Beyond GNN
—图神经网络及认知推理

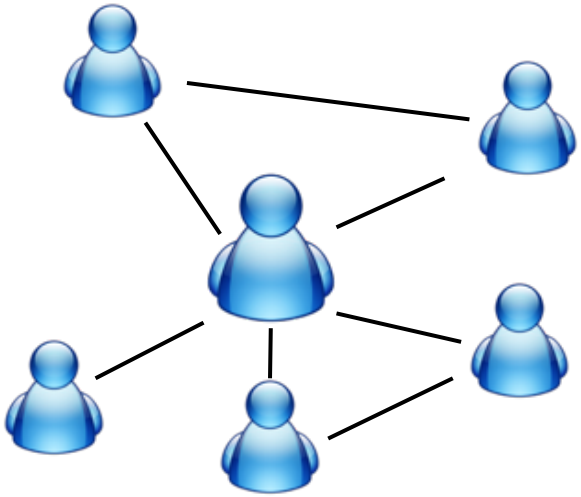
Jie Tang
Computer Science
Tsinghua University

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

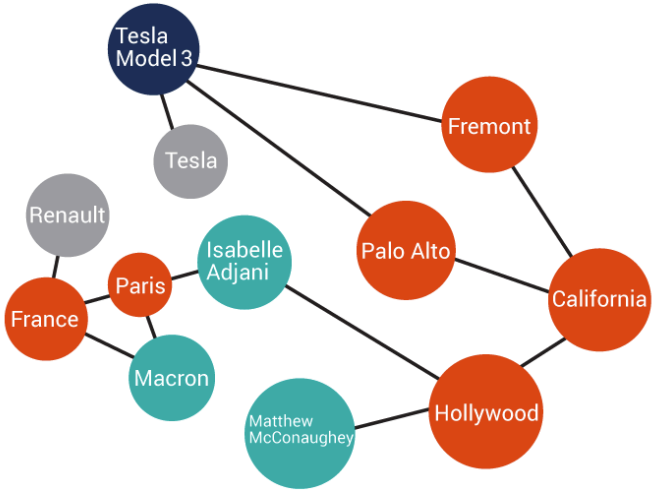
AI Open

- 人工智能研究关乎开放和速度，如果你不与别人分享，你的成果可能毫无意义
  - 《Nature》采访
- 感谢邀请到这里来“开放共享”

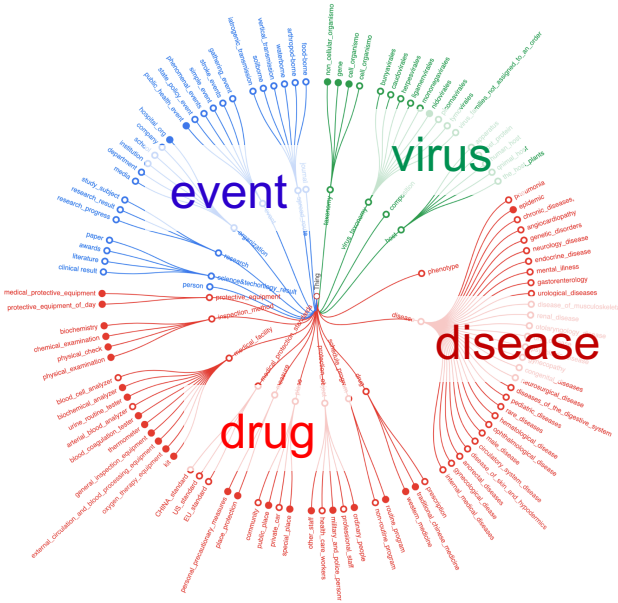
Networked World



Social Network



Knowledge Graph



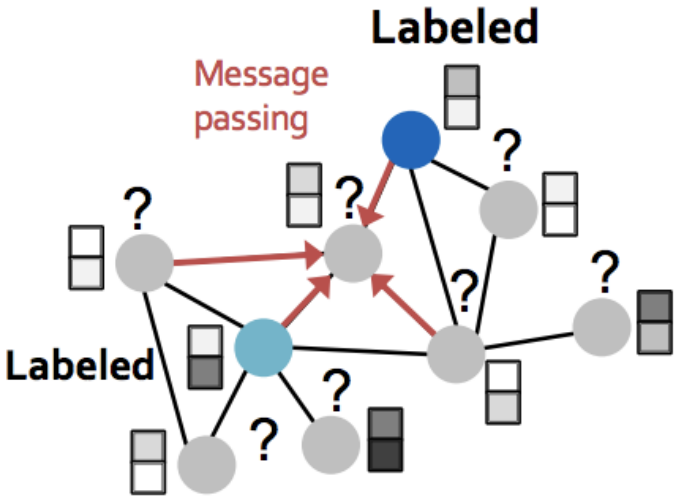
COVID Graph

图数据上的机器学习问题

- 图数据上的机器学习任务:
  - 节点分类
    - Predict a type of a given node
  - 链接预测
    - Predict whether two nodes are linked
  - 子图聚类
    - Identify densely linked clusters of nodes
  - 子图相似
    - How similar are two (sub)graphs?

Setting: Semi-supervised Learning on Graphs

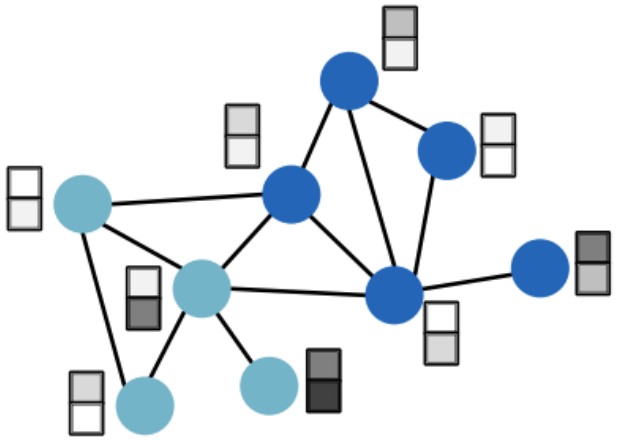
Input: Partially labeled attributed graph



?: Unlabeled

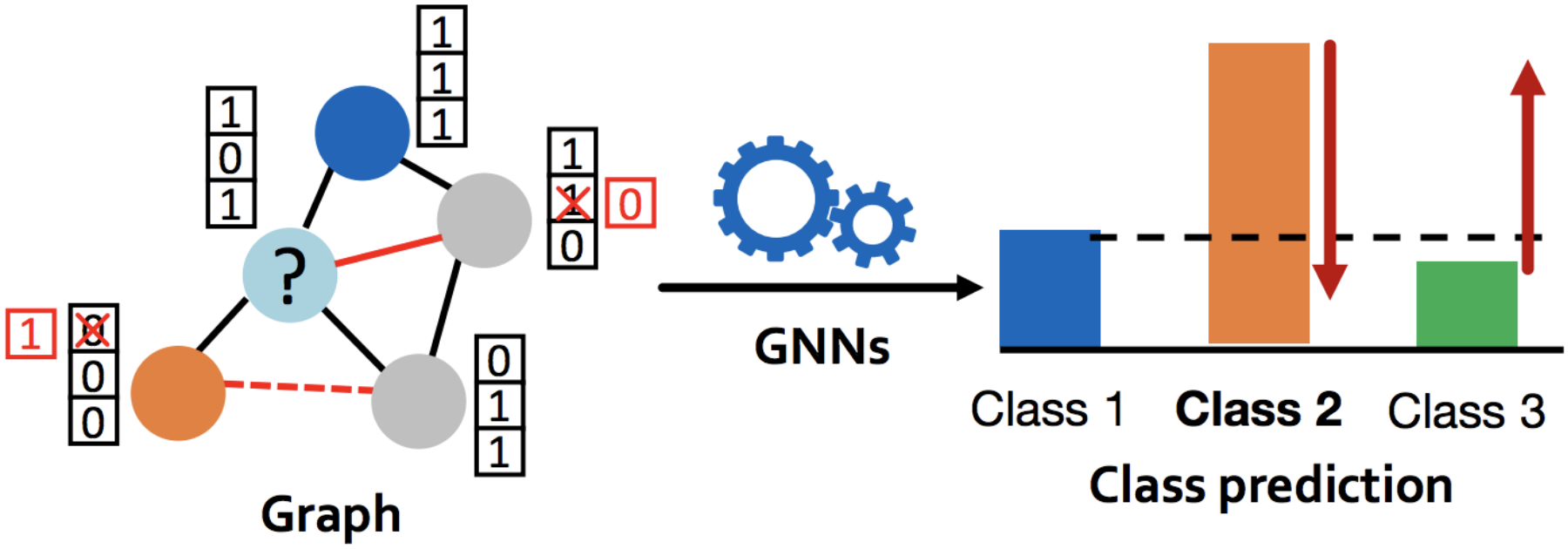
GCN

Goal: Predict labels of unlabeled nodes



Challenges: non-robust

- $\mathbf{H}^{(l+1)} = \sigma(\hat{\mathbf{A}}\mathbf{H}^{(l)}\mathbf{W}^{(l)})$: Deterministic propagation
- Nodes are highly dependent with its neighborhoods making GNNs non-robust.

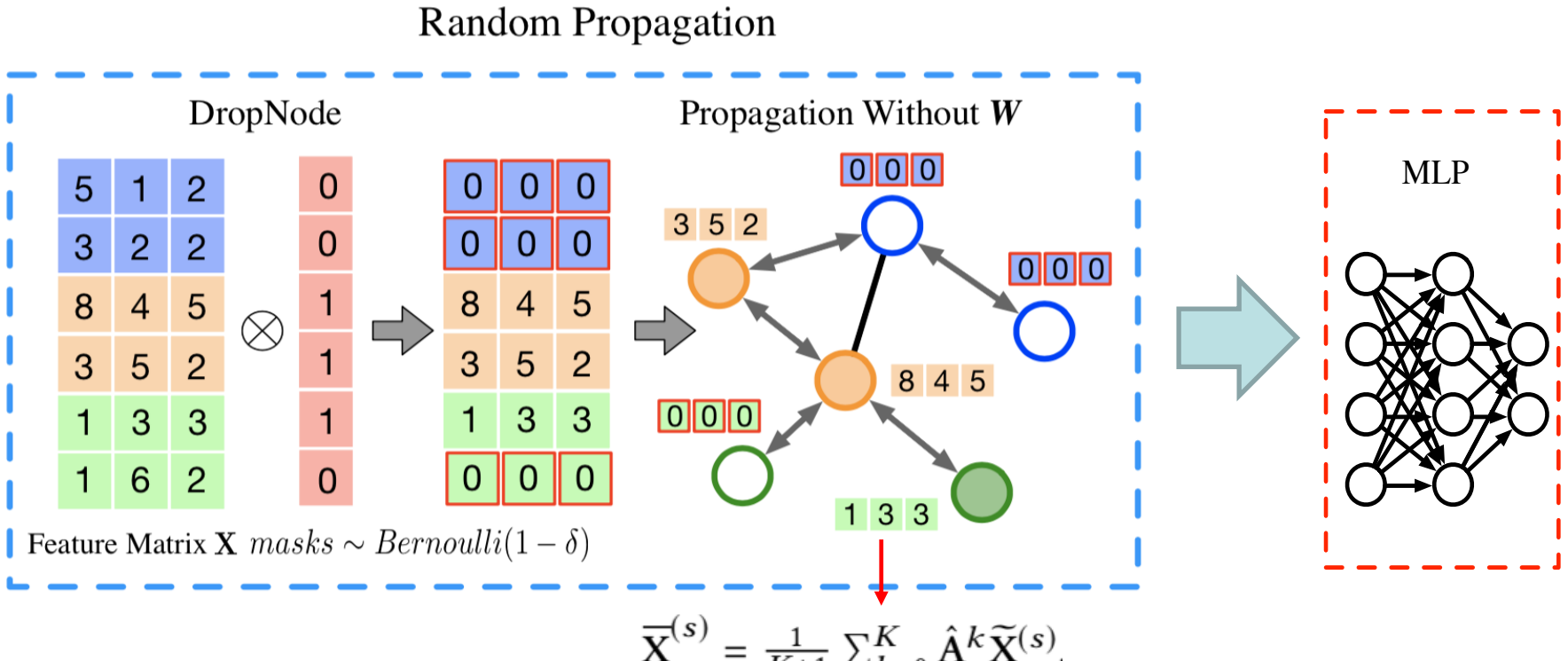


Motivation: some problems of GCN

- $\mathbf{H}^{(l+1)} = \sigma(\hat{\mathbf{A}}\mathbf{H}^{(l)}\mathbf{W}^{(l)})$: Propagation is coupled with transformation.
- Stacking many layers may cause over-fitting and over-smoothing.

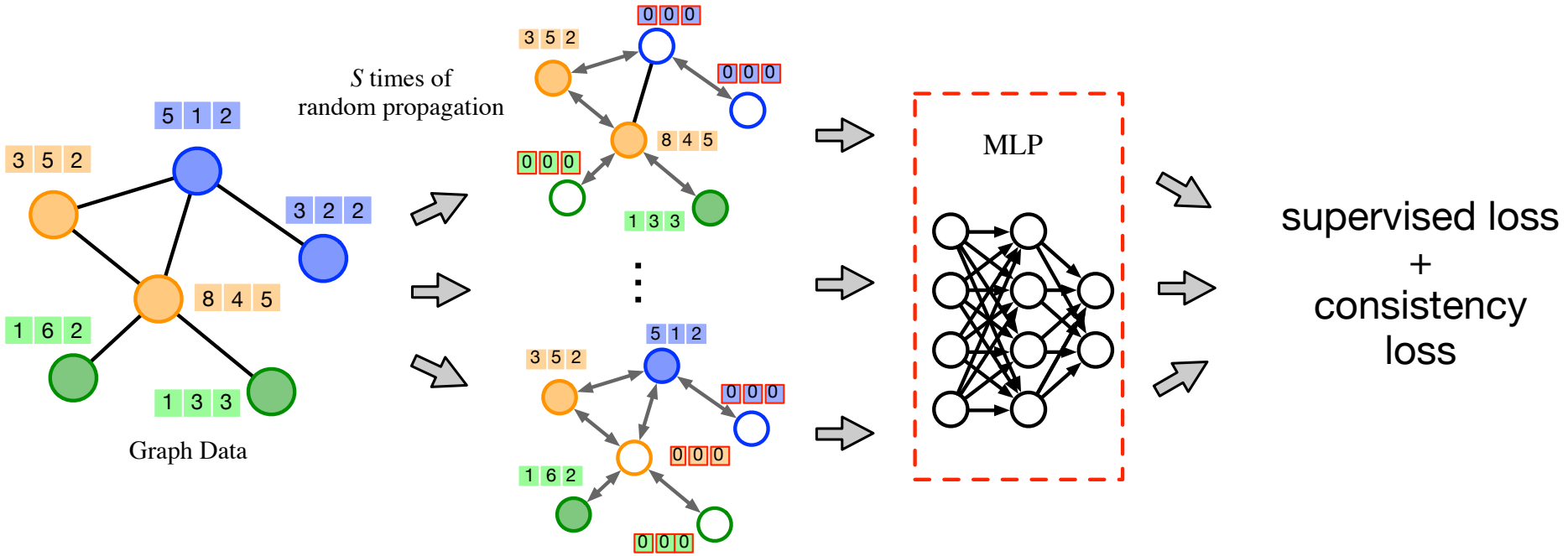
Grand: Graph Random Neural Network

- Random Propagation (DropNode + Propagation):
  - Each node is enabled to be not sensitive to specific neighborhoods.
  - Decouple propagation from feature transformation.



Graph Random Neural Network(Grand)

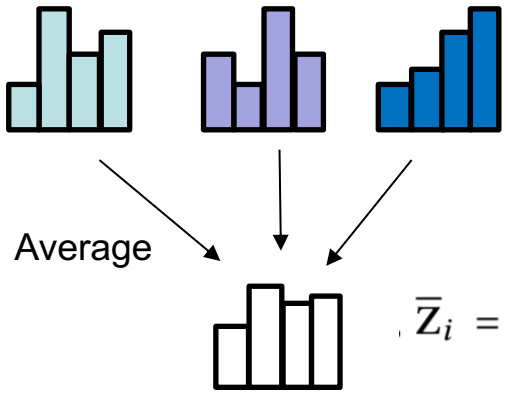
- Consistency Regularized Training:
  - Random propagation serves as graph data augmentation
  - Consistency Regularization: Optimizing the consistency among *S* augmentations of the graph.



Grand training details

Consistency Loss：

Distributions of a node:



$$\overline{\mathbf{Z}}_i = \frac{1}{S}\sum_{s=1}^{S}\widetilde{\mathbf{Z}}_i^{(s)}$$

$$\overline{\mathbf{Z}}'_{ik} = \overline{\mathbf{Z}}_{ik}^{\frac{1}{T}} \Big/ \sum_{j=0}^{C-1}\overline{\mathbf{Z}}_{ij}^{\frac{1}{T}}$$

$$\mathcal{L}_{con} = \frac{1}{S}\sum_{s=1}^{S}\sum_{i=0}^{n-1}\mathcal{D}(\overline{\mathbf{Z}}'_i, \widetilde{\mathbf{Z}}_i^{(s)}).$$

Algorithm 2 Consistency Regularized Training for GRAND

Input:

Adjacency matrix $\hat{\mathbf{A}}$, feature matrix $\mathbf{X} \in \mathbb{R}^{n\times d}$, times of augmentations in each epoch S, DropNode probability δ.

Output:

Prediction $\mathbf{Z}$.

while not convergence **do**
  for $s = 1 : S$ **do**
    Apply DropNode via Algorithm 1: $\widetilde{\mathbf{X}}^{(s)} \sim \text{DropNode}(\mathbf{X}, \delta)$.
    Perform propagation: $\overline{\mathbf{X}}^{(s)} = \frac{1}{K+1}\sum_{k=0}^{K}\hat{\mathbf{A}}^k\widetilde{\mathbf{X}}^{(s)}$.
    Predict class distribution using MLP: $\widetilde{\mathbf{Z}}^{(s)} = P(\mathbf{Y}|\overline{\mathbf{X}}^{(s)};\Theta)$.
  end for
  Compute supervised classification loss $\mathcal{L}_{sup}$ via Eq. 4 and consistency regularization loss via Eq. 6.
  Update the parameters Θ by gradients descending:

$$\nabla_\Theta \mathcal{L}_{sup} + \lambda \mathcal{L}_{con}$$

end while
Output prediction $\mathbf{Z}$ via Eq. 8.

Theoretical Analysis

- With Consistency Regularization Loss:
  - Random propagation can enforce the consistency of the classification confidence between each node and its all multi-hop neighborhoods.

- With Supervised Cross-entropy Loss:
  - Random propagation can enforce the consistency of the classification confidence between each node and its labeled multi-hop neighborhoods.

Different between DropNode and Dropout

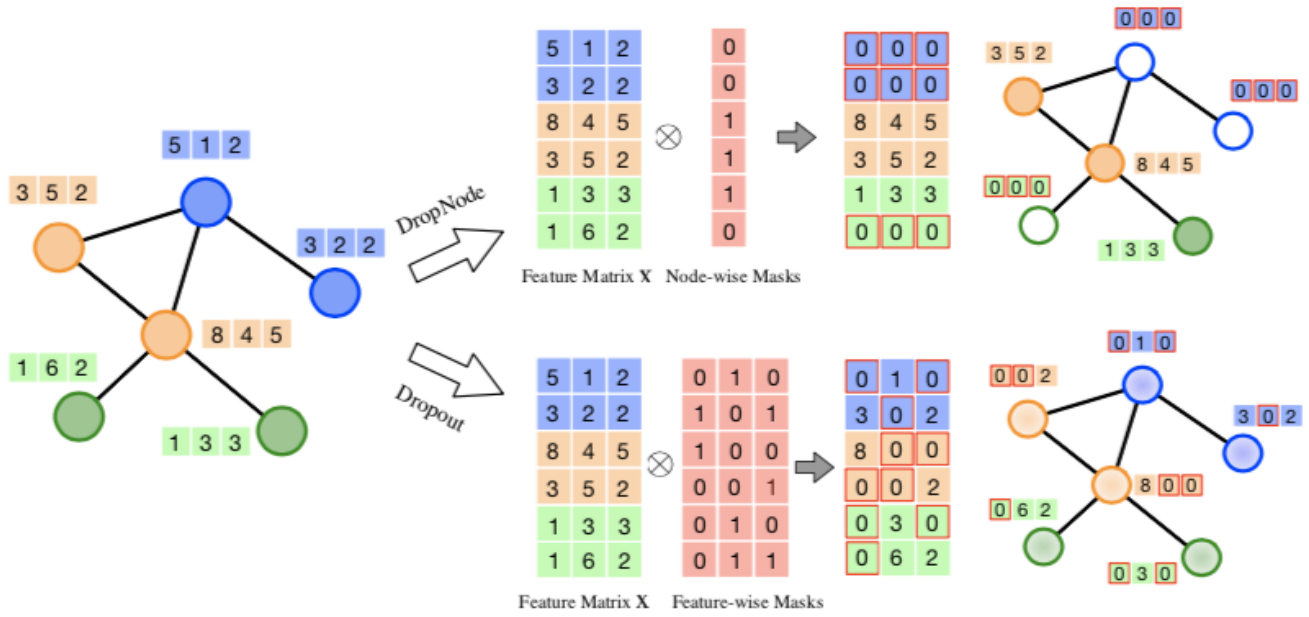


Figure 3: Difference between dropnode and dropout. Dropout drops each element in **X** independently, while DropNode drops the entire features of selected nodes, i.e., the row vectors of **X**, randomly.

- Theoretically, Dropout is an adaptive L2 regularization.

Results

Table 2: Summary of classification accuracy (%).

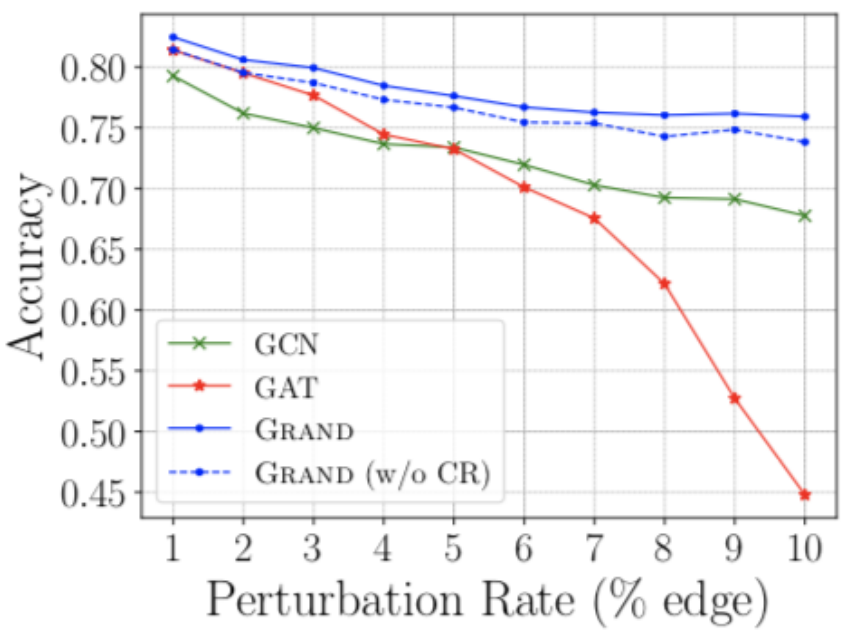
Category	Method	Cora	Citeseer	Pubmed
Graph Convolution	GCN [27]	81.5	70.3	79.0
	GAT [42]	83.0±0.7	72.5±0.7	79.0 ±0.3
	Graph U-Net [15]	84.4±0.6	73.2±0.5	79.6±0.2
	MixHop [2]	81.9 ± 0.4	71.4±0.8	80.8±0.6
	GMNN [36]	83.7	72.9	81.8
	GraphNAS [16]	84.2±1.0	73.1±0.9	79.6±0.4
Regularization based GCNs[2]	VBAT [13]	83.6 ± 0.5	74.0 ± 0.6	79.9 ± 0.4
	G^3NN [31]	82.5 ±0.2	74.4±0.3	77.9 ± 0.4
	GraphMix [43]	83.9 ± 0.6	74.5 ±0.6	81.0 ± 0.6
	DropEdge [37]	82.8	72.3	79.6
Sampling based GCNs[3]	GraphSAGE [22]	78.9±0.8	67.4±0.7	77.8±0.6
	FastGCN [9]	81.4±0.5	68.8±0.9	77.6±0.5
Our methods	GRAND	**85.4±0.4**	**75.4±0.4**	**82.7±0.6**
	GRAND_GCN	84.5±0.3	74.2±0.3	80.0±0.3
	GRAND_GAT	84.3±0.4	73.2± 0.4	79.2±0.6
	GRAND_dropout	84.9±0.4	75.0±0.3	81.7±1.0

Ablation Study

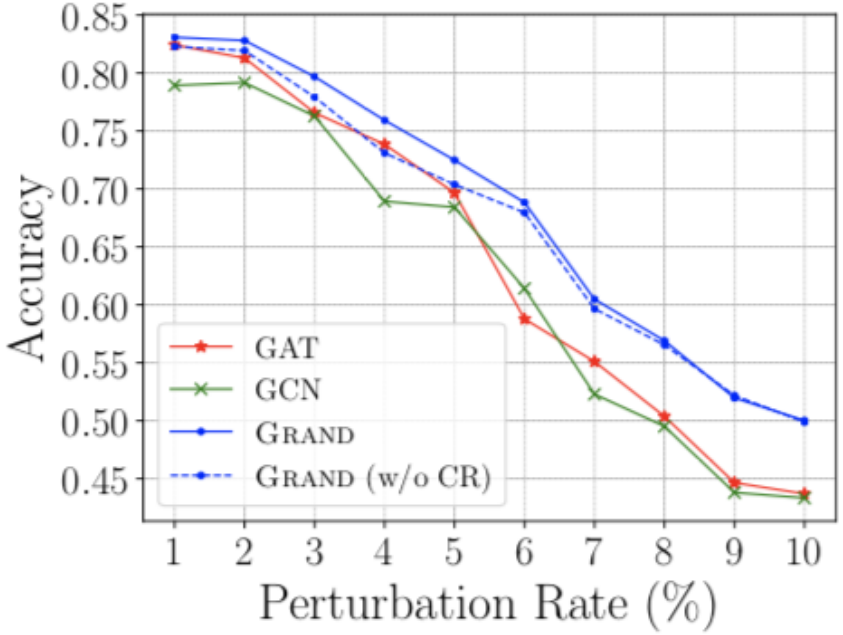
Table 3: Ablation study results (%).

Model	Cora	Citeseer	Pubmed
GRAND	**85.4±0.4**	**75.4±0.4**	**82.7±0.6**
without CR	84.4 ±0.5	73.1 ±0.6	80.9 ±0.8
without multiple DropNode	84.7 ±0.4	74.8±0.4	81.0±1.1
without sharpening	84.6 ±0.4	72.2±0.6	81.6 ± 0.8
without CR and DropNode	83.2 ± 0.5	70.3 ± 0.6	78.5± 1.4

Robustness Analysis



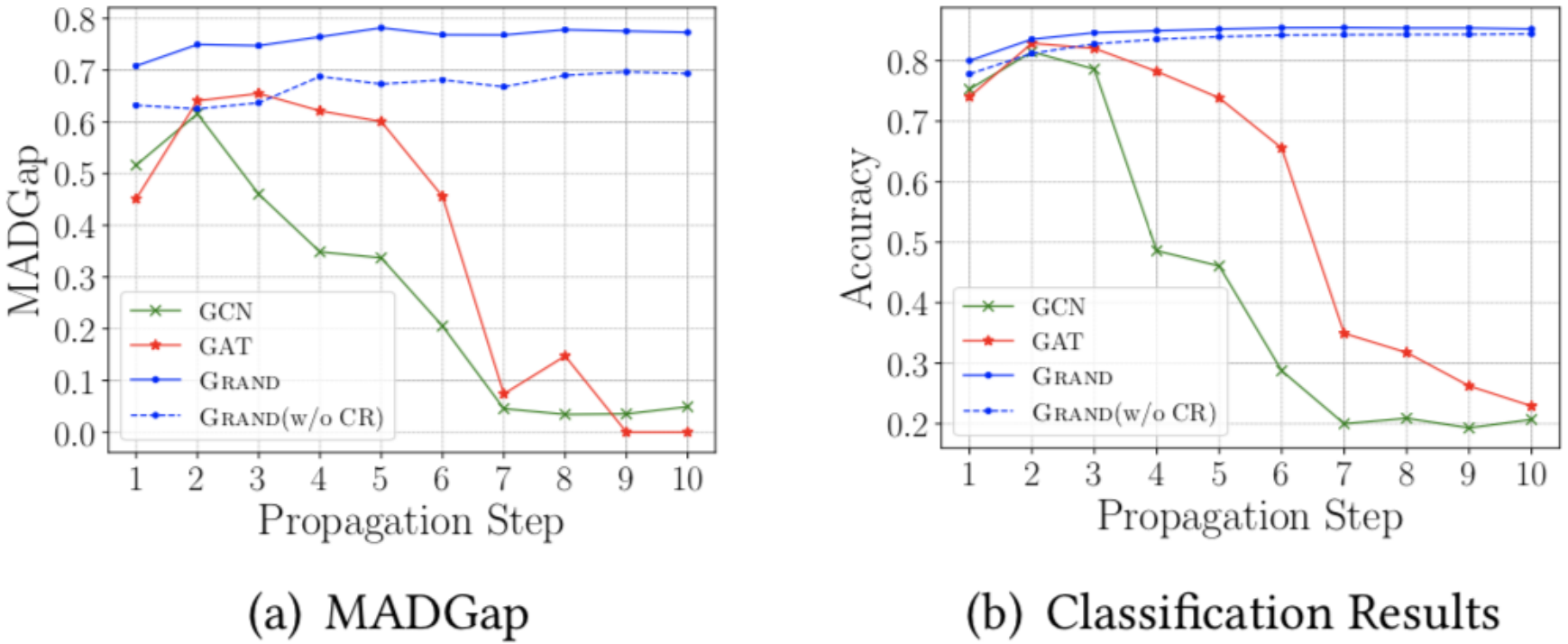
(a) Random Attack



(b) Metattack

Figure 4: Robustness: results under attacks on Cora.

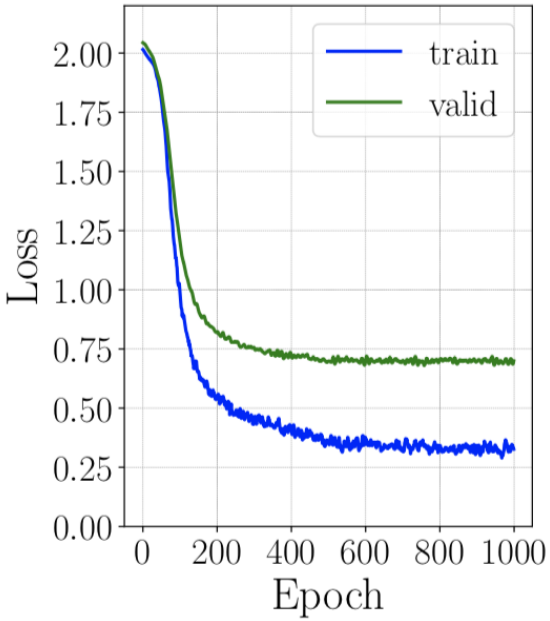
Relieving Over-smoothing



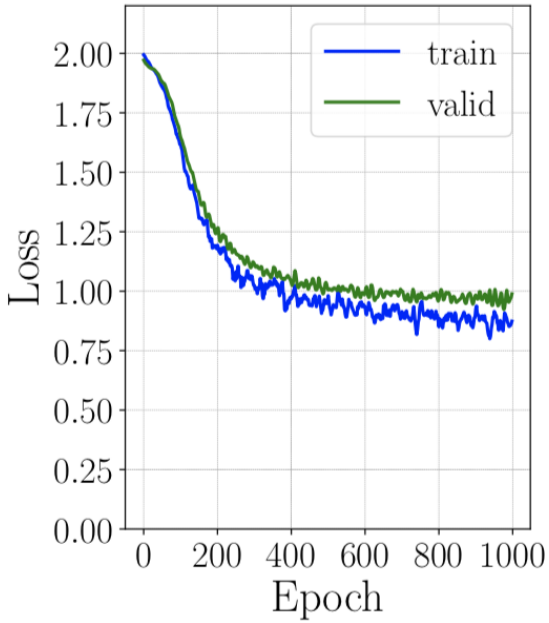
(a) MADGap (b) Classification Results

Figure 5: Over-smoothing: GRAND vs. GCN & GAT on Cora.

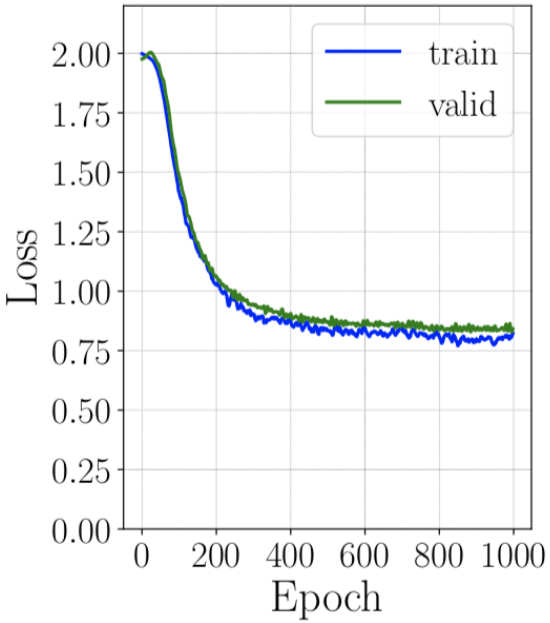
Generalization Improvement



(a) Without CR and RP



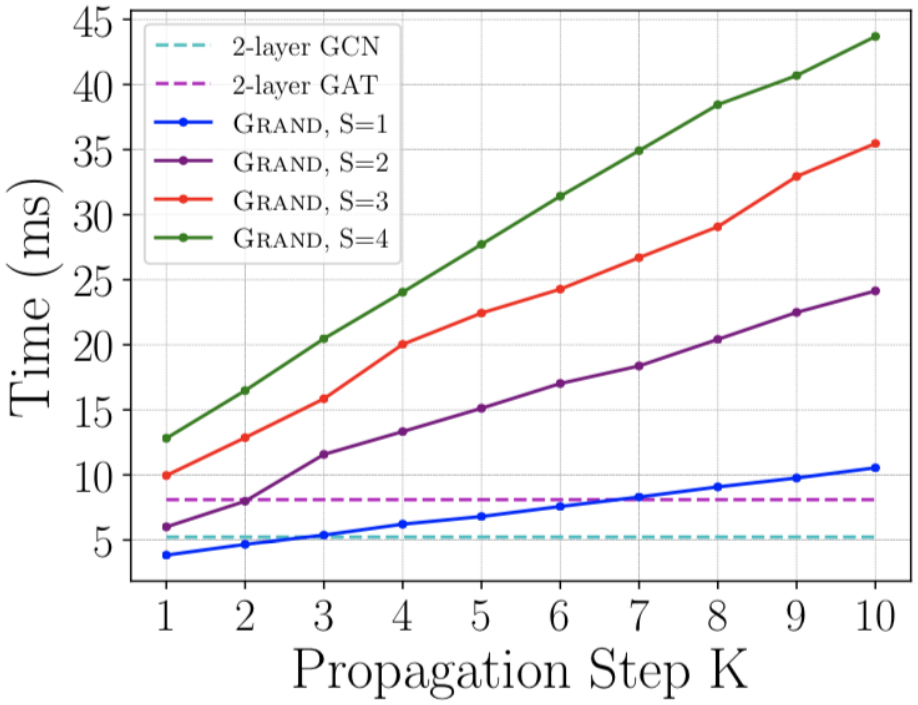
(b) Without CR



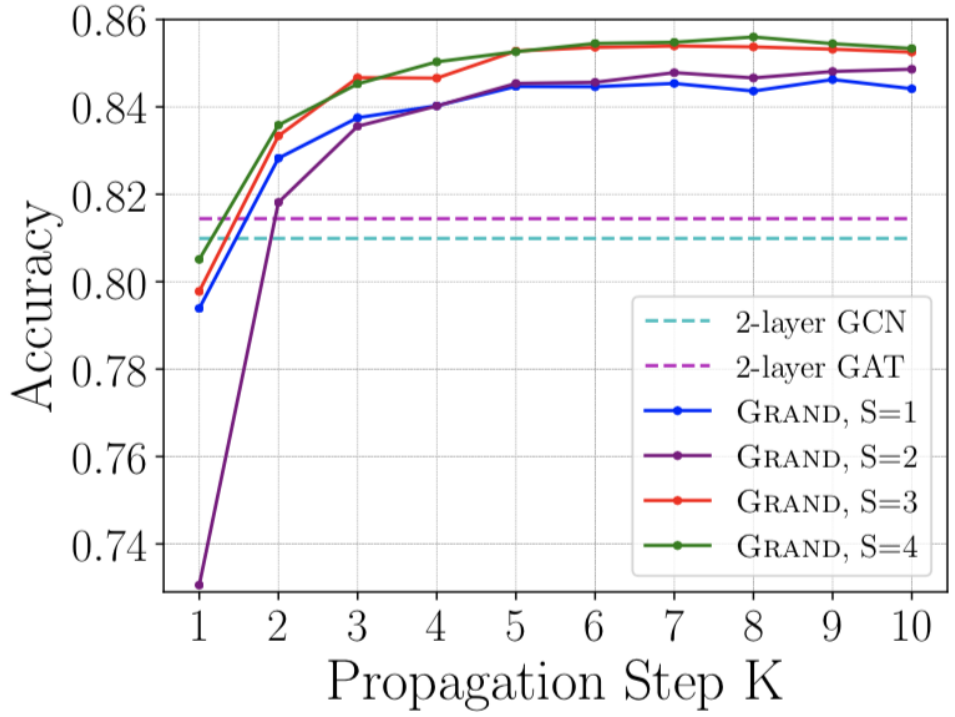
(c) GRAND

Figure 6: Generalization: Training/validation losses on Cora.

Efficiency Analysis



(a) Per-epoch Training Time



(b) Classification Accuracy

Figure 7: Efficiency Analysis for GRAND.

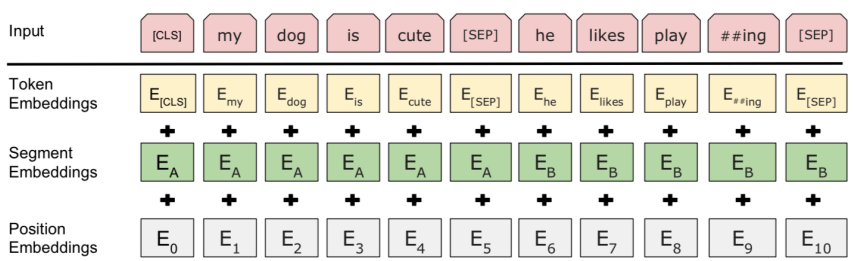
Results on Large Datasets

Table 6: Results on large datasets.

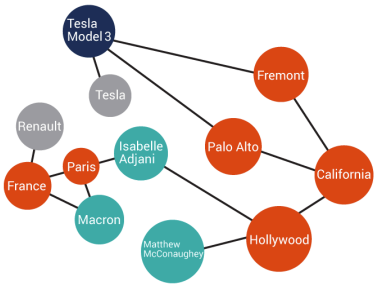
Method	Cora Full	Coauthor CS	Coauthor Physics	Amazon Computer	Amazon Photo	AMiner CS
GCN	62.2 ± 0.6	91.1 ± 0.5	92.8 ± 1.0	82.6 ± 2.4	91.2 ± 1.2	49.4 ± 1.7
GAT	51.9 ± 1.5	90.5 ± 0.6	92.5 ± 0.9	78.0 ± 19.0	85.7 ± 20.3	49.7 ± 1.6
GRAND	**63.5 ±0.6**	**92.9 ± 0.5**	**94.6 ± 0.5**	**85.7 ± 1.8**	**92.5 ± 1.7**	**52.2 ± 1.3**

Challenges

- How to combine Graph with Pre-Training (BERT/CL)?
- Graph Neural Network Pre-Training



+

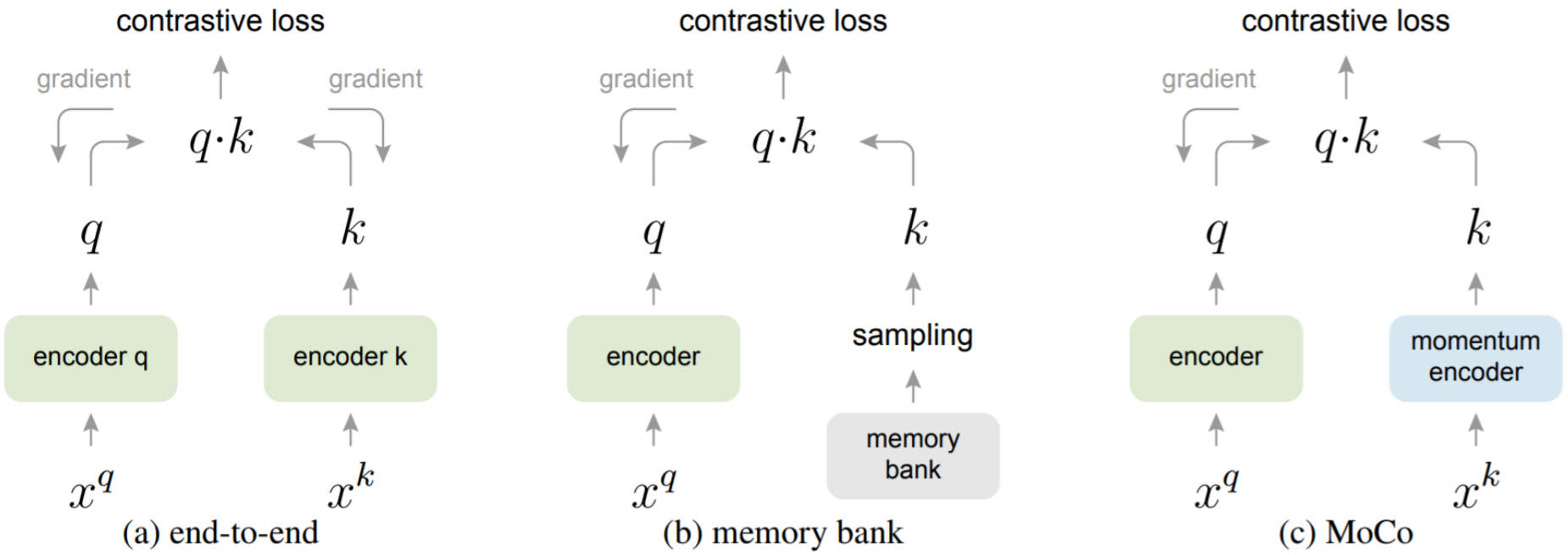


GCC: Graph Contrastive Coding for Graph Neural Network Pre-Training

- Momentum Contrast for Unsupervised Visual Representation Learning https://arxiv.org/pdf/1911.05722.pdf
- Train ResNet on ImageNet without image labels
  - Competitive results on ImageNet classification
  - Outperform supervised ResNet in 7 downstream tasks

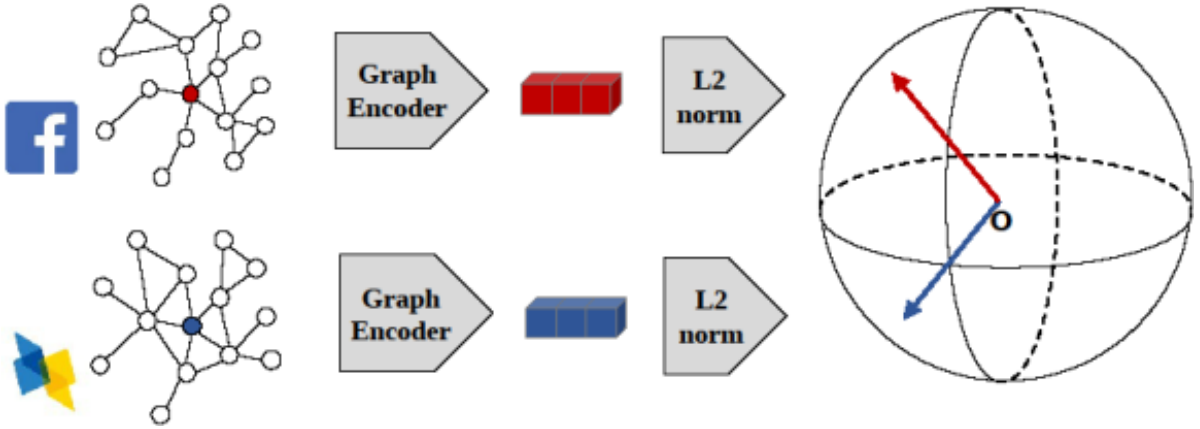
1. J. Qiu, Q. Chen, Y. Dong, J. Zhang, H. Yang, M. Ding, K. Wang, and J. Tang. GCC: Graph Contrastive Coding for Graph Neural Network Pre-Training. *KDD'20*.

Revisit MoCo



Graph Contrastive Coding (GCC)

- Problem formulation:
  - Measuring vertex structural similarity

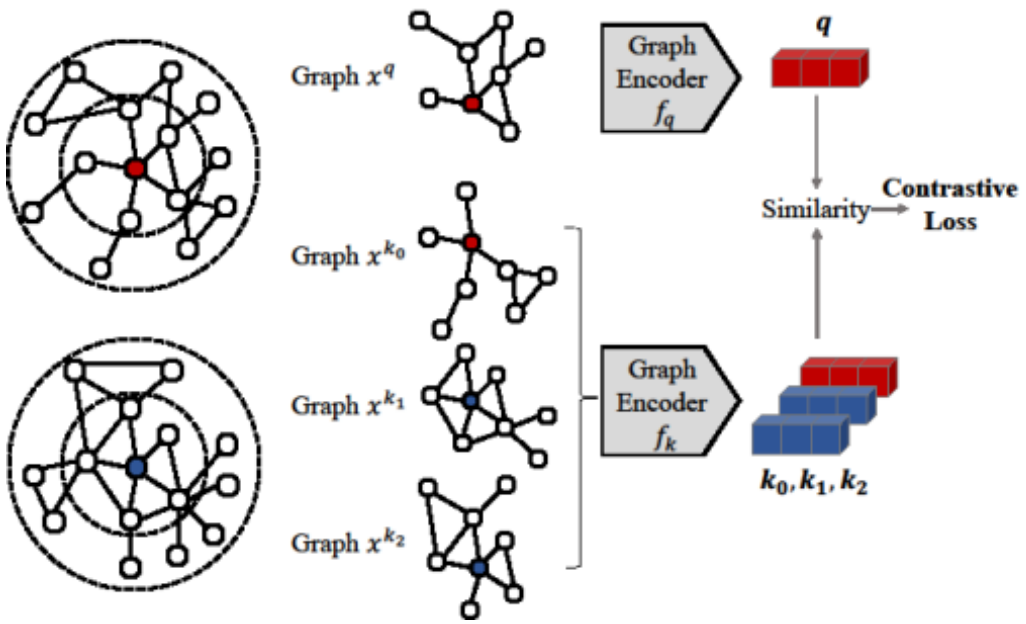


  - New wine in old bottles:
    - motif, centrality, clustering coefficient, structural diversity, edge density, etc

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*.

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*.

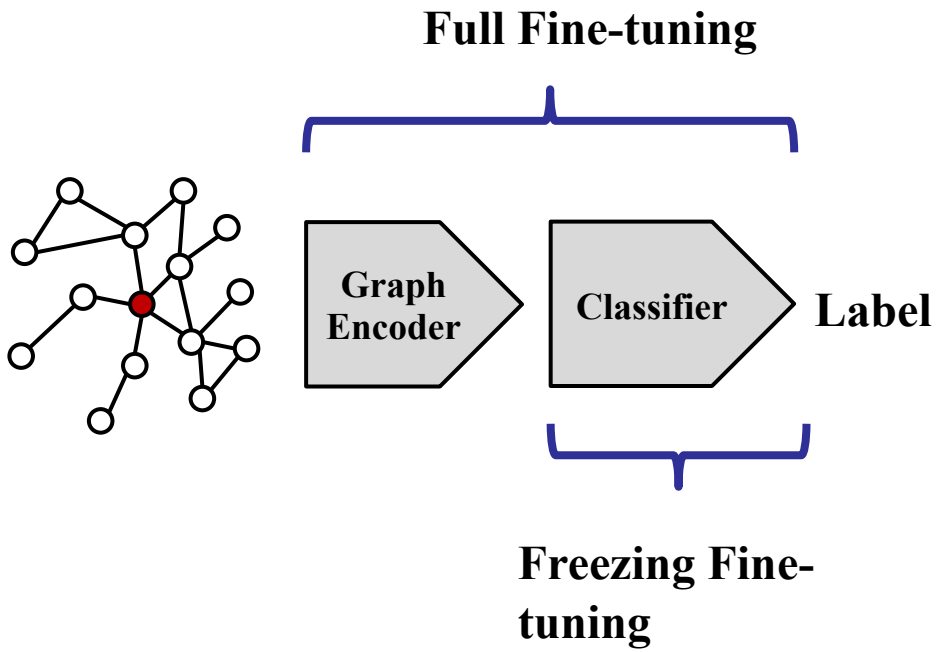
GCC Pre-Training / Fine-tuning

- 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

Takeaway Messages

- Tackle non-robust, over-fitting, and over-smoothing
- Understanding GNNs as Signal Rescaling
  - Unifying GCNs
  - Unifying GAT
- Pre-training and Self-supervised learning is becoming more and more important

GNN与推理

认知推理

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

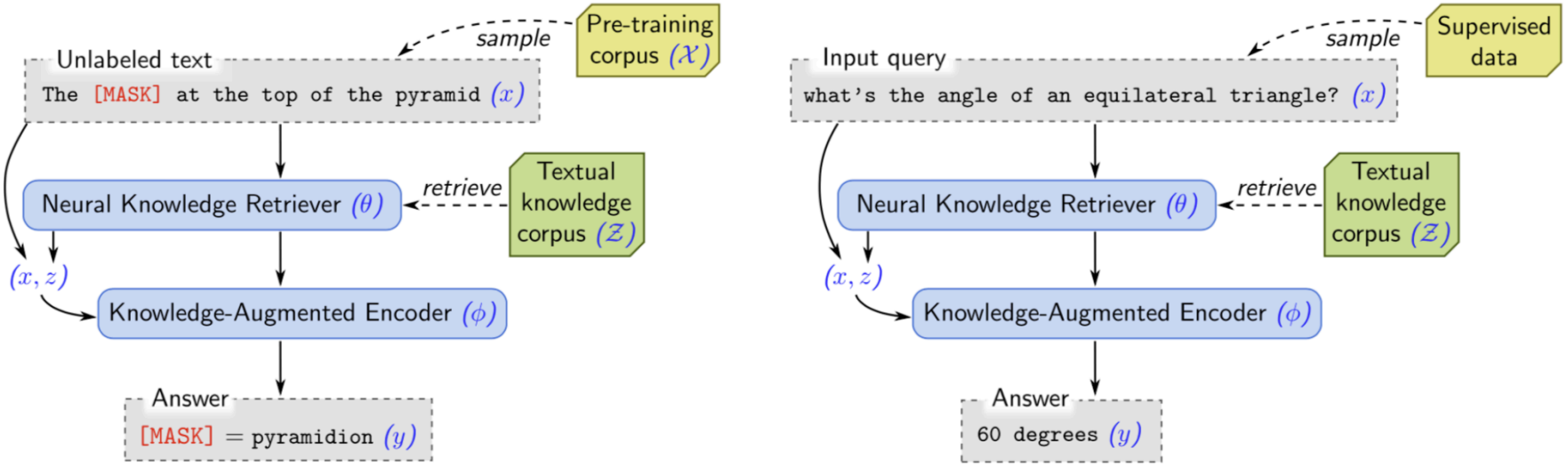
Tsinghua University is a major research university in Beijing and dedicated to academic excellence and global development. Tsinghua is perennially ranked as one of the top academic institutions in China, Asia, and worldwide...

REALM: Retrieval-Augmented LM

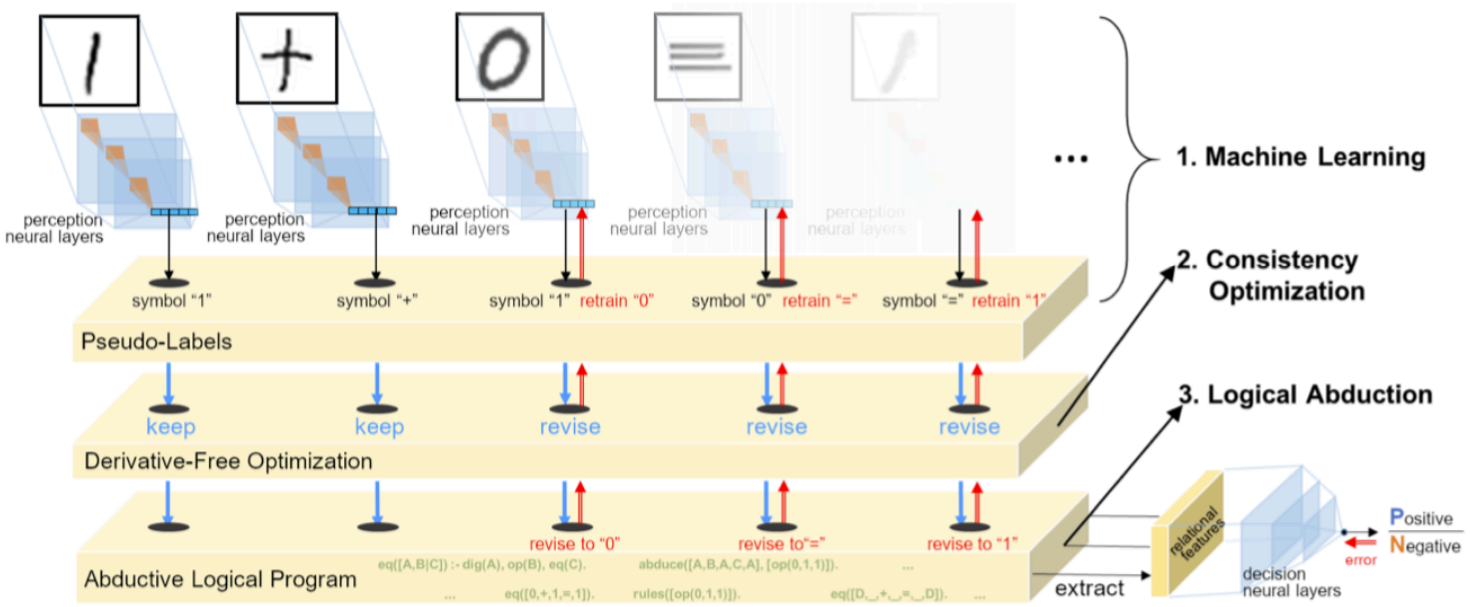
$$\text{BERT} : p(y|x)$$

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

- where Z is the supporting set.



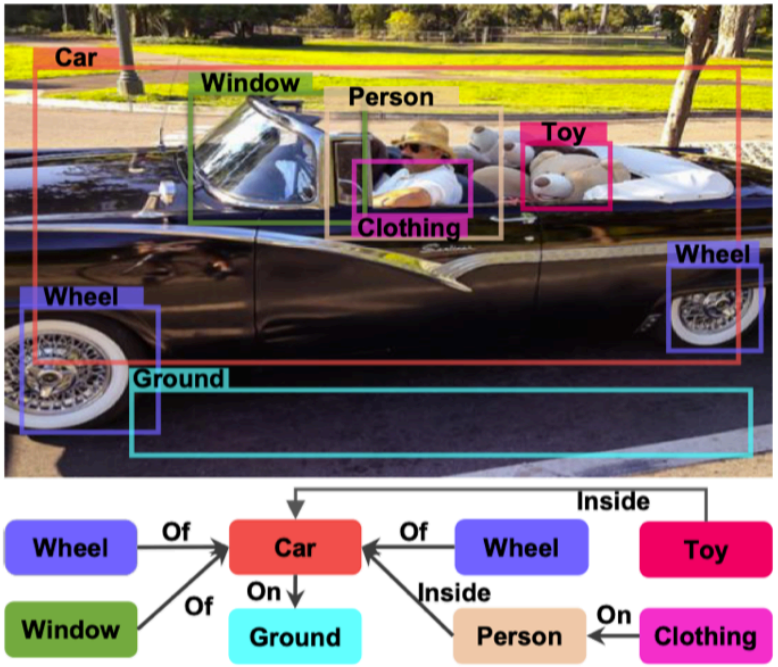
Abductive Learning

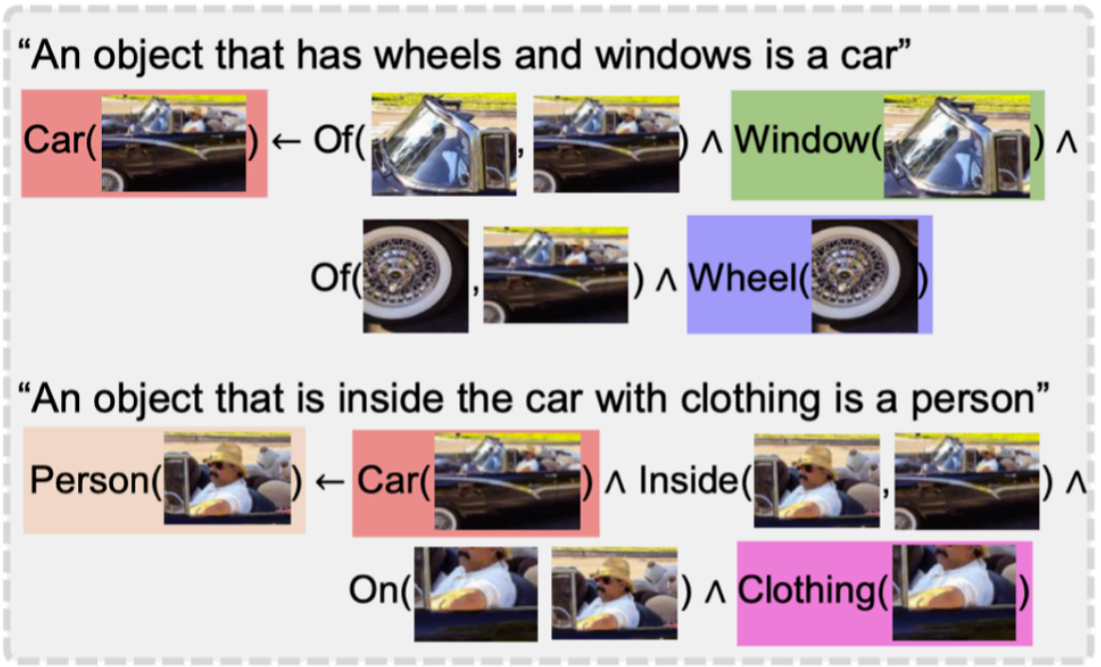


They use deep learning to extract high-level concepts and logical rules to make predictions based on concepts.
The optimization is difficult since it involves non-differentiable logic learning.

1. W. Dai et al. Bridging Machine Learning and Logical Reasoning by Abductive Learning. NeurIPS. 2019, pp. 2811–2822.

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

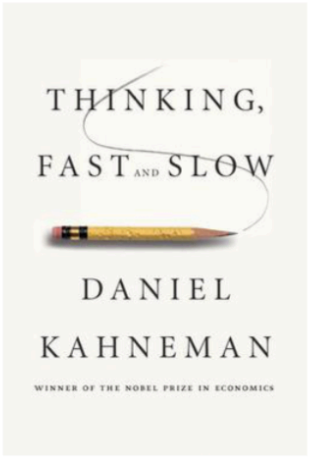
SYSTEM 1 VS. SYSTEM 2 COGNITION

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

认知图谱(System 2 DL)：算法与认知的结合

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.

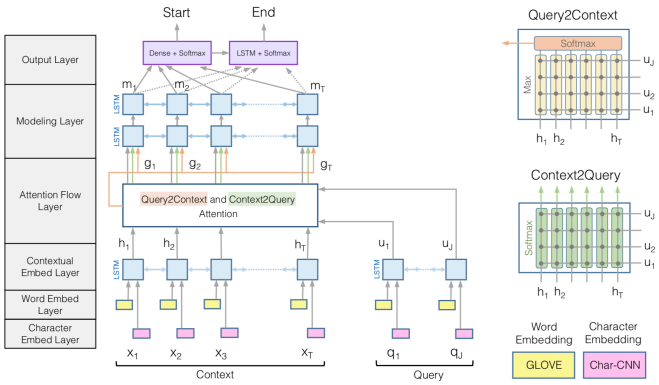
Tsinghua University

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...

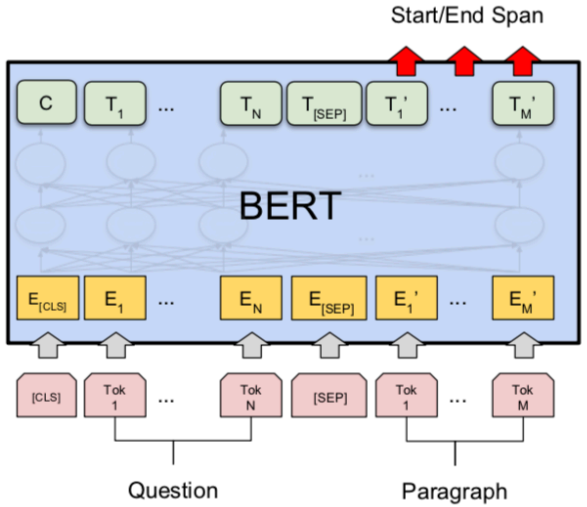
算法：BIDAF, BERT, XLNet

- 目标：理解整个文档，而不仅仅是局部片段
- 但仍然缺乏在知识层面上的推理能力

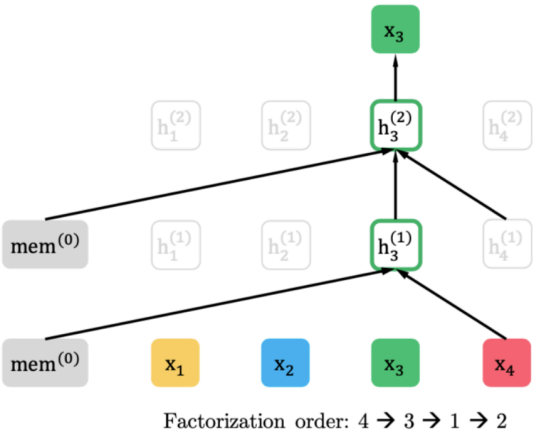
BiDAF



BERT



XLNet

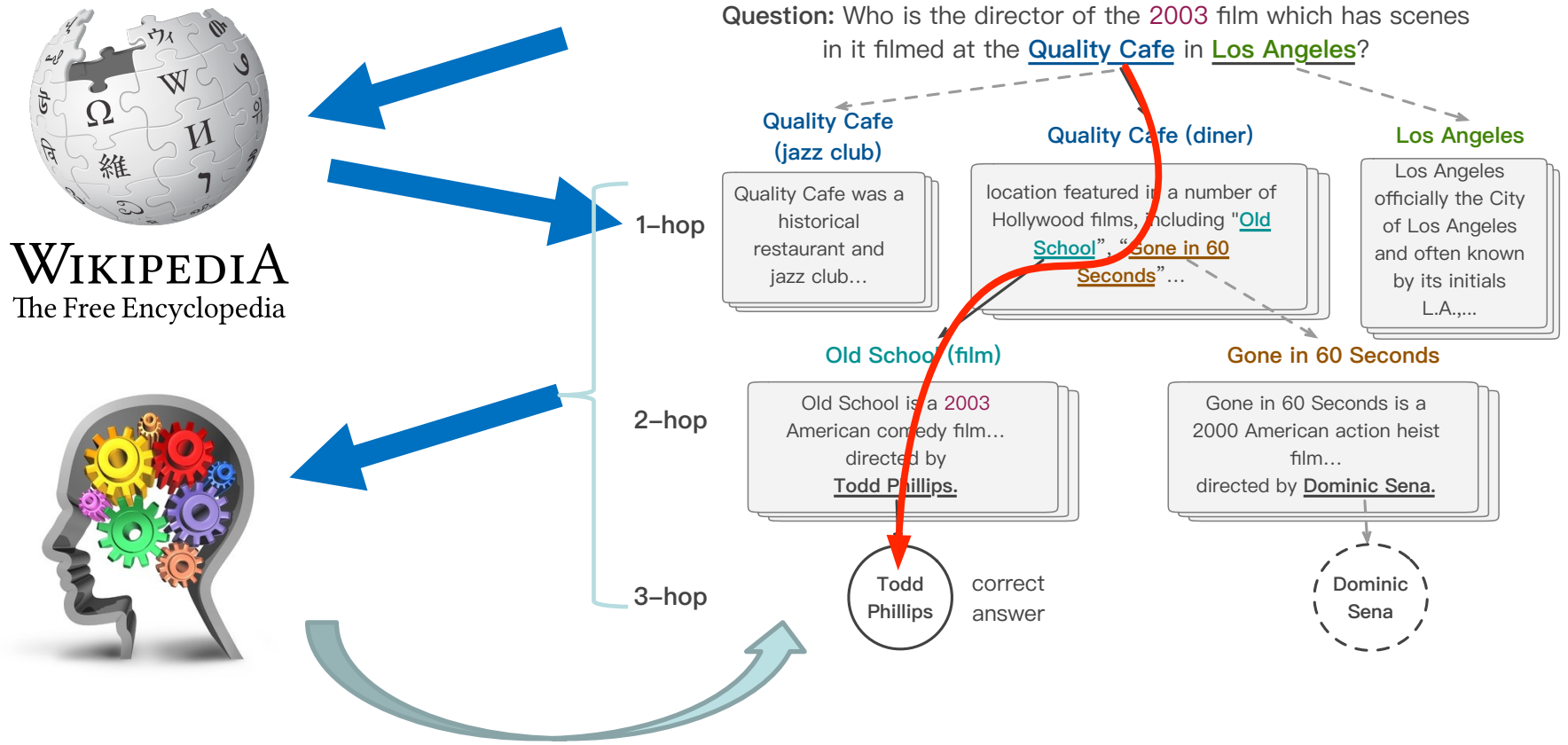


挑战：可解释性

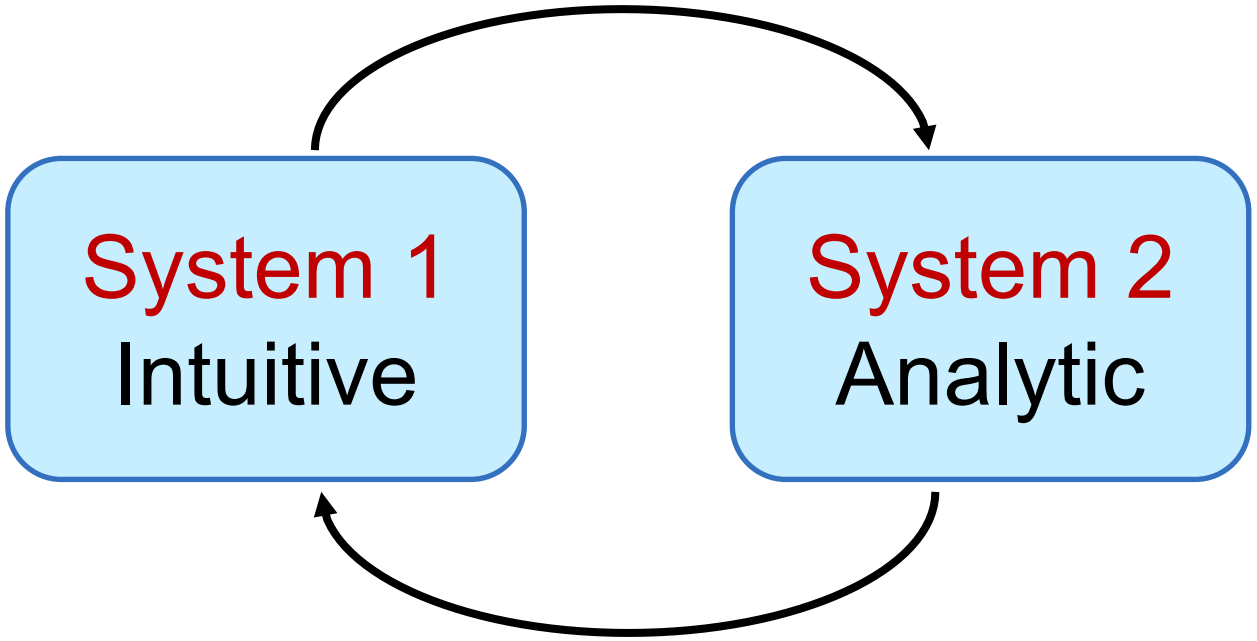
- 大部分阅读理解方法都只能看做**黑盒**：
  - 输入：问题和文档
  - 输出：答案文本块（在文档中的起止位置）

- 如何让用户可以验证答案的对错：
  - 推理路径或者子图
  - 每个推理节点上的支撑事实
  - 用于对比的其他可能答案和推理路径

认知图谱：知识表示，推理和决策



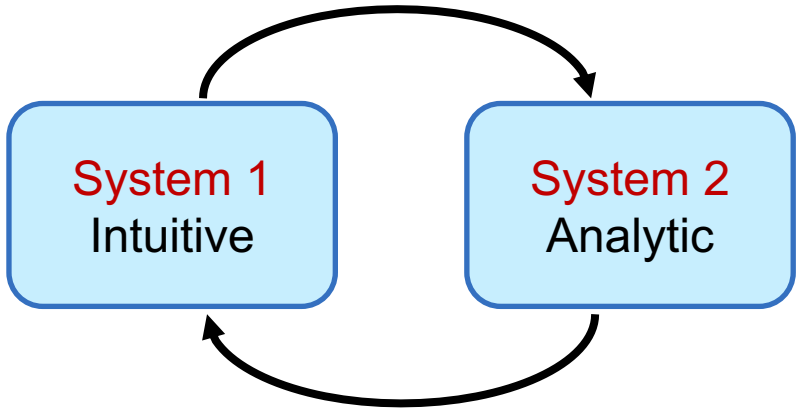
和认知科学的结合



Dual Process Theory (Cognitive Science)

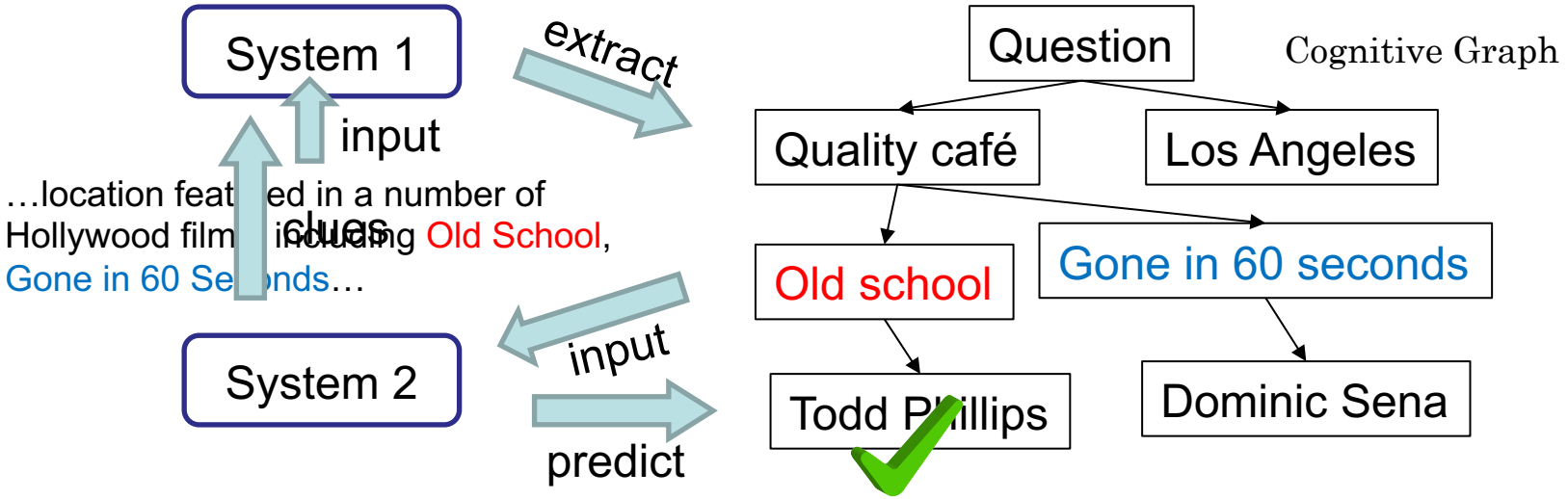
Reasoning w/ Cognitive Graph

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



CogQA: Cognitive Graph for QA

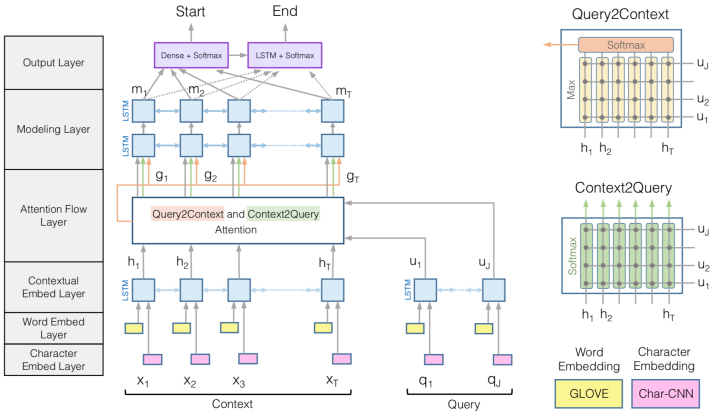
- An iterative framework corresponding to dual process theory
- System 1
  - extract entities to build the cognitive graph
  - generate semantic vectors for each node
- System 2
  - Do reasoning based on semantic vectors and graph
  - Feed clues to System 1 to extract next-hop entities



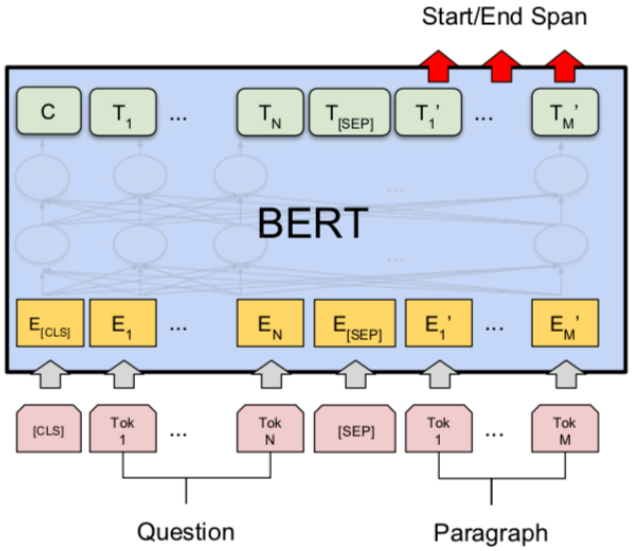
System 1: BIDAF, BERT

- reading comprehension: target at understanding the whole paragraph

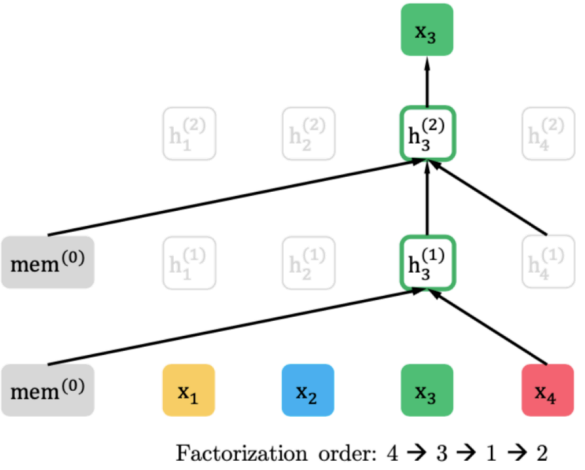
BiDAF



BERT

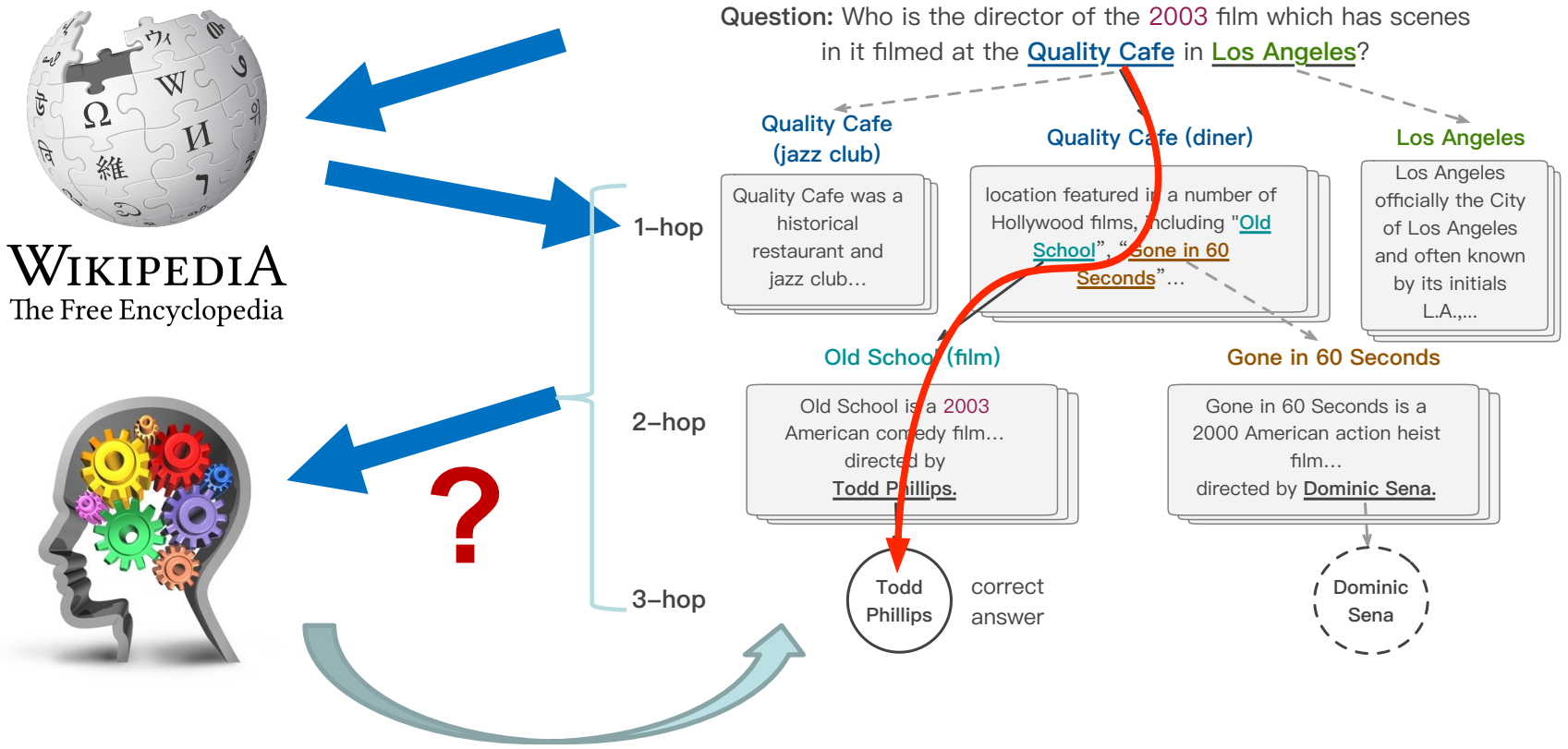


XLNet

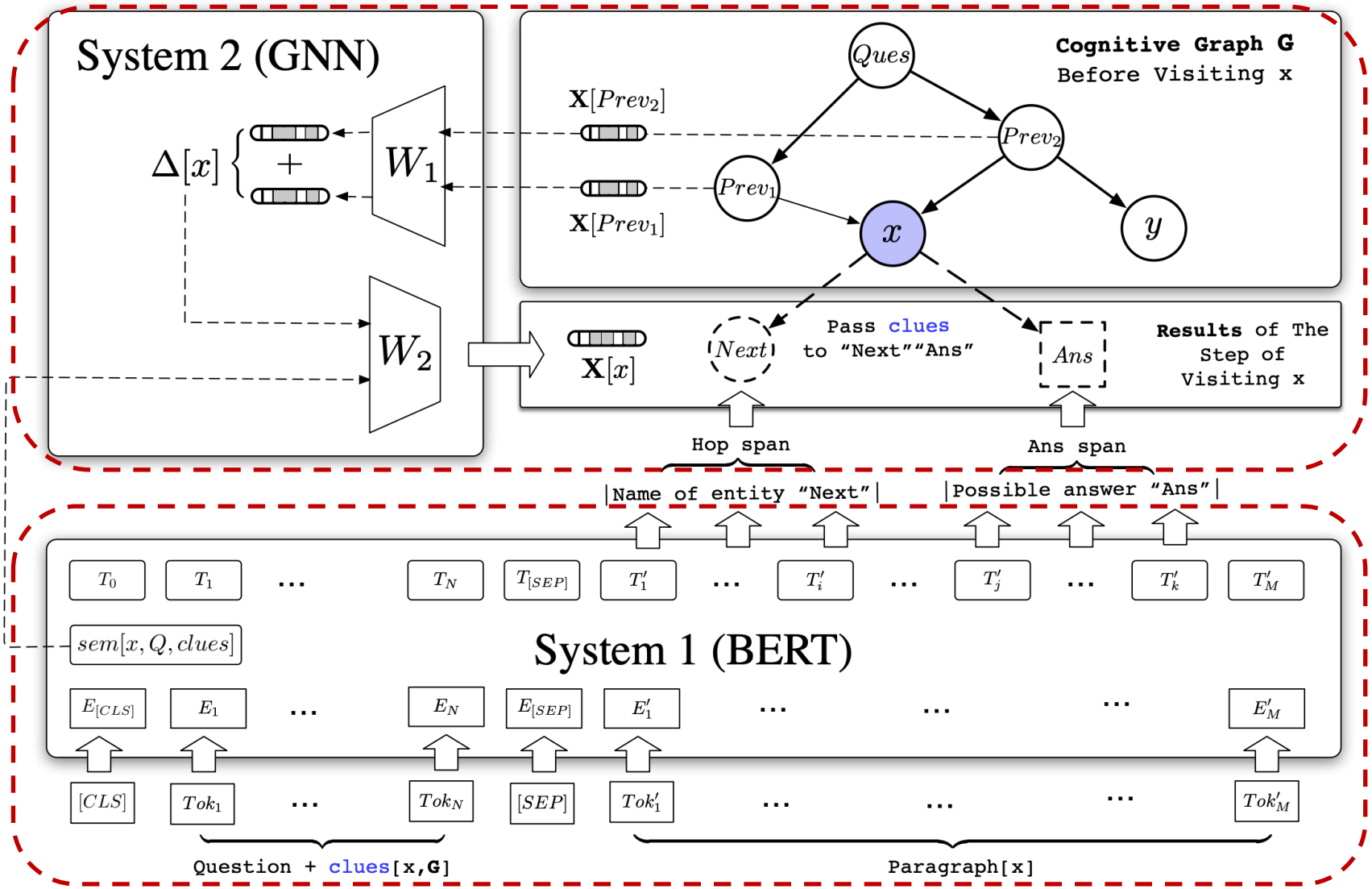


1. Rajpurkar, Pranav, et al. "SQuAD: 100,000+ Questions for Machine Comprehension of Text." EMNLP. 2016.

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.

Performance

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

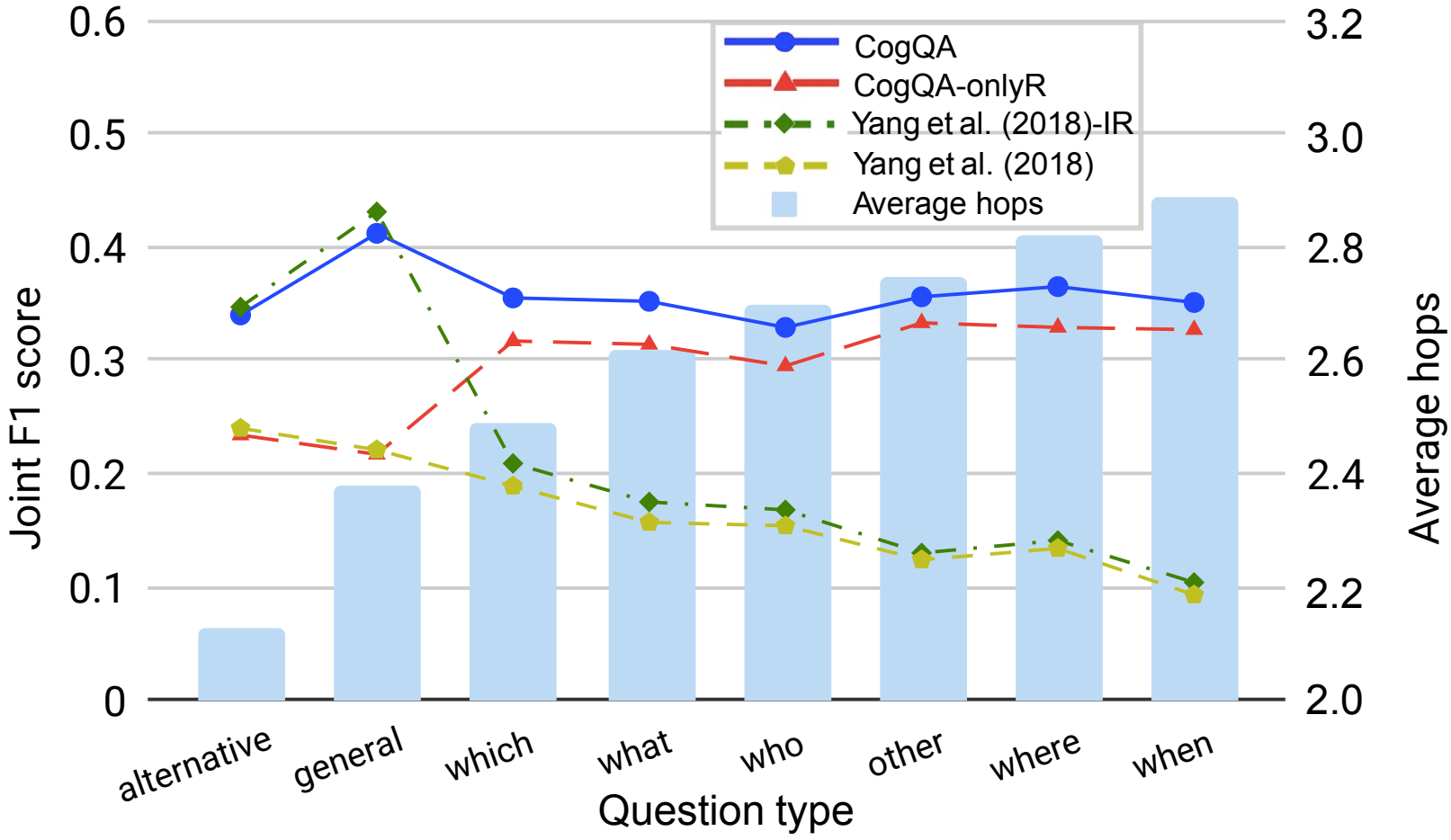
	Model	Ans				Sup				Joint			
		EM	F_1	Prec	Recall	EM	F_1	Prec	Recall	EM	F_1	Prec	Recall
Dev	Yang et al. (2018)	23.9	32.9	34.9	33.9	5.1	40.9	47.2	40.8	2.5	17.2	20.4	17.8
	Yang et al. (2018)-IR	24.6	34.0	35.7	34.8	10.9	49.3	52.5	52.1	5.2	21.1	22.7	23.2
	BERT	22.7	31.6	33.4	31.9	6.5	42.4	54.6	38.7	3.1	17.8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

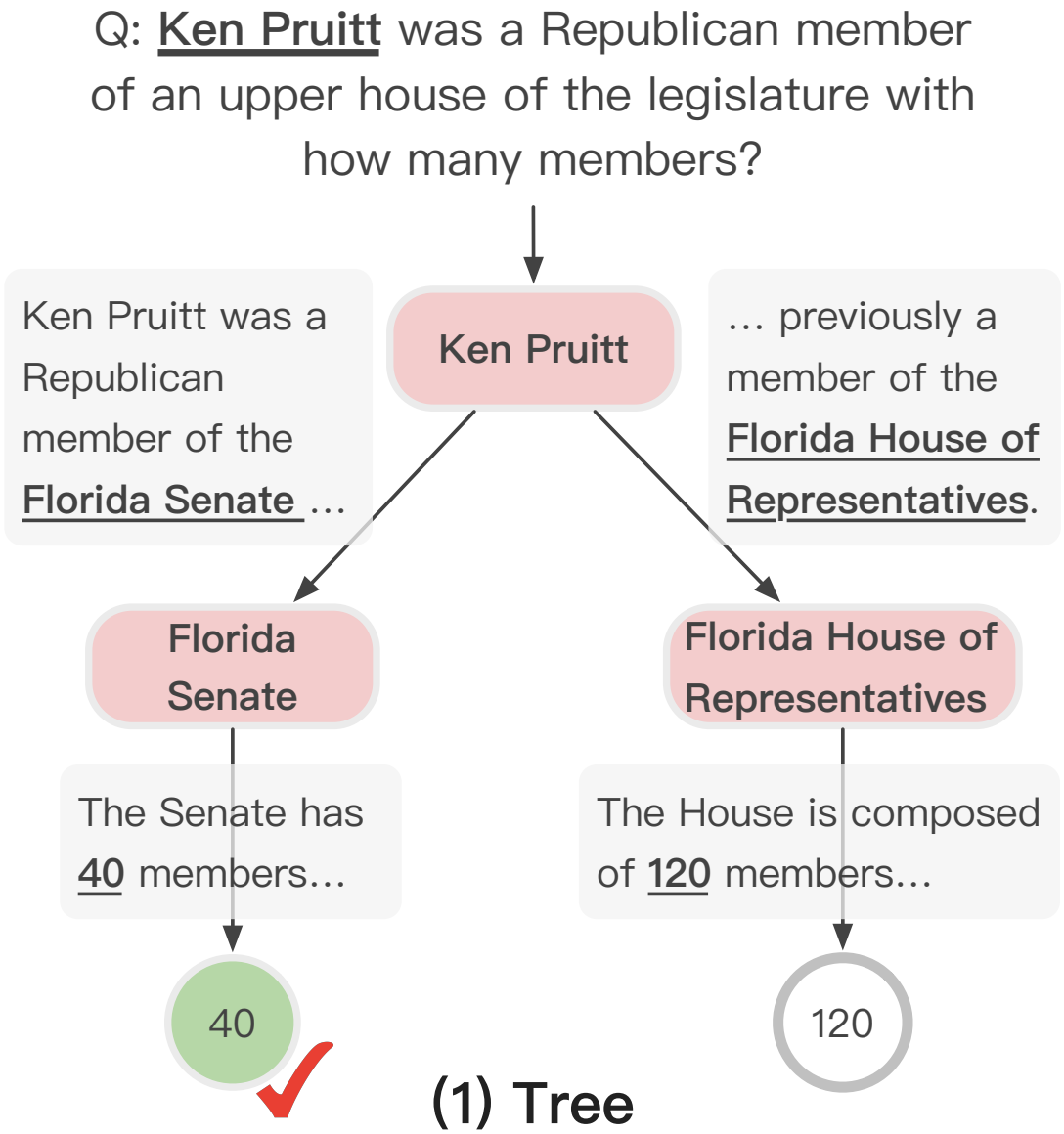
** Code available at https://github.com/THUDM/CogQA

Reasoning Power

CogQA Performs much better on question with more hops !



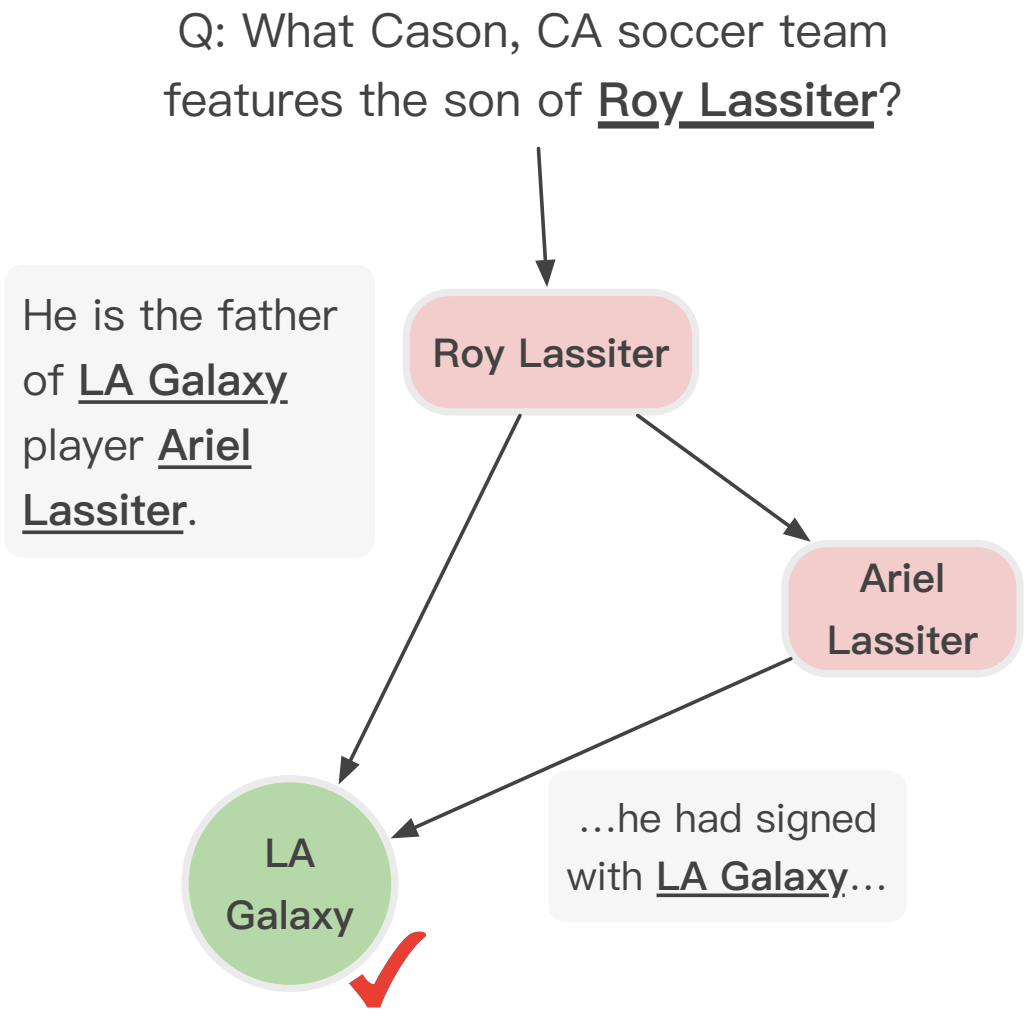
Case Study



(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*"

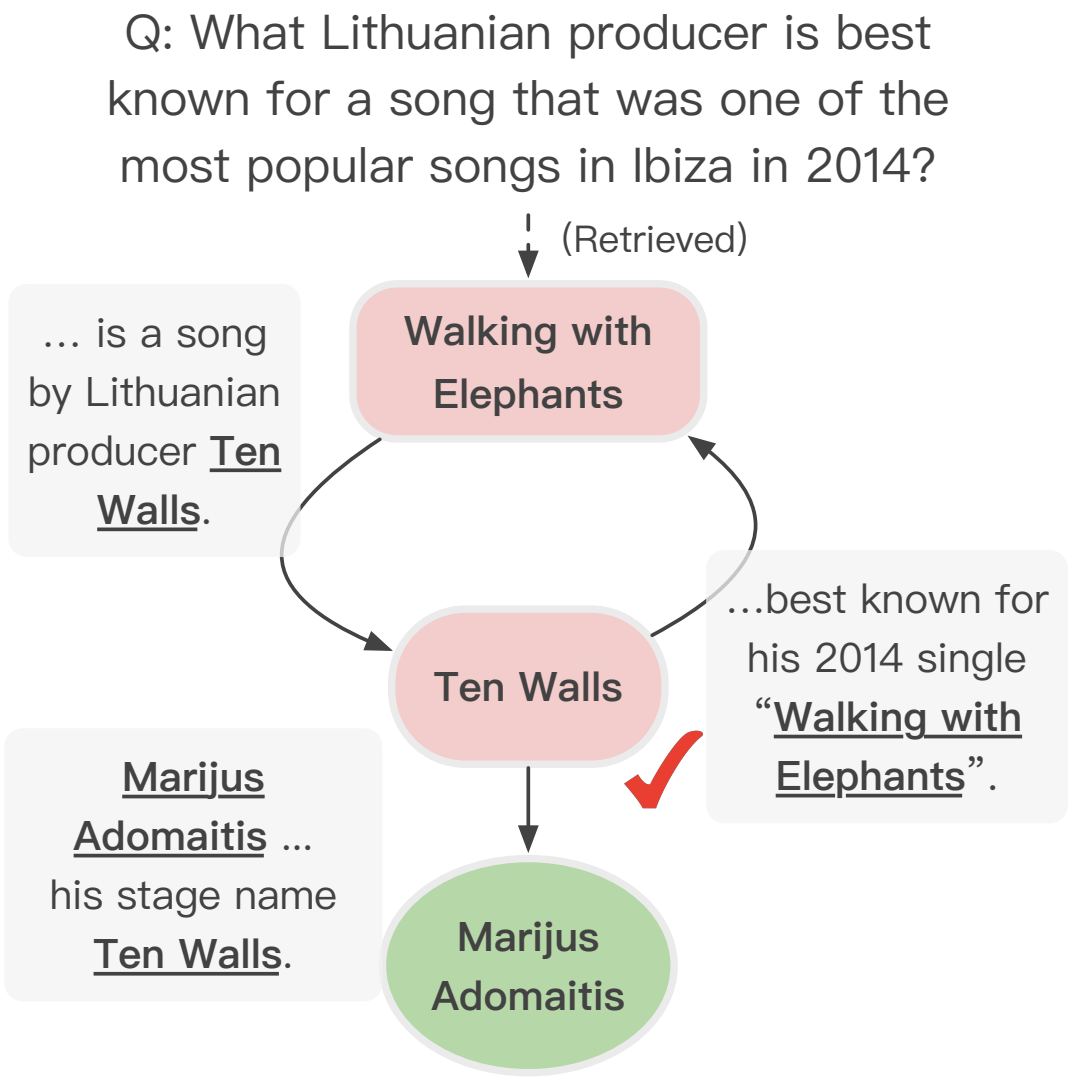
Case Study



(2) DAG

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

Case Study



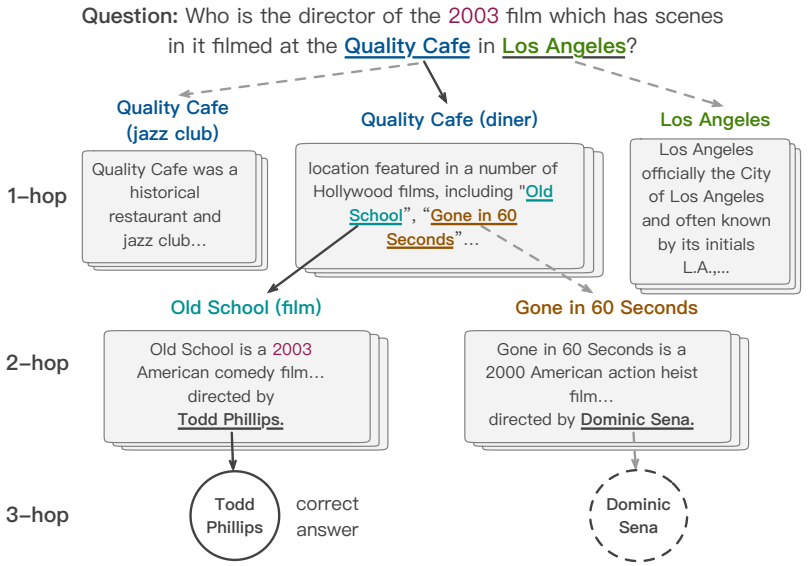
(3) Cyclic Graph

- 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.

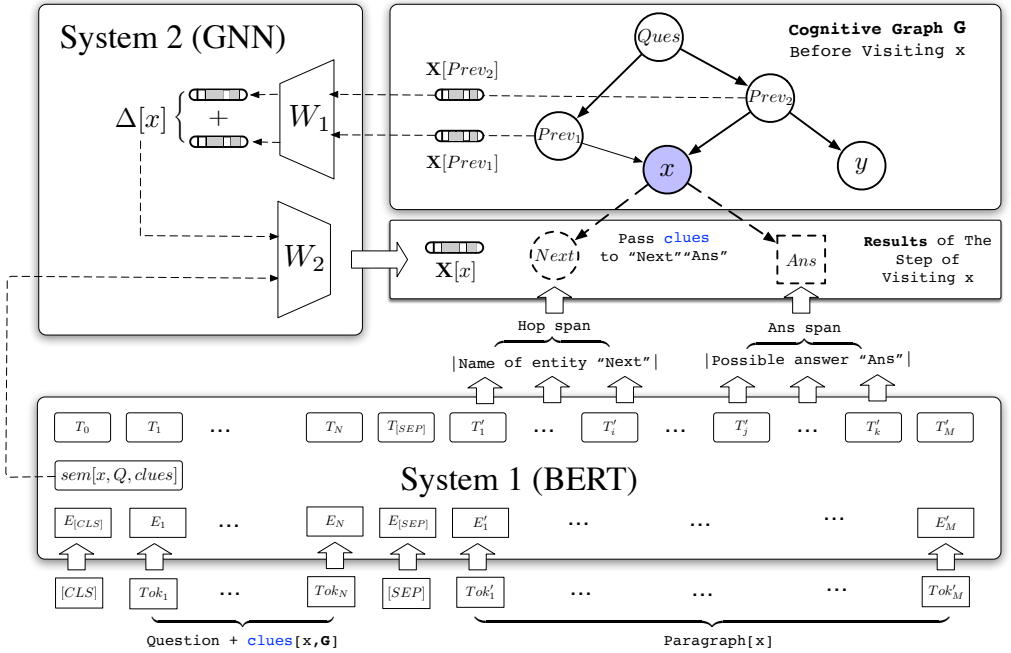
Summary

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

Cognitive graph



CogQA framework



CogDL: 基于图的深度学习开源工具包

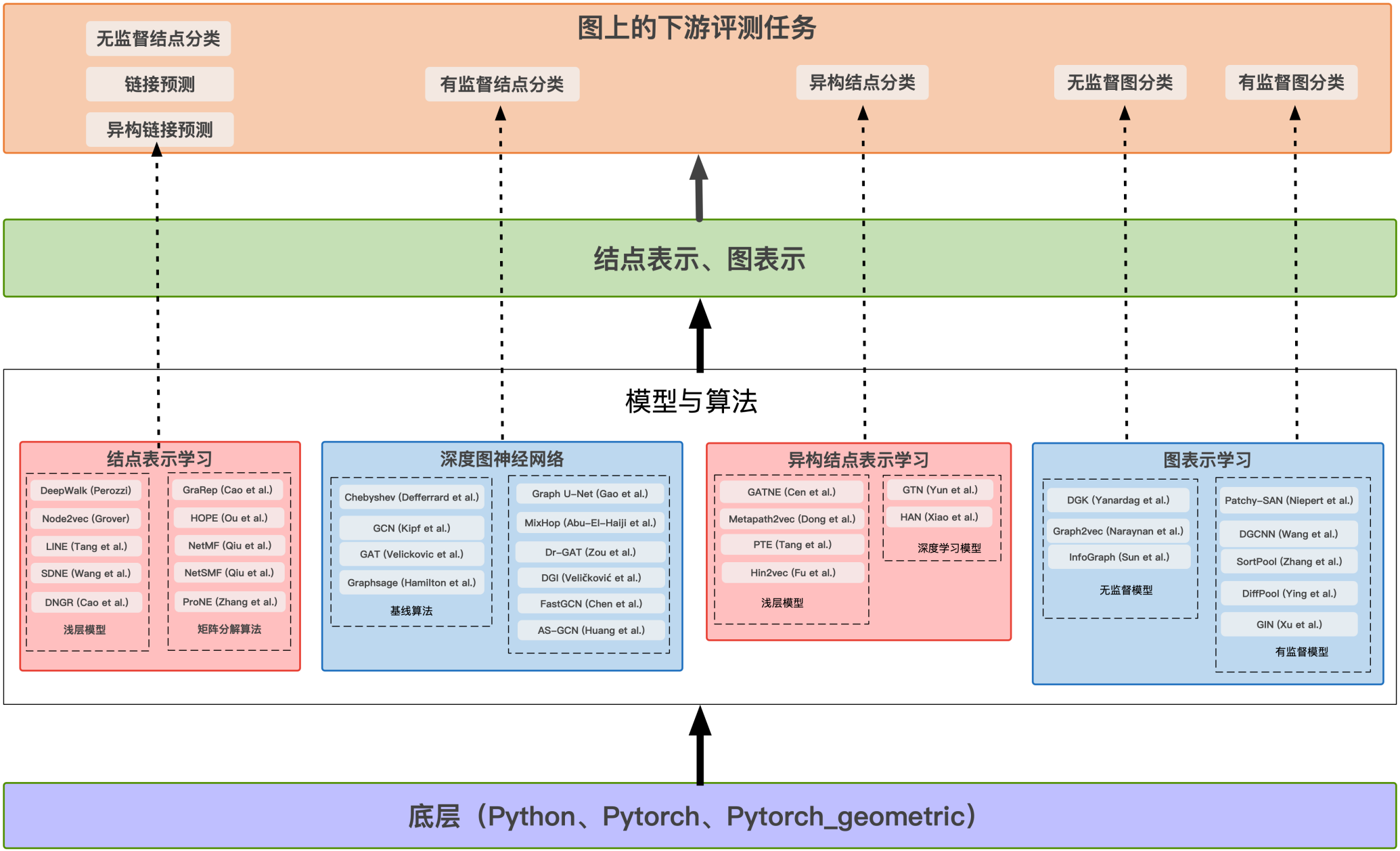
Yukuo Cen, Yan Wang, Zhenyu Hou, Jie Tang

主页: http://keg.cs.tsinghua.edu.cn/cogdl
代码库：https://github.com/THUDM/cogdl
文档：https://cogdl.readthedocs.io

Google/Baidu “CogDL”

清华大学知识工程实验室（KEG）& 北京智源人工智能研究院（BAAI）

CogDL


图上的下游评测任务
无监督结点分类
链接预测
异构链接预测
有监督结点分类
异构结点分类
无监督图分类
有监督图分类
结点表示、图表示
模型与算法
结点表示学习
DeepWalk (Perozzi)
Node2vec (Grover)
LINE (Tang et al.)
SDNE (Wang et al.)
DNGR (Cao et al.)
浅层模型
GraRep (Cao et al.)
HOPE (Ou et al.)
NetMF (Qiu et al.)
NetSMF (Qiu et al.)
ProNE (Zhang et al.)
矩阵分解算法
深度图神经网络
Chebyshev (Defferrard et al.)
GCN (Kipf et al.)
GAT (Velickovic et al.)
Graphsage (Hamilton et al.)
基线算法
Graph U-Net (Gao et al.)
MixHop (Abu-El-Haiji et al.)
Dr-GAT (Zou et al.)
DGI (Veličković et al.)
FastGCN (Chen et al.)
AS-GCN (Huang et al.)
异构结点表示学习
GATNE (Cen et al.)
Metapath2vec (Dong et al.)
PTE (Tang et al.)
Hin2vec (Fu et al.)
浅层模型
GTN (Yun et al.)
HAN (Xiao et al.)
深度学习模型
图表示学习
DGK (Yanardag et al.)
Graph2vec (Naraynan et al.)
InfoGraph (Sun et al.)
无监督模型
Patchy-SAN (Niepert et al.)
DGCNN (Wang et al.)
SortPool (Zhang et al.)
DiffPool (Ying et al.)
GIN (Xu et al.)
有监督模型
底层（Python、Pytorch、Pytorch_geometric）

CogDL 特性

构建面向图数据的表示学习和分类的灵活、易用的工具包

✓任务导向： CogDL以图上的任务为主，提供了相关的模型、数据集以及我们得到的排行榜

✓一键运行： CogDL支持用户使用多个GPU同时运行同一个任务下多个模型在多个数据集上的多组实验

✓多类任务： CogDL支持同构/异构网络中的节点分类和链接预测任务以及图分类任务

✓可扩展性：用户可以基于CogDL已有的框架来实现和提交新的数据集、模型和任务

COGDL TOOLKIT - Home

Not Secure | keg.cs.tsinghua.edu.cn/cogdl/

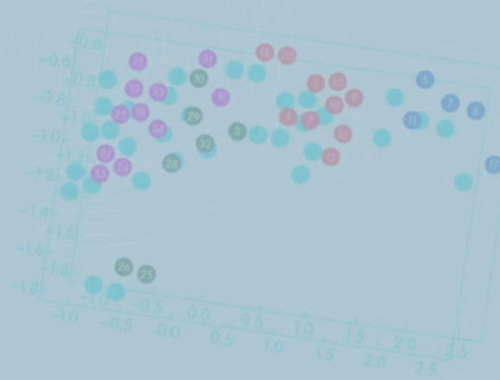
COGDL TOOLKIT

HOME DOCUMENTATION LEADERBOARD METHODS DATASETS FAQ

CogDL: An Extensive Research Toolkit for Deep Learning on Graphs

KEG, Tsinghua University

中文版



What is CogDL?

CogDL is a graph representation learning toolkit that allows researchers and developers to easily train and compare baseline or custom models for node classification, link prediction and other tasks on graphs. It provides implementations of many popular models, including: non-GNN Baselines like Deepwalk, LINE, NetMF, GNN Baselines like GCN, GAT, GraphSAGE.

Task-Oriented

CogDL focuses on tasks on graphs and provides corresponding models, datasets, and leaderboards.

Easy-Running

CogDL supports running multiple experiments simultaneously on multiple models and datasets under a specific task using multiple GPUs.

CogDL代码

- 通过git clone来下载CogDL的代码

```
cogdl
├── cogdl
│   ├── data                # 数据处理脚本目录
│   ├── datasets            # 数据集标准化目录
│   ├── layers              # 神经网络层定义目录
│   ├── models              # 模型实现目录
│   ├── tasks               # 任务实现目录
│   ├── options.py          # 标准参数设置文件
│   └── utils.py            # 工具函数文件
├── data                    # 数据集存放目录
│   ├── Blogcatalog
│   └── ...
├── docs                    # 文档定义和生成目录
├── scripts
│   ├── display_data.py     # 数据集打印信息脚本
│   ├── download.py         # 数据集下载脚本
│   ├── parallel_train.py   # 实验运行主函数的并行版本
│   └── train.py            # 实验运行主函数
├── tests                   # 测试代码目录
├── README.md               # 英文说明文档
├── README_CN.md            # 中文说明文档
└── setup.py                # cogdl的安装文件
```

CogDL的安装与使用

安装PyTorch和其他依赖项:

- https://github.com/pytorch/pytorch#installation
- https://github.com/rusty1s/pytorch_geometric/#installation
- pip install -e .

基本用法:

python train.py --task example_task --dataset example_dataset --model example_method

CogDL使用方法

- 在Cora数据集上运行GCN、GAT模型。

```
(torch) ➜  cogdl git:(master) python scripts/train.py --task node_classification --dataset cora --model pyg_gcn pyg_gat --seed 0 1 2 3 4
Namespace(cpu=False, dataset=['cora'], device_id=[0], dropout=0.6, enhance=False, hidden_size=8, lr=0.005, max_epoch=500, model=['pyg_gcn', 'pyg_g
at'], num_classes=None, num_features=None, num_heads=8, num_layers=2, patience=100, save_dir='.', seed=[0, 1, 2, 3, 4], task='node_classification'
, weight_decay=0.0005)
Epoch: 390, Train: 1.0000, Val: 0.7360:  76%| | 380/500 [00:02<00:00, 139.66it/s]
Test accuracy = 0.779
Epoch: 220, Train: 1.0000, Val: 0.7620:  43%| | 215/500 [00:01<00:01, 154.84it/s]
Test accuracy = 0.798
Epoch: 395, Train: 1.0000, Val: 0.7680:  76%| | 378/500 [00:02<00:00, 159.16it/s]
Test accuracy = 0.786
Epoch: 252, Train: 1.0000, Val: 0.7720:  49%| | 244/500 [00:01<00:01, 130.41it/s]
Test accuracy = 0.802
Epoch: 303, Train: 1.0000, Val: 0.7580:  57%| | 287/500 [00:01<00:01, 155.30it/s]
Test accuracy = 0.782
Epoch: 173, Train: 1.0000, Val: 0.7820:  34%| | 172/500 [00:02<00:04, 67.48it/s]
Test accuracy = 0.803
Epoch: 198, Train: 1.0000, Val: 0.7740:  39%| | 195/500 [00:03<00:04, 61.49it/s]
Test accuracy = 0.818
Epoch: 147, Train: 1.0000, Val: 0.7760:  28%| | 140/500 [00:02<00:05, 68.21it/s]
Test accuracy = 0.805
Epoch: 195, Train: 1.0000, Val: 0.7800:  39%| | 195/500 [00:02<00:04, 70.55it/s]
Test accuracy = 0.811
Epoch: 156, Train: 1.0000, Val: 0.7780:  30%| | 151/500 [00:02<00:05, 67.44it/s]
| Variant             | Acc           |
|---------------------|---------------|
| ('cora', 'pyg_gcn') | 0.7894±0.0090 |
| ('cora', 'pyg_gat') | 0.8092±0.0052 |
[13] 0:~/cogdl*                                                    "ubuntu" 14:08 23-Jun-20
```

CogDL使用方法 – 自定义数据集

- 需要先在datasets目录下注册一个的数据集类。
- 这是一个将PyG的数据集导入到CogDL的示例。

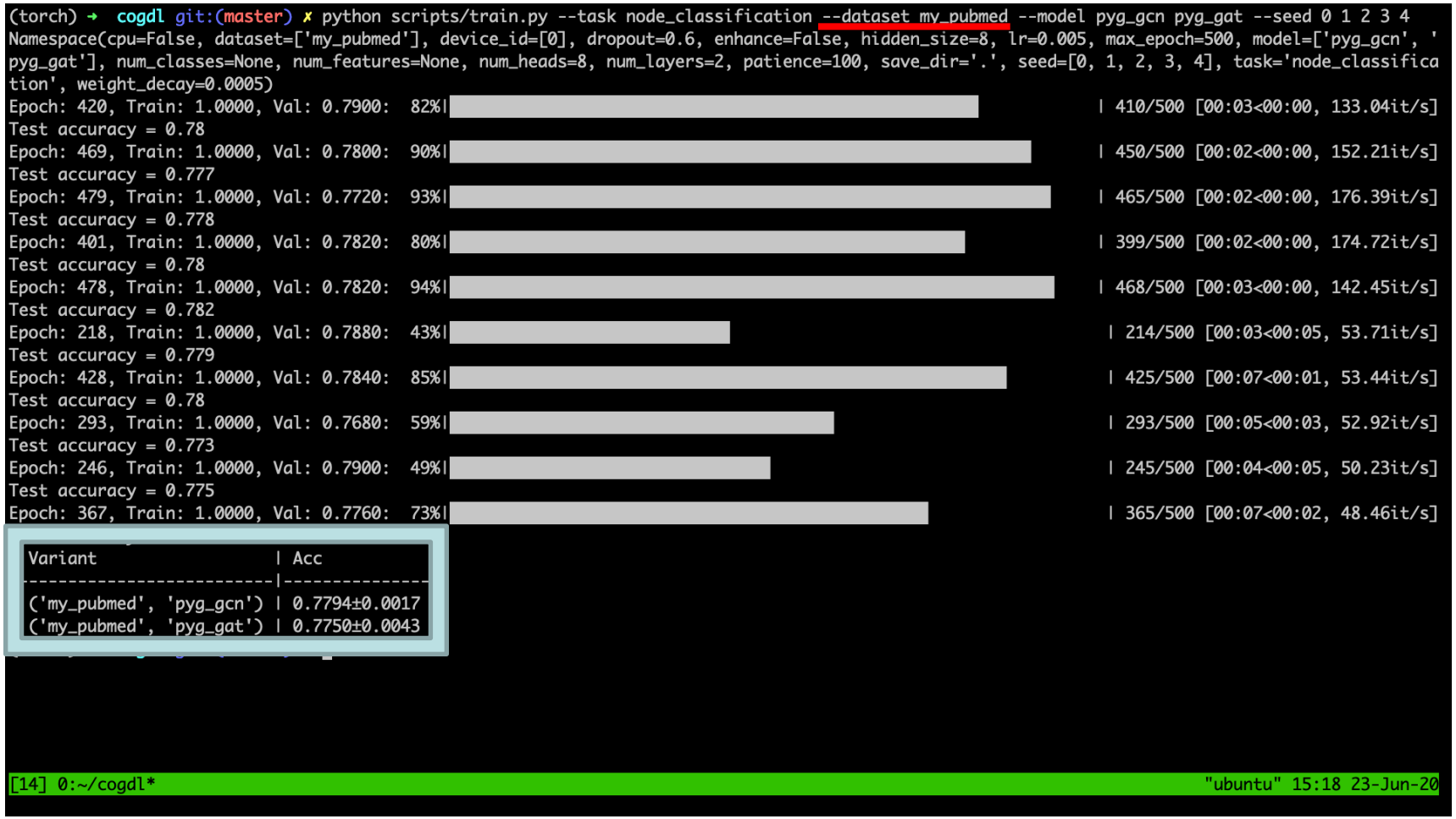
```python
import os.path as osp

import torch

import torch_geometric.transforms as T
from torch_geometric.datasets import Planetoid

from . import register_dataset


@register_dataset("my_pubmed")
class MyPubmedDataset(Planetoid):
    def __init__(self):
        dataset = "Pubmed"
        path = osp.join(
            osp.dirname(osp.realpath(__file__)), "../..", "data", "MyPubmed"
        )

        super(MyPubmedDataset, self).__init__(
            path, dataset, transform=T.TargetIndegree()
        )
```

注册一个名为my_pubmed的数据集

调用PyG的基类来完成数据集的下载和导入

CogDL使用方法 – 自定义数据集

- 在命令行中指定自定义数据集的名称。



CogDL使用方法 – 自定义模型

- 自己实现一个两层的GCN的示例。

```python
import torch
import torch.nn as nn
import torch.nn.functional as F
from torch_geometric.nn.conv import GCNConv

from .. import BaseModel, register_model

@register_model("my_gcn")
class MyGCN(BaseModel):
    @staticmethod
    def add_args(parser):
        parser.add_argument("--num-features", type=int)
        parser.add_argument("--num-classes", type=int)

    @classmethod
    def build_model_from_args(cls, args):
        return cls(
            args.num_features,
            args.num_classes,
        )

    def __init__(self, num_features, num_classes):
        super(MyGCN, self).__init__()

        self.num_features = num_features
        self.num_classes = num_classes
        self.conv1 = GCNConv(num_features, 64)
        self.conv2 = GCNConv(64, num_classes)

    def forward(self, x, edge_index):
        x = F.relu(self.conv1(x, edge_index))
        x = F.dropout(x, p=0.5, training=self.training)
        x = self.conv2(x, edge_index)
        return F.log_softmax(x, dim=1)

    def loss(self, data):
        return F.nll_loss(
            self.forward(data.x, data.edge_index)[data.train_mask],
            data.y[data.train_mask],
        )

    def predict(self, data):
        return self.forward(data.x, data.edge_index)
```

定义模型超参

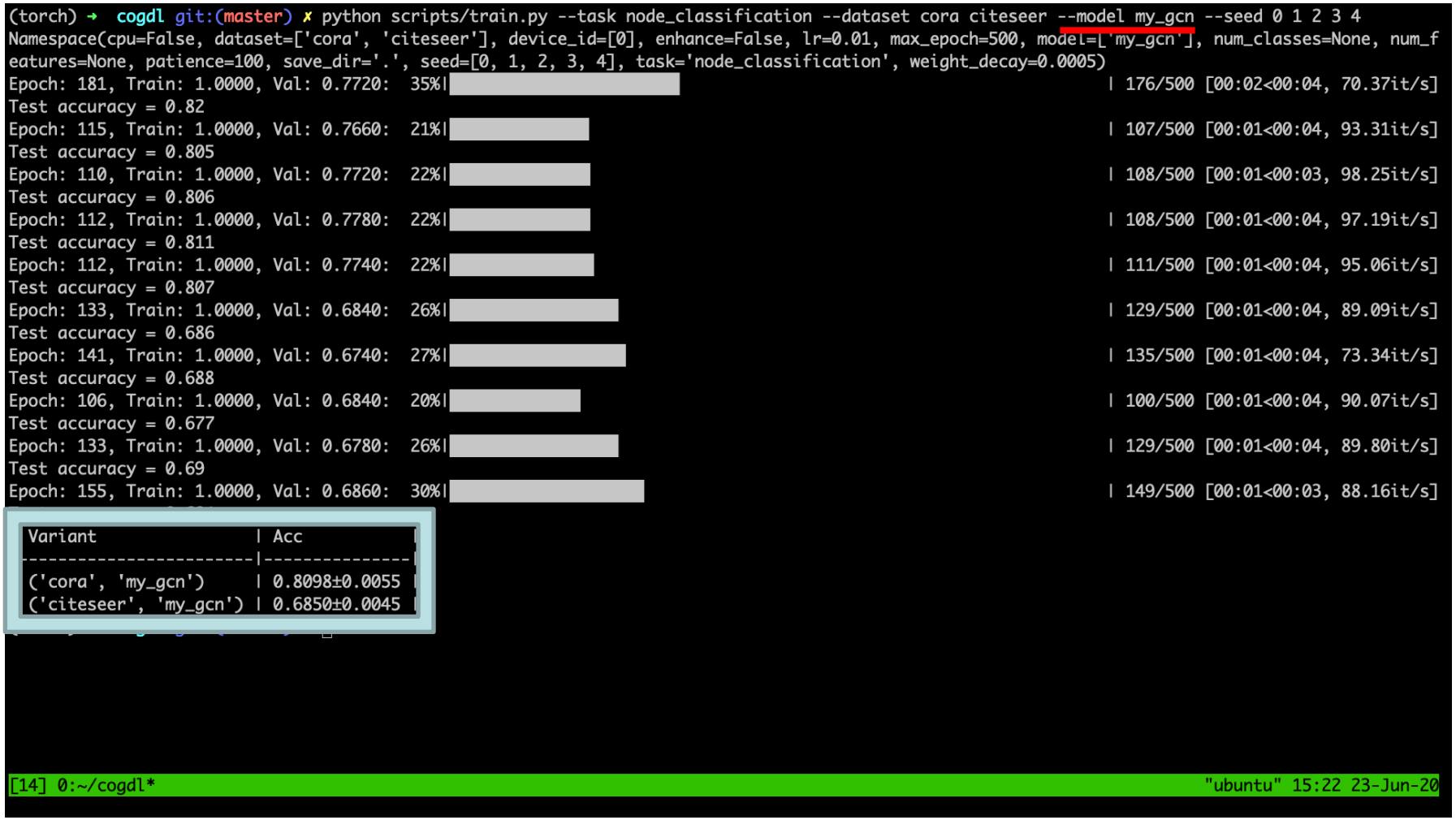
定义图卷积层

定义模型流程

损失函数和预测方式

CogDL使用方法 – 自定义模型

- 在命令行中指定自定义模型的名称。



CogDL算法流程

```python
class BaseModel(nn.Module):
    @staticmethod
    def add_args(parser):
        """Add model-specific arguments to the parser."""
        pass

    @classmethod
    def build_model_from_args(cls, args):
        """Build a new model instance."""
        raise NotImplementedError(
            "Models must implement the build_model_from_args method"
        )
```

```python
@register_model("spectral")
class Spectral(BaseModel):
    @staticmethod
    def add_args(parser):
        """Add model-specific arguments to the parser."""
        pass

    @classmethod
    def build_model_from_args(cls, args):
        return cls(args.hidden_size)

    def __init__(self, dimension):
        super(Spectral, self).__init__()
        self.dimension = dimension

    def train(self, G):
        matrix = nx.normalized_laplacian_matrix(G).todense()
        matrix = np.eye(matrix.shape[0]) - np.asarray(matrix)
        ut, s, _ = sp.linalg.svds(matrix, self.dimension)
        emb_matrix = ut * np.sqrt(s)
        emb_matrix = preprocessing.normalize(emb_matrix, "l2")
        return emb_matrix
```

```python
class BaseTask(object):
    @staticmethod
    def add_args(parser):
        """Add task-specific arguments to the parser."""
        pass

    def __init__(self, args):
        pass

    def train(self, num_epoch):
        raise NotImplementedError
```

```python
@register_dataset("ppi")
class PPIDataset(MatlabMatrix):
    def __init__(self):
        dataset, filename = "ppi", "Homo_sapiens"
        url = "http://snap.stanford.edu/node2vec/"
        path = osp.join(osp.dirname(osp.realpath(__file__)), "../..", "data", dataset)
        super(PPIDataset, self).__init__(path, filename, url)
```

```python
class MatlabMatrix(Dataset):
    def __init__(self, root, name, url):
        self.name = name
        self.url = url
        super(MatlabMatrix, self).__init__(root)
        self.data = torch.load(self.processed_paths[0])
```

算法流程划分

模块组合

```python
if __name__ == "__main__":
    parser = options.get_training_parser()
    args, _ = parser.parse_known_args()
    args = options.parse_args_and_arch(parser, args)
    variants = list(
        gen_variants(dataset=args.dataset, model=args.model, seed=args.seed)
    )

    # Collect results
    results_dict = defaultdict(list)
    results = [main(args) for args in variant_args_generator(args, variants)]
    for variant, result in zip(variants, results):
        results_dict[variant[:-1]].append(result)

    col_names = ["Variant"] + list(results_dict[variant[:-1]][-1].keys())
    tab_data = tabulate_results(results_dict)
    print(tabulate(tab_data, headers=col_names, tablefmt="github"))
```

```python
@register_task("unsupervised_node_classification")
class UnsupervisedNodeClassification(BaseTask):
    """Node classification task."""

    @staticmethod
    def add_args(parser):
        """Add task-specific arguments to the parser."""
        # fmt: off
        parser.add_argument("--hidden-size", type=int, default=128)
        parser.add_argument("--num-shuffle", type=int, default=5)
        # fmt: on

    def __init__(self, args):
        super(UnsupervisedNodeClassification, self).__init__(args)
        dataset = build_dataset(args)
```

执行命令

结果

Python scripts/train.py --task unsupervised_node_classification --dataset ppi --model spectral

Variant	Micro-F1 0.1	Micro-F1 0.3	Micro-F1 0.5	Micro-F1 0.7	Micro-F1 0.9
('ppi', 'spectral')	0.1730±0.0000	0.2135±0.0000	0.2229±0.0000	0.2254±0.0000	0.2278±0.0000

自定义数据集或模型

- **提交你的前沿算法**：如果您有一个性能优异的算法并愿意发布出来，你可以在我们的代码仓库里提出一个issue。在验证该算法的原创性，创造性和效果后，我们将该算法的效果添加到我们的排行榜上。

- **添加你自己的数据集**：如果您有一个独特，有研究价值的数据集并且愿意发布出来，你可以在我们的代码仓库里提出一个issue，我们将把所以适合的模型在您的数据集上运行并更新我们的排行榜。

- **实现你自己的模型**：如果您有一个性能优秀的算法，并愿意在我们的工具包中实现它，以帮助更多的人，您可以创建一个pull request。

CogDL支持的算法

按照类型划分:

- 无监督结点表示学习算法
- 半监督结点表示学习算法
- 异构结点表示学习算法
- 图表示学习算法

算法

无监督结点表示学习算法

Algorithm	Directed	Weight	Shallow network	Matrix factorization	Sampling	Reproducibility
DeepWalk			✓			✓
LINE	✓	✓	✓		✓	✓
Node2vec	✓	✓	✓		✓	✓
SDNE	✓	✓	✓			✓
DNGR	✓	✓	✓			
HOPE	✓	✓		✓		✓
GraRep	✓	✓		✓		
NetMF	✓	✓		✓		✓
NetSMF		✓		✓	✓	✓
ProNE	✓	✓		✓		✓

算法

半监督结点表示学习算法

Algorithm	Weight	Sampling	Attention	Inductive	Reproducibility
Graph U-Net	✓	✓			✓
MixHop	✓				✓
Dr-GAT			✓	✓	✓
GAT			✓	✓	✓
DGI	✓	✓		✓	✓
GCN	✓			✓	✓
GraphSAGE	✓	✓		✓	✓
Chebyshev	✓			✓	✓

算法

异构结点表示学习算法

Algorithm	Multi-Node	Multi-Edge	Attribute	Supervised	MetaPath	Reproducibility
GATNE	✓	✓	✓		✓	✓
Metapath2vec	✓				✓	✓
PTE	✓					✓
Hin2vec	✓				✓	✓
GTN	✓		✓	✓	✓	✓
HAN	✓		✓	✓	✓	✓

算法

图表示学习算法

Algorithm	Node feature	Unsupervised	Graph kernel	Shallow network	Reproducibility
Infograph	✓	✓			✓
Diffpool	✓				✓
Graph2Vec		✓	✓	✓	✓
Sortpool	✓				✓
GIN	✓				✓
PATCHY_SAN	✓		✓		✓
DGCNN	✓				✓
DGK		✓	✓	✓	

排行榜

评测任务:

- 无监督结点分类
- 半监督结点分类
- 异构结点分类
- 链接预测
- 异构链接预测
- 图分类

排行榜

无监督结点分类

Rank	Method	PPI	Blogcatalog	Wikipedia
1	ProNE (Zhang et al, IJCAI'19)	**26.32**	**43.63**	57.64
2	NetMF (Qiu et al, WSDM'18)	24.86	43.49	**58.46**
3	Node2vec (Grover et al, KDD'16)	23.86	42.51	53.68
4	NetSMF (Qiu et at, WWW'19)	24.39	43.21	51.42
5	DeepWalk (Perozzi et al, KDD'14)	22.72	42.26	50.42
6	LINE (Tang et al, WWW'15)	23.15	39.29	49.83
7	Hope (Ou et al, KDD'16)	23.24	35.52	52.96
8	SDNE (Wang et al, KDD'16)	20.14	40.32	48.24
9	GraRep (Cao et al, CIKM'15)	20.96	34.35	51.84
10	DNGR (Cao et al, AAAI'16)	16.45	28.54	48.57

排行榜

半监督结点分类

Rank	Method	Cora	Citeseer	Pubmed
1	Graph U-Net (Gao et al., 2019)	**84.4 ± 0.6**	**73.2 ± 0.5**	79.6 ± 0.2
2	MixHop (Abu-El-Haija et al., ICML'19)	81.9 ± 0.4	71.4 ± 0.8	**80.8 ± 0.6**
3	DR-GAT (Zou et al., 2019)	83.6 ± 0.5	72.8 ± 0.8	79.1 ± 0.3
4	GAT (Veličković et al., ICLR'18)	83.0 ± 0.7	72.5 ± 0.7	79.0 ± 0.3
5	DGI (Veličković et al., ICLR'19)	82.3 ± 0.6	71.8 ± 0.7	76.8 ± 0.6
6	GCN (Kipf et al., ICLR'17)	81.4 ± 0.5	70.9 ± 0.5	79.0 ± 0.3
7	GraphSAGE (Hamilton et al., NeurIPS'17)	80.1 ± 0.2	66.2 ± 0.4	76.9 ± 0.7
8	Chebyshev (Defferrard et al., NeurIPS'16)	79.2 ± 1.4	69.3 ± 1.3	68.5 ± 1.2

排行榜

异构结点分类

Rank	Method	DBLP	ACM	IMDB
1	GTN (Yun et al, NeurIPS'19)	**92.03**	**90.85**	**59.24**
2	HAN (Xiao et al, WWW'19)	91.21	87.25	53.94
3	PTE (Tang et al, KDD'15)	78.65	87.44	48.91
4	Metapath2vec (Dong et al, KDD'17)	75.18	88.79	43.10
5	Hin2vec (Fu et al, CIKM'17)	74.31	84.66	44.04

排行榜

链接预测

Rank	Method	PPI	Wikipedia
1	ProNE (Zhang et al, IJCAI'19)	79.93	**82.74**
2	NetMF (Qiu et al, WSDM'18)	79.04	73.24
3	Hope (Ou et al, KDD'16)	**80.21**	68.89
4	LINE (Tang et al, WWW'15)	73.75	66.51
5	Node2vec (Grover et al, KDD'16)	70.19	66.60
6	NetSMF (Qiu et at, WWW'19)	68.64	67.52
7	DeepWalk (Perozzi et al, KDD'14)	69.65	65.93
8	SDNE (Wang et al, KDD'16)	54.87	60.72

排行榜

异构链接预测

Rank	Method	Amazon	YouTube	Twitter
1	GATNE (Cen et al, KDD'19)	97.44	**84.61**	**92.30**
2	NetMF (Qiu et al, WSDM'18)	**97.72**	82.53	73.75
3	ProNE (Zhang et al, IJCAI'19)	96.51	78.96	81.32
4	Node2vec (Grover et al, KDD'16)	86.86	74.01	78.30
5	DeepWalk (Perozzi et al, KDD'14)	92.54	74.31	60.29
6	LINE (Tang et al, WWW'15)	92.56	73.40	60.36
7	Hope (Ou et al, KDD'16)	94.39	74.66	70.61
8	GraRep (Cao et al, CIKM'15)	83.88	71.37	49.64

排行榜

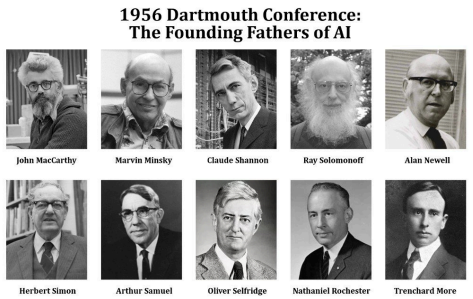
图分类

Rank	Method	MUTAG	IMDB-B	IMDB-M	PROTEINS	COLLAB
1	Infograph (Sun et al, ICLR'20)	88.95	74.50	51.33	73.93	78.14
2	GIN (Xu et al, ICLR'19)	88.33	76.70	50.80	72.86	79.52
3	DiffPool (Ying et al, NeuIPS'18)	85.18	74.30	50.73	75.30	77.20
4	SortPool (Zhang et al, AAAI'18)	85.61	75.20	51.07	74.11	79.98
5	Graph2Vec (Narayanan et al, CoRR'17)	83.68	73.90	52.27	73.30	85.58
6	PATCH_SAN (Niepert et al, ICML'16)	85.12	76.00	46.20	75.50	75.42
7	DGCNN (Wang et al, ACM Transactions on Graphics'17)	83.33	69.50	46.33	66.67	77.45
8	DGK (Yanardag et al, KDD'15)	83.68	55.00	40.40	72.59	/

Next

第三代人工智能，开启认知图谱新篇章

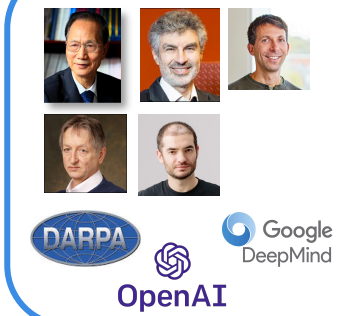
符号AI



符号模型/规则模型/感知机

第一代

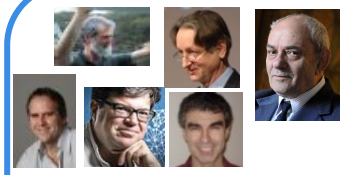
认知智能



张钹院士2016年提出第三代人工智能雏形，DARPA 2018年发布AI Next计划。推进数据与知识推理融合的计算、脑认知机理的融合

第三代

第二代



大数据驱动的统计学习方法初步实现了针对文本、图像、语音等的感知与识别

感知智能

目前急需的是高质量超大规模知识图谱（AI的基础设施）以及对知识的理解能力（面向认知的深度学习算法）

挑战与未来(Next 30)

意识

—让计算机具有自我意识

- 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.

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 Graph Neural Network 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