Graph Neural Networks

CPSC 677 Advanced NLP
7th Oct 2020

Irene Li
Outline

Introduction

A Brief History, Earlier Research: Pagerank, DeepWalk

Recent Papers on NLP

Future Directions

Discussion
Introduction: Graphs

Social networks

Economic networks

Biomedical networks

Information networks: Web & citations

Internet

Networks of neurons
Introduction: Graphs

Event Graphs

Knowledge Graphs

Disease pathways

Molecules

Scene Graphs

Cell-cell similarity networks
A brief history...

A Graph Neural Network is a type of Neural Network which directly operates on the Graph structure.
PageRank works by counting the number and quality of links to a page to determine a rough estimate of how important the website is. The underlying assumption is that more important websites are likely to receive more links from other websites.
Apply Pagerank to summarization: Textrank

1. Separate the text into sentences
2. Build a sparse matrix of words and the count it appears in each sentence to calculate tf-idf matrix.
3. Construct the similarity matrix between sentences
4. Use Pagerank to score the sentences in graph

1. Separate the Text into Sentences

“Hi world! Hello world! This is Andrew.”

[“Hi world!”, “Hello world!”, “This is Andrew.”]
2. Build a sparse matrix of words and the count it appears in each sentence, get tf-idf
3. Construct the similarity matrix between sentences

TF-IDF matrix

\[
\begin{pmatrix}
1 & 0.366 & 0 \\
0.366 & 1 & 0 \\
0 & 0 & 1 \\
\end{pmatrix}
\]

similarity matrix

4. Use Pagerank to score the sentences in graph


- Rank the sentences with underlying assumption that “summary sentences” are similar to most other sentences
Introduction: Graphs

Vertices V
Edges E

\[ G = (V, E) \]
Early Research: DeepWalk (2014)

Goal: Learning latent representations of vertices in a network.

Idea: Skip-Gram Model (Word2vec)
Node representation

(a) Input: Karate Graph  (b) Output: Representation

(a) Random walk generation.
GCN: Graph **Convolutional** Network (ICLR, 2017)

Focus on **Graph structure data & convolution**.

How to understand traditional CNN?
From CNN to GCN (1)

Filter: shared with all locations

http://petar-v.com/GAT/
From CNN to GCN (2)

to go to the next layer using “graph convolution”
**GCN**: formulation

Input:

- **A** is adjacency matrix
- **X** is the feature matrix, shape $N \times D$, can be one-hot

$H$ is the outputs of the current layer $l$

Then choosing $f$.

Every neural network layer:

$$H^{(l+1)} = f(H^{(l)}, A)$$
**GCN**: formulation

Simple version of $f$: a single layer NN, with a ReLU as $f$.

$W$ is the parameter matrix.

\[
H^{(l+1)} = f(H^{(l)}, A)
\]

\[
H^{(l+1)} = \sigma(AH^{(l)}W^{(l)})
\]

For the first layer, $H^0$:

\[
H^{(0)} = X
\]
**GCN**: formulation

Two tricks:

1) Add information of the node from itself: $A = A + I$
2) Normalize the adjacency matrix $A$

Multiplication with $A$ will completely change the scale of the feature vectors (especially in neural networks)

Degree matrix $D$

$$D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$$
**GCN: formulation**

**Definition 1** The normalized adjacency matrix is

\[
\mathcal{A} \equiv D^{-1/2}AD^{-1/2},
\]

where \( A \) is the adjacency matrix of \( G \) and \( D = \text{diag}(d) \) for \( d(i) \) the degree of node \( i \).

For a graph \( G \) (with no isolated vertices), we can see that

\[
D^{-1/2} = \left( \begin{array}{cccc}
\frac{1}{\sqrt{d(1)}} & 0 & \cdots & 0 \\
0 & \frac{1}{\sqrt{d(2)}} & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & \frac{1}{\sqrt{d(n)}}
\end{array} \right).
\]

\[
\hat{A} = A + I
\]

\[
f(H^{(l)}, A) = \sigma \left( \hat{D}^{-\frac{1}{2}} \hat{A} \hat{D}^{-\frac{1}{2}} H^{(l)} W^{(l)} \right)
\]
**GCN**: formulation

\[
H^{(l+1)} = f(H^{(l)}, A)
\]

\[
H^{(l+1)} = \sigma(AH^{(l)}W^{(l)})
\]

\[
f(H^{(l)}, A) = \sigma(\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}H^{(l)}W^{(l)})
\]

**Node classification**: an MLP Layer at the end.

PyTorch: [https://github.com/tkipf/pygcn](https://github.com/tkipf/pygcn)

[http://opennmt.net/OpenNMT-py/main.html](http://opennmt.net/OpenNMT-py/main.html)
Results: [supervised learning] classify the nodes in citation network

<table>
<thead>
<tr>
<th>Method</th>
<th>CiteSeer</th>
<th>Cora</th>
<th>Pubmed</th>
<th>NELL</th>
</tr>
</thead>
<tbody>
<tr>
<td>ManiReg [3]</td>
<td>60.1</td>
<td>59.5</td>
<td>70.7</td>
<td>21.8</td>
</tr>
<tr>
<td>SemiEmb [28]</td>
<td>59.6</td>
<td>59.0</td>
<td>71.1</td>
<td>26.7</td>
</tr>
<tr>
<td>LP [32]</td>
<td>45.3</td>
<td>68.0</td>
<td>63.0</td>
<td>26.5</td>
</tr>
<tr>
<td>DeepWalk [22]</td>
<td>43.2</td>
<td>67.2</td>
<td>65.3</td>
<td>58.1</td>
</tr>
<tr>
<td>ICA [18]</td>
<td>69.1</td>
<td>75.1</td>
<td>73.9</td>
<td>23.1</td>
</tr>
<tr>
<td>Planetoid* [29]</td>
<td>64.7 (26s)</td>
<td>75.7 (13s)</td>
<td>77.2 (25s)</td>
<td>61.9 (185s)</td>
</tr>
<tr>
<td>GCN (this paper)</td>
<td><strong>70.3 (7s)</strong></td>
<td><strong>81.5 (4s)</strong></td>
<td><strong>79.0 (38s)</strong></td>
<td><strong>66.0 (48s)</strong></td>
</tr>
</tbody>
</table>
Variational Graph Auto-Encoders for link prediction

Recall: encoder and decoder

Similar models on texts, images, ...

Can we do the same thing on graph-structure data?
Details of VGAE

Two layers of GCN as the **encoder**.

$$\text{GCN}(X, A) = \tilde{A} \text{ReLU}(\tilde{A}XW_0)W_1$$

$$\mu = \text{GCN}_\mu(X, A) \quad \log \sigma = \text{GCN}_\sigma(X, A)$$

$$q(Z | X, A) = \prod_{i=1}^{N} q(z_i | X, A), \quad \text{with} \quad q(z_i | X, A) = \mathcal{N}(z_i | \mu_i, \text{diag}(\sigma_i^2))$$

An inner product as the **Decoder**

$$p(A | Z) = \prod_{i=1}^{N} \prod_{j=1}^{N} p(A_{ij} | z_i, z_j), \quad \text{with} \quad p(A_{ij} = 1 | z_i, z_j) = \sigma(z_i^\top z_j)$$

$$\mathcal{L} = \mathbb{E}_{q(Z|X, A)}[\log p(A | Z)] - \text{KL}[q(Z | X, A) \| p(Z)]$$
A simple version (without variation) ...

Graph Auto-Encoders (GAE)

For a simple GAE, we will get rid of the distribution restrictions, simply take a GCN as the encoder and an inner product function as the decoder:

\[
\hat{A} = \sigma (ZZ^\top) \\
Z = GCN(X, A)
\]
Results: [Semi-supervised] Classify edges/non-edges

The models are trained on an incomplete version of these datasets where parts of the citation links (edges) have been removed, while all node features are kept. Complete X; incomplete A → A

Table 1: Link prediction task in citation networks. See [1] for dataset details.

<table>
<thead>
<tr>
<th>Method</th>
<th>Cora (AUC, AP)</th>
<th>Citeseer (AUC, AP)</th>
<th>Pubmed (AUC, AP)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SC [5]</td>
<td>84.6 ± 0.01, 88.5 ± 0.00</td>
<td>80.5 ± 0.01, 85.0 ± 0.01</td>
<td>84.2 ± 0.02, 87.8 ± 0.01</td>
</tr>
<tr>
<td>DW [6]</td>
<td>83.1 ± 0.01, 85.0 ± 0.00</td>
<td>80.5 ± 0.02, 83.6 ± 0.01</td>
<td>84.4 ± 0.00, 84.1 ± 0.00</td>
</tr>
<tr>
<td>GAE*</td>
<td>84.3 ± 0.02, 88.1 ± 0.01</td>
<td>78.7 ± 0.02, 84.1 ± 0.02</td>
<td>82.2 ± 0.01, 87.4 ± 0.00</td>
</tr>
<tr>
<td>VGAE*</td>
<td>84.0 ± 0.02, 87.7 ± 0.01</td>
<td>78.9 ± 0.03, 84.1 ± 0.02</td>
<td>82.7 ± 0.01, 87.5 ± 0.01</td>
</tr>
<tr>
<td>GAE</td>
<td>91.0 ± 0.02, 92.0 ± 0.03</td>
<td>89.5 ± 0.04, 89.9 ± 0.05</td>
<td>96.4 ± 0.00, 96.5 ± 0.00</td>
</tr>
<tr>
<td>VGAE</td>
<td>91.4 ± 0.01, 92.6 ± 0.01</td>
<td>90.8 ± 0.02, 92.0 ± 0.02</td>
<td>94.4 ± 0.02, 94.7 ± 0.02</td>
</tr>
</tbody>
</table>
Question: Better ways to “move” to the next layer?

Attention!
Recall Attention in seq2seq

Source image: https://speech.ee.ntu.edu.tw/~tlkaqk/
Attention in graph: looking at the neighbors

http://petar-v.com/GAT/
Graph Attention Networks (ICLR, 2019)

**Self-attention**: by looking at the neighbors with different weights.

Which neighbor has a larger impact/similar to it? Calculate a node pair (i,j):

$$e_{ij} = a(\tilde{h}_i, \tilde{h}_j)$$

Normalize over all the neighbors:

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k \in N_i} \exp(e_{ik})}$$

Use neighbors to represent the current node: weighted average

$$\tilde{h}_i' = \sigma \left( \sum_{j \in N_i} \alpha_{ij}^k W^k \tilde{h}_j \right)$$
Results

Table 2: Summary of results in terms of classification accuracies, for Cora, Citeseer and Pubmed. GCN-64* corresponds to the best GCN result computing 64 hidden features (using ReLU or ELU).

<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>79.0%</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>
About GATs

Better representation ability than GCN with attention mechanism.

Shared attention module: locally, efficient.

More: http://petar-v.com/GAT/
4 things you must know about GNNs

1. Graph-structural data?
2. Adjacency matrix A?
3. Node representation X?
4. Propagation Rule f?

Fit in a proper scenario!

How could we utilize GNNs in NLP?
Paper 1: **Graph Convolutional Networks for Text Classification** (AAAI, 2019)

Highlights:

- Text classification: normally treated as sequences;
- Investigate solving this task using GCN-based models;
- No complicated embeddings are needed to initialize the nodes: 1-hot vectors. Then it jointly learns the embeddings for both words and documents;
- Efficiency and robustness.

Graph Convolutional Networks for Text Classification

Liang Yao, Chengsheng Mao, Yuan Luo*
Northwestern University
Chicago IL 60611
{liang.yao, chengsheng.mao, yuan.luo}@northwestern.edu
Background: document classification

**Representation learning methods:** TF-IDF, Bag-of-word, word2vec, contextualized word embeddings...

**Recent methods for classification:** CNN, LSTM, BERT

**GCNs:** Graph neural networks have been effective at tasks thought to have rich relational structure and can preserve global structure information of a graph in graph embeddings.

Treat each individual sentence/document as sequences; To some extent, each training sample is independent.

Utilizing relations BETWEEN the corpus?
Building a graph for the corpus

Nodes: docs and tokens; Task: node classification.
Adjacency Matrix $A$

Word node and document node.

Pointwise mutual information (PMI)

$$A_{ij} = \begin{cases} 
\text{PMI}(i, j) & \text{i, j are words, PMI}(i, j) > 0 \\
\text{TF-IDF}_{ij} & \text{i is document, j is word} \\
1 & i = j \\
0 & \text{otherwise}
\end{cases}$$

$$\text{PMI}(i, j) = \log \frac{p(i, j)}{p(i)p(j)}$$

$$p(i, j) = \frac{\#W(i, j)}{\#W}$$

$$p(i) = \frac{\#W(i)}{\#W}$$
Node representation $X$

1-hot vector to initialize each single node: tokens and documents

Propagation rule

$$Z = \text{softmax}(\tilde{A} \text{ReLU}(\tilde{A}XW_0)W_1)$$

$$\mathcal{L} = - \sum_{d \in \mathcal{Y}_D} \sum_{f=1}^{F} Y_{df} \ln Z_{df}$$

Two GCN layers: A two-layer GCN can allow message passing among nodes that are at maximum two steps away.
Experiments: public datasets

<table>
<thead>
<tr>
<th>Dataset</th>
<th># Docs</th>
<th># Training</th>
<th># Test</th>
<th># Words</th>
<th># Nodes</th>
<th># Classes</th>
<th>Average Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>20NG</td>
<td>18,846</td>
<td>11,314</td>
<td>7,532</td>
<td>42,757</td>
<td>61,603</td>
<td>20</td>
<td>221.26</td>
</tr>
<tr>
<td>R8</td>
<td>7,674</td>
<td>5,485</td>
<td>2,189</td>
<td>7,688</td>
<td>15,362</td>
<td>8</td>
<td>65.72</td>
</tr>
<tr>
<td>R52</td>
<td>9,100</td>
<td>6,532</td>
<td>2,568</td>
<td>8,892</td>
<td>17,992</td>
<td>52</td>
<td>69.82</td>
</tr>
<tr>
<td>Ohsumed</td>
<td>7,400</td>
<td>3,357</td>
<td>4,043</td>
<td>14,157</td>
<td>21,557</td>
<td>23</td>
<td>135.82</td>
</tr>
<tr>
<td>MR</td>
<td>10,662</td>
<td>7,108</td>
<td>3,554</td>
<td>18,764</td>
<td>29,426</td>
<td>2</td>
<td>20.39</td>
</tr>
</tbody>
</table>
## Experimental results

<table>
<thead>
<tr>
<th>Model</th>
<th>20NG</th>
<th>R8</th>
<th>R52</th>
<th>Ohsumed</th>
<th>MR</th>
</tr>
</thead>
<tbody>
<tr>
<td>TF-IDF + LR</td>
<td>0.8319 ± 0.0000</td>
<td>0.9374 ± 0.0000</td>
<td>0.8695 ± 0.0000</td>
<td>0.5466 ± 0.0000</td>
<td>0.7459 ± 0.0000</td>
</tr>
<tr>
<td>CNN-rand</td>
<td>0.7693 ± 0.0061</td>
<td>0.9402 ± 0.0057</td>
<td>0.8537 ± 0.0047</td>
<td>0.4387 ± 0.0100</td>
<td>0.7498 ± 0.0070</td>
</tr>
<tr>
<td>CNN-non-static</td>
<td>0.8215 ± 0.0052</td>
<td>0.9571 ± 0.0052</td>
<td>0.8759 ± 0.0048</td>
<td>0.5844 ± 0.0106</td>
<td><strong>0.7775 ± 0.0072</strong></td>
</tr>
<tr>
<td>LSTM</td>
<td>0.6571 ± 0.0152</td>
<td>0.9368 ± 0.0082</td>
<td>0.8554 ± 0.0113</td>
<td>0.4113 ± 0.0117</td>
<td>0.7506 ± 0.0044</td>
</tr>
<tr>
<td>LSTM (pretrain)</td>
<td>0.7543 ± 0.0172</td>
<td>0.9609 ± 0.0019</td>
<td>0.9048 ± 0.0086</td>
<td>0.5110 ± 0.0150</td>
<td>0.7733 ± 0.0089</td>
</tr>
<tr>
<td>Bi-LSTM</td>
<td>0.7318 ± 0.0185</td>
<td>0.9631 ± 0.0033</td>
<td>0.9054 ± 0.0091</td>
<td>0.4927 ± 0.0107</td>
<td>0.7768 ± 0.0086</td>
</tr>
<tr>
<td>PV-DBOW</td>
<td>0.7436 ± 0.0018</td>
<td>0.8587 ± 0.0010</td>
<td>0.7829 ± 0.0011</td>
<td>0.4665 ± 0.0019</td>
<td>0.6109 ± 0.0010</td>
</tr>
<tr>
<td>PV-DM</td>
<td>0.5114 ± 0.0022</td>
<td>0.5207 ± 0.0004</td>
<td>0.4492 ± 0.0005</td>
<td>0.2950 ± 0.0007</td>
<td>0.5947 ± 0.0038</td>
</tr>
<tr>
<td>PTE</td>
<td>0.7674 ± 0.0029</td>
<td>0.9669 ± 0.0013</td>
<td>0.9071 ± 0.0014</td>
<td>0.5358 ± 0.0029</td>
<td>0.7023 ± 0.0036</td>
</tr>
<tr>
<td>fastText</td>
<td>0.7938 ± 0.0039</td>
<td>0.9613 ± 0.0021</td>
<td>0.9281 ± 0.0009</td>
<td>0.5770 ± 0.0049</td>
<td>0.7514 ± 0.0020</td>
</tr>
<tr>
<td>fastText (bigrams)</td>
<td>0.7967 ± 0.0029</td>
<td>0.9474 ± 0.0011</td>
<td>0.9099 ± 0.0005</td>
<td>0.5569 ± 0.0039</td>
<td>0.7624 ± 0.0012</td>
</tr>
<tr>
<td>SWEM</td>
<td>0.8516 ± 0.0029</td>
<td>0.9532 ± 0.0026</td>
<td>0.9294 ± 0.0024</td>
<td>0.6312 ± 0.0055</td>
<td>0.7665 ± 0.0063</td>
</tr>
<tr>
<td>LEAM</td>
<td>0.8191 ± 0.0024</td>
<td>0.9331 ± 0.0024</td>
<td>0.9184 ± 0.0023</td>
<td>0.5858 ± 0.0079</td>
<td>0.7695 ± 0.0045</td>
</tr>
<tr>
<td>Graph-CNN-C</td>
<td>0.8142 ± 0.0032</td>
<td>0.9699 ± 0.0012</td>
<td>0.9275 ± 0.0022</td>
<td>0.6386 ± 0.0053</td>
<td>0.7722 ± 0.0027</td>
</tr>
<tr>
<td>Graph-CNN-S</td>
<td>–</td>
<td>0.9680 ± 0.0020</td>
<td>0.9274 ± 0.0024</td>
<td>0.6282 ± 0.0037</td>
<td>0.7699 ± 0.0014</td>
</tr>
<tr>
<td>Graph-CNN-F</td>
<td>–</td>
<td>0.9689 ± 0.0006</td>
<td>0.9320 ± 0.0004</td>
<td>0.6304 ± 0.0077</td>
<td>0.7674 ± 0.0021</td>
</tr>
<tr>
<td>Text GCN</td>
<td><strong>0.8634 ± 0.0009</strong></td>
<td><strong>0.9707 ± 0.0010</strong></td>
<td><strong>0.9356 ± 0.0018</strong></td>
<td><strong>0.6836 ± 0.0056</strong></td>
<td><strong>0.7674 ± 0.0020</strong></td>
</tr>
</tbody>
</table>
Test accuracy by varying training data proportions.

Effects of the Size of Labeled Data:

GCN can perform quite well with low label rate (Kipf and Welling, 2017)
Conclusion

- Text Classification: **sequential data** vs. **graph-structured data**
- Why GCN works so well: the text graph can capture both document-word relations and global word-word relations.
- When GCN is not better than CNN/LSTM: GCN ignores word orders that are very useful in **sentiment classification**. [In MR dataset]
Paper 2: **Heterogeneous Graph Neural Networks for Extractive Document Summarization** (ACL, 2020)

**Highlights:**

As a crucial step in extractive document summarization, learning cross-sentence relations has been explored by a plethora of approaches.

Propose a heterogeneous graph-based neural network for extractive summarization (HeterSumGraph): contains **semantic nodes** of different granularity levels apart from sentences.

Evaluated on three benchmark datasets, both single-doc and multi-doc summarization.

---

**Heterogeneous Graph Neural Networks for Extractive Document Summarization**

Danqing Wang; Pengfei Liu; Yining Zheng, Xipeng Qiu†, Xuanjing Huang  
Shanghai Key Laboratory of Intelligent Information Processing, Fudan University  
School of Computer Science, Fudan University  
825 Zhangheng Road, Shanghai, China  
{dqwang18,pfliu14,ynzheng19,xpqiuxjhuang}@fudan.edu.cn
Related Work

Extractive Document Summarization
   Traditional methods.
   Transformers, pre-trained models
   Graph-based models: (Yasunaga et al., 2017; Xu et al., 2019).

Heterogeneous Graph for NLP
   Homogeneous vs. Heterogeneous graph: single type of nodes vs. multiple types of nodes
   Recent explorations:
      a heterogeneous graph consisting of topic, word and sentence nodes and uses the markov chain model for the iterative update (Wei, 2012)
Model Overview: HeterSumGraph

Sequence labeling

\[ D = \{ s_1, \cdots, s_n \} \]
\[ y_1, \cdots, y_n (y_i \in \{0, 1\}) \]

Heterogeneous graph

Relay nodes: basic semantic nodes
(e.g. words, concepts, etc.)

Supernodes: other units of discourse
(e.g. phrases, sentences, documents, etc.)
Document as a Heterogeneous Graph

- Word: 300d GloVe
- Each sentence: CNN and Bi-LSTM
- Edge feature: importance of relationships between word and sentence nodes (TF-IDF)

Heterogeneous Graph Layer (w. attention)

\[
\begin{align*}
z_{ij} &= \text{LeakyReLU}\left( W_a [W_q h_i; W_k h_j] \right), \\
\alpha_{ij} &= \frac{\exp(z_{ij})}{\sum_{l \in N_i} \exp(z_{il})}, \\
u_i &= \sigma\left( \sum_{j \in N_i} \alpha_{ij} W_v h_j \right), \\
u_i &= \|_{k=1}^{K} \sigma\left( \sum_{j \in N_i} \alpha_{ij}^k W^k h_i \right)
\end{align*}
\]
Sentence selector

Node classification on sentence nodes.

Cross-entropy.

Single-doc extractive summarization within a het-graph.
Experiments: single-doc summarization

GloVe embeddings;
Limits to 50 sentences for each doc.
NYT50: news data
100,834 and 9706 for training and testing

Trigram Blocking: remove redundancy.

<table>
<thead>
<tr>
<th>Model</th>
<th>R-1</th>
<th>R-2</th>
<th>R-L</th>
</tr>
</thead>
<tbody>
<tr>
<td>First sentence (Durrett et al., 2016)</td>
<td>28.60</td>
<td>17.30</td>
<td>-</td>
</tr>
<tr>
<td>First k words (Durrett et al., 2016)</td>
<td>35.70</td>
<td>21.60</td>
<td>-</td>
</tr>
<tr>
<td>LEAD-3</td>
<td>38.99</td>
<td>18.74</td>
<td>35.35</td>
</tr>
<tr>
<td>ORACLE</td>
<td>60.54</td>
<td>40.75</td>
<td>57.22</td>
</tr>
<tr>
<td>COMPRESS (Durrett et al., 2016)</td>
<td>42.20</td>
<td>24.90</td>
<td>-</td>
</tr>
<tr>
<td>SUMO (Liu et al., 2019)</td>
<td>42.30</td>
<td>22.70</td>
<td>38.60</td>
</tr>
<tr>
<td>PG* (See et al., 2017)</td>
<td>43.71</td>
<td>26.40</td>
<td>-</td>
</tr>
<tr>
<td>DRM (Paulus et al., 2017)</td>
<td>42.94</td>
<td>26.02</td>
<td>-</td>
</tr>
<tr>
<td>Ext-BiLSTM</td>
<td>46.32</td>
<td>25.84</td>
<td>42.16</td>
</tr>
<tr>
<td>Ext-Transformer</td>
<td>45.07</td>
<td>24.72</td>
<td>40.85</td>
</tr>
<tr>
<td>HSG</td>
<td>46.89</td>
<td>26.26</td>
<td>42.58</td>
</tr>
<tr>
<td>HSG + Tri-Blocking</td>
<td>46.57</td>
<td>25.94</td>
<td>42.25</td>
</tr>
</tbody>
</table>

Table 2: Limited-length ROUGE Recall on NYT50 test set. The results of models with * are copied from Liu and Lapata (2019b) and '-' means that the original paper did not report the result.
Extending to multi-doc summarization

- Establish the **document-level** relationship in the same way as the sentence-level by just adding supernodes for documents.
- Word nodes become bridges of doc and sent nodes.
- Doc node takes the mean-pooling of its sentence node features as its initial state.

Figure 3: Graph structure of HETERDocSUMGraph for multi-document summarization (corresponding to the Graph Layer part of Figure 1). Green, blue and orange boxes represent word, sentence and document nodes respectively. $d_1$ consists of $s_{11}$ and $s_{12}$ while $d_2$ contains $s_{21}$ and $s_{22}$. As a relay node, the relation of **document-document**, **sentence-sentence**, and **sentence-document** can be built through the common word nodes. For example, sentence $s_{11}$, $s_{12}$ and $s_{21}$ share the same word $w_1$, which connects them across documents.
Experiments: multi-doc summarization

Multi-News (Fabbri, ACL 2019)
2-10 sources; each limits to 500 tokens
44,972/5,622/5,622 for training, validation and test

<table>
<thead>
<tr>
<th>Model</th>
<th>R-1</th>
<th>R-2</th>
<th>R-L</th>
</tr>
</thead>
<tbody>
<tr>
<td>First-1</td>
<td>25.44</td>
<td>7.06</td>
<td>22.12</td>
</tr>
<tr>
<td>First-2</td>
<td>35.70</td>
<td>10.28</td>
<td>31.71</td>
</tr>
<tr>
<td>First-3</td>
<td>40.21</td>
<td>12.13</td>
<td>37.13</td>
</tr>
<tr>
<td>ORACLE</td>
<td>52.32</td>
<td>22.23</td>
<td>47.93</td>
</tr>
<tr>
<td>LexRank* (Erkan and Radev, 2004)</td>
<td>41.77</td>
<td>13.81</td>
<td>37.87</td>
</tr>
<tr>
<td>TextRank* (Mihalcea and Tarau, 2004)</td>
<td>41.95</td>
<td>13.86</td>
<td>38.07</td>
</tr>
<tr>
<td>MMR* (Carbonell and Goldstein, 1998)</td>
<td>44.72</td>
<td>14.92</td>
<td>40.77</td>
</tr>
<tr>
<td>PG† (Lebanoff et al., 2018)</td>
<td>44.55</td>
<td>15.54</td>
<td>40.75</td>
</tr>
<tr>
<td>BottomUp† (Gehrmann et al., 2018)</td>
<td>45.27</td>
<td>15.32</td>
<td>41.38</td>
</tr>
<tr>
<td>Hi-MAP† (Fabbri et al., 2019)</td>
<td>45.21</td>
<td>16.29</td>
<td>41.39</td>
</tr>
<tr>
<td>HSG</td>
<td>45.66</td>
<td>16.22</td>
<td>41.80</td>
</tr>
<tr>
<td>HSG + Tri-Blocking</td>
<td>44.92</td>
<td>15.59</td>
<td>40.89</td>
</tr>
<tr>
<td>HDSG</td>
<td><strong>46.05</strong></td>
<td><strong>16.35</strong></td>
<td><strong>42.08</strong></td>
</tr>
<tr>
<td>HDSG + Tri-Blocking</td>
<td>45.55</td>
<td>15.78</td>
<td>41.29</td>
</tr>
</tbody>
</table>

Table 4: Results on the test set of Multi-News. We reproduce models with “*” via the released code and directly use the outputs of † provided by Fabbri et al. (2019) for evaluation.
Conclusion

- Very convenient to adapt single-document graph to multi-document with document nodes.
- Learning **cross-sentence relations**: fine-grained semantic units in the summarization graph.
- No pre-trained models are used (GloVe), but needs more analysis.
- **Missing relations?**
A Graph-based Coarse-to-fine Method for Unsupervised Bilingual Lexicon Induction

Shuo Ren†‡,* Shujie Liu§, Ming Zhou§, Shuai Ma†‡
†SKLSDE Lab, Beihang University, Beijing, China
‡Beijing Advanced Innovation Center for Big Data and Brain Computing, China
§Microsoft Research Asia, Beijing, China
†{shuoren,mashuai}@buaa.edu.cn §{shujliu,mingzhou}@microsoft.com
Bilingual Lexicon Induction (BLI)

Inducing word translations from monolingual corpora of two languages

Input:
- Chinese corpus
- English corpus

Output:
- Dictionary

Becomes an essential part of recent unsupervised machine translation approaches (Lample et al., 2018; Artetxe et al., 2018c; Marie and Fujita, 2018; Ren et al., 2019; Artetxe et al., 2019)

Related Work for BLI

Unsupervised cross-lingual word embeddings.

Start with a seed dictionary -> expand by nearest neighbors:

\[
W^* = \arg \min_{W \in M_d(\mathbb{R})} \|WX - Y\|_F
\]
A Graph-based Coarse-to-fine Method

- Word Graph Construction
- Clique Extraction and Mapping
- Seed Dictionary Induction
- Cross-lingual Embedding Learning
- Inference
Word Graph Construction

Construct a “word graph” for each language

**Vertices**: pre-trained **monolingual embeddings** (top 10k)

**Edges**:

- Cos similarities based on embeddings.
- No self-loops.

CSLS similarity: [Word Translation Without Parallel Data](https://arxiv.org/abs/1802.01610), ICLR 2018
Clique Extraction

“Clique” means a maximum complete subgraph where every two distinct vertices in the clique are adjacent.

Bron-Kerbosch (BK) algorithm is able to recognize cliques in graphs: *more efficient than clustering.*

Find central word of each clique:

1. Average all embeddings;
2. Find the closest word.

Match Cliques on the two languages.
Clique Mapping & Seed Dictionary Induction

Map cliques using the central word <- fully unsupervised!

Monolingual embeddings: on the same space, look at the closest clique.
Cross-lingual Embedding Learning & Inference

Old method: learn a single mapping $W$ to match all words

$$W^* = \underset{W}{\text{argmin}} \|WX - Y\|_F, W^\top W = I$$

Group mapping: learn an individual mapping $M_i$ for each clique $i$:

$$W_{i^*} = \underset{W_i}{\text{argmin}} \|W_iX_i - Y_i\|_F, W_{i^\top}W_i = I$$

Given a new word: find its embedding $x$, find clique, calculate $y$ using $W_i$. 
Experiments

<table>
<thead>
<tr>
<th>Method</th>
<th>en-fr</th>
<th>en-de</th>
<th>en-es</th>
<th>en-it</th>
<th>en-ru</th>
<th>en-zh</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>→</td>
<td>←</td>
<td>→</td>
<td>←</td>
<td>←</td>
<td>←</td>
</tr>
<tr>
<td><strong>Supervised</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Smith et al., 2017)</td>
<td>81.1</td>
<td>82.4</td>
<td>73.5</td>
<td>72.4</td>
<td>81.4</td>
<td>82.9</td>
</tr>
<tr>
<td>(Artetxe et al., 2018a)</td>
<td>80.5</td>
<td>83.1</td>
<td>73.5</td>
<td>73.5</td>
<td>80.5</td>
<td>83.8</td>
</tr>
<tr>
<td>(Joulin et al., 2018)</td>
<td><strong>83.3</strong></td>
<td>84.1</td>
<td><strong>79.1</strong></td>
<td>76.3</td>
<td><strong>84.1</strong></td>
<td><strong>86.3</strong></td>
</tr>
<tr>
<td>(Jawanpuria et al., 2019)</td>
<td>82.1</td>
<td><strong>84.2</strong></td>
<td>74.9</td>
<td><strong>76.7</strong></td>
<td>81.9</td>
<td>85.5</td>
</tr>
<tr>
<td></td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td><strong>Unsupervised</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Conneau et al., 2017)</td>
<td>82.3</td>
<td>81.1</td>
<td>74.0</td>
<td>72.2</td>
<td>81.7</td>
<td>83.3</td>
</tr>
<tr>
<td>(Xu et al., 2018)</td>
<td>77.9</td>
<td>75.5</td>
<td>69.3</td>
<td>67.0</td>
<td>79.5</td>
<td>77.8</td>
</tr>
<tr>
<td>(Alvarez-Melis and Jaakkola, 2018)</td>
<td>81.3</td>
<td>78.9</td>
<td>71.9</td>
<td>72.8</td>
<td>81.7</td>
<td>80.4</td>
</tr>
<tr>
<td>(Artetxe et al., 2018b)</td>
<td>82.3</td>
<td>83.6</td>
<td>75.1</td>
<td>74.3</td>
<td>82.3</td>
<td>84.7</td>
</tr>
<tr>
<td>(Hoshen and Wolf, 2018)</td>
<td>82.3</td>
<td><strong>84.1</strong></td>
<td>74.7</td>
<td>73.0</td>
<td>82.1</td>
<td>84.1</td>
</tr>
<tr>
<td>Ours (without GM)</td>
<td>82.7</td>
<td>83.4</td>
<td><strong>75.5</strong></td>
<td>75.7</td>
<td>82.6</td>
<td>84.8</td>
</tr>
<tr>
<td>Ours (with GM)</td>
<td><strong>82.9</strong></td>
<td>83.9</td>
<td>75.3</td>
<td><strong>76.1</strong></td>
<td><strong>82.9</strong></td>
<td><strong>85.3</strong></td>
</tr>
</tbody>
</table>

Table 1: Precision@1 for the MUSE BLI task. All baselines leverage CSLS to be the retrieve metric during inference except for Xu et al. (2018) which uses cosine similarity. The bold numbers indicate the best results of supervised and unsupervised methods. “GM” means applying the group mapping technique described in §3.4.
### Extracted Cliques

<table>
<thead>
<tr>
<th>id</th>
<th>words</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>, . - ) (</td>
</tr>
<tr>
<td>2</td>
<td><strong>and</strong> also both well addition additionally besides</td>
</tr>
<tr>
<td>3</td>
<td><strong>his</strong> himself him he her</td>
</tr>
<tr>
<td>4</td>
<td>northeastern west <strong>south</strong> southeastern southeast east southwest northeast northwest southwestern north</td>
</tr>
<tr>
<td>5</td>
<td>january march <strong>august</strong> july september october june april december november february</td>
</tr>
<tr>
<td>6</td>
<td>science <strong>scientists</strong> scientific biology mathematics physics chemistry sciences</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Table 5: Examples of English cliques extracted from the word graph in the first iteration. The bold words are the central words in their respective cliques.
## Seed dictionary generation

<table>
<thead>
<tr>
<th>en</th>
<th>fr</th>
<th>VecMap</th>
<th>Ours</th>
<th>zh</th>
<th>VecMap</th>
<th>Ours</th>
</tr>
</thead>
<tbody>
<tr>
<td>and</td>
<td>part(share)</td>
<td>établir(establish)</td>
<td>et(and)</td>
<td>也(too)</td>
<td>/</td>
<td>和(and)</td>
</tr>
<tr>
<td>his</td>
<td>n</td>
<td>matin(morning)</td>
<td>lui(him)</td>
<td>此(now)</td>
<td>第六(sixth)</td>
<td>他(he)</td>
</tr>
<tr>
<td>south</td>
<td>un (a)</td>
<td>avait(had)</td>
<td>ouest(west)</td>
<td>台北(Taipei)</td>
<td>(prize)</td>
<td>北(north)</td>
</tr>
<tr>
<td>august</td>
<td>flotte(fleet)</td>
<td>mars(march)</td>
<td>mars (march)</td>
<td>电影(film)</td>
<td>第五(fifth)</td>
<td>三月(march)</td>
</tr>
<tr>
<td>build</td>
<td>paris(Paris)</td>
<td>seule(alone)</td>
<td>faire(make)</td>
<td>用作(used as)</td>
<td>了解(understand)</td>
<td>形成(form)</td>
</tr>
</tbody>
</table>

Table 4: Examples of seeds produced with different methods. Inside the brackets is the interpretation of the words.
Conclusion

- A Graph-based Coarse-to-fine Method for BLI
- Focus on word-word relationships: word embeddings contain rich information [semantic embeddings promote this method]
- Unsupervised task.
Future Directions

**Heterogeneity**: handle different types of nodes and edges; various forms of inputs, such as images and texts!

[Heterogeneous Graph Attention Network](#)

**Dynamicity**: node/edge inputs may change time by time (dynamic spatial relations).

Social network in a period of time…

**Deep and large graphs, pre-trained graphs**: Super large graphs: graph partitioning

[Strategies for Pre-training Graph Neural Networks](#)
Discussion (1)

What other tasks in NLP can be solved?

Parse trees, knowledge graphs…
Discussion (2)

Is it possible to have multiple edges (A)?

Is it possible to have multiple node representations (X)?

Multiple node representations: [Is a Single Embedding Enough? Learning Node Representations that Capture Multiple Social Contexts](#)
Discussion (3)

What are the drawbacks of GNNs compared with LSTMs and/or CNNs?

How do you choose them?
Related resources

https://sites.google.com/view/deep-gcns


CS224W: Machine Learning with Graphs (Fall 2019)
Thanks

Q&A