Lightweight, Dynamic Graph Convolutional Networks for AMR-to-Text Generation

Yan Zhang*, Zhijiang Guo*, Zhiyang Teng, Wei Lu Shay B. Cohen, Zuozhu Liu, Lidong Bing
I will join this kind of game.
AMR-to-Text Generation: Main Challenge

How to learn a good representation of the AMR graph?

Source: AMR Graph
Progress in AMR-to-Text Generation

Large gap

LSTM
GRN
GCN
SAN

Konstas, '17
Song, '18
Ribeiro, '19
Guo, '19
Wang, '20
Cai, '20
Zhu, '19

AMR 1.0
LDC2015E86
Progress in AMR-to-Text Generation

This work
Graph Convolution v.s. Self-Attention

**GCN**  
Guo et al., 2019  
Damonte et al., 2019

\[ h_i = \sigma(\sum_{j=1}^{n} A_{i,j} Wh_j + b) \]

**SAN**  
vaswani et al., 2017

\[ h_i = \sigma(\sum_{j=1}^{n} A_{i,j} Wh_j + b) \]

**Structured SAN**  
Zhu et al., 2019  
Cai et al., 2020  
Wang et al., 2020

\[ h_i = \sigma(\sum_{j=1}^{n} A_{i,j} Wh_j + b) \]

\[ A_{i,j} = f(h_i, h_j) \]

\[ A_{i,j} = f(h_i, h_j, r_{ij}) \]

---

**Game**

- **ARG0**: This
- **ARG1**: Kind

**Observed**

- **ARG0**: This
- **ARG1**: Kind

**Structured SAN**

- **ARG0**: This
- **ARG1**: Kind

---

**Game**

- **ARG0**: This
- **ARG1**: Kind

**Observed**

- **ARG0**: This
- **ARG1**: Kind

**Structured SAN**

- **ARG0**: This
- **ARG1**: Kind
Graph Convolution v.s. Self-Attention

**GCN** (Guo et al., 2019; Damonte et al., 2019)

\[ h_i = \sigma(\sum_{j=1}^{n} A_{i,j} Wh_j + b) \]

**SAN** (vaswani et al., 2017)

\[ h_i = \sigma(\sum_{j=1}^{n} A_{i,j} Wh_j + b) \]

**Structured SAN** (Zhu et al., 2019; Cai et al., 2020; Wang et al., 2020)

\[ h_i = \sigma(\sum_{j=1}^{n} A_{i,j} Wh_j + b) \]

\[ A_{i,j} = f(h_i, h_j) \]

\[ A_{i,j} = f(h_i, h_j, r_{ij}) \]

- \( A_{i,j} = 1 \) if there is an edge between \( i \) and \( j \)
Graph Convolution v.s. Self-Attention

**GCN**  
Guo et al., 2019  
Damonte et al., 2019

\[ h_i = \sigma(\sum_{j=1}^{n} A_{i,j} Wh_j + b) \]

**SAN**  
vaswani et al., 2017

\[ h_i = \sigma(\sum_{j=1}^{n} A_{i,j} Wh_j + b) \]

**Structured SAN**  
Zhu et al., 2019  
Cai et al., 2020  
Wang et al., 2020

\[ h_i = \sigma(\sum_{j=1}^{n} A_{i,j} Wh_j + b) \]

\[ A_{i,j} = f(h_i, h_j) \]

\[ A_{i,j} = f(h_i, h_j, r_{ij}) \]

\[ A_{i,j} = 1 \text{ if there is an edge between } i \text{ and } j \]
Graph Convolution v.s. Self-Attention

<table>
<thead>
<tr>
<th>Model</th>
<th>Time Complexity</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>GCN</td>
<td>$O(n)$</td>
<td>Guo et al., 2019</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Damonte et al., 2019</td>
</tr>
<tr>
<td>SAN</td>
<td>$O(n^2)$</td>
<td>vaswani et al., 2017</td>
</tr>
<tr>
<td>Structured SAN</td>
<td>$O(n^2+n)$</td>
<td>Zhu et al., 2019</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Cai et al., 2020</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Wang et al., 2020</td>
</tr>
</tbody>
</table>
Graph Convolution v.s. Self-Attention

**GCN**  Guo et al., 2019  
Damonte et al., 2019

**SAN**  vaswani et al., 2017

**Structured SAN**  Zhu et al., 2019  
Cai et al., 2020  
Wang et al., 2020

<table>
<thead>
<tr>
<th></th>
<th>GCN</th>
<th>SAN</th>
<th>Structured SAN</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Time Complexity</strong></td>
<td>O(n)</td>
<td>O(n^2)</td>
<td>O(n^2+n)</td>
</tr>
<tr>
<td><strong>Neighbour Info</strong></td>
<td>1-order</td>
<td>n-order</td>
<td>n-order</td>
</tr>
</tbody>
</table>

**Graph Structures**

1. **GCN**
   - **ARG0**
   - **ARG1**
   - 1-order

2. **SAN**
   - **ARG0**
   - **ARG1**
   - n-order

3. **Structured SAN**
   - **ARG0**
   - **ARG1**
   - n-order
Graph Convolution v.s. Self-Attention

<table>
<thead>
<tr>
<th></th>
<th>GCN</th>
<th>SAN</th>
<th>Structured SAN</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Time Complexity</strong></td>
<td>O(n)</td>
<td>O(n^2)</td>
<td>O(n^2+n)</td>
</tr>
<tr>
<td><strong>Neighbour Info</strong></td>
<td>l-order</td>
<td>n-order</td>
<td>n-order</td>
</tr>
<tr>
<td><strong>Layers</strong></td>
<td>36</td>
<td>8</td>
<td>8</td>
</tr>
</tbody>
</table>

ARG0, ARG1

36

8

8

Guo et al., 2019
Damonte et al., 2019
vaswani et al., 2017
Zhu et al., 2019
Cai et al., 2020
Wang et al., 2020

O(n)

O(n^2)

O(n^2+n)

36

8

8

O(n)

O(n^2)

O(n^2+n)
**Graph Convolution v.s. Self-Attention**

**LDGCN** *(This work)*

- **Time Complexity**: $O(n)$
- **Neighbour Info**: $k$-order ($k<n$)
- **Layers**: 36 *(fewer #params)*

**SAN** *vaswani et al., 2017*

- **Time Complexity**: $O(n^2)$
- **Neighbour Info**: $n$-order
- **Layers**: 8

**Structured SAN** *Zhu et al., 2019 Cai et al., 2020 Wang et al., 2020*

- **Time Complexity**: $O(n^2+n)$
- **Neighbour Info**: $n$-order
- **Layers**: 8
Research Questions

• **Q1**: Can we build a more effective graph encoder solely based on graph convolution?

• **Q2**: Existing models require large model size to maintain model capacity. Can we build a model with fewer parameters while have similar model capacity?
Research Questions

• **Q1:** Can we build a more effective graph encoder solely based on graph convolution?

• **A:** Dynamic fusion mechanism is introduced to graph convolutions to integrate information from higher order neighbors
Dynamic Fusion Mechanism

- *Vanilla graph convolutional layer* only takes 1-order adjacency matrix as the input.
Dynamic Fusion Mechanism

- Each graph convolutional layer takes $k$ number of $k$-order adjacency matrices as inputs (here $k=3$).
Dynamic Fusion Mechanism

- Each graph convolutional layer takes $k$ number of $k$-order adjacency matrices as inputs (here $k=3$).
- The dynamic fusion mechanism is able to integrate information from 1- to $k$-hop neighbors.

Vanilla GCN Layer

LDGCN Layer
**Dynamic Fusion Mechanism**

- **Computational Overhead**: In practice, there is no need to calculate or store $A^k$. $A^kH_l$ is computed with right-to-left multiplication.
- If $k=3$, we can calculate $A^3H_l$ as $(A(A(AH_l)))$. $A$ is stored as vanilla GCNs.

![Diagram showing Vanilla GCN Layer and LDGCN Layer with dynamic fusion](attachment:image.png)
Research Questions

- **Q2**: Can we build a lighter model that still achieves competitive performance?

- **A**: Weight sharing strategies are proposed to *reduce memory usage* and *speed up inference*. 
Research Questions

- **Q2**: Existing models require large model size to maintain model capacity. Can we build a model with fewer parameters while having similar model capacity?

- **A**: Weight sharing strategies are proposed to **reduce memory usage** and **speed up inference**.
  - Group Graph Convolution
  - Weight Tied Convolution
Group Graph Convolution: Depthwise

- **Depthwise graph convolution**: the input and output representation of the $l$-th layer $H_l$ and $H_{l+1}$ are partitioned into $N$ disjoint groups (here $N=3$).
Group Graph Convolution: Layerwise

- **Densely Connected graph convolution:** each layer takes the concatenation of outputs from all preceding layers as its input
- **Layerwise graph convolution:** The input representation $H_0$ is partitioned into $M=3$ disjoint groups.
We further adopt a more aggressive strategy where parameters are shared across all layers.

Theoretically, weight tied networks can be unrolled to any depth, typically with improved feature abstractions as depth increases (Bai et al., 2019).

Vanilla Graph Convolutions

Weight Tied Graph Convolutions
## Dataset Statistics

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Train</th>
<th>Dev</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>AMR 1.0 (LDC2015E86)</td>
<td>16,833</td>
<td>1,368</td>
<td>1,371</td>
</tr>
<tr>
<td>AMR 2.0 (LDC2017T10)</td>
<td>36,521</td>
<td>1,368</td>
<td>1,371</td>
</tr>
<tr>
<td>AMR 3.0 (LDC2020T02)</td>
<td>55,635</td>
<td>1,722</td>
<td>1,898</td>
</tr>
</tbody>
</table>
Main Results

Results on AMR 1.0

- **LDGCN_WT**: our model with weight tied convolution;
- **LDGCN_GC**: our model with group graph convolution;
Main Results

Results on AMR 2.0

- **LDGCN_WT**: our model with weight tied convolution;
- **LDGCN_GC**: our model with group graph convolution;
Main Results

Results on AMR 3.0

- **LDGCN_WT**: our model with weight tied convolution;
- **LDGCN_GC**: our model with group graph convolution;
## AMR 1.0-With External Training Data

<table>
<thead>
<tr>
<th>Model</th>
<th>External</th>
<th>BLEU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Seq2Seq (Konstas et al., 2017)</td>
<td>2M</td>
<td>32.3</td>
</tr>
<tr>
<td>GraphLSTM (Song et al., 2018)</td>
<td>2M</td>
<td>28.2</td>
</tr>
<tr>
<td>Transformer (Wang et al., 2020)</td>
<td>2M</td>
<td>31.6</td>
</tr>
<tr>
<td>GT_Dual (Wang et al., 2020)</td>
<td>2M</td>
<td>36.4</td>
</tr>
<tr>
<td>Our LDGCN_GC</td>
<td>0.5M</td>
<td>36.0</td>
</tr>
<tr>
<td>Our LDGCN_WT</td>
<td>0.5M</td>
<td>36.8</td>
</tr>
<tr>
<td>Model</td>
<td>Speed</td>
<td></td>
</tr>
<tr>
<td>---------------------</td>
<td>--------</td>
<td></td>
</tr>
<tr>
<td>Transformer</td>
<td>1.00x</td>
<td></td>
</tr>
<tr>
<td>DeepGCN</td>
<td>1.21x</td>
<td></td>
</tr>
<tr>
<td>Our LDGCN_GC</td>
<td>1.17x</td>
<td></td>
</tr>
<tr>
<td>Our LDGCN_WT</td>
<td>1.22x</td>
<td></td>
</tr>
</tbody>
</table>
Performance against Graph Size

CHRF+++ vs. Graph Sizes

- GT_SAN
- LDGCN_GC
Summary

- We propose the novel LDGCN model, which maintains a better balance between parameter efficiency and model capacity.
- Extensive experiments show that the LDGCN model outperforms state-of-the-art approaches with significantly fewer parameters.
Thank You

Code Available

https://github.com/yanzhangnlp/LDGCNs