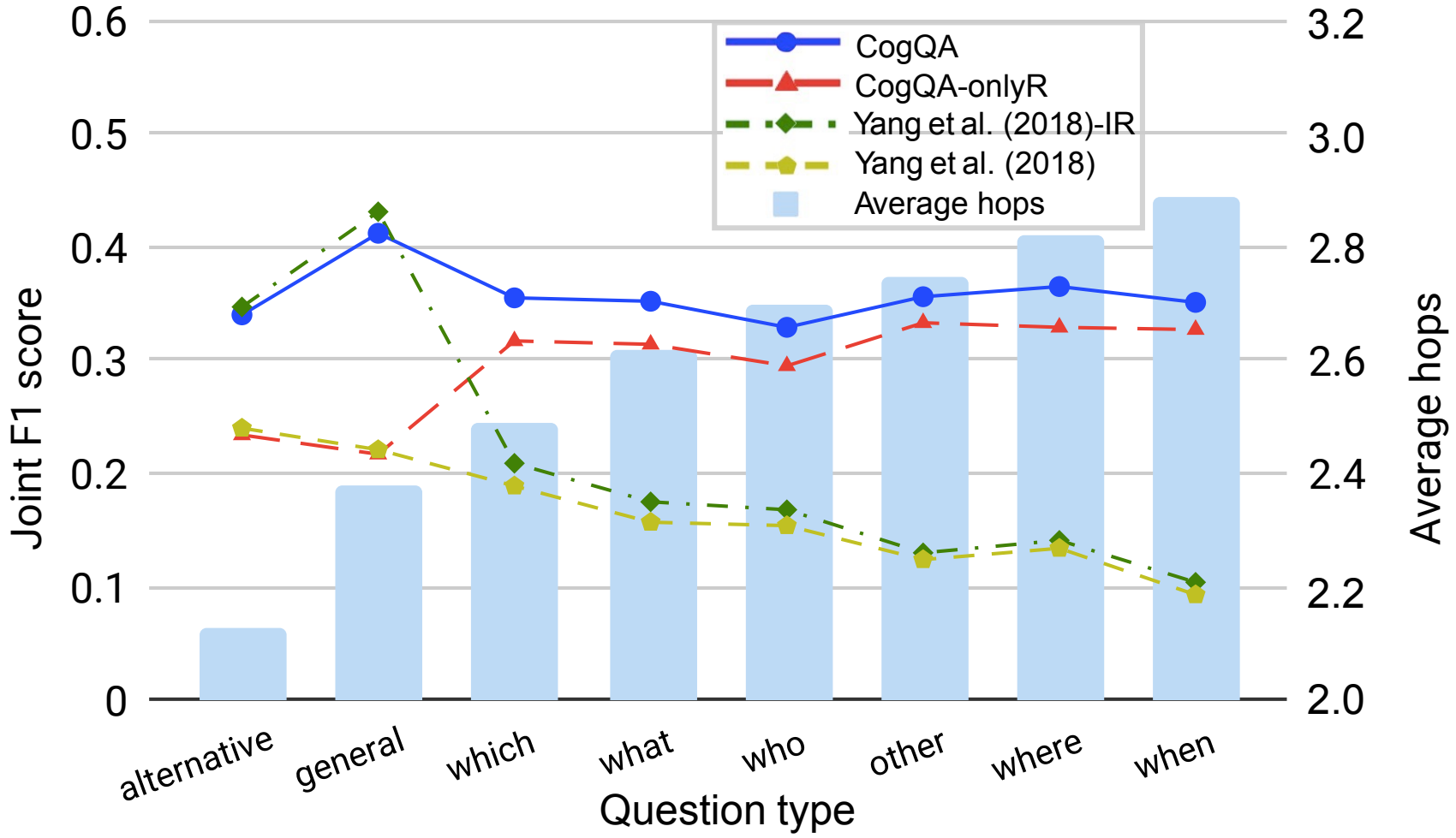
	Model	Ans				Sup				Joint			
		EM	F_1	Prec	Recall	EM	F_1	Prec	Recall	EM	F_1	Prec	Recall
Dev	Yang et al. (2018)	23.9	32.9	34.9	33.9	5.1	40.9	47.2	40.8	2.5	17.2	20.4	17.8
	Yang et al. (2018)-IR	24.6	34.0	35.7	34.8	10.9	49.3	52.5	52.1	5.2	21.1	22.7	23.2
	BERT	22.7	31.6	33.4	31.9	6.5	42.4	54.6	38.7	3.1	17.8	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.

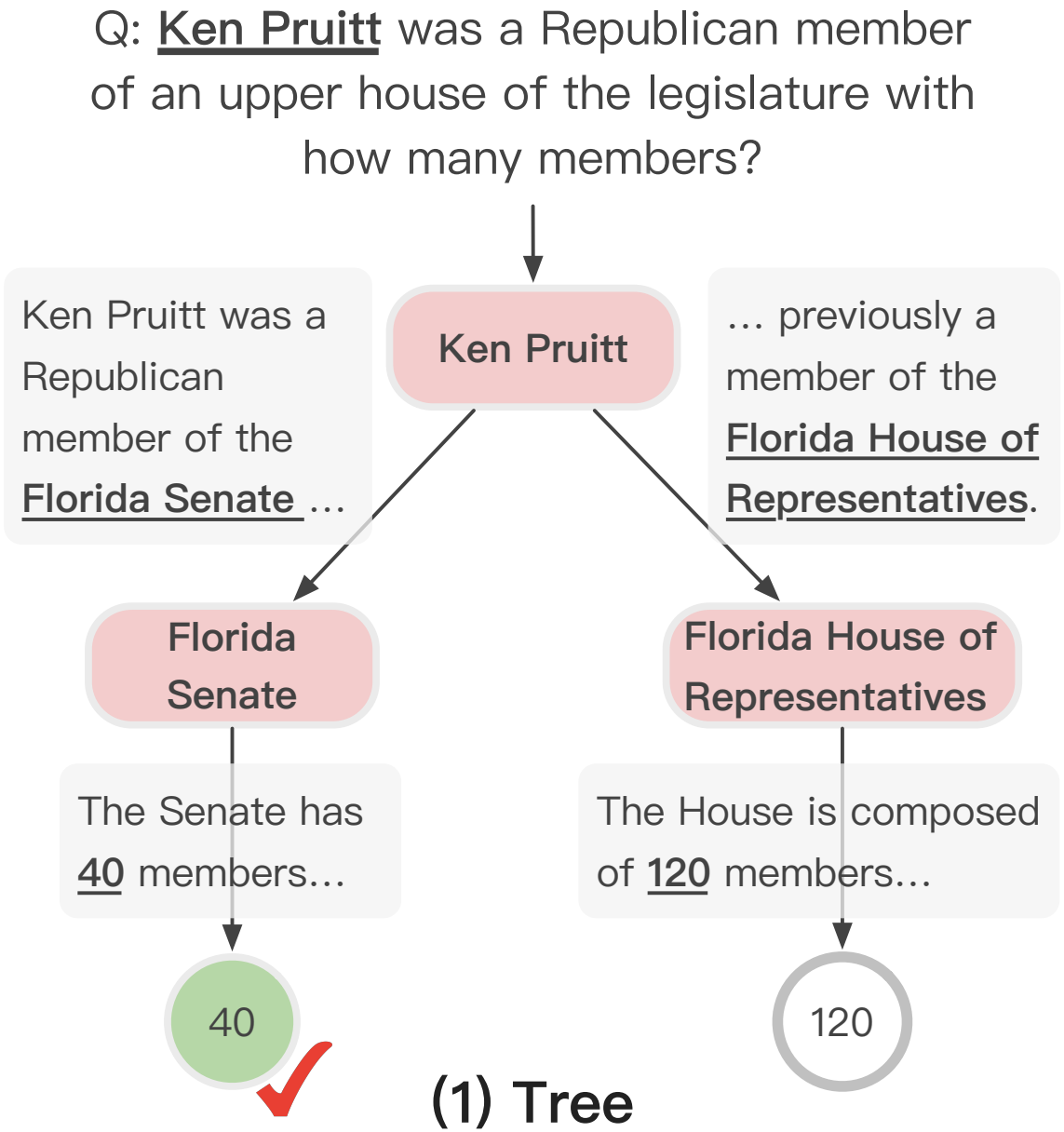
** Code available at https://github.com/THUDM/CogQA

Reasoning Power

CogQA Performs much better on question with more hops !



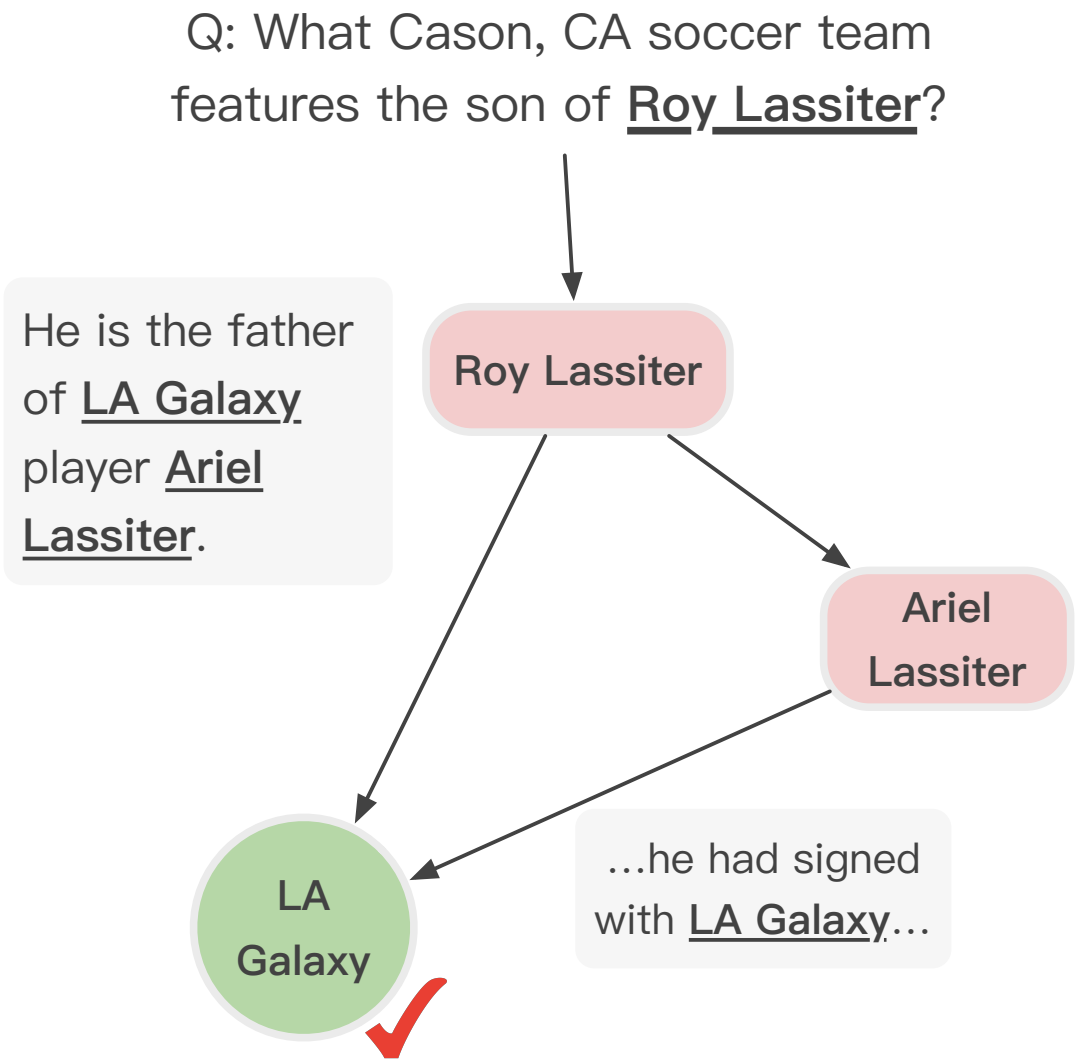
Case Study



(1) Tree

- 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*"

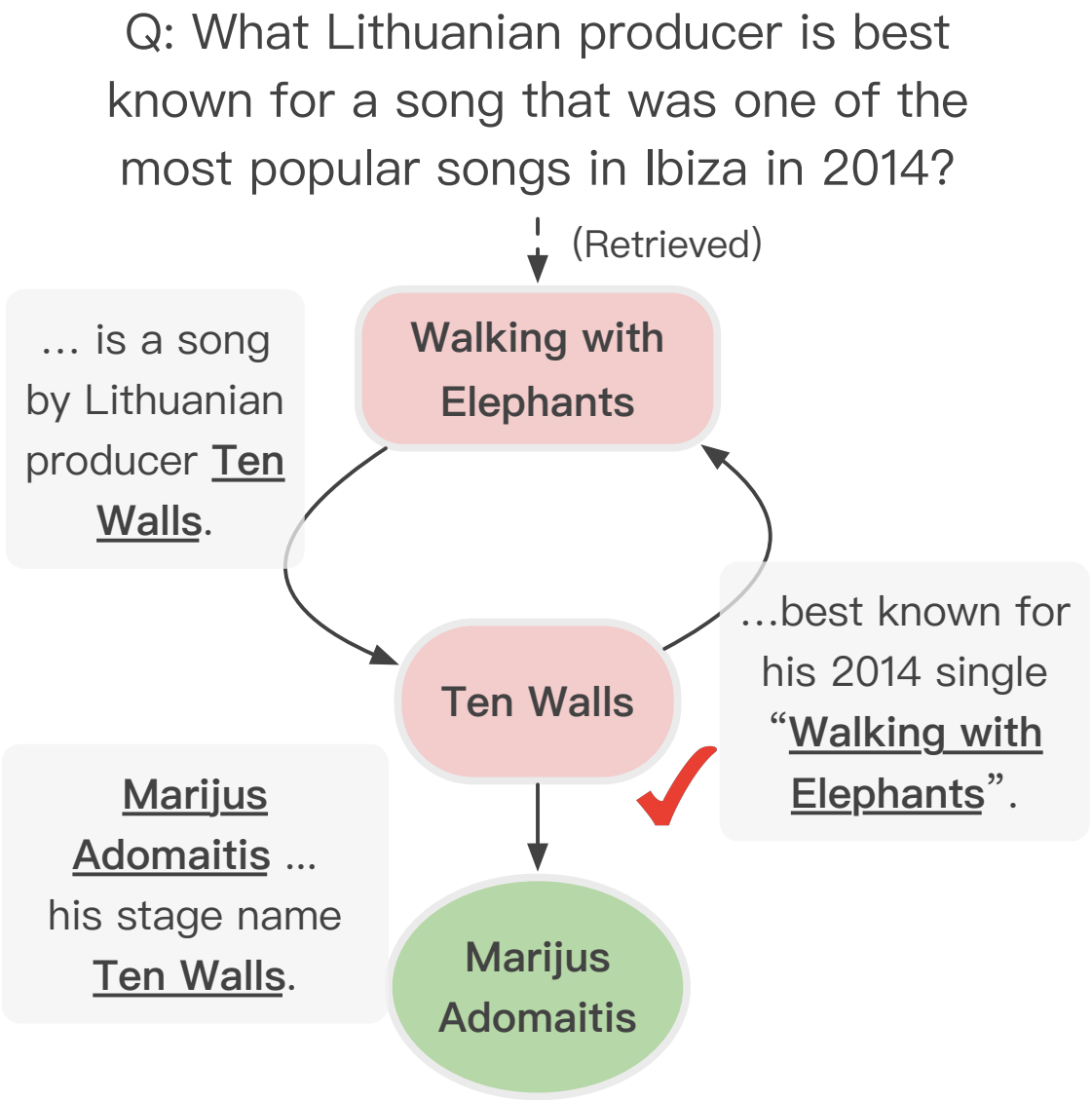
Case Study



(2) DAG

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

Case Study



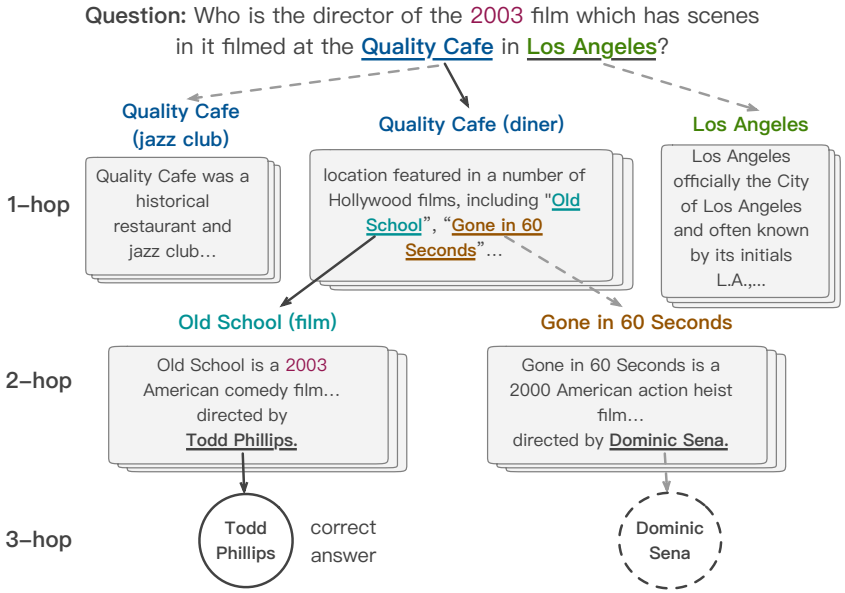
(3) Cyclic Graph

- 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.

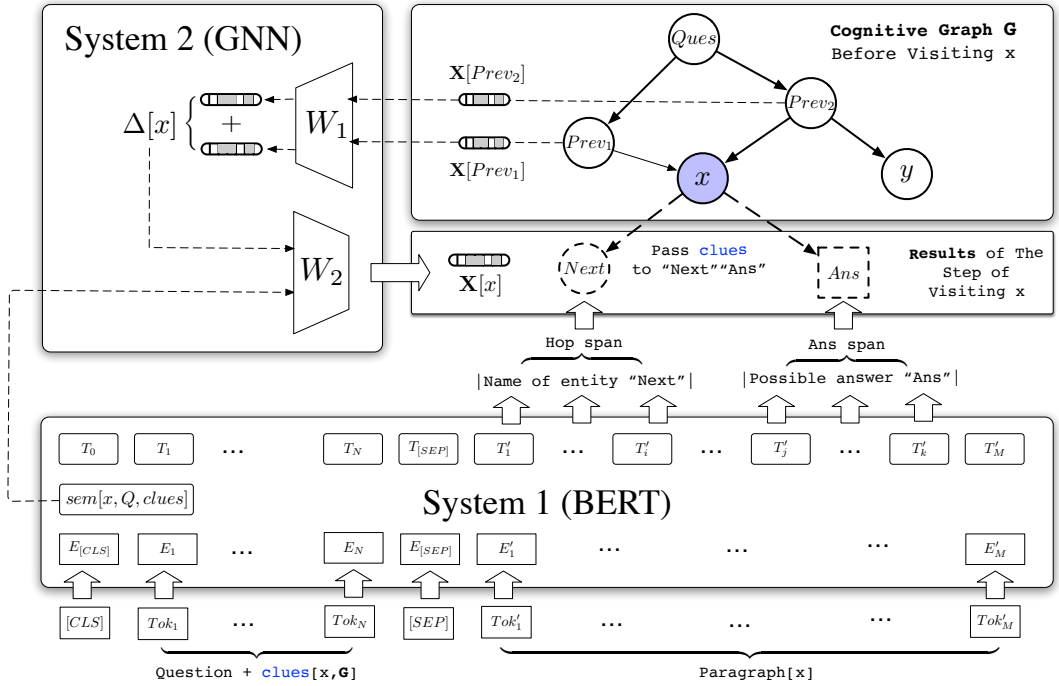
Summary

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

Cognitive graph

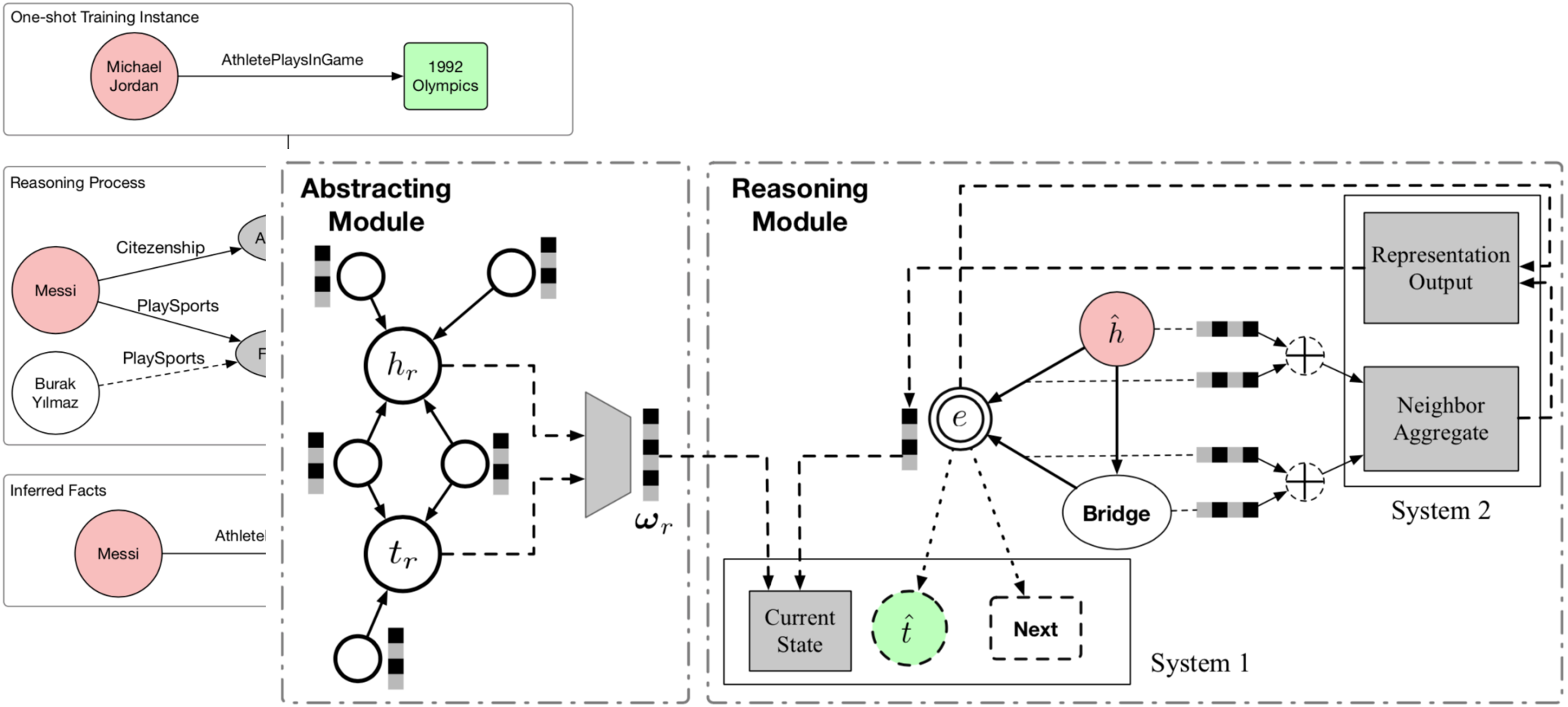


CogQA framework



More Applications: KG completion

- Completing knowledge graph with cognitive graph



Related Publications

- 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.
- 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.
- 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.

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

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