Network Embedding, Graph Neural Networks and Reasoning

Jie Tang
Computer Science
Tsinghua University
Networked World

Social Network

Knowledge Graph

COVID Graph
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?
Agenda

- Network Embedding
- Revisiting Network Embedding
- Graph Neural Network
- Revisiting Graph Neural Network
- GNN&Reasoning
- Revisiting GNN&Reasoning
Representation Learning on Networks

**Representation Learning/Graph Embedding**

- **d-dimensional vector, \( d << |V| \)**
  - Example: \( 0.8 \ 0.2 \ 0.3 \ \ldots \ 0.0 \ 0.0 \)

- Users with the same label are located in closer

- **e.g., node classification**

  - **label1**
  - **label2**
DeepWalk: Random Walk + Word2Vec

Random walk

One example RW path

SkipGram with Hierarchical softmax

Hierarchical softmax

\[
P(\{v_{i-w}, \ldots, v_{i+w}\} | v_i | \Phi(v_i))) = \prod_{j=i-w, j \neq i}^{i+w} P(v_j | \Phi(v_i))
\]

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

Parameter Learning

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

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

\[
p(c|v) = \frac{\exp(z_v^T z_c)}{\sum_{u \in V} \exp(z_v^T z_u)}
\]
Results: BlogCatalog

<table>
<thead>
<tr>
<th>% Labeled Nodes</th>
<th>10%</th>
<th>20%</th>
<th>30%</th>
<th>40%</th>
<th>50%</th>
<th>60%</th>
<th>70%</th>
<th>80%</th>
<th>90%</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DEEPWALK</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SpectralClustering</td>
<td>31.06</td>
<td>34.95</td>
<td>37.27</td>
<td>38.93</td>
<td>39.97</td>
<td>40.99</td>
<td>41.66</td>
<td>42.42</td>
<td>42.62</td>
</tr>
<tr>
<td>EdgeCluster</td>
<td>27.94</td>
<td>30.76</td>
<td>31.85</td>
<td>32.99</td>
<td>34.12</td>
<td>35.00</td>
<td>34.63</td>
<td>35.99</td>
<td>36.29</td>
</tr>
<tr>
<td>Modularity</td>
<td>27.35</td>
<td>30.74</td>
<td>31.77</td>
<td>32.97</td>
<td>34.09</td>
<td>36.13</td>
<td>36.08</td>
<td>37.23</td>
<td>38.18</td>
</tr>
<tr>
<td>wvRN</td>
<td>19.51</td>
<td>24.34</td>
<td>25.62</td>
<td>28.82</td>
<td>30.37</td>
<td>31.81</td>
<td>32.19</td>
<td>33.33</td>
<td>34.28</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>% Labeled Nodes</th>
<th>10%</th>
<th>20%</th>
<th>30%</th>
<th>40%</th>
<th>50%</th>
<th>60%</th>
<th>70%</th>
<th>80%</th>
<th>90%</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DEEPWALK</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SpectralClustering</td>
<td>19.14</td>
<td>23.57</td>
<td>25.97</td>
<td>27.46</td>
<td>28.31</td>
<td>29.46</td>
<td>30.13</td>
<td>31.38</td>
<td>31.78</td>
</tr>
<tr>
<td>EdgeCluster</td>
<td>16.16</td>
<td>19.16</td>
<td>20.48</td>
<td>22.00</td>
<td>23.00</td>
<td>23.64</td>
<td>23.82</td>
<td>24.61</td>
<td>24.92</td>
</tr>
<tr>
<td>Modularity</td>
<td>17.36</td>
<td>20.00</td>
<td>20.80</td>
<td>21.85</td>
<td>22.65</td>
<td>23.41</td>
<td>23.89</td>
<td>24.20</td>
<td>24.97</td>
</tr>
<tr>
<td>wvRN</td>
<td>6.25</td>
<td>10.13</td>
<td>11.64</td>
<td>14.24</td>
<td>15.86</td>
<td>17.18</td>
<td>17.98</td>
<td>18.86</td>
<td>19.57</td>
</tr>
<tr>
<td>Majority</td>
<td>2.52</td>
<td>2.55</td>
<td>2.52</td>
<td>2.58</td>
<td>2.58</td>
<td>2.63</td>
<td>2.61</td>
<td>2.48</td>
<td>2.62</td>
</tr>
</tbody>
</table>

- Feed the learned representation for node classification
- DeepWalk (node representation learning) *performs well*, especially when *labels are sparse*
Results: YouTube

<table>
<thead>
<tr>
<th>% Labeled Nodes</th>
<th>1%</th>
<th>2%</th>
<th>3%</th>
<th>4%</th>
<th>5%</th>
<th>6%</th>
<th>7%</th>
<th>8%</th>
<th>9%</th>
<th>10%</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEEPWALK</td>
<td>37.95</td>
<td>39.28</td>
<td>40.08</td>
<td>40.78</td>
<td>41.32</td>
<td>41.72</td>
<td>42.12</td>
<td>42.48</td>
<td>42.78</td>
<td>43.05</td>
</tr>
<tr>
<td>SpectralClustering</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EdgeCluster</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Modularity</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>wvRN</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Majority</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| Micro-F1(%)     | 23.90 | 31.68 | 35.53 | 36.76 | 37.81 | 38.63 | 38.94 | 39.46 | 39.92 | 40.07 |
| DeepWalk        |       |       |       |       |       |       |       |       |       |       |
| SpectralClustering |       |       |       |       |       |       |       |       |       |       |
| EdgeCluster     |       |       |       |       |       |       |       |       |       |       |
| Modularity      |       |       |       |       |       |       |       |       |       |       |
| wvRN            |       |       |       |       |       |       |       |       |       |       |
| Majority        |       |       |       |       |       |       |       |       |       |       |

| Macro-F1(%)     | 19.48 | 25.01 | 28.15 | 29.17 | 29.82 | 30.65 | 30.75 | 31.23 | 31.45 | 31.54 |
| DeepWalk        |       |       |       |       |       |       |       |       |       |       |
| SpectralClustering |       |       |       |       |       |       |       |       |       |       |
| EdgeCluster     |       |       |       |       |       |       |       |       |       |       |
| Modularity      |       |       |       |       |       |       |       |       |       |       |
| wvRN            | 13.15 | 15.78 | 19.66 | 20.9  | 23.31 | 25.43 | 27.08 | 26.48 | 28.33 | 28.89 |

- Similar results on YouTube
- Spectral Clustering does not scale to large graphs
• DeepWalk utilizes fixed-length, unbiased random walks to generate context for each node, can we do better?
Later...

- **LINE**\[^1\]**: explicitly preserves both *first-order* and *second-order* proximities.
- **PTE**\[^2\]**: learn heterogeneous text network embedding via a semi-supervised manner.
- **Node2vec**\[^3\]**: use a *biased* random walk to better explore node’s neighborhood.
- **Metapath2vec**\[^4\]**: *meta-path-based* random walks for heterogeneous networks.
- **GATNE**\[^5\]**: *inductive learning* for heterogeneous networks.

LINE: First-order proximity

Definition

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

Definition

- The **second-order** proximity between a pair of vertices \((v_i, v_j)\) in a network is the similarity between their neighborhood network structures.

- If no vertex is linked from/to both \(v_i\) and \(v_j\), the **second-order** proximity between \(v_i\) and \(v_j\) is 0.
LINE: Information Network Embedding

- Given a large network $G = (V, E)$, LINE aims to represent each vertex $v \in V$ into a low-dimensional space $R^d$
- Goal: learning a function $f_G : V \rightarrow R^d$, $d \ll |V|$. In $R^d$, both the first-order proximity and the second-order proximity between the vertices are preserved
- For each node $v \in V$, we use $u_i \in R^d$ to represent the corresponding low-dimensional vector representation.
LINE with First-order Proximity

- For each undirected $e_{ij}$, we define the joint probability between vertex $v_i$ and $v_j$ as follows:

$$p_1(v_i, v_j) = \frac{1}{1 + \exp(-\mathbf{u}_i^T \cdot \mathbf{u}_j)}$$

- It defines distribution $p(\cdot, \cdot)$ over the space $V \times V$, and its empirical probability can be defined as $\hat{p}_1(i, j) = \frac{w_{i,j}}{W}$ where $W = \sum_{i,j \in E} w_{i,j}$

**Objective Function:** Minimize the following objective function, where $d(\cdot, \cdot)$ is the distance between two distributions.

$$O_1 = d(\hat{p}_1(\cdot, \cdot), p_1(\cdot, \cdot))$$

- Instantiate $d(\cdot, \cdot)$ as KL-divergence:

$$O_1 = -\sum_{(i,j) \in E} w_{ij} \log p_1(v_i, v_j)$$
LINE with Second-order Proximity

- We first define the probability of “context” $v_j$ generated by vertex $v_i$ as:

$$p_2(v_j \mid v_i) = \frac{\exp(u'_j \cdot \bar{u}_i)}{\sum_{k=1}^{\lvert V \rvert} \exp(u'_k \cdot \bar{u}_i)}$$

- We minimize the following objective function:

$$O_2 = \sum_{i \in V} \lambda_i d(\hat{p}_2(\cdot \mid v_i), p_2(\cdot \mid v_i))$$

The empirical probability $\hat{p}_2(v_j \mid v_i)$ defined as:

$$\hat{p}_2(v_j \mid v_i) = \frac{w_{ij}}{d_i}$$

where $d_i$ is the out-degree of vertex $i$.

- Replacing $d(\cdot, \cdot)$ with KL-divergence, setting $\lambda_i = d_i$, we have:

$$O_2 = - \sum_{(i,j) \in E} w_{ij} \log p_2(v_j \mid v_i)$$
Model Optimization

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

$$\log\sigma(\vec{u}_j^T \cdot \vec{u}_i) + \sum_{i=1}^{K} E_{v_n} \sim P_n(v) [\log\sigma(-\vec{u}_n^T \cdot \vec{u}_i)]$$

  - the observed edges
  - the negative edges drawn from the noise distribution

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

- **LINE**\[^1\]**: explicitly preserves both *first-order* and *second-order* proximities.
- **PTE**\[^2\]**: learn heterogeneous text network embedding via a semi-supervised manner.
- **Node2vec**\[^3\]**: use a *biased* random walk to better explore node’s neighborhood.
- **Metapath2vec**\[^4\]**: *meta-path-based* random walks for heterogeneous networks.
- **GATNE**\[^5\]**: *inductive learning* for heterogeneous networks.

---

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

---

node2vec

- Biased random walk $R$ that given a node $v$ generates random walk neighborhood $N_{rw}(v)$
- Return parameter $p$:
  - Return back to the previous node
- In-out parameter $q$:
  - Moving outwards (DFS) vs. inwards (BFS)
node2vec

• Biased random walk $R$ that given a node $v$ generates random walk neighborhood $N_{rw}(v)$

• Return parameter $p$:
  • Return back to the previous node

• In-out parameter $q$:
  • Moving outwards (DFS) vs. inwards (BFS)

Interactions of characters in a novel:

$\begin{align*}
p=1, q=2 & \quad \text{Microscopic view of the network neighbourhood} \\
p=1, q=0.5 & \quad \text{Macroscopic view of the network neighbourhood}
\end{align*}$
Metapath2vec: Heterogeneous Random Walk

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

- How do we random walk over heterogeneous networks?
- How do we apply skip-gram over different types of nodes?
Metapath2vec: Heterogeneous Random Walk

Metapath2vec: Heterogeneous Random Walk

• Given a meta-path scheme

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

• The transition probability at step \( i \) is defined as

\[
p(v_{i+1} | v_i, \mathcal{P}) = \begin{cases} 
\frac{1}{|N_{t+1}(v_i)|} & (v_{i+1}, v_i) \in E, \phi(v_{i+1}) = t+1 \\
0 & (v_{i+1}, v_i) \in E, \phi(v_{i+1}) \neq t+1 \\
0 & (v_{i+1}, v_i) \notin E 
\end{cases}
\]

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

\[ p(v_{i+1} | v_i) = p(v_{i+1} | v_1), \text{ if } t = l \]
Metapath2vec: Heterogeneous Random Walk

\[ \mathcal{O}(X) = \log \sigma(X_{ct} \cdot X_v) + \sum_{k=1}^{K} \mathbb{E}_{u_t^k \sim P_t(u_t)} \log \sigma(-X_{u_t^k} \cdot X_v) \]
Application: Embedding Academic Graph

Microsoft Academic Graph & AMiner
Application 2: Node Clustering
Visualization

(a) DeepWalk/node2vec

(b) PTE

(c) metapath2vec

(d) metapath2vec++

word2vec [Mikolov, 2013]

http://projector.tensorflow.org/
GATNE: Attributed Multiplex Heterogeneous Network Embedding

- Recommend items to users by considering *Attributed Multiplex Heterogeneous Networks (AMHEN)*
## Different Types of Network Embedding

<table>
<thead>
<tr>
<th>Network Type</th>
<th>Method</th>
<th>Heterogeneity</th>
<th>Attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>Homogeneous Network (HON)</td>
<td>DeepWalk [27]</td>
<td>Single</td>
<td>/</td>
</tr>
<tr>
<td></td>
<td>LINE [35]</td>
<td>Single</td>
<td>/</td>
</tr>
<tr>
<td></td>
<td>node2vec [10]</td>
<td>Single</td>
<td>/</td>
</tr>
<tr>
<td></td>
<td>NetMF [29]</td>
<td>Single</td>
<td>/</td>
</tr>
<tr>
<td></td>
<td>NetSMF [28]</td>
<td>Single</td>
<td>/</td>
</tr>
<tr>
<td>Attributed Homogeneous Network (AHON)</td>
<td>TADW [41]</td>
<td>Single</td>
<td>Attributed</td>
</tr>
<tr>
<td></td>
<td>LANE [16]</td>
<td>Single</td>
<td>Attributed</td>
</tr>
<tr>
<td></td>
<td>AANE [15]</td>
<td>Single</td>
<td>Attributed</td>
</tr>
<tr>
<td></td>
<td>SNE [20]</td>
<td>Single</td>
<td>Attributed</td>
</tr>
<tr>
<td></td>
<td>DANE [9]</td>
<td>Single</td>
<td>Attributed</td>
</tr>
<tr>
<td></td>
<td>ANRL [44]</td>
<td>Single</td>
<td>Attributed</td>
</tr>
<tr>
<td>Heterogeneous Network (HEN)</td>
<td>PTE [34]</td>
<td>Multi</td>
<td>/</td>
</tr>
<tr>
<td></td>
<td>metapath2vec [7]</td>
<td>Single</td>
<td>/</td>
</tr>
<tr>
<td></td>
<td>HERec [31]</td>
<td>Single</td>
<td>/</td>
</tr>
<tr>
<td>Multiplex Heterogeneous Network (MHEN)</td>
<td>PMNE [22]</td>
<td>Single</td>
<td>/</td>
</tr>
<tr>
<td></td>
<td>MNE [43]</td>
<td>Multi</td>
<td>/</td>
</tr>
<tr>
<td></td>
<td>mvn2vec [32]</td>
<td>Multi</td>
<td>/</td>
</tr>
<tr>
<td></td>
<td>GATNE-T</td>
<td>Multi</td>
<td>Multi</td>
</tr>
<tr>
<td>Attributed MHEN (AMHEN)</td>
<td>GATNE-I</td>
<td>Multi</td>
<td>Multi</td>
</tr>
</tbody>
</table>
Multiplex Heterogeneous Graph Embedding

## Recommendation Results

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

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

GATNE outperforms all sorts of baselines in the various datasets.

**Code available at [https://github.com/THUDM/GATNE](https://github.com/THUDM/GATNE)**
Alibaba Offline A/B Tests

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

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

• Network Embedding
• Revisiting Network Embedding
• Graph Neural Network
• Revisiting Graph Neural Network
• GNN&Reasoning
• Revisiting GNN&Reasoning
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

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Closed Matrix Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>DeepWalk</td>
<td>[ \log \left( \text{vol}(G) \left( \frac{1}{T} \sum_{r=1}^{T} (D^{-1} A)^r \right) D^{-1} \right) - \log b ]</td>
</tr>
<tr>
<td>LINE</td>
<td>[ \log \left( \text{vol}(G) D^{-1} A D^{-1} \right) - \log b ]</td>
</tr>
</tbody>
</table>
| PTE          | \[ \log \left( \alpha \text{vol}(G_{ww}) (D_{row}^{ww})^{-1} A_{ww} (D_{col}^{ww})^{-1} \right) \]  
|              | \[ \beta \text{vol}(G_{dw}) (D_{row}^{dw})^{-1} A_{dw} (D_{col}^{dw})^{-1} \]  
|              | \[ \gamma \text{vol}(G_{lw}) (D_{row}^{lw})^{-1} A_{lw} (D_{col}^{lw})^{-1} \]  |
| node2vec     | \[ \log \left( \frac{1}{2T} \sum_{r=1}^{T} \left( \sum_{u} X_{w,u} P_{c,w,u}^{r} + \sum_{u} X_{c,u} P_{w,c,u}^{r} \right) \right) \]  
|              | \[ \left( \sum_{u} X_{w,u} \right) \left( \sum_{u} X_{c,u} \right) \]  

- **A**: \( A \in \mathbb{R}^{V \times V}_+ \) is G’s adjacency matrix with \( A_{i,j} \) as the edge weight between vertices \( i \) and \( j \);
- **D_{col}**: \( D_{col} = \text{diag}(A^\top e) \) is the diagonal matrix with column sum of \( A \);
- **D_{row}**: \( D_{row} = \text{diag}(Ae) \) is the diagonal matrix with row sum of \( A \);
- **D**: For undirected graphs \( A^\top = A \), \( D_{col} = D_{row} \). For brevity, \( D \) represents both \( D_{col} \) & \( D_{row} \).
- **D**: \( D = \text{diag}(d_1, \ldots, d_{|V|}) \), where \( d_i \) represents generalized degree of vertex \( i \);
- **vol(G)**: \( \text{vol}(G) = \sum_i \sum_j A_{i,j} = \sum_i d_i \) 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

\[
\log \left( \frac{\text{vol}(G)}{b} \left( \frac{1}{T} \sum_{r=1}^{T} (D^{-1}A)^r \right) D^{-1} \right)
\]

\[
\text{vol}(G) = \sum_i \sum_j A_{ij}
\]

A: Adjacency matrix    \( b \): #negative samples
D: Degree matrix        \( T \): context window size
Skip gram with negative sampling

Skip-gram with negative sampling (SGNS)

- SGNS maintains a multiset $\mathcal{D}$ that counts the occurrence of each word-context pair $(w, c)$

- Objective

$$
\mathcal{L} = \sum_w \sum_c (\#(w, c) \log g(x_w^T x_c) + \frac{b \#(w) \#(c)}{|\mathcal{D}|} \log g(-x_w^T x_c))
$$

- For sufficiently large dimension $d$, the objective above is equivalent to factorizing the PMI matrix

$$
\log \frac{\#(w, c)|\mathcal{D}|}{b \#(w) \#(c)}
$$

Levy and Goldberg. Neural word embeddings as implicit matrix factorization. In NIPS 2014
Understanding random walk + skip gram

1. for $n = 1, 2, \ldots, N$ do
2.    Pick $w_1^n$ according to a probability distribution $P(w_1)$;
3.    Generate a vertex sequence $(w_1^n, \ldots, w_L^n)$ of length $L$ by a random walk on network $G$;
4.   for $j = 1, 2, \ldots, L - T$ do
5.     for $r = 1, \ldots, T$ do
6.     \hspace{1em} Add vertex-context pair $(w_j^n, w_{j+r}^n)$ to multiset $D$;
7.     \hspace{1em} Add vertex-context pair $(w_{j+r}^n, w_j^n)$ to multiset $D$;
8.  Run SGNS on $D$ with $b$ negative samples.
Understanding random walk + skip gram
Suppose the multiset $\mathcal{D}$ is constructed based on random walk on graphs, can we interpret $\log \frac{\#(w,c)\mid \mathcal{D}\mid}{b\#(w)\#(c)}$ with graph structures?
Understanding random walk + skip gram

- Partition the multiset $\mathcal{D}$ into several sub-multisets according to the way in which each node and its context appear in a random walk node sequence.
- More formally, for $r = 1, 2, \ldots, T$, we define

$$\mathcal{D}_{\rightarrow} = \{(w, c) : (w, c) \in \mathcal{D}, w = w^n_j, c = w^n_{j+r}\}$$

$$\mathcal{D}_{\leftarrow} = \{(w, c) : (w, c) \in \mathcal{D}, w = w^n_{j+r}, c = w^n_j\}$$
Understanding random walk + skip gram

\[
\log \left( \frac{\#(w, c) |D|}{b \#(w) \cdot \#(c)} \right) = \log \left( \frac{\#(w, c) |D|}{\#(w) |D| \#(c) |D|} \right)
\]

the length of random walk \( L \to \infty \)

\[
P = D^{-1} A
\]

\[
\begin{align*}
\frac{\#(w, c) |D|}{\#(w) \cdot \#(c)} & \quad \frac{\#(w, c) |D|}{\#(w) |D| \cdot \#(c) |D|} \\
\frac{d_w}{vol(G)} & \quad \frac{d_c}{vol(G)} \\
\frac{1}{2T} \sum_{r=1}^{T} \left( \frac{d_w}{vol(G)} (P^r)_{w, c} + \frac{d_c}{vol(G)} (P^r)_{c, w} \right) & \quad \frac{d_w}{vol(G)} \cdot \frac{d_c}{vol(G)} \\
\frac{1}{2T} \sum_{r=1}^{T} (P^r)_{w, c} + \frac{1}{d_w} \sum_{r=1}^{T} (P^r)_{c, w} & \quad \frac{1}{d_c} \sum_{r=1}^{T} (P^r)_{w, c} + \frac{1}{d_w} \sum_{r=1}^{T} (P^r)_{c, w} \\
\end{align*}
\]
\[
\frac{\#(w, c)}{\#(w) \cdot \#(c)} \xrightarrow{p} \frac{\text{vol}(G)}{2T} \left( \frac{1}{d_c} \sum_{r=1}^{T} (P^r)_{w,c} + \frac{1}{d_w} \sum_{r=1}^{T} (P^r)_{c,w} \right)
\]

\[
= \frac{\text{vol}(G)}{2T} \left( \sum_{r=1}^{T} P^r D^{-1} + \sum_{r=1}^{T} D^{-1} (P^r)^T \right)
\]

\[
= \frac{\text{vol}(G)}{2T} \left( \sum_{r=1}^{T} D^{-1} A \times \cdots \times D^{-1} A D^{-1} + \sum_{r=1}^{T} D^{-1} A D^{-1} \times \cdots \times A D^{-1} \right)
\]

\[
= \frac{\text{vol}(G)}{T} \sum_{r=1}^{T} D^{-1} A \times \cdots \times D^{-1} A D^{-1}
\]

\[
= \text{vol}(G) \left( \frac{1}{T} \sum_{r=1}^{T} P^r \right) D^{-1}.
\]
DeepWalk is factorizing a matrix

DeepWalk is asymptotically and implicitly factorizing

\[
\log \left( \frac{\text{vol}(G)}{b} \left( \frac{1}{T} \sum_{r=1}^{T} (D^{-1}A)^{r} \right) D^{-1} \right)
\]

\[\text{vol}(G) = \sum_{i} \sum_{j} A_{ij}\]

- **A**: Adjacency matrix
- **D**: Degree matrix
- **b**: #negative samples
- **T**: context window size
LINE

Objective of LINE:

\[ \mathcal{L} = \sum_{i=1}^{|V|} \sum_{j=1}^{|V|} \left( A_{i,j} \log g ( x_i^T y_j ) + \frac{b d_i d_j}{\text{vol}(G)} \log g (-x_i^T y_j) \right). \]

Align it with the Objective of SGNS:

\[ \mathcal{L} = \sum_{w} \sum_{c} \left( \#(w, c) \log g ( x_w^T y_c ) + \frac{b \#(w) \#(c)}{|\mathcal{D}|} \log g (-x_w^T y_c) \right). \]

LINE is actually factorizing

\[ \log \left( \frac{\text{vol}(G)}{b} D^{-1} A D^{-1} \right) \]

Recall DeepWalk’s matrix form:

\[ \log \left( \frac{\text{vol}(G)}{b} \left( \frac{1}{T} \sum_{r=1}^{T} (D^{-1} A)^r \right) D^{-1} \right). \]

Observation: LINE is a special case of DeepWalk \((T = 1)\).
PTE

Figure 2: Heterogeneous Text Network.

$$\log \left( \begin{bmatrix} \alpha \text{vol}(G_{ww}) (D_{row}^{ww})^{-1} A_{ww} (D_{col}^{ww})^{-1} \\ \beta \text{vol}(G_{dw}) (D_{row}^{dw})^{-1} A_{dw} (D_{col}^{dw})^{-1} \\ \gamma \text{vol}(G_{lw}) (D_{row}^{lw})^{-1} A_{lw} (D_{col}^{lw})^{-1} \end{bmatrix} \right) - \log b,$$
Understanding node2vec

\[ T_{u,v,w} = \begin{cases} \frac{1}{p} & (u, v) \in E, (v, w) \in E, u = w; \\ 1 & (u, v) \in E, (v, w) \in E, u \neq w, (w, u) \in E; \\ \frac{1}{q} & (u, v) \in E, (v, w) \in E, u \neq w, (w, u) \notin E; \\ 0 & \text{otherwise}. \end{cases} \]

\[ P_{u,v,w} = \text{Prob} (w_{j+1} = u | w_j = v, w_{j-1} = w) = \frac{T_{u,v,w}}{\sum_u T_{u,v,w}}. \]

Stationary Distribution

\[ \sum_w P_{u,v,w} X_{v,w} = X_{u,v} \]

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

Understanding node2vec

Theorem

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

\[
\log \left( \frac{\frac{1}{2T} \sum_{\tau=1}^{T} \left( \sum_{u} X_{w,u} P_{c,w,u}^\tau + \sum_{u} X_{c,u} P_{w,c,u}^\tau \right)}{b \left( \sum_{u} X_{w,u} \right) \left( \sum_{u} X_{c,u} \right)} \right)
\]
Can we directly factorize the derived matrices for learning embeddings?
NetMF

• DeepWalk is implicitly factorizing

\[ M = \log \left( \frac{\text{vol}(G)}{b} \left( \frac{1}{T} \sum_{r=1}^{T} (D^{-1} A)^r \right) D^{-1} \right) \]

• NetMF is explicitly factorizing

\[ M = \log \left( \frac{\text{vol}(G)}{b} \left( \frac{1}{T} \sum_{r=1}^{T} (D^{-1} A)^r \right) D^{-1} \right) \]
NetMF

• DeepWalk is implicitly factorizing

\[ M = \log \left( \frac{\text{vol}(G)}{b} \right) \left( \frac{1}{T} \sum_{r=1}^{T} \left( D^{-1} A \right)^r \right) D^{-1} \]

Recall that in random walk + skip-gram based embedding models:

\( z_v^T z_c \rightarrow \) the probability that node \( v \) and context \( c \) appear on a random walk path

• NetMF is explicitly factorizing

\[ M = \log \left( \frac{\text{vol}(G)}{b} \right) \left( \frac{1}{T} \sum_{r=1}^{T} \left( D^{-1} A \right)^r \right) D^{-1} \]

\( z_v^T z_c \rightarrow \) the similarity score \( M_{vc} \) between node \( v \) and context \( c \) defined by this matrix
NetMF

Approximate $D^{-1/2}AD^{-1/2}$ with its top-$h$ eigenpairs $U_h \Lambda_h U_h^T$

The Arnoldi algorithm [1] for significant time reduction

1. Eigen-decomposition $D^{-1/2}AD^{-1/2} \approx U_h \Lambda_h U_h^T$;
2. Approximate $\hat{M}$ with
   $$\hat{M} = \frac{\text{vol}(G)}{b} D^{-1/2} U_h \left( \frac{1}{T} \sum_{r=1}^{T} \Lambda_h^r \right) U_h^T D^{-1/2};$$
3. Compute $\hat{M}' = \max(\hat{M}, 1)$;
4. Rank-$d$ approximation by SVD: $\log \hat{M}' = U_d \Sigma_d V_d^T$;
5. return $U_d \sqrt{\Sigma_d}$ as network embedding.

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

- According to Frobenius norm’s property

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

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

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

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

- Also because the property of NGL,

$$\sigma_s \left( \left( \frac{1}{T} \sum_{r=1}^{T} P^r \right) D^{-1} \right) \leq \frac{1}{T} \sum_{r=1}^{T} \lambda_{ps}^r \leq \frac{\text{vol}(G)}{bd_{\text{min}}} \sqrt{\sum_{j=k+1}^{n} \left( \frac{1}{T} \sum_{r=1}^{T} \lambda_j^r \right)^2}$$
Experimental Results

Predictive performance on varying the ratio of training data;
The $x$-axis represents the ratio of labeled data (%)
Network Embedding as Matrix Factorization

- DeepWalk
  \[ \log \left( \frac{\text{vol}(G)}{b} \left( \frac{1}{T} \sum_{r=1}^{T} (D^{-1}A)^r \right) D^{-1} \right) \]

- LINE
  \[ \log \left( \frac{\text{vol}(G)}{b} D^{-1}AD^{-1} \right) \]

- PTE
  \[ \log \left( \begin{bmatrix} \alpha \text{vol}(G_{ww})(D_{row}^{ww})^{-1} A_{ww}(D_{col}^{ww})^{-1} \\ \beta \text{vol}(G_{dw})(D_{row}^{dw})^{-1} A_{dw}(D_{col}^{dw})^{-1} \\ \gamma \text{vol}(G_{lw})(D_{row}^{lw})^{-1} A_{lw}(D_{col}^{lw})^{-1} \end{bmatrix} \right) - \log b \]

- node2vec
  \[ \log \left( \frac{1}{2T} \sum_{r=1}^{T} \left( \frac{\sum_u X_{w,u}P^{r}_{c,w,u} + \sum_u X_{c,u}P^{r}_{w,c,u}}{b (\sum_u X_{w,u}) (\sum_u X_{c,u})} \right) \right) \]

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

Small world

Academic graph

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

For random-walk matrix polynomial

$$L = D - \sum_{r=1}^{T} \alpha_r D \left( D^{-1} A \right)^r$$

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

One can construct a $(1 + \epsilon)$-spectral sparsifier $\tilde{L}$ with non-zeros

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

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

---

Sparsify $S$

For random-walk matrix polynomial  

\[ L = D - \sum_{r=1}^{T} \alpha_r D \left( D^{-1} A \right)^r \]

where \( \sum_{r=1}^{T} \alpha_r = 1 \) and \( \alpha_r \) non-negative

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

Suppose \( G = (V, E, A) \) and \( \tilde{G} = (V, \tilde{E}, \tilde{A}) \) are two weighted undirected networks. Let \( L = D_G - A \) and \( \tilde{L} = D_{\tilde{G}} - \tilde{A} \) be their Laplacian matrices, respectively. We define \( G \) and \( \tilde{G} \) are (1 + \( \epsilon \))-spectrally similar if

\[
\forall x \in \mathbb{R}^n, (1 - \epsilon) \cdot x^T \tilde{L} x \leq x^T L x \leq (1 + \epsilon) \cdot x^T \tilde{L} x.
\]

NetSMF --- Sparse

- Construct a random walk matrix polynomial sparsifier, $\tilde{L}$
- Construct a NetMF matrix sparsifier.
  \[
  \text{trunc}_\log^\circ \left( \frac{\text{vol}(G)}{b} D^{-1} (D - \tilde{L}) D^{-1} \right)
  \]
- Factorize the constructed matrix

Results

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

** Code available at [https://github.com/xptree/NetSMF](https://github.com/xptree/NetSMF)**
NE as Sparse Matrix Factorization

• node-context set $\mathcal{D} = E$ (sparsity)

• To avoid the trivial solution $(r_i = c_j, r_i^T c_j \to \infty, s.t. \hat{p} \to 1)$

• Local negative samples drawn from

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

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

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

- Let the partial derivative w.r.t. $r_i^T c_j$ be zero

$$r_i^T c_j = \ln p_{i,j} - \ln(\lambda P_{D,j}), \quad (v_i, v_j) \in \mathcal{D}$$

- Matrix to be factorized (sparse)

$$M_{i,j} = \begin{cases} 
\ln p_{i,j} - \ln(\lambda P_{D,j}) & , (v_i, v_j) \in \mathcal{D} \\
0 & , (v_i, v_j) \notin \mathcal{D}
\end{cases}$$
NE as Sparse Matrix Factorization

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

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

- **Sparsity** (local structure and local negative samples) \(\rightarrow\) much faster and scalable (e.g., randomized tSVD, \(O(|E|)\))

- The optimization (single thread) is much **faster** than SGD used in DeepWalk, LINE, etc. and is still **scalable**!!!

- Challenge: may **lose** high order information!

- Improvement via **spectral propagation**
Higher-order Cheeger’s inequality

- \( L = U \Lambda U^{-1} \), where \( \Lambda = \text{diag}([\lambda_1, \ldots, \lambda_n]) \) with \( 0 = \lambda_1 \leq \cdots \leq \lambda_n \)

- Bridge graph spectrum and graph partitioning

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

- \( k \)-way Cheeger constant \( \rho_G(k) \): reflects the effect of the graph partitioned into \( k \) parts. A smaller value of \( \rho_G(k) \) means a better partitioning effect.

Higher-order Cheeger’s inequality

- Bridge graph spectrum and graph partition
  \[ \frac{\lambda_k}{2} \leq \rho_G(k) \leq O(k^2)\sqrt{\lambda_k} \]

- Low eigenvalues control the **global** information

- High eigenvalues control the **local** information

- Example: \( \lambda_2 = 0 \iff \text{Graph is disconnected} \)

- Spectral propagation: propagate the initial network embedding in the **spectrally modulated** network!
NE Enhancement via Spectral Propagation

• the form of the spectral filter

\[ \widetilde{L} = U \text{diag}([g(\lambda_1), \ldots, g(\lambda_n)])U^T \]

\[ g(\lambda) = e^{-\frac{1}{2}[(\lambda-\mu)^2-1]\theta} \]

• Band-pass (low-pass, high-pass)

• pass eigenvalues within a certain range and weaken eigenvalues outside that range

• amplify local and global network information
Chebyshev Expansion for Efficiency

- Utilize truncated Chebyshev expansion to avoid explicit eigendecomposition and Fourier transform

\[
\widetilde{L} \approx U \sum_{i=0}^{k-1} c_i(\theta)T_i(\tilde{\Lambda})U^T = \sum_{i=0}^{k-1} c_i(\theta)T_i(\tilde{L})
\]

\[
\tilde{\Lambda} = -\frac{1}{2}[(\Lambda - \mu E_n)^2 - E_n], \quad \tilde{L} = -\frac{1}{2}[(L - \mu E_n)^2 - E_n]. \quad \tilde{\lambda} = \frac{1}{2}[(\lambda - \mu)^2 - 1] \in [-1, 1].
\]

- Calculate the coefficient of Chebyshev expansion

\[
c_i(\theta) = \begin{cases} 
\frac{1}{\pi} \int_{-1}^{1} \frac{T_i(x)e^{-x\theta}}{\sqrt{1-x^2}} \, dx = (-)^i I_i(\theta) & , i = 0 \\
\frac{2}{\pi} \int_{-1}^{1} \frac{T_i(x)e^{-x\theta}}{\sqrt{1-x^2}} \, dx = 2(-)^i I_i(\theta) & , i \neq 0
\end{cases}
\]

where \( I_i(\theta) \) is the modified Bessel function of the first kind
Chebyshev Expansion for Efficiency

• NE

\[ R_d \leftarrow D^{-1} A(E_n - \tilde{L})R_d \]

\[ = D^{-1} A\{E_n - [I_0(\theta)T_0(\tilde{L}) + 2 \sum_{i=1}^{k-1} (-)^{-1} I_i(\theta)T_i(\tilde{L})]\}R_d \]

• Denote \( \bar{R}_d^{(i)} = T_i(\tilde{L})R_d \) and calculate the Equation in the following recursive way

\[
\begin{cases}
\bar{R}_d^{(i)} = 2\tilde{L}\bar{R}_d^{(i-1)} - \bar{R}_d^{(i-2)} \\
\bar{R}_d^{(0)} = R_d \\
\bar{R}_d^{(1)} = \tilde{L}R_d
\end{cases}
\]
Complexity of ProNE

• Spectral propagation only involves sparse matrix multiplication! The complexity is linear!

• sparse matrix factorization + spectral propagation = $O(|V|d^2 + k|E|)$
Results

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

* 10 minutes on Youtube (~1M nodes)

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

<table>
<thead>
<tr>
<th>Dataset</th>
<th>training ratio</th>
<th>0.1</th>
<th>0.3</th>
<th>0.5</th>
<th>0.7</th>
<th>0.9</th>
</tr>
</thead>
<tbody>
<tr>
<td>DeepWalk</td>
<td></td>
<td>16.4</td>
<td>19.4</td>
<td>21.1</td>
<td>22.3</td>
<td>22.7</td>
</tr>
<tr>
<td>LINE</td>
<td></td>
<td>16.3</td>
<td>20.1</td>
<td>21.5</td>
<td>22.7</td>
<td>23.1</td>
</tr>
<tr>
<td>node2vec</td>
<td></td>
<td>16.2</td>
<td>19.7</td>
<td>21.6</td>
<td>23.1</td>
<td>24.1</td>
</tr>
<tr>
<td>GraRep</td>
<td></td>
<td>15.4</td>
<td>18.9</td>
<td>20.2</td>
<td>20.4</td>
<td>20.9</td>
</tr>
<tr>
<td>HOPE</td>
<td></td>
<td>16.4</td>
<td>19.8</td>
<td>21.0</td>
<td>21.7</td>
<td>22.5</td>
</tr>
<tr>
<td>ProNE (SMF)</td>
<td></td>
<td>15.8</td>
<td>20.6</td>
<td>22.7</td>
<td>23.7</td>
<td>24.2</td>
</tr>
<tr>
<td>ProNE</td>
<td>(±σ)</td>
<td>18.2</td>
<td>22.7</td>
<td>24.6</td>
<td>25.4</td>
<td>25.9</td>
</tr>
<tr>
<td></td>
<td>(±0.5)</td>
<td>(±0.3)</td>
<td>(±0.7)</td>
<td>(±1.0)</td>
<td>(±1.1)</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dataset</th>
<th>training ratio</th>
<th>0.01</th>
<th>0.03</th>
<th>0.05</th>
<th>0.07</th>
<th>0.09</th>
</tr>
</thead>
<tbody>
<tr>
<td>DeepWalk</td>
<td></td>
<td>49.3</td>
<td>55.0</td>
<td>57.1</td>
<td>57.9</td>
<td>58.4</td>
</tr>
<tr>
<td>LINE</td>
<td></td>
<td>48.7</td>
<td>52.6</td>
<td>53.5</td>
<td>54.1</td>
<td>54.5</td>
</tr>
<tr>
<td>node2vec</td>
<td></td>
<td>48.9</td>
<td>55.1</td>
<td>57.0</td>
<td>58.0</td>
<td>58.4</td>
</tr>
<tr>
<td>GraRep</td>
<td></td>
<td>50.5</td>
<td>52.6</td>
<td>53.2</td>
<td>53.5</td>
<td>53.8</td>
</tr>
<tr>
<td>HOPE</td>
<td></td>
<td>52.2</td>
<td>55.0</td>
<td>55.9</td>
<td>56.3</td>
<td>56.6</td>
</tr>
<tr>
<td>ProNE (SMF)</td>
<td></td>
<td>50.8</td>
<td>54.9</td>
<td>56.1</td>
<td>56.7</td>
<td>57.0</td>
</tr>
<tr>
<td>ProNE</td>
<td>(±σ)</td>
<td>48.8</td>
<td>52.2</td>
<td>58.0</td>
<td>58.8</td>
<td>59.2</td>
</tr>
<tr>
<td></td>
<td>(±1.0)</td>
<td>(±0.5)</td>
<td>(±0.2)</td>
<td>(±0.2)</td>
<td>(±0.1)</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dataset</th>
<th>training ratio</th>
<th>0.01</th>
<th>0.03</th>
<th>0.05</th>
<th>0.07</th>
<th>0.09</th>
</tr>
</thead>
<tbody>
<tr>
<td>DeepWalk</td>
<td></td>
<td>38.0</td>
<td>40.1</td>
<td>41.3</td>
<td>42.1</td>
<td>42.8</td>
</tr>
<tr>
<td>LINE</td>
<td></td>
<td>33.2</td>
<td>35.5</td>
<td>37.0</td>
<td>38.2</td>
<td>39.3</td>
</tr>
<tr>
<td>node2vec</td>
<td></td>
<td>36.5</td>
<td>40.2</td>
<td>41.2</td>
<td>41.7</td>
<td>42.1</td>
</tr>
<tr>
<td>ProNE (SMF)</td>
<td></td>
<td>38.2</td>
<td>41.4</td>
<td>42.3</td>
<td>42.9</td>
<td>43.3</td>
</tr>
<tr>
<td>ProNE</td>
<td>(±σ)</td>
<td>38.2</td>
<td>41.4</td>
<td>42.3</td>
<td>42.9</td>
<td>43.3</td>
</tr>
<tr>
<td></td>
<td>(±0.8)</td>
<td>(±0.3)</td>
<td>(±0.2)</td>
<td>(±0.2)</td>
<td>(±0.2)</td>
<td></td>
</tr>
</tbody>
</table>

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

* ProNE (SMF) = ProNE w/ only sparse matrix factorization

** Code available at https://github.com/THUDM/ProNE
Spectral Propagation: $k$-way Cheeeger

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

(a) ProDeepWalk
(b) ProLINE
(c) ProNode2vec
(d) ProGraRep
(e) ProHOPE
Input: Adjacency Matrix $A$

Output: Vector $Z$

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
Agenda

• Network Embedding
• Revisiting Network Embedding
• Graph Neural Network
• Revisiting Graph Neural Network
• GNN&Reasoning
• Revisiting GNN&Reasoning
From Shallow to Deep

• So far we have focused on shallow encoders, i.e. embedding lookups:
From Shallow to Deep

• Limitations of shallow encoding:
  – $O(|V|)$ parameters are needed: there is no parameter sharing and every node has its own unique embedding vector
  – Inherently “transductive”: It is impossible to generate embeddings for nodes that were not seen during training
  – Do not incorporate node features: Many graphs have features that we can and should leverage.

• We will now discuss deeper methods based on graph neural networks, i.e., encoder is a complex function that depends on graph structure.
A Naive Approach With Deep Neural Network

• Taking adjacency matrix $A$ and feature matrix $X$ into deep (fully connected neural network).

• But there are a huge number of parameters and you have to re-train the model if the network structure changes.

• So we need weight sharing (CNNs)!
GCN

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

- **FastGCN** (2018)
- **GraphSAGE** (2017)
- **GCN** (2017)
- **GraphSGAN** (2018)
- **GAT** (2018)

Diagram showing the timeline of recent GCN research with key milestones:

1. Sample neighborhood
2. Aggregate feature information from neighbors
3. Predict graph context and label using aggregated information

Diagram illustrates the propagation of information through the graph, from input layers to output layers, highlighting the advancements in GCN architecture.
GNN/GCN Model Architecture

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

\[ X = H^{(0)} \]

\[ \begin{align*}
H^{(l+1)} &= \sigma \left( \hat{A}H^{(l)}W^{(l)} \right) \\
Z &= H^{(N)}
\end{align*} \]

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

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

\[ h_v = f(h_a, h_b, h_c, h_d, h_e) \]
Convolutional neural network

\[ \sum wx \]
GNN: Graph Convolutional Networks

\[ h_v^k = \sigma(W_k) \sum_{u \in N(v) \cup v} \frac{h_u^{k-1}}{\sqrt{|N(u)||N(v)|}} \]

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

the example is from Yizhou's slides
GNN: Graph Convolutional Networks

\[ h^k_v = \sigma(W_k \sum_{u \in N(v) \cup v} \frac{h^{k-1}_u}{\sqrt{|N(u)||N(v)|}}) \]

- node \( v \)'s embedding at layer \( k \)
- the neighbors of node \( v \)
- Non-linear activation function (e.g., ReLU)
- parameters in layer \( k \)
GNN: Graph Convolutional Networks

\[ h^k_v = \sigma(W_k \sum_{u \in N(v)} \frac{h^k_u}{\sqrt{|N(u)||N(v)|}} + W_k \sum_v \frac{h^k_v}{\sqrt{|N(v)||N(v)|}}) \]

\[ D^{-1/2} A D^{-1/2} H^{(k)} W^{(k)} \]

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

\[ H^k = \sigma \left( D^{-\frac{1}{2}} (A + I) D^{-\frac{1}{2}} H^{(k)} W^{(k)} \right) \]

\[ Z = H^K \]
GNN: Graph Convolutional Networks

\[ H^k = \sigma \left( D^{-\frac{1}{2}}(A + I)D^{-\frac{1}{2}}H^{(k)}W^{(k)} \right) \]

GCN is one way of neighbor aggregations
- GraphSage
- Graph Attention
- \ldots \ldots
GraphSAGE Model Architecture

- 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

GraphSage

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

\[ h_v^k = \sigma([A_k \cdot \text{AGG}([h_u^{k-1}, \forall u \in N(v)]), B_k h_v^{k-1}]) \]

**GCN**

\[ h_v^k = \sigma(W_k \sum_{u \in N(v) \cup v} \frac{h_u^{k-1}}{\sqrt{|N(u)| |N(v)|})} \]

2. Slide snipping from “Hamilton & Tang, AAAI 2019 Tutorial on Graph Representation Learning”
GraphSage

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

### GCN

$$h_v^k = \sigma(W_k \sum_{u \in N(v) \cup v} \frac{h_u^{k-1}}{\sqrt{|N(u)| \cdot |N(v)|}}$$

### GraphSage

Instead of summation, it concatenate neighbor & self embeddings

$$h_v^k = \sigma([A_k \cdot \text{AGG}([h_u^{k-1}, \forall u \in N(v)]), B_k h_v^{k-1}])$$

---

2. Slide snipping from “Hamilton & Tang, AAAI 2019 Tutorial on Graph Representation Learning”
**Performance**

Table 1: Prediction results for the three datasets (micro-averaged F1 scores). Results for unsupervised and fully supervised GraphSAGE are shown. Analogous trends hold for macro-averaged scores.

<table>
<thead>
<tr>
<th>Name</th>
<th>Citation</th>
<th>Reddit</th>
<th>PPI</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Unsup. F1</td>
<td>Sup. F1</td>
<td>Unsup. F1</td>
</tr>
<tr>
<td>Random</td>
<td>0.206</td>
<td>0.206</td>
<td>0.043</td>
</tr>
<tr>
<td>Raw features</td>
<td>0.575</td>
<td>0.575</td>
<td>0.585</td>
</tr>
<tr>
<td>DeepWalk</td>
<td>0.565</td>
<td>0.565</td>
<td>0.324</td>
</tr>
<tr>
<td>DeepWalk + features</td>
<td>0.701</td>
<td>0.701</td>
<td>0.691</td>
</tr>
<tr>
<td>GraphSAGE-GCN</td>
<td>0.742</td>
<td>0.772</td>
<td><strong>0.908</strong></td>
</tr>
<tr>
<td>GraphSAGE-mean</td>
<td>0.778</td>
<td>0.820</td>
<td>0.897</td>
</tr>
<tr>
<td>GraphSAGE-LSTM</td>
<td>0.788</td>
<td>0.832</td>
<td><strong>0.907</strong></td>
</tr>
<tr>
<td>GraphSAGE-pool</td>
<td><strong>0.798</strong></td>
<td><strong>0.839</strong></td>
<td>0.892</td>
</tr>
</tbody>
</table>

% gain over feat. 39% 46% 55% 63% 19% 45%
Realistically, neighbors play different influences
Graph Attention Networks

$$h_v^k = \sigma(W_k \sum_{u \in N(v) \cup v} \frac{h_u^{k-1}}{\sqrt{|N(u)||N(v)|}})$$

GCN

Graph Attention

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

Learned attention weights

1. Velickovic et al. Graph Attention Networks. ICLR 2018
### GAT Results

#### Transductive

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

#### Inductive

<table>
<thead>
<tr>
<th>Method</th>
<th>PPI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random</td>
<td>0.396</td>
</tr>
<tr>
<td>MLP</td>
<td>0.422</td>
</tr>
<tr>
<td>GraphSAGE-GCN (Hamilton et al., 2017)</td>
<td>0.500</td>
</tr>
<tr>
<td>GraphSAGE-mean (Hamilton et al., 2017)</td>
<td>0.598</td>
</tr>
<tr>
<td>GraphSAGE-LSTM (Hamilton et al., 2017)</td>
<td>0.612</td>
</tr>
<tr>
<td>GraphSAGE-pool (Hamilton et al., 2017)</td>
<td>0.600</td>
</tr>
<tr>
<td>GraphSAGE*</td>
<td>0.768</td>
</tr>
<tr>
<td>Const-GAT (ours)</td>
<td>0.934 ± 0.006</td>
</tr>
<tr>
<td>GAT (ours)</td>
<td><strong>0.973 ± 0.002</strong></td>
</tr>
</tbody>
</table>
Agenda

- Network Embedding
- Revisiting Network Embedding
- Graph Neural Network
- Revisiting Graph Neural Network
- GNN&Reasoning
- Revisiting GNN&Reasoning
Setting: Semi-supervised Learning on Graphs

Input: Partially labeled attributed graph

Goal: Predict labels of unlabeled nodes

GCN
Challenges: non-robust

- \( H^{(l+1)} = \sigma(\hat{A}H^{(l)}W^{(l)}) \): Deterministic propagation
- Nodes are highly dependent with its neighborhoods, making GNNs non-robust.
Motivation: some problems of GCN

- $H^{(l+1)} = \sigma(\hat{A}H^{(l)}W^{(l)})$: 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.

\[
\text{supervised loss} + \text{consistency loss}
\]
Grand training details

Consistency Loss:

Distributions of a node:

\[ \overline{Z}_i = \frac{1}{S} \sum_{s=1}^{S} \overline{Z}_i^{(s)} \]

\[ \overline{Z}_{ik}' = \overline{Z}_{ik}' \bigg/ \sum_{j=0}^{C-1} \overline{Z}_{ij}' \]

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

Algorithm 2 Consistency Regularized Training for GRAND

Input:
- Adjacency matrix \( \hat{A} \), feature matrix \( X \in \mathbb{R}^{n \times d} \), times of augmentations in each epoch \( S \), DropNode probability \( \delta \).

Output:
- Prediction \( Z \).

1: while not convergence do
2:     for \( s = 1 : S \) do
3:         Apply DropNode via Algorithm 1: \( \tilde{X}^{(s)} \sim \text{DropNode}(X, \delta) \).
4:         Perform propagation: \( \overline{X}^{(s)} = \frac{1}{K+1} \sum_{k=0}^{K} \hat{A}^k \tilde{X}^{(s)} \).
5:         Predict class distribution using MLP: \( \tilde{Z}^{(s)} = P(Y|\overline{X}^{(s)}; \Theta) \).
6:     end for
7:     Compute supervised classification loss \( \mathcal{L}_{sup} \) via Eq. 4 and consistency regularization loss via Eq. 6.
8:     Update the parameters \( \Theta \) by gradients descending:
\[ \nabla_\Theta \mathcal{L}_{sup} + \lambda \mathcal{L}_{con} \]
9: end while
10: Output prediction \( Z \) 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

- Theoretically, Dropout is an adaptive L2 regularization.

**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.
## Results

<table>
<thead>
<tr>
<th>Category</th>
<th>Method</th>
<th>Cora</th>
<th>CiteSeer</th>
<th>Pubmed</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Graph Convolution</strong></td>
<td>GCN [27]</td>
<td>81.5</td>
<td>70.3</td>
<td>79.0</td>
</tr>
<tr>
<td></td>
<td>GAT [42]</td>
<td>83.0±0.7</td>
<td>72.5±0.7</td>
<td>79.0±0.3</td>
</tr>
<tr>
<td></td>
<td>Graph U-Net [15]</td>
<td>84.4±0.6</td>
<td>73.2±0.5</td>
<td>79.6±0.2</td>
</tr>
<tr>
<td></td>
<td>MixHop [2]</td>
<td>81.9±0.4</td>
<td>71.4±0.8</td>
<td>80.8±0.6</td>
</tr>
<tr>
<td></td>
<td>GMNN [36]</td>
<td>83.7</td>
<td>72.9</td>
<td>81.8</td>
</tr>
<tr>
<td></td>
<td>GraphNAS [16]</td>
<td>84.2±1.0</td>
<td>73.1±0.9</td>
<td>79.6±0.4</td>
</tr>
<tr>
<td><strong>Regularization based GCNs²</strong></td>
<td>VBAT [13]</td>
<td>83.6±0.5</td>
<td>74.0±0.6</td>
<td>79.9±0.4</td>
</tr>
<tr>
<td></td>
<td>G³NN [31]</td>
<td>82.5±0.2</td>
<td>74.4±0.3</td>
<td>77.9±0.4</td>
</tr>
<tr>
<td></td>
<td>GraphMix [43]</td>
<td>83.9±0.6</td>
<td>74.5±0.6</td>
<td>81.0±0.6</td>
</tr>
<tr>
<td></td>
<td>DropEdge [37]</td>
<td>82.8</td>
<td>72.3</td>
<td>79.6</td>
</tr>
<tr>
<td><strong>Sampling based GCNs³</strong></td>
<td>GraphSAGE [22]</td>
<td>78.9±0.8</td>
<td>67.4±0.7</td>
<td>77.8±0.6</td>
</tr>
<tr>
<td></td>
<td>FastGCN [9]</td>
<td>81.4±0.5</td>
<td>68.8±0.9</td>
<td>77.6±0.5</td>
</tr>
<tr>
<td><strong>Our methods</strong></td>
<td><strong>GRAND</strong></td>
<td><strong>85.4±0.4</strong></td>
<td><strong>75.4±0.4</strong></td>
<td><strong>82.7±0.6</strong></td>
</tr>
<tr>
<td></td>
<td>GRAND_GCN</td>
<td>84.5±0.3</td>
<td>74.2±0.3</td>
<td>80.0±0.3</td>
</tr>
<tr>
<td></td>
<td>GRAND_GAT</td>
<td>84.3±0.4</td>
<td>73.2±0.4</td>
<td>79.2±0.6</td>
</tr>
<tr>
<td></td>
<td>GRAND_dropout</td>
<td>84.9±0.4</td>
<td>75.0±0.3</td>
<td>81.7±1.0</td>
</tr>
</tbody>
</table>

Table 2: Summary of classification accuracy (%).
Ablation Study

Table 3: Ablation study results (%).

<table>
<thead>
<tr>
<th>Model</th>
<th>Cora</th>
<th>Citeseer</th>
<th>Pubmed</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>GRAND</strong></td>
<td>85.4±0.4</td>
<td>75.4±0.4</td>
<td>82.7±0.6</td>
</tr>
<tr>
<td>without CR</td>
<td>84.4 ±0.5</td>
<td>73.1 ±0.6</td>
<td>80.9 ±0.8</td>
</tr>
<tr>
<td>without multiple DropNode</td>
<td>84.7 ±0.4</td>
<td>74.8±0.4</td>
<td>81.0±1.1</td>
</tr>
<tr>
<td>without sharpening</td>
<td>84.6 ±0.4</td>
<td>72.2±0.6</td>
<td>81.6 ± 0.8</td>
</tr>
<tr>
<td>without CR and DropNode</td>
<td>83.2 ± 0.5</td>
<td>70.3 ± 0.6</td>
<td>78.5 ± 1.4</td>
</tr>
</tbody>
</table>
Robustness Analysis

Figure 4: Robustness: results under attacks on Cora.
Relieving Over-smoothing

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

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>
<thead>
<tr>
<th>Method</th>
<th>Cora Full</th>
<th>Coauthor CS</th>
<th>Coauthor Physics</th>
<th>Amazon Computer</th>
<th>Amazon Photo</th>
<th>AMiner CS</th>
</tr>
</thead>
<tbody>
<tr>
<td>GCN</td>
<td>62.2 ± 0.6</td>
<td>91.1 ± 0.5</td>
<td>92.8 ± 1.0</td>
<td>82.6 ± 2.4</td>
<td>91.2 ± 1.2</td>
<td>49.4 ± 1.7</td>
</tr>
<tr>
<td>GAT</td>
<td>51.9 ± 1.5</td>
<td>90.5 ± 0.6</td>
<td>92.5 ± 0.9</td>
<td>78.0 ± 19.0</td>
<td>85.7 ± 20.3</td>
<td>49.7 ± 1.6</td>
</tr>
<tr>
<td>GRAND</td>
<td>63.5 ± 0.6</td>
<td>92.9 ± 0.5</td>
<td>94.6 ± 0.5</td>
<td>85.7 ± 1.8</td>
<td>92.5 ± 1.7</td>
<td>52.2 ± 1.3</td>
</tr>
</tbody>
</table>
• Again, what are the **fundamentals** underlying the different models?

• Understanding GNNs as **Signal Rescaling**
Case Study - GAT

- Message-Passing as attention coefficients

\[ \alpha_{ij} = \frac{\exp \left( \text{LeakyReLU} \left( \tilde{a}^T [W \vec{h}_i || W \vec{h}_j] \right) \right)}{\sum_k \tilde{A}_{ik} \neq 0 \exp \left( \text{LeakyReLU} \left( \tilde{a}^T [W \vec{h}_i || W \vec{h}_k] \right) \right)} \]

split \( \tilde{a} \) into \( \tilde{p} \) and \( \tilde{q} \)

\[ \alpha_{ij} = \frac{\tilde{A}_{ij} \cdot \exp \left( \text{LeakyReLU} \left( (WH\tilde{p})_i + (WH\tilde{q})_j \right) \right)}{\sum_k \tilde{A}_{ik} \cdot \exp \left( \text{LeakyReLU} \left( (WH\tilde{p})_i + (WH\tilde{q})_k \right) \right)} \]

convert to matrix form

\[
\begin{align*}
H' &= \sigma(\mathcal{N}(P\mathcal{L} + \mathcal{L}Q)HW) \\
\mathcal{L} &= A + I_N \\
\mathcal{N}(\cdot) &= \text{SparseRowSoftmax}(\text{LeakyReLU}(\cdot)) \\
P_{ii} &= (HW\tilde{p})_i, \quad Q_{ii} = (HW\tilde{q})_i
\end{align*}
\]
Case Study – GraphSAGE-mean

• Message-Passing as heuristic sampling and aggregating scheme

\[ h^k_v \leftarrow \sigma \left( W \cdot \text{MEAN} \left( \{h^k_{v-1}\} \cup \{h^k_{u-1}, j \in \text{Sample}(i)\} \right) \right) \]

convert to matrix form

\[
\begin{align*}
H' &= \sigma \left( \mathcal{N} \left( \mathcal{L} \odot Q \right) HW \right) \\
\mathcal{L} &= A + I_N \\
\mathcal{N}(M) &= D^{-1}M, \quad D_{ii} = \sum_j M_{ij} \\
\{Q_{ij} | \hat{A}_{ij} \neq 0\} &\sim \text{Sample (deg}(i), S_i) \end{align*}
\]
Case Study - FastGCN

- Message-Passing as heuristic sampling and aggregating scheme

\[
H^{(l+1)}(v,:) = \sigma \left( \frac{1}{t_l} \sum_{j=1}^{t_l} \frac{\hat{A}(v, u_j^{(l)}) H^{(l)}(u_j^{(l)},:), W^{(l)}}{q(u_j^{(l)})} \right)
\]

\[
q(u) = ||\hat{A}(:, u)||^2 / \sum_{u' \in V} ||\hat{A}(:, u')||^2, u \in V
\]

\[\text{convert to matrix form}\]

\[
H' = \sigma (N (LQ) HW)
\]

\[
N(M) = \frac{1}{t_l} M
\]

\[
\{q(u_1)Q_{11}, q(u_2)Q_{22}, \cdots, q(u_n)Q_{nn}\} \sim \text{Sample}(N, t_l)
\]
Case Study - ASGCN

- Message-Passing as learned sampling and aggregating scheme

\[
q^*(u_j) = \frac{\sum_{i=1}^{n} p(u_j|v_i)g(x(u_j))}{\sum_{k=1}^{N} \sum_{i=1}^{n} p(u_k|v_i)g(x(u_k))}
\]

\[
h^{l+1}(v_i) = \sigma \left( N(v_i) \frac{1}{n} \sum_{j=1}^{n} \frac{p(u_j|v_i)}{q^*(u_j)} h^{l}(u_j) \right)
\]

\[u_j \sim q^*(u_j)\]

convert to matrix form

\[
\begin{align*}
H' &= \sigma (\mathcal{N}(LQ)HW) \\
\mathcal{N}(M) &= \frac{1}{n} M \\
Q_{ii} &= \frac{q_i}{q^*(u_i)} , q_i \sim \text{Ber}(n, q^*(u_i))
\end{align*}
\]
Summarization

<table>
<thead>
<tr>
<th>Methods</th>
<th>Rescaling and Propagation Rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>vanilla GCN</td>
<td>$H' = \sigma (\mathcal{L}HW)$</td>
</tr>
<tr>
<td>GAT</td>
<td>$H' = \sigma (\mathcal{N} (P \mathcal{L} + \mathcal{L}Q) HW)$</td>
</tr>
<tr>
<td>GraphSAGE$_{mean}$</td>
<td>$H' = \sigma (\mathcal{N} (\mathcal{L} \odot Q) HW)$</td>
</tr>
<tr>
<td>FastGCN</td>
<td>$H' = \sigma (\mathcal{N} (\mathcal{L}Q) HW)$</td>
</tr>
<tr>
<td>ASGCN</td>
<td>$H' = \sigma (\mathcal{N} (\mathcal{L}Q) HW)$</td>
</tr>
</tbody>
</table>

- $P$ and $Q$ are rescaling coefficients matrices.
- $\mathcal{N}(\cdot)$ denotes some matrix normalization function.
- $\mathcal{L}$ is the model specified Laplacian matrix.
- $\sigma(\cdot)$ can be any activation function.
Observations

• Propagation – by Laplacian
  – Laplacians are normalized adjacency matrix with self-loop
  
  \[ \mathcal{L} \text{ can be } D^{-1}(A + I), D^{-\frac{1}{2}}(A + I)D^{-\frac{1}{2}} \text{ or even } A + I. \]

• Signal Rescaling
  – Before Propagation (Pre-Prop)
  – After Propagation (Post-Prop)
  – During Propagation (Edge-Wise)
  – Re-normalization (For numerical stability)

A combination gives approximation for
Observations

• In this way, we
  – disentangle rescaling from message-passing
  – decouple rescaling operations into different operators

• Based on our observation, we are able to design a general / flexible GCN framework with decoupled rescaling operators
Building Blocks of Signal Rescaling

• Pre-propagation Rescaling
  – Vanilla GCN: \( H' = \sigma(\mathcal{L}HW) \)
  – Matrix row scaling operator as Diagonal matrix \( P \) and \( Q \)

  – Rescaling before propagation: \( H' = \sigma(\mathcal{L}QHW) \)
  – Re-normalization after aggregation: \( H' = \sigma(P\mathcal{L}QHW) \)

  – \( Q \) are extracted from pre-prop feature by a learnable vector \( \hat{\mathbf{p}} \):
    \[ Q_{ii} = \text{sigmoid}(HW\hat{q})_i \]
  – \( P \) makes the row sum of \( P\mathcal{L}Q \) to be 1.
Building Blocks of Signal Rescaling

• Post-propagation Rescaling
  – Vanilla GCN: \( H' = \sigma(\mathcal{L}HW) \)
  – Negative signal rescaling by node adaptive encoder:
    \[
    \sigma_{\vec{r}}(x) = \begin{cases} 
    \sigma(x), & x \geq 0 \\
    \sigma(\vec{r}_i x), & x < 0 
    \end{cases}
    \]
  – Rescaling after propagation \( H' = \sigma_{\vec{r}}(\mathcal{L}HW) \)

  – \( \vec{r} \) are extracted from post-prop feature by a learnable vector \( \vec{p} \):
    \[
    \vec{r}_i = \text{sigmoid}(\mathcal{L}HW\vec{q})_i
    \]
  – Re-normalization are opted out for \( x\text{LU}(\cdot) \) activation’s output mostly concentrate on the positive signal.
Building Blocks of Signal Rescaling

• Edge-wise Rescaling
  – Approximated by the combination of Pre-prop rescaling and Post-prop rescaling

• Node Sampling Strategy
  – (Schematic) Dropout applied to Pre-propagation Matrix
General Framework

Diagonal Matrix $Q$ as Pre-prop rescaling operator
Node Adaptive Encoder $\sigma_{\tilde{r}}(\cdot)$ as Post-prop rescaling operator
Analysis

• The pre-propagation rescaling redistributes the signal energy. It functions as noise reducer.

• The post-propagation rescaling reduce the variance before activation. It functions like Layer Normalization.

• The Node Sampling Strategy functions like small batch trick (Regularization).

• The flexibility of Signal Rescaling framework is favorable by Graph AutoML.
Graph Attention as Signal Rescaling

- **GAT**
  - $H' = \sigma(\mathcal{N}(PL + \mathcal{L}Q)HW)$
  - $\mathcal{L} = A + I_N$
  - $\mathcal{N}(\cdot) = \text{SparseRowSoftmax}(\text{LeakyReLU}(\cdot))$
  - $P_{ii} = (HW\vec{p})_i, \quad Q_{ii} = (HW\vec{q})_i$
Graph Attention as Signal Rescaling

- **Node Attention**
  \[ H'_i = \sigma_i(P\mathcal{L}QHW)_i \]
  \[ P_{ii} = \frac{1}{q_i} \]

- **Edge Attention**
  \[ H'_i = \sigma_i(P\mathcal{L}QHW)_i \]
  \[ P_{ii} = \frac{1}{\mathcal{L}Q} \]

- **k-hop Edge Attention**
  \[ H'_i = \sigma_i(P\mathcal{L}^k QHW)_i \]
  \[ P_{ii} = \frac{1}{\mathcal{L}^k Q} \]

- **k-hop Path Attention**
  \[ H'_i = \sigma_i(P\mathcal{L}Q_k\mathcal{L}Q_{k-1} \cdots \mathcal{L}Q_1HW)_i \]
  \[ P_{ii} = \frac{1}{\mathcal{L}Q_k\mathcal{L}Q_{k-1} \cdots \mathcal{L}Q_1} \]
## Performance – Citation Networks

<table>
<thead>
<tr>
<th>Category</th>
<th>Method</th>
<th>Cora</th>
<th>Citeseer</th>
<th>Pubmed</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Chebyshev</td>
<td>81.2</td>
<td>69.8</td>
<td>74.4</td>
</tr>
<tr>
<td></td>
<td>GCN</td>
<td>81.5</td>
<td>70.3</td>
<td>79.0</td>
</tr>
<tr>
<td></td>
<td>MixHop</td>
<td>81.9±0.4</td>
<td>71.4±0.8</td>
<td>80.8±0.6</td>
</tr>
<tr>
<td></td>
<td>FastGCN*</td>
<td>80.0±0.6</td>
<td>69.5±0.6</td>
<td>77.8±0.4</td>
</tr>
<tr>
<td></td>
<td>ASGCN*</td>
<td>78.5±0.5</td>
<td>68.9±0.5</td>
<td>75.5±0.4</td>
</tr>
<tr>
<td></td>
<td>GAT</td>
<td>83.0±0.7</td>
<td>72.5±0.7</td>
<td>79.0±0.3</td>
</tr>
<tr>
<td></td>
<td>GAT-SR**</td>
<td>83.4±0.4</td>
<td>72.4±0.6</td>
<td>78.7±0.4</td>
</tr>
<tr>
<td></td>
<td>GAT-16**</td>
<td>83.4±0.4</td>
<td>72.3±0.6</td>
<td>78.8±0.3</td>
</tr>
<tr>
<td></td>
<td>SRGCN</td>
<td>84.1±0.5</td>
<td>73.4±0.7</td>
<td>79.5±0.4</td>
</tr>
<tr>
<td></td>
<td>SRGCN (Pre)</td>
<td>83.8±0.6</td>
<td>73.2±0.6</td>
<td>79.3±0.6</td>
</tr>
<tr>
<td></td>
<td>SRGCN (Post)</td>
<td>84.1±0.5</td>
<td>73.3±0.7</td>
<td>79.4±0.5</td>
</tr>
<tr>
<td></td>
<td>SRGCN (1/2)</td>
<td>83.9±0.6</td>
<td>73.3±0.9</td>
<td>79.3±0.6</td>
</tr>
<tr>
<td></td>
<td>SRGCN (1/4)</td>
<td>84.0±0.6</td>
<td>73.0±0.9</td>
<td>79.0±0.8</td>
</tr>
<tr>
<td></td>
<td>SRGCN (1/8)</td>
<td>84.0±0.6</td>
<td>72.5±1.4</td>
<td>77.5±1.3</td>
</tr>
</tbody>
</table>
## Performance – PPI networks

<table>
<thead>
<tr>
<th>Method</th>
<th>F1</th>
<th>Method</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random</td>
<td>39.6</td>
<td>GeniePath</td>
<td>97.9</td>
</tr>
<tr>
<td>MLP</td>
<td>42.2</td>
<td>GAT</td>
<td>97.3±0.2</td>
</tr>
<tr>
<td>GraphSAGE-GCN</td>
<td>50.0</td>
<td>SRGCN</td>
<td>98.4±0.2</td>
</tr>
<tr>
<td>GraphSAGE-mean</td>
<td>59.8</td>
<td>SRGCN (Pre)</td>
<td>98.4±0.3</td>
</tr>
<tr>
<td>GraphSAGE-LSTM</td>
<td>61.2</td>
<td>SRGCN (Post)</td>
<td>97.3±0.2</td>
</tr>
<tr>
<td>GraphSAGE-pool</td>
<td>60.0</td>
<td>SRGCN (1/2)</td>
<td>98.1±0.4</td>
</tr>
<tr>
<td>GraphSAGE</td>
<td>76.8</td>
<td>SRGCN (1/4)</td>
<td>96.7±0.5</td>
</tr>
</tbody>
</table>
Takeaway Messages & Future Works

- The *decoupled rescaling operations* have their separated utilities.
- Node sampling strategy functions as small batch trick.
  - Can we find a theory?
  - How to better use this?
- The graph *AutoML* has its potential to further enhance of the community of GNNs.
  - Realization?
• Let us now revisit the negative sampling?
SampledNCE Framework

• An trainable encoder $E_\theta$ -- generate node embeddings
  – An embedding lookup table
  – A variant of Graph Neural Networks
• A positive sampler -- sample positive nodes
• A negative sampler -- sample negative nodes
Negative Sampling

• Negative Sampling
  – serve as a simplified version of noise contrastive estimation.
  – a efficient way to accelerate the training of word2vec.
  – negative sampling has a huge influence on the performance.
  –the best choice varies largely on different datasets.
  –few papers systematically analyzed or discussed negative sampling in graph representation learning.
Related Work

- **Degree-based NS**: utilize degree to sample negative samples
  - Power of Degree (Mikolov, Sutskever & Dean, 2013)
  - RNS (Caselles-Dupré, Lesaint & Royo-Letelier, 2018)
  - WRMF (Hu, Koren & Volinsky, 2008)

- **Hard-samples NS**: mine the hard negative samples by rejection
  - DNS (Zhang, Chen & Yu, 2013)
  - PinSAGE (Rex, Ruining & Jure, 2018)
  - WARP (Weston, Bengio & Usunier, 2011)

- **GAN-based NS**: utilize GAN to adversarially generate “difficult” negative samples
  - IRGAN (Wang, Yu & Xu, 2017)
  - KBGAN (Cai & Wang, 2018)
  - NMRN (Wang, Yin & Wang, 2018)
Understanding Negative Sampling--Objective

The objective function for embedding:

\[ J = \mathbb{E}_{(u,v) \sim p_d} \log \sigma(\mathbf{u}^T \mathbf{v}) + \mathbb{E}_{v \sim p_d(v)} [k \mathbb{E}_{u' \sim p_n(u'|v)} \log \sigma(-\mathbf{u'}^T \mathbf{v})] \]

For each \((u, v)\) pair, define two Bernoulli distribution:

- \textbf{P}: \(P_{u,v}(x = 1) = \frac{p_d(u|v)}{p_d(u|v) + kp_n(u|v)}\)

- \textbf{Q}: \(Q_{u,v}(x = 1) = \sigma(\mathbf{u}^T \mathbf{v})\)

The objective function can be simplified as follows:

\[ J = - \sum_u (p_d(u|v) + kp_n(u|v)) H(P_{u,v}, Q_{u,v}), \text{where } H(p, q) \text{ is the cross entropy.} \]

According to \textit{Gibbs Inequality}, the optimal embedding for each node pair \((u, v)\) :

\[ \mathbf{u}^T \mathbf{v} = - \log \frac{k \cdot p_n(u|v)}{p_d(u|v)} \]
Understanding Negative Sampling--Risk

The loss function to minimize empirical risk as follows:

\[ J_T(v) = \frac{1}{T} \sum_{i=1}^{T} \log \sigma(\tilde{u}_i^T \tilde{v}) + \frac{1}{T} \sum_{i=1}^{kT} \log \sigma(-\tilde{u}'_i^T \tilde{v}), \]

where \( \{u_1, ..., u_T\} \) are sampled from \( p_d(u|v) \) and \( \{u'_1, ..., u'_{kT}\} \) are sampled from \( p_n(u|v) \).

Theorem: the random variable \( \sqrt{T}(\theta_T - \theta^*) \) asymptotically converges to a distribution with zero mean vector and covariance matrix.

\[
\text{Cov}(\sqrt{T}(\theta_T - \theta^*)) = \text{diag}(m)^{-1} - (1 + 1/k)1^T 1
\]

where \( m = \left[ \frac{k p_d(u_0|v) p_n(u_0|v)}{p_d(u_0|v) + k p_n(u_0|v)}, ..., \frac{k p_d(u_{N-1}|v) p_n(u_{N-1}|v)}{p_d(u_{N-1}|v) + k p_n(u_{N-1}|v)} \right]^T \) and \( 1 = [1, ..., 1]^T \).

According to Theorem, the mean squared error of \( \tilde{u}^T \tilde{v} \):

\[
\mathbb{E} \left[ \| (\theta_T - \theta^*)_u \|^2 \right] = \frac{1}{T} \left( \frac{1}{p_d(u|v)} - 1 + \frac{1}{k p_n(u|v) - \frac{1}{k}} \right)
\]
The Principle of Negative Sampling

A simple solution is to sample negative nodes \textit{positively but sub-linearly} correlated to their positive sampling distribution.

\[ p_n(u|v) \propto p_d(u|v)\alpha, \quad 0 < \alpha < 1 \]

- **Monotonicity**: from the perspective of objective, if we have \( p_d(u_i|v) > p_d(u_j|v) \),

\[
\tilde{u}_i^T \tilde{v} = \log p_d(u_i|v) - \alpha \log p_d(u_i|v) + c \\
> (1 - \alpha) \log p_d(u_j|v) + c = \tilde{u}_j^T \tilde{v}
\]

- **Accuracy**: from the perspective of risk, if \( p_n(u|v) \propto p_d(u|v)^\alpha \),

\[
\mathbb{E}[\|\theta_T - \theta^*\|_u^2] = \frac{1}{T} \left( \frac{1}{p_d(u|v)} \left(1 + \frac{p_d(u|v)^{1-\alpha}}{c}\right) - 1 - \frac{1}{k}\right)
\]
MCNS Model

• self-contrast approximation for positive sampling distribution:
  – replacing $p_d$ by inner products based on the current encoder

\[
p_n(u|v) \propto p_d(u|v)^\alpha \approx \frac{(E_\theta(u) \cdot E_\theta(v))^\alpha}{\sum_{u' \in U} (E_\theta(u') \cdot E_\theta(v))^\alpha}
\]

• Metropolis-Hastings to accelerate negative sampling

\[
\mathcal{L} = \max(0, E_\theta(v) \cdot E_\theta(x) - E_\theta(u) \cdot E_\theta(v) + \gamma)
\]
# Experiments

## Recommendation Task

**MovieLens:**
- 2,625 nodes
- 100,000 links

**Amazon:**
- 255,404 nodes
- 1,689,188 links

**Alibaba:**
- 159,633 nodes
- 907,470 links

## Evaluation Metric:
- MRR & Hits@30

<table>
<thead>
<tr>
<th></th>
<th>DeepWalk</th>
<th>MovieLens MRR</th>
<th>MovieLens Hits@30</th>
<th>Amazon MRR</th>
<th>Amazon Hits@30</th>
<th>Alibaba MRR</th>
<th>Alibaba Hits@30</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Deg</strong>&lt;sup&gt;0.75&lt;/sup&gt;</td>
<td>0.025±.001</td>
<td>0.062±.001</td>
<td>0.063±.001</td>
<td>0.041±.001</td>
<td>0.057±.001</td>
<td>0.037±.001</td>
<td>0.064±.001</td>
</tr>
<tr>
<td>WRMF</td>
<td>0.022±.001</td>
<td>0.038±.001</td>
<td>0.040±.001</td>
<td>0.034±.001</td>
<td>0.043±.001</td>
<td>0.036±.001</td>
<td>0.057±.002</td>
</tr>
<tr>
<td>RNS</td>
<td>0.031±.001</td>
<td>0.082±.002</td>
<td>0.079±.001</td>
<td>0.046±.003</td>
<td>0.079±.003</td>
<td>0.035±.001</td>
<td>0.078±.003</td>
</tr>
<tr>
<td>PinSAGE</td>
<td>0.036±.001</td>
<td>0.091±.002</td>
<td>0.090±.002</td>
<td>0.057±.004</td>
<td>0.080±.001</td>
<td>0.054±.001</td>
<td>0.081±.001</td>
</tr>
<tr>
<td>WARP</td>
<td>0.041±.003</td>
<td>0.114±.003</td>
<td>0.111±.003</td>
<td>0.061±.001</td>
<td>0.098±.002</td>
<td>0.067±.001</td>
<td>0.106±.001</td>
</tr>
<tr>
<td>DNS</td>
<td>0.040±.003</td>
<td>0.113±.003</td>
<td>0.115±.003</td>
<td>0.063±.001</td>
<td>0.101±.003</td>
<td>0.067±.001</td>
<td>0.090±.002</td>
</tr>
<tr>
<td>IRGAN</td>
<td>0.047±.002</td>
<td>0.111±.002</td>
<td>0.101±.002</td>
<td>0.059±.001</td>
<td>0.091±.001</td>
<td>0.061±.001</td>
<td>0.083±.001</td>
</tr>
<tr>
<td>KBGAN</td>
<td>0.049±.001</td>
<td>0.114±.003</td>
<td>0.100±.001</td>
<td>0.060±.001</td>
<td>0.089±.001</td>
<td>0.065±.001</td>
<td>0.087±.002</td>
</tr>
<tr>
<td>MCNS</td>
<td>0.053±.001</td>
<td>0.122±.004</td>
<td>0.114±.001</td>
<td>0.065±.001</td>
<td>0.108±.001</td>
<td>0.070±.001</td>
<td>0.116±.001</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>DeepWalk</th>
<th>MovieLens MRR</th>
<th>MovieLens Hits@30</th>
<th>Amazon MRR</th>
<th>Amazon Hits@30</th>
<th>Alibaba MRR</th>
<th>Alibaba Hits@30</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Deg</strong>&lt;sup&gt;0.75&lt;/sup&gt;</td>
<td>0.115±.002</td>
<td>0.270±.002</td>
<td>0.270±.001</td>
<td>0.161±.003</td>
<td>0.238±.002</td>
<td>0.138±.003</td>
<td>0.249±.004</td>
</tr>
<tr>
<td>WRMF</td>
<td>0.110±.003</td>
<td>0.187±.002</td>
<td>0.181±.002</td>
<td>0.139±.002</td>
<td>0.188±.001</td>
<td>0.121±.003</td>
<td>0.227±.004</td>
</tr>
<tr>
<td>RNS</td>
<td>0.143±.004</td>
<td>0.362±.004</td>
<td>0.356±.001</td>
<td>0.171±.004</td>
<td>0.317±.004</td>
<td>0.132±.004</td>
<td>0.302±.005</td>
</tr>
<tr>
<td>PinSAGE</td>
<td>0.158±.003</td>
<td>0.379±.005</td>
<td>0.383±.005</td>
<td>0.176±.004</td>
<td>0.333±.005</td>
<td>0.146±.003</td>
<td>0.312±.005</td>
</tr>
<tr>
<td>WARP</td>
<td>0.164±.005</td>
<td>0.406±.002</td>
<td>0.404±.005</td>
<td>0.181±.004</td>
<td>0.340±.004</td>
<td>0.178±.004</td>
<td>0.342±.004</td>
</tr>
<tr>
<td>DNS</td>
<td>0.166±.005</td>
<td>0.404±.006</td>
<td>0.410±.006</td>
<td>0.182±.003</td>
<td>0.358±.004</td>
<td>0.186±.005</td>
<td>0.336±.004</td>
</tr>
<tr>
<td>IRGAN</td>
<td>0.207±.002</td>
<td>0.415±.004</td>
<td>0.408±.004</td>
<td>0.183±.004</td>
<td>0.342±.003</td>
<td>0.175±.003</td>
<td>0.320±.002</td>
</tr>
<tr>
<td>KBGAN</td>
<td>0.198±.003</td>
<td>0.420±.003</td>
<td>0.401±.005</td>
<td>0.181±.003</td>
<td>0.347±.003</td>
<td>0.181±.003</td>
<td>0.331±.004</td>
</tr>
<tr>
<td>MCNS</td>
<td>0.230±.003</td>
<td>0.426±.005</td>
<td>0.413±.003</td>
<td>0.207±.003</td>
<td>0.386±.004</td>
<td>0.201±.003</td>
<td>0.387±.002</td>
</tr>
</tbody>
</table>
Experiments

Link Prediction Task
Arxiv: 5,242 nodes & 28,980 links
Evaluation Metric: AUC

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>DeepWalk</th>
<th>GCN</th>
<th>GraphSAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>WRMF</td>
<td>65.3±0.1</td>
<td>80.3±0.4</td>
<td>79.1±0.2</td>
</tr>
<tr>
<td>RNS</td>
<td>62.2±0.2</td>
<td>74.3±0.5</td>
<td>74.7±0.5</td>
</tr>
<tr>
<td>PinSAGE</td>
<td>67.2±0.4</td>
<td>80.4±0.3</td>
<td>80.1±0.4</td>
</tr>
<tr>
<td>WARP</td>
<td>70.5±0.3</td>
<td>81.6±0.3</td>
<td>82.7±0.4</td>
</tr>
<tr>
<td>DNS</td>
<td>70.4±0.3</td>
<td>81.5±0.3</td>
<td>82.6±0.4</td>
</tr>
<tr>
<td>IRGAN</td>
<td>71.1±0.2</td>
<td>82.0±0.4</td>
<td>82.2±0.3</td>
</tr>
<tr>
<td>KBGAN</td>
<td>71.6±0.3</td>
<td>81.7±0.3</td>
<td>82.1±0.3</td>
</tr>
<tr>
<td>MCNS</td>
<td>73.1±0.4</td>
<td>82.6±0.4</td>
<td>83.5±0.5</td>
</tr>
</tbody>
</table>

Node Classification Task
BlogCatalog: 10,312 nodes & 333,983 links
Evaluation Metric: Micro-F1

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>DeepWalk</th>
<th>GCN</th>
<th>GraphSAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>WRMF</td>
<td>30.9</td>
<td>35.8</td>
<td>37.5</td>
</tr>
<tr>
<td>RNS</td>
<td>29.8</td>
<td>34.1</td>
<td>36.0</td>
</tr>
<tr>
<td>PinSAGE</td>
<td>32.0</td>
<td>37.4</td>
<td>40.1</td>
</tr>
<tr>
<td>WARP</td>
<td>35.1</td>
<td>40.3</td>
<td>42.1</td>
</tr>
<tr>
<td>DNS</td>
<td>35.2</td>
<td>40.4</td>
<td>42.5</td>
</tr>
<tr>
<td>IRGAN</td>
<td>34.3</td>
<td>39.6</td>
<td>41.8</td>
</tr>
<tr>
<td>KBGAN</td>
<td>34.6</td>
<td>40.0</td>
<td>42.3</td>
</tr>
<tr>
<td>MCNS</td>
<td>36.1</td>
<td>41.2</td>
<td>43.3</td>
</tr>
</tbody>
</table>
Efficiency Comparison

The runtime of MCNS and hard-samples or GAN-based strategies with GraphSAGE encoder in recommendation task.

Runtime Comparisons:

![Runtime Comparisons Graph](image-url)
Summary of MCNS

• Analyze **the role of negative sampling** from the perspectives of both **objective** and **risk**.

• Derive the theory and quantify that the negative sampling distribution should be **positively but sub-linearly** correlated to their positive sampling distribution.

• Approximate the positive distribution with self-contrast approximation and accelerating negative sampling by Metropolis-Hastings.

• Achieve **start-of-art performance** in recommendation, link prediction and node classification, on a total of 19 experimental settings.
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

Jiezhong Qiu*, Qibin Chen*, Yuxiao Dong$, Jing Zhang+, Hongxia Yang#, Ming Ding*, Kuansan Wang$, Jie Tang*
*Tsinghua University
$Microsoft Research, Redmond
#DAMO Academy, Alibaba Group
+Renmin University

Revisit MoCo (CVPR 20)

• Momentum Contrast for Unsupervised Visual Representation Learning

• Train ResNet on ImageNet without image labels
  – Competitive results on ImageNet classification
  – Outperform supervised ResNet in 7 downstream tasks
Revisit MoCo

(a) end-to-end

(b) memory bank

(c) 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
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.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Academia</th>
<th>DBLP (SNAP)</th>
<th>DBLP (NetRep)</th>
<th>IMDB</th>
<th>Facebook</th>
<th>LiveJournal</th>
</tr>
</thead>
<tbody>
<tr>
<td>$</td>
<td>V</td>
<td>\rangle$</td>
<td>137,969</td>
<td>317,080</td>
<td>540,486</td>
<td>896,305</td>
</tr>
<tr>
<td>$</td>
<td>E</td>
<td>\rangle$</td>
<td>739,384</td>
<td>2,099,732</td>
<td>30,491,458</td>
<td>7,564,894</td>
</tr>
</tbody>
</table>

• Fine-tuning Tasks:
  – Node classification
  – Graph classification
  – Similarity search

• Two fine-tuning settings.
## Node Classification

<table>
<thead>
<tr>
<th>Datasets</th>
<th>US-Airport</th>
<th>H-index</th>
</tr>
</thead>
<tbody>
<tr>
<td>(</td>
<td>V</td>
<td>)</td>
</tr>
<tr>
<td>(</td>
<td>E</td>
<td>)</td>
</tr>
<tr>
<td>ProNE</td>
<td>62.3</td>
<td>69.1</td>
</tr>
<tr>
<td>GraphWave</td>
<td>60.2</td>
<td>70.3</td>
</tr>
<tr>
<td>Struc2vec</td>
<td><strong>66.2</strong></td>
<td>&gt; 1 Day</td>
</tr>
<tr>
<td>GCC (E2E, freeze)</td>
<td>64.8</td>
<td><strong>78.3</strong></td>
</tr>
<tr>
<td>GCC (MoCo, freeze)</td>
<td>65.6</td>
<td>75.2</td>
</tr>
<tr>
<td>GCC (rand, full)</td>
<td>64.2</td>
<td>76.9</td>
</tr>
<tr>
<td>GCC (E2E, full)</td>
<td><strong>68.3</strong></td>
<td>80.5</td>
</tr>
<tr>
<td>GCC (MoCo, full)</td>
<td>67.2</td>
<td><strong>80.6</strong></td>
</tr>
</tbody>
</table>
Graph Classification

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

<table>
<thead>
<tr>
<th></th>
<th>KDD-ICDM</th>
<th>SIGIR-CIKM</th>
<th>SIGMOD-ICDE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2,867</td>
<td>2,607</td>
<td>2,851</td>
</tr>
<tr>
<td></td>
<td>7,637</td>
<td>4,774</td>
<td>6,354</td>
</tr>
<tr>
<td># groud truth</td>
<td>697</td>
<td>874</td>
<td>874</td>
</tr>
<tr>
<td>k</td>
<td>20</td>
<td>40</td>
<td>20</td>
</tr>
<tr>
<td>Random</td>
<td>0.0198</td>
<td>0.0566</td>
<td>0.0223</td>
</tr>
<tr>
<td>RoIX</td>
<td>0.0779</td>
<td>0.1288</td>
<td>0.0548</td>
</tr>
<tr>
<td>Panther++</td>
<td>0.0892</td>
<td>0.1558</td>
<td><strong>0.0782</strong></td>
</tr>
<tr>
<td>GraphWave</td>
<td>0.0846</td>
<td><strong>0.1693</strong></td>
<td>0.0549</td>
</tr>
<tr>
<td>GCC (E2E)</td>
<td><strong>0.1047</strong></td>
<td>0.1564</td>
<td>0.0549</td>
</tr>
<tr>
<td>GCC (MoCo)</td>
<td>0.0904</td>
<td>0.1521</td>
<td>0.0652</td>
</tr>
</tbody>
</table>
Agenda

• Network Embedding
• Revisiting Network Embedding
• Graph Neural Network
• Revisiting Graph Neural Network
• GNN&Reasoning
• Revisiting GNN&Reasoning
Question: Who is the director of the 2003 film which has scenes in it filmed at the **Quality Café** in **Los Angeles**?

**Quality Café**
The Quality Café 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.

**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.

**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.

**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

**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...
AI趋势：从感知到认知

- 面向认知的GNN

  - Storage & Computing
  - Network embedding, GCN, GNN, knowledge
  - reasoning, planning, logical, expression

计算 | 感知 | 认知

Stochastic vs Deterministic
Uncertainty!
graph_net

• By DeepMind

Box 3: Our definition of “graph”

Here we use “graph” to mean a directed, attributed multi-graph with a global attribute. In our terminology, a node is denoted as $v_i$, an edge as $e_k$, and the global attributes as $u$. We also use $s_k$ and $r_k$ to indicate the indices of the sender and receiver nodes (see below), respectively, for edge $k$. To be more precise, we define these terms as:

- Directed: one-way edges, from a “sender” node to a “receiver” node.
- Attribute: properties that can be encoded as a vector, set, or even another graph.
- Attributed: edges and vertices have attributes associated with them.
- Global attribute: a graph-level attribute.
- Multi-graph: there can be more than one edge between vertices, including self-edges.

Figure 2 shows a variety of different types of graphs corresponding to real data that we may be interested in modeling, including physical systems, molecules, images, and text.

https://arxiv.org/abs/1806.01261
REALM: Retrieval-Augmented LM

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

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

- 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.

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

Agenda

• Network Embedding
• Revisiting Network Embedding
• Graph Neural Network
• Revisiting Graph Neural Network
• GNN&Reasoning
• Revisiting GNN&Reasoning
Challenge: only System 1 DL

SYSTEM 1 VS. SYSTEM 2 COGNITION

2 systems (and categories of cognitive tasks):

System 1
- Intuitive, fast, UNCONSCIOUS, non-linguistic, habitual
- Current DL

System 2
- Slow, logical, sequential, CONSCIOUS, linguistic, algorithmic, planning, reasoning
- Future DL

1. From Bengio’s NIPS’2019 Keynote
**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.

**Old School**
Old School is a 2003 American comedy film released by DreamWorks 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.

**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).

**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...

**Wikipedia**
The Free Encyclopedia
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
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

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

Quality Cafe (jazz club)
Quality Cafe was a historical restaurant and jazz club...

Quality Cafe (diner)
location featured in a number of Hollywood films, including "Old School", "Gone in 60 Seconds"...

Old School (film)
Old School is a 2003 American comedy film... directed by Todd Phillips.

Gone in 60 Seconds
Gone in 60 Seconds is a 2000 American action heist film... directed by Dominic Sena.

Todd Phillips
correct answer

Los Angeles
Los Angeles officially the City of Los Angeles and often known by its initials L.A.,...
和认知科学的结合

Dual Process Theory (Cognitive Science)

System 1
Intuitive

System 2
Analytic
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

...location featured in a number of Hollywood films, including Old School, Gone in 60 Seconds...
System 1: BIDAF, BERT

• reading comprehension: target at understanding the whole paragraph

Cognitive Graph: DL + Dual Process Theory


System 1 (BERT):
- Implicit knowledge expansion
- Question + clues[x, G]
- Paragraph[x]

System 2:
- Explicit decision
- Possible answer “Ans”
- Name of entity “Next”

Diagram: A series of interconnected nodes representing a cognitive graph, with arrows indicating the flow of information between different parts of the system.
System 2: Reasoning, and Decision

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

Quality Cafe (jazz club)
Quality Cafe was a historical restaurant and jazz club...

Quality Cafe (diner)
location featured in a number of Hollywood films, including "Old School," "Gone in 60 Seconds"...

Old School (film)
Old School is a 2003 American comedy film... directed by Todd Phillips.

Gone in 60 Seconds
Gone in 60 Seconds is a 2000 American action heist film... directed by Dominic Sena.

Todd Phillips
correct answer

Los Angeles
Los Angeles officially the City of Los Angeles and often known by its initials L.A.,...
System 1: the BERT Implementation

- 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

\[
P_{\text{start}}[i] = \frac{e^{\text{Sans} \cdot T_i}}{\sum_j e^{\text{Sans} \cdot T_j}}
\]
\[
end_k = \arg \max_{\text{start}_k \leq j \leq \text{start}_k + \max L} P_{\text{end}}[j]
\]
At each step, hidden representations $X$ for nodes are updated according to the propagation rules:

$$\Delta = \sigma((AD^{-1})^T\sigma(XW_1))$$
$$X' = \sigma(XW_2 + \Delta)$$

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

$$answer = \arg \max \mathcal{F}(X[x])$$
Performance

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

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

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](https://github.com/THUDM/CogQA)
CogQA Performs much **better** on question with more hops!
Case Study

Q: **Ken Pruitt** was a Republican member of an upper house of the legislature with how many members?

- **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

Q: What Cason, CA soccer team features the son of Roy Lassiter?

He is the father of LA Galaxy player Ariel Lassiter.

• DAG-shape Cognitive Graph

• Multiple supporting facts provides richer information, increasing the credibility of the answer.

(2) DAG
Case Study

Q: What Lithuanian producer is best known for a song that was one of the most popular songs in Ibiza in 2014?

- Walking with Elephants
- Ten Walls
- Marijus Adomaitis

• 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

1-hop
- Quality Cafe was a historical restaurant and jazz club...
- Los Angeles
- "Old School", "Gone in 60 Seconds"
- Quality Cafe (jazz club)
- Quality Cafe (diner)
- Los Angeles

2-hop
- Old School is a 2003 American comedy film... directed by Todd Phillips.
- Gone in 60 Seconds is a 2000 American action heist film... directed by Dominic Sena.

3-hop
- Todd Phillips
- Dominic Sena
- correct answer

CogQA framework

System 1 (BERT)

System 2 (GNN)

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

Result: Todd Phillips

Question + clues[x,G]

Paragraph[x]
More Applications: KG completion

• Completing knowledge graph with cognitive graph
Takeaway Messages

• Revisit NE, GNN & Reasoning
  – Network Embedding
  – Graph Neural Network
  – GNN&Reasoning

• Summary
  – Pre-training and Self-supervised learning is becoming more and more important
  – System 2 DL for reasoning and logistical

• What is the Next?
挑战与未来(Next 10)

认知与推理
—Trillion-scale common-sense knowledge graph

* AI = Knowledge + Intelligence

挑战与未来(Next 30)

意识
—让计算机具有自我意识

认知推理 ⇔ 记忆 ⇔ 自我意识

• Next AI = Reasoning + Memory + Consciousness

Related Publications

• Jiezhong Qiu, Qibin Chen, Yuxiao Dong, Jing Zhang, Hongxia Yang, Ming Ding, Kuansan Wang, and Jie Tang. GCC: Graph Contrastive Coding for Structural Graph Representation Pre-Training. KDD’20.
• Zhen Yang, Ming Ding, Chang Zhou, Hongxia Yang, Jingren Zhou, and Jie Tang. Understanding Negative Sampling in Graph Representation Learning. KDD’20.
• 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.

For more, check 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