Graph Neural Networks

CPSC 677 Advanced NLP
7th Oct 2020

Irene Li

Yale University

Outline

Introduction

A Brief History, Earlier Research: Pagerank, DeepWalk

Recent Papers on NLP

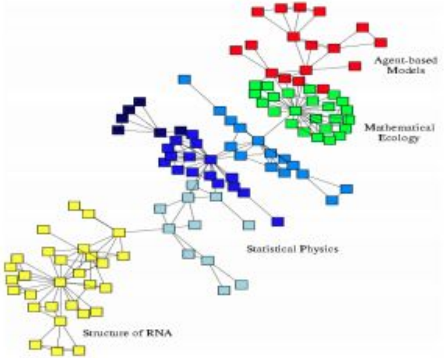
Future Directions

Discussion

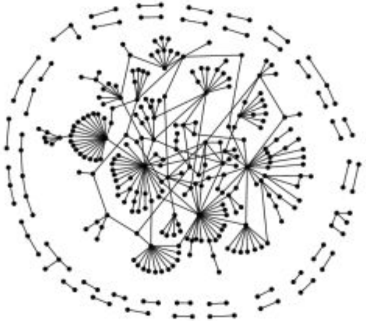
Introduction: Graphs

[Source](#)

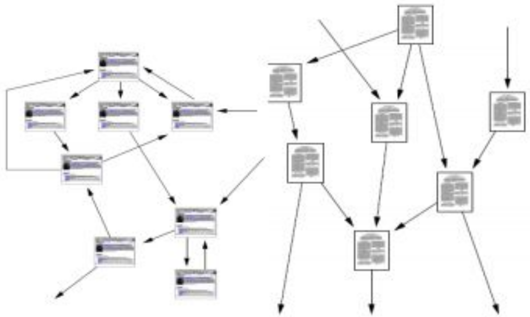
Social networks



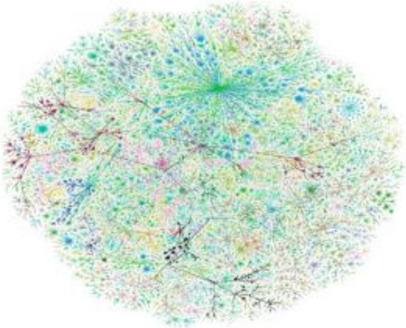
Economic networks



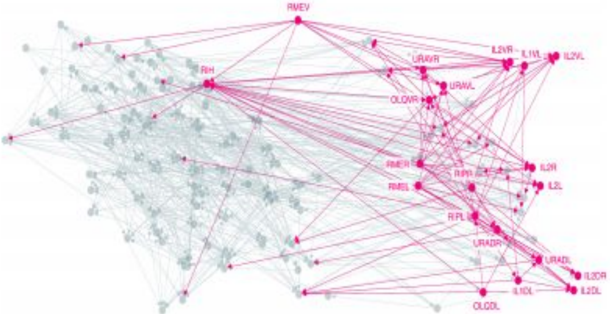
Biomedical networks



Information networks:
Web & citations



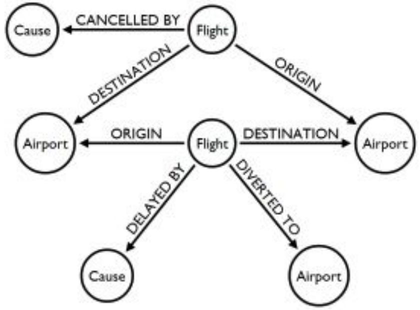
Internet



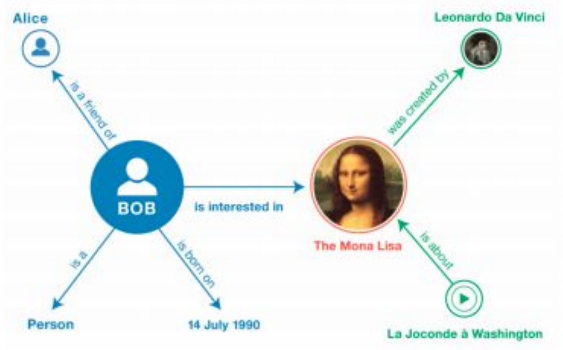
Networks of neurons

Introduction: Graphs

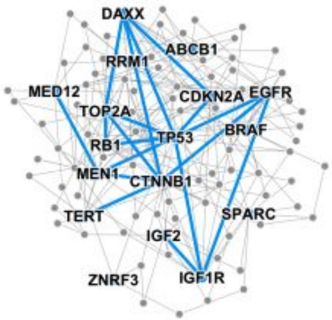
Source



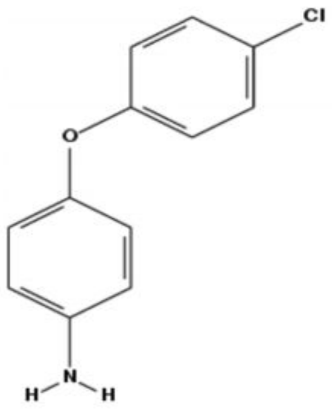
Event Graphs



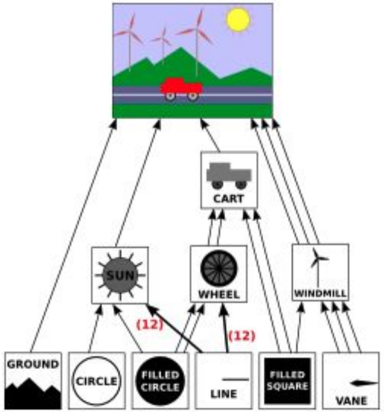
Knowledge Graphs



Disease pathways



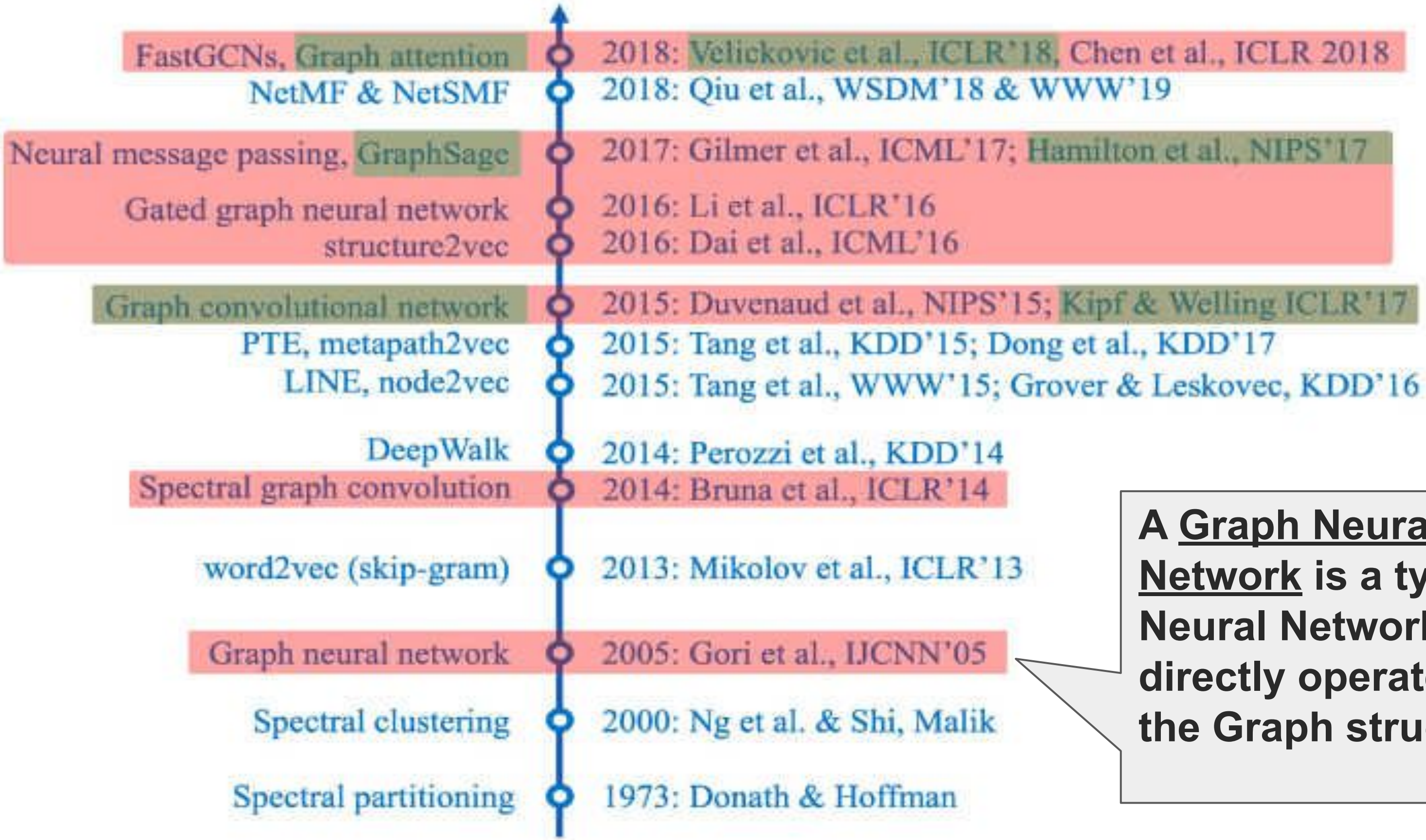
Molecules



Scene Graphs

Cell-cell similarity networks

A brief history...



A <u>Graph Neural Network</u> is a type of Neural Network which directly operates on the Graph structure.

Pagerank

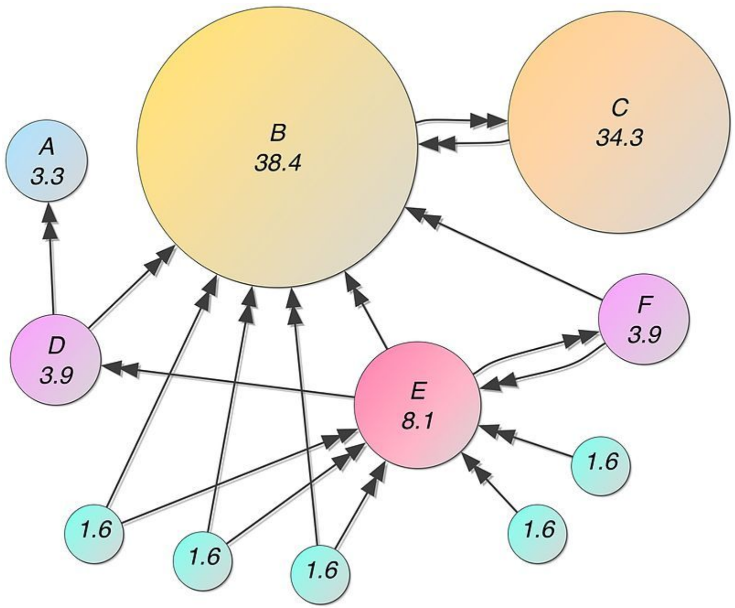
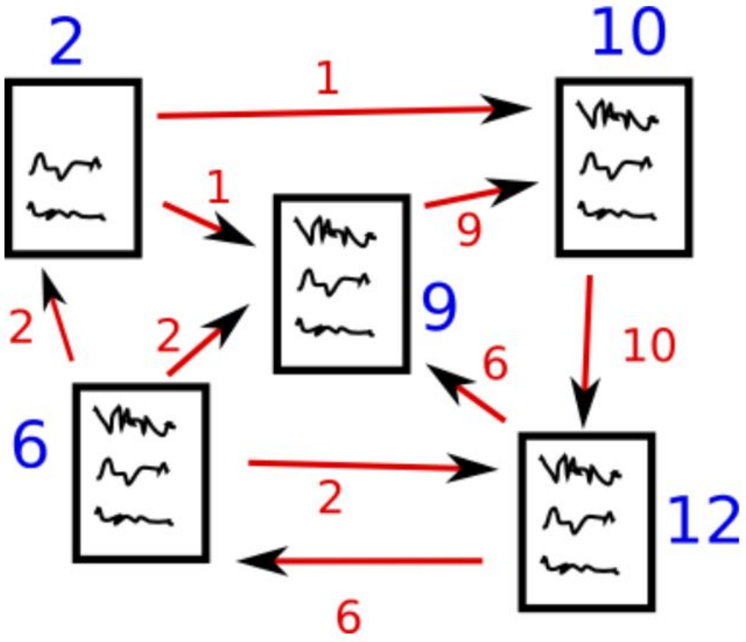


Image source

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.

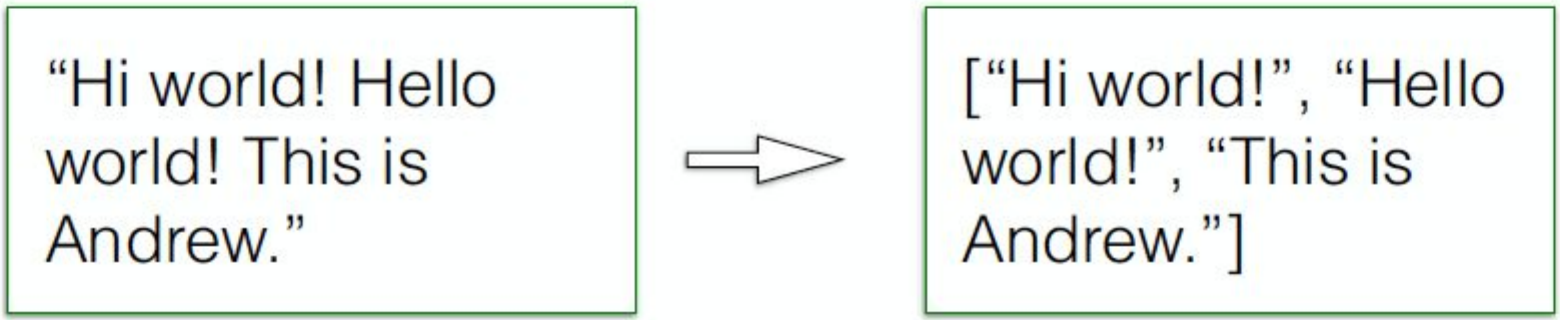
Apply Pagerank to summarization: Textrank

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



https://cmci.colorado.edu/~mpaul/files/emnlp2010.slides.pdf

1. Separate the Text into Sentences



https://cmci.colorado.edu/~mpaul/files/emnlp2010.slides.pdf

2. Build a sparse matrix of words and the count it appears in each sentence, get tf-idf

["Hi world!", "Hello world!", "This is Andrew."] → TF-IDF matrix

https://cmci.colorado.edu/~mpaul/files/emnlp2010.slides.pdf

3. Construct the similarity matrix between sentences

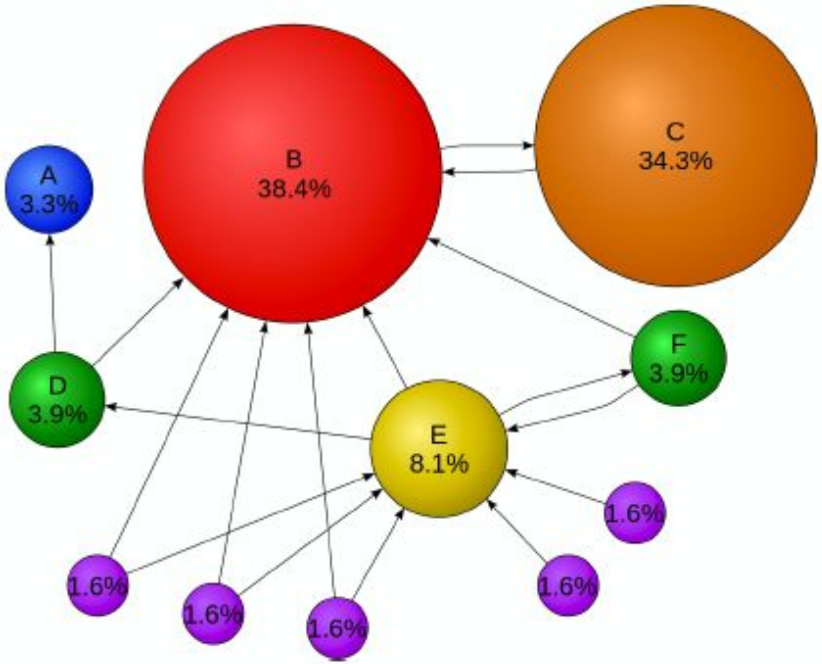
TF-IDF matrix $\Rightarrow$

$$\begin{matrix} 1 & 0.366 & 0 \\ 0.366 & 1 & 0 \\ 0 & 0 & 1 \end{matrix}$$

similarity matrix

https://cmci.colorado.edu/~mpaul/files/emnlp2010.slides.pdf

4. Use Pagerank to score the sentences in graph



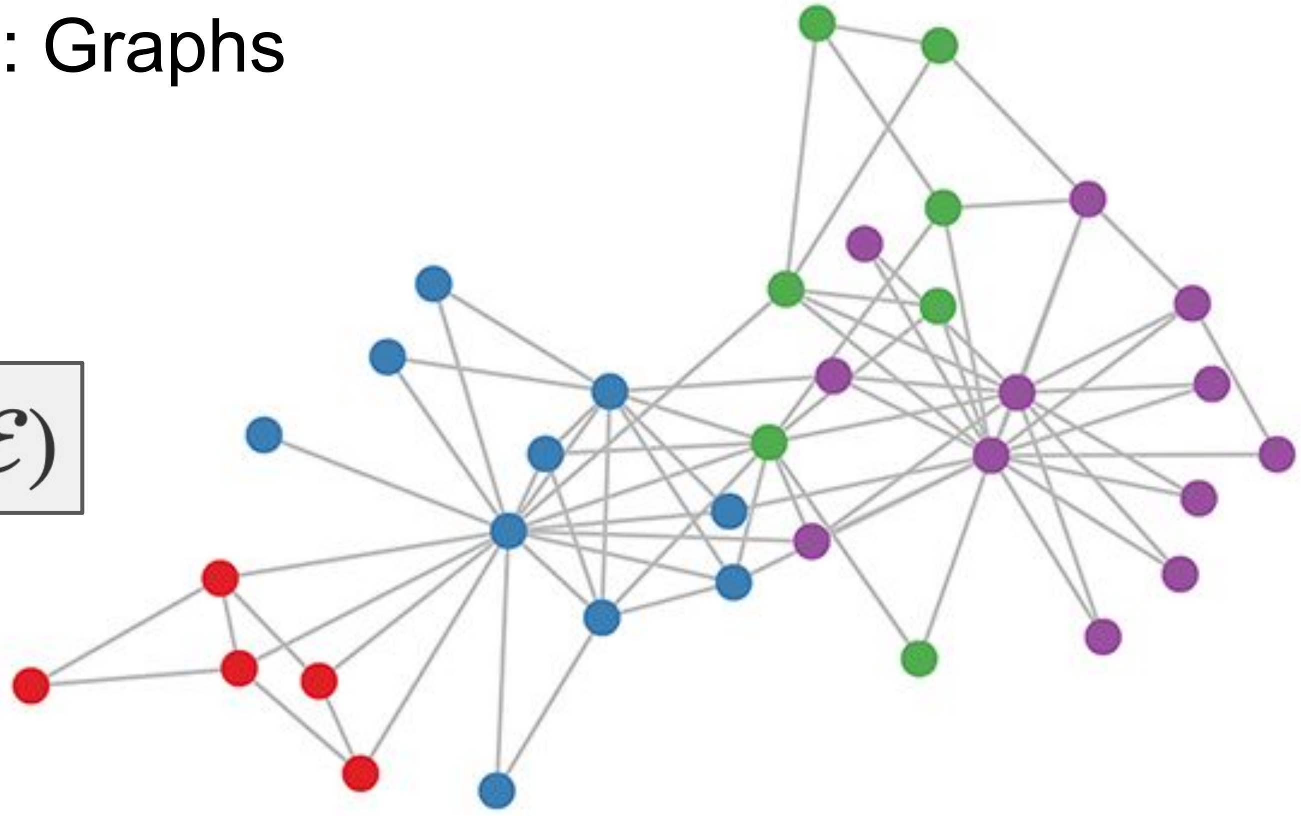
- Rank the sentences with underlying assumption that “summary sentences” are similar to most other sentences

https://cmci.colorado.edu/~mpaul/files/emnlp2010.slides.pdf

Introduction: Graphs

Vertices V
Edges E

$$\mathcal{G} = (\mathcal{V}, \mathcal{E})$$



https://images.app.goo.gl/mnaK9vc1547Sd5UU8 Karate Network

Early Research: DeepWalk (2014)

Goal: Learning latent representations of vertices in a network.

Idea: Skip-Gram Model (Word2vec)

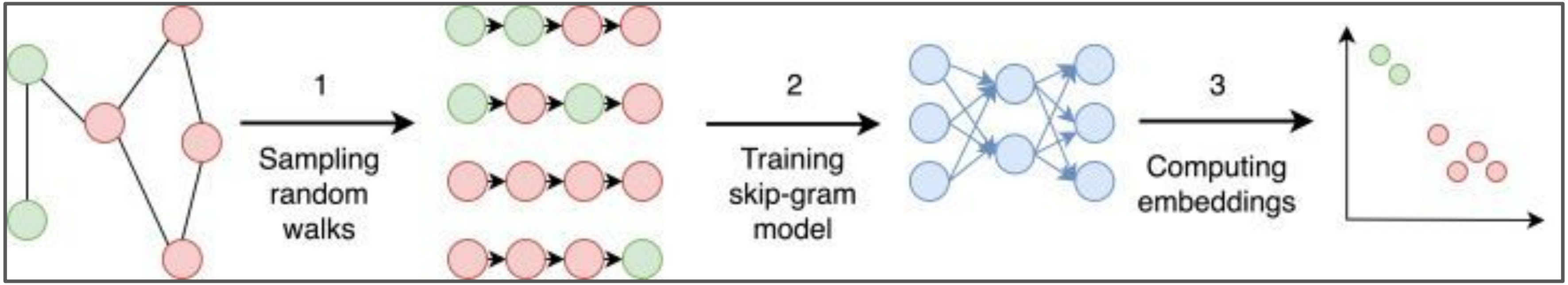
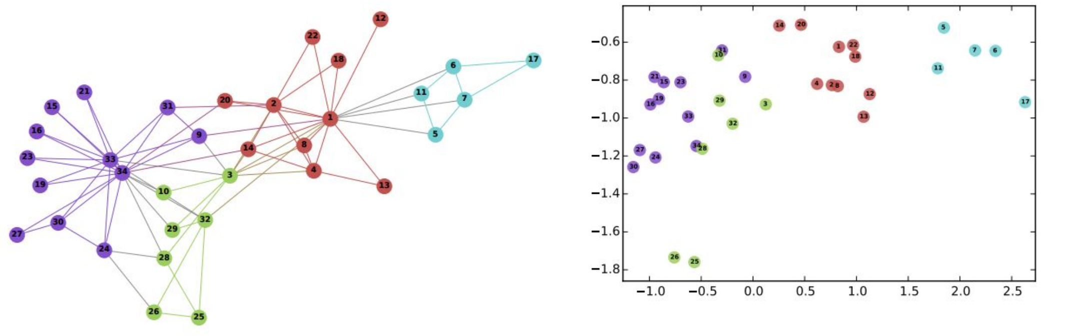
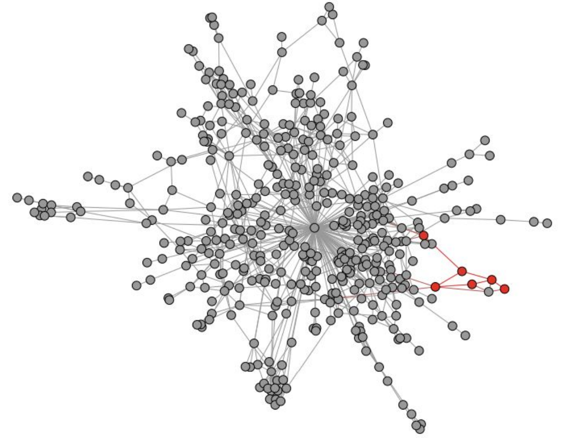


Image source

Node representation



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

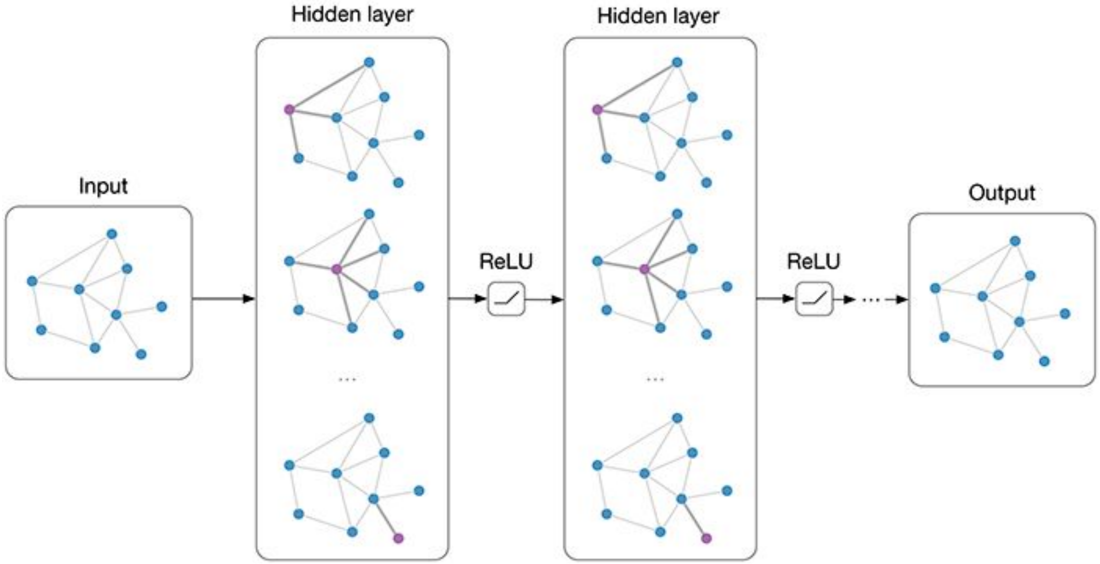


(a) Random walk generation.

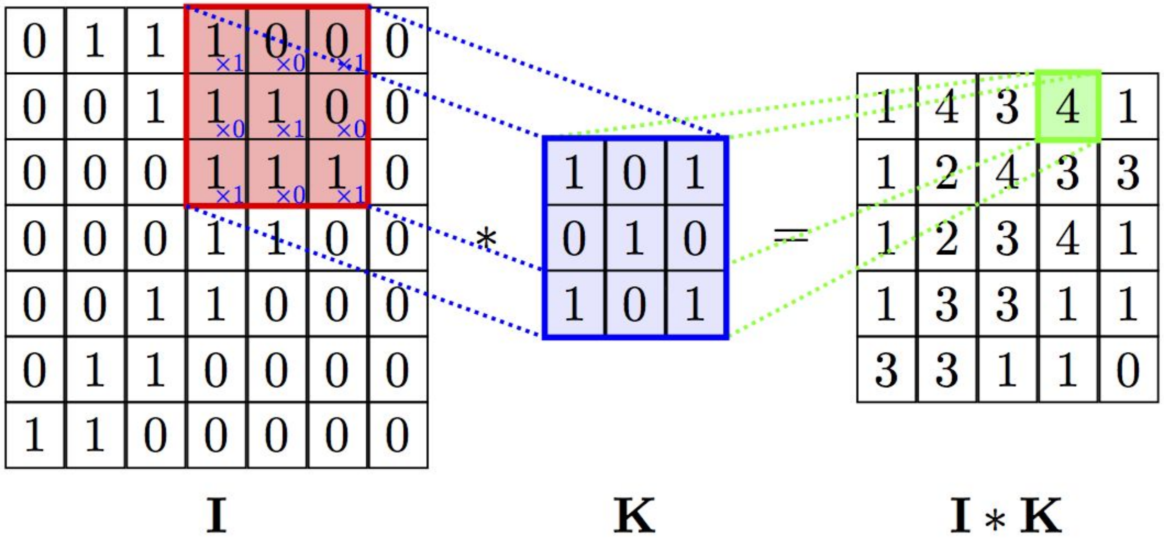
GCN: Graph **Convolutional** Network (ICLR, 2017)

Focus on Graph structure data & convolution.

How to understand traditional CNN?



From CNN to GCN (1)

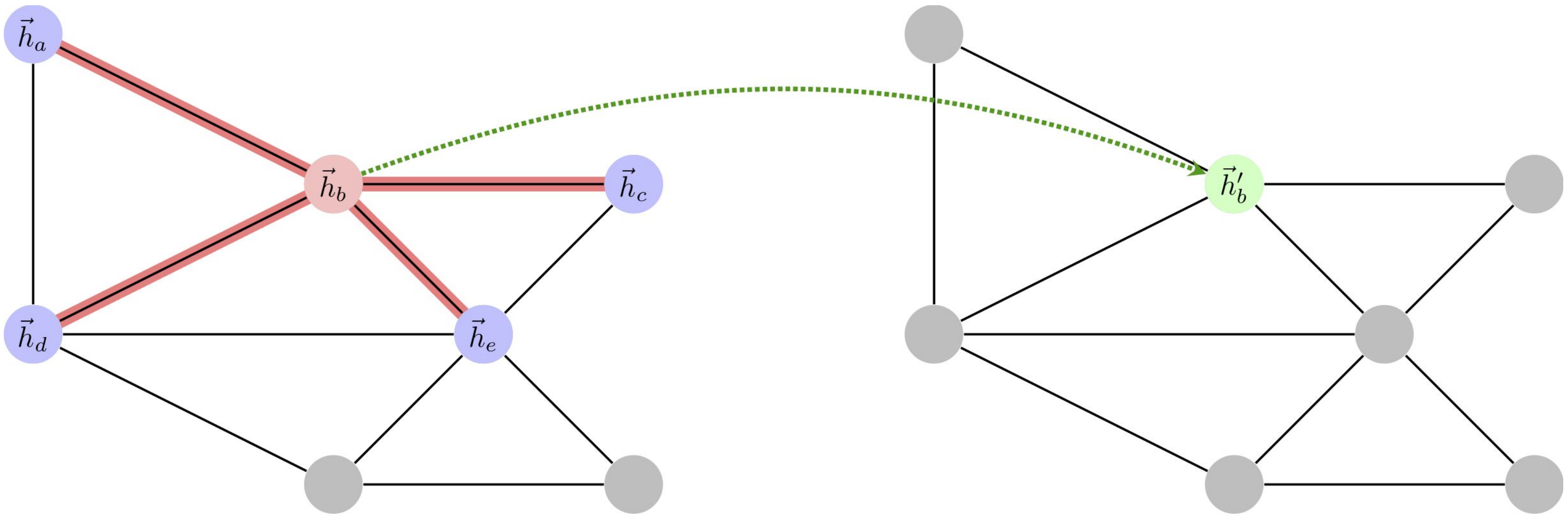


Filter: shared with all locations

http://petar-v.com/GAT/

From CNN to GCN (2)

to go to the next layer using “graph convolution”



http://petar-v.com/GAT/

GCN: formulation

$$\mathcal{G} = (\mathcal{V}, \mathcal{E})$$

Input:

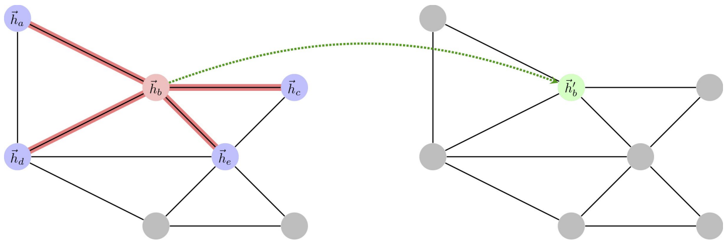
A is adjacency matrix

X is the feature matrix, shape N×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, H0:

$$H^{(0)} = X$$

GCN: formulation

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

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

$$D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$$

GCN: formulation

Definition 1 *The* **normalized adjacency matrix** *is*

$$\mathscr{A} \equiv D^{-1/2} A D^{-1/2},$$

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

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

$$D^{-1/2} = \begin{pmatrix} \frac{1}{\sqrt{d(1)}} & 0 & \cdots & 0 \\ 0 & \frac{1}{\sqrt{d(2)}} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{1}{\sqrt{d(n)}} \end{pmatrix}.$$

$$\hat{A} = A + I$$

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

GCN: formulation

$$H^{(l+1)} = f(H^{(l)}, A)$$

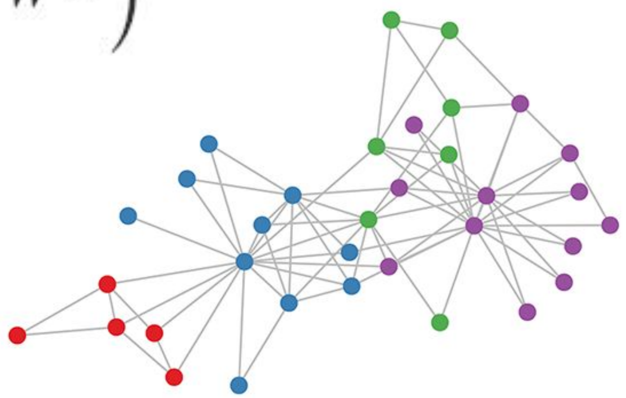
$$H^{(l+1)} = \sigma(AH^{(l)}W^{(l)})$$

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

Node classification: an MLP Layer at the end.

PyTorch: https://github.com/tkipf/pygcn

http://opennmt.net/OpenNMT-py/main.html



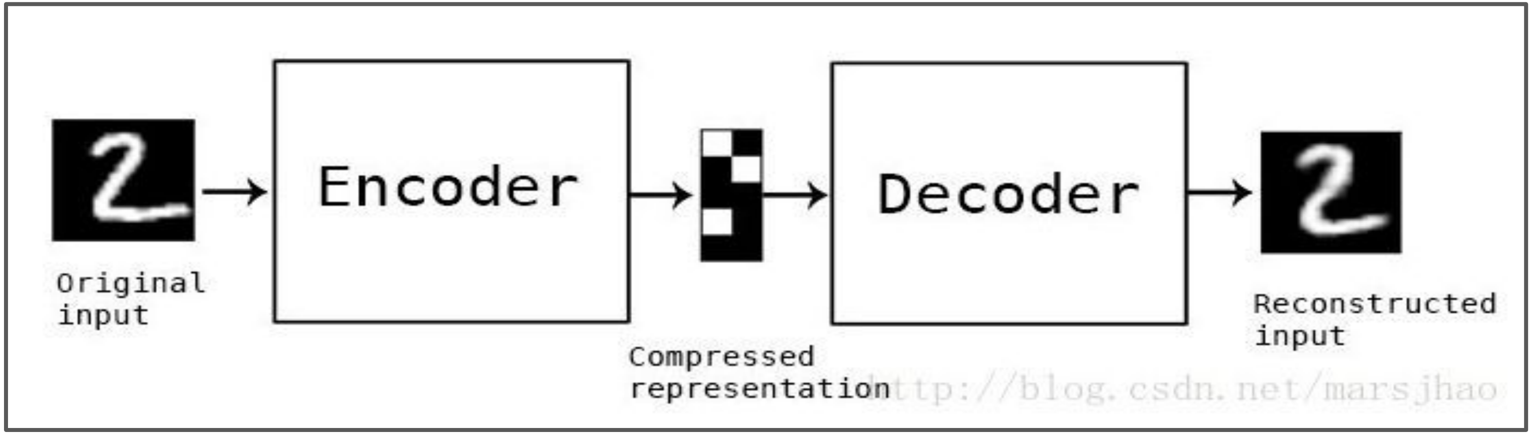
Results: [supervised learning] classify the nodes in citation network

Table 2: Summary of results in terms of classification accuracy (in percent).

Method	Citeseer	Cora	Pubmed	NELL
ManiReg [3]	60.1	59.5	70.7	21.8
SemiEmb [28]	59.6	59.0	71.1	26.7
LP [32]	45.3	68.0	63.0	26.5
DeepWalk [22]	43.2	67.2	65.3	58.1
ICA [18]	69.1	75.1	73.9	23.1
Planetoid* [29]	64.7 (26s)	75.7 (13s)	77.2 (25s)	61.9 (185s)
GCN (this paper)	**70.3** (7s)	**81.5** (4s)	**79.0** (38s)	**66.0** (48s)

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**.

$$\mathrm{GCN}(\mathbf{X}, \mathbf{A}) = \tilde{\mathbf{A}}\, \mathrm{ReLU}\big(\tilde{\mathbf{A}}\mathbf{X}\mathbf{W}_0\big)\mathbf{W}_1$$

$$\boldsymbol{\mu} = \mathrm{GCN}_{\boldsymbol{\mu}}(\mathbf{X}, \mathbf{A}) \qquad \log \boldsymbol{\sigma} = \mathrm{GCN}_{\boldsymbol{\sigma}}(\mathbf{X}, \mathbf{A})$$

$$q(\mathbf{Z} \,|\, \mathbf{X}, \mathbf{A}) = \textstyle\prod_{i=1}^{N} q(\mathbf{z}_i \,|\, \mathbf{X}, \mathbf{A})\,, \quad \text{with} \quad q(\mathbf{z}_i \,|\, \mathbf{X}, \mathbf{A}) = \mathcal{N}(\mathbf{z}_i \,|\, \boldsymbol{\mu}_i, \mathrm{diag}(\boldsymbol{\sigma}_i^2))$$

An inner product as the **Decoder**

$$p\,(\mathbf{A} \,|\, \mathbf{Z}) = \textstyle\prod_{i=1}^{N} \prod_{j=1}^{N} p\,(A_{ij} \,|\, \mathbf{z}_i, \mathbf{z}_j)\,, \quad \text{with} \quad p\,(A_{ij} = 1 \,|\, \mathbf{z}_i, \mathbf{z}_j) = \sigma(\mathbf{z}_i^\top \mathbf{z}_j)$$

$$\mathcal{L} = \mathbb{E}_{q(\mathbf{Z}|\mathbf{X},\mathbf{A})}\big[\log p\,(\mathbf{A} \,|\, \mathbf{Z})\big] - \mathrm{KL}\big[q(\mathbf{Z} \,|\, \mathbf{X}, \mathbf{A}) \,||\, p(\mathbf{Z})\big]$$

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:

$$\hat{\mathbf{A}} = \sigma\left(\mathbf{Z}\mathbf{Z}^\top\right)$$
$$\mathbf{Z} = GCN(\mathbf{X}, \mathbf{A})$$

Results: [Semi-supervised] Classify edges/non-edges

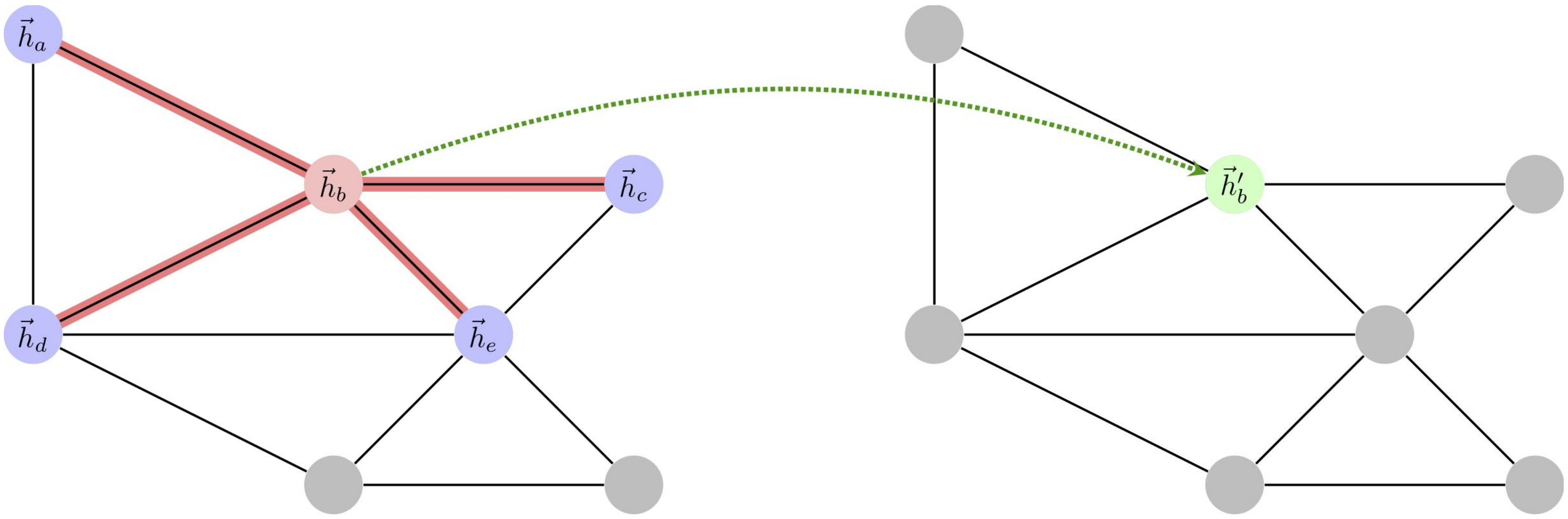
The models are trained on an incomplete version of these datasets where parts of the citation links (edges) have been removed, while all node features are kept. Complete X; incomplete A → A

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

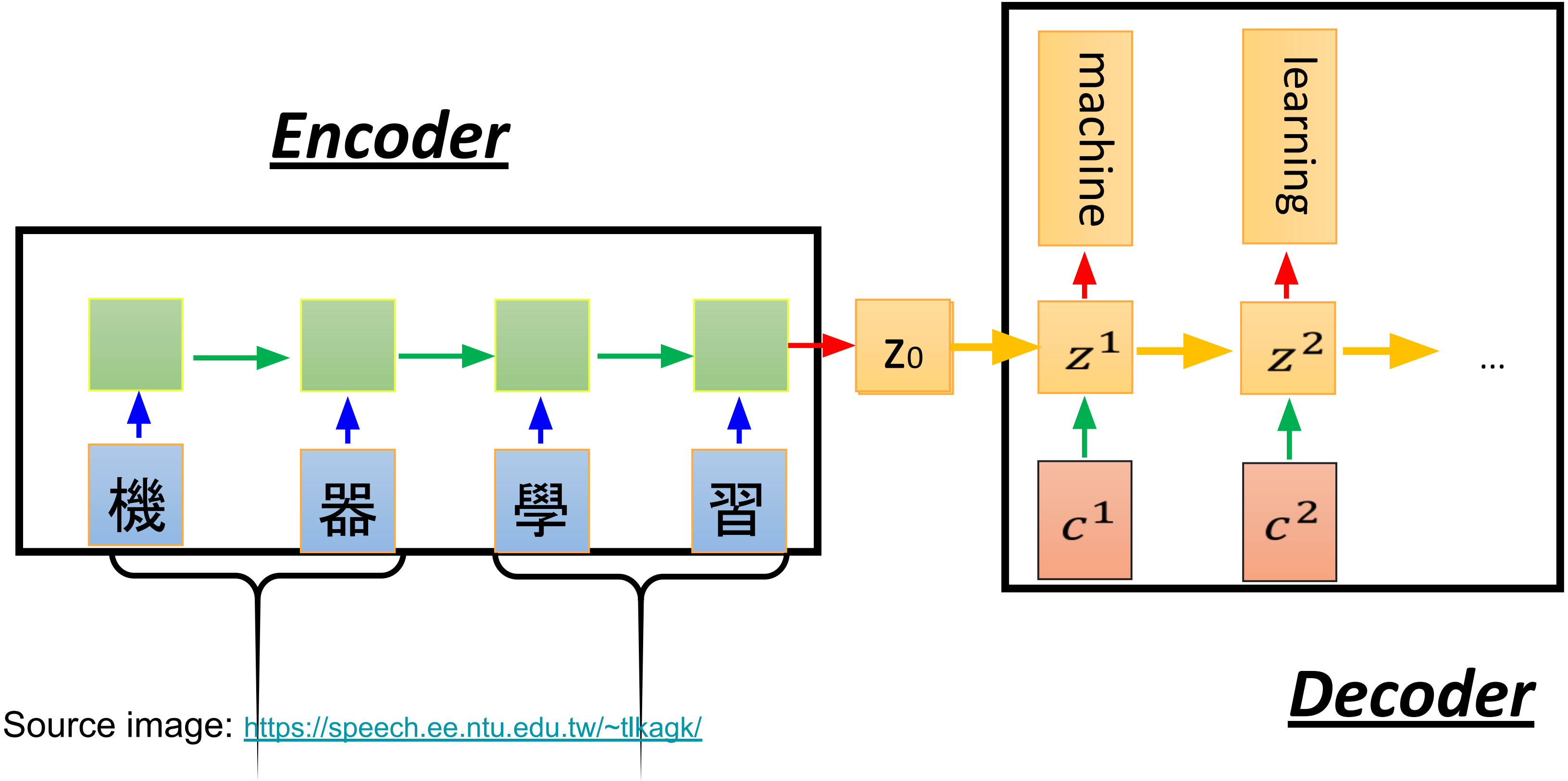
Method	Cora AUC	Cora AP	Citeseer AUC	Citeseer AP	Pubmed AUC	Pubmed AP
SC [5]	84.6 ± 0.01	88.5 ± 0.00	80.5 ± 0.01	85.0 ± 0.01	84.2 ± 0.02	87.8 ± 0.01
DW [6]	83.1 ± 0.01	85.0 ± 0.00	80.5 ± 0.02	83.6 ± 0.01	84.4 ± 0.00	84.1 ± 0.00
GAE*	84.3 ± 0.02	88.1 ± 0.01	78.7 ± 0.02	84.1 ± 0.02	82.2 ± 0.01	87.4 ± 0.00
VGAE*	84.0 ± 0.02	87.7 ± 0.01	78.9 ± 0.03	84.1 ± 0.02	82.7 ± 0.01	87.5 ± 0.01
GAE	91.0 ± 0.02	92.0 ± 0.03	89.5 ± 0.04	89.9 ± 0.05	$\mathbf{96.4} \pm 0.00$	$\mathbf{96.5} \pm 0.00$
VGAE	$\mathbf{91.4} \pm 0.01$	$\mathbf{92.6} \pm 0.01$	$\mathbf{90.8} \pm 0.02$	$\mathbf{92.0} \pm 0.02$	94.4 ± 0.02	94.7 ± 0.02

Question: Better ways to “move” to the next layer?

Attention!

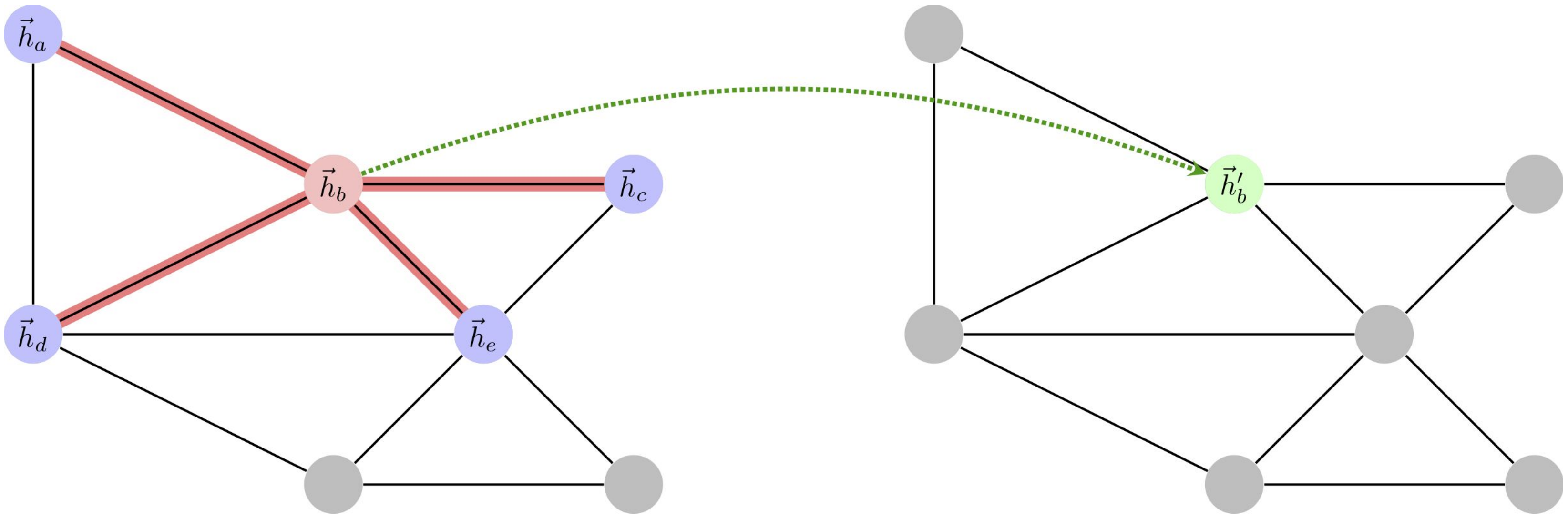


Recall Attention in seq2seq



Source image: https://speech.ee.ntu.edu.tw/~tlkagk/

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):

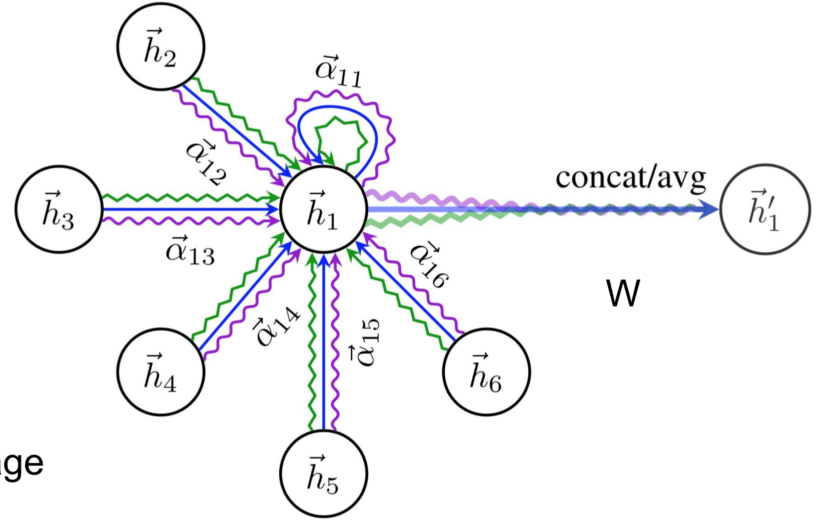
$$e_{ij} = a(\vec{h}_i, \vec{h}_j)$$

Normalize over all the neighbors:

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

Use neighbors to represent the current node: weighted average

$$\vec{h}'_i = \Big\Vert_{k=1}^{K} \sigma\left(\sum_{j \in \mathcal{N}_i} \alpha_{ij}^k \mathbf{W}^k \vec{h}_j\right)$$



Multi-head

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).

Transductive

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

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

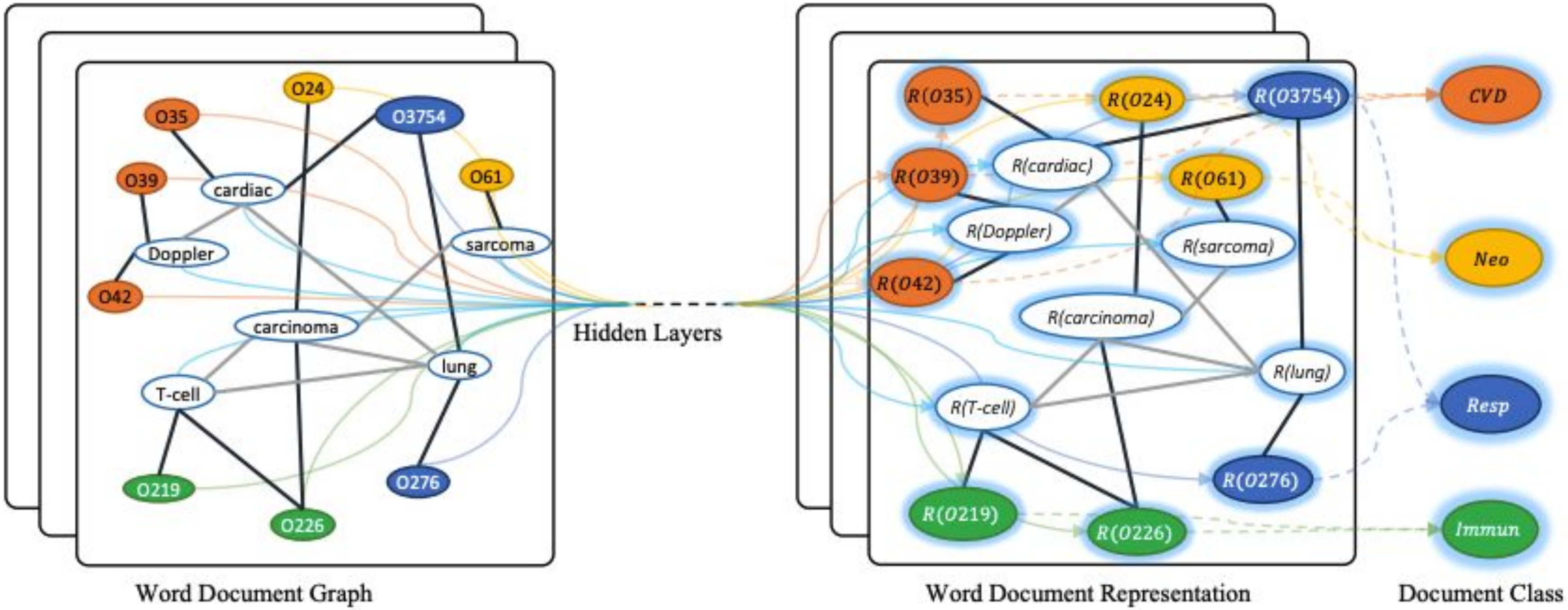
GCNs: Graph neural networks have been effective at tasks thought to have rich relational structure and can preserve global structure information of a graph in graph embeddings.

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)

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

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

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

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

Node representation X

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

Propagation rule

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

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

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

Experiments: public datasets

Table 1: Summary statistics of datasets.

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

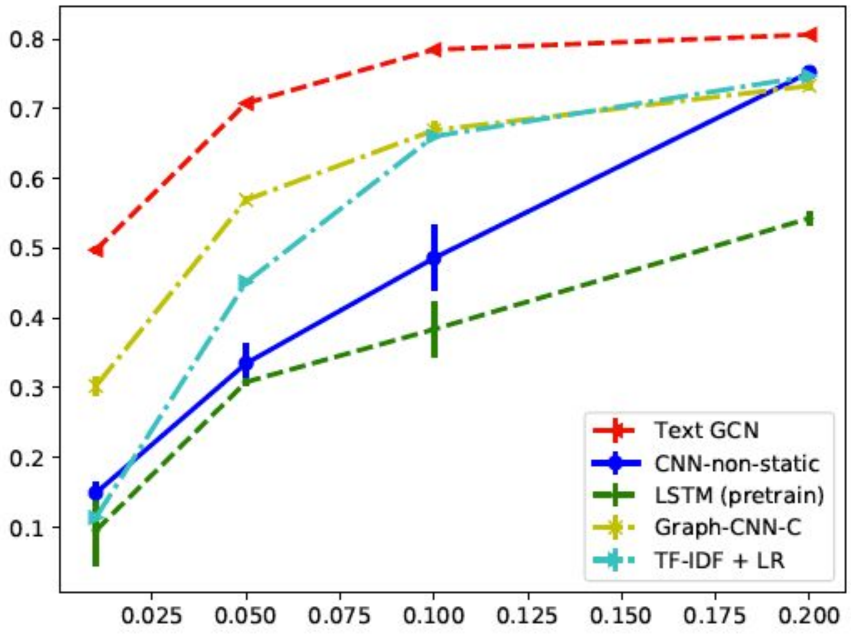
Experimental results

Model	20NG	R8	R52	Ohsumed	MR
TF-IDF + LR	0.8319 ± 0.0000	0.9374 ± 0.0000	0.8695 ± 0.0000	0.5466 ± 0.0000	0.7459 ± 0.0000
CNN-rand	0.7693 ± 0.0061	0.9402 ± 0.0057	0.8537 ± 0.0047	0.4387 ± 0.0100	0.7498 ± 0.0070
CNN-non-static	0.8215 ± 0.0052	0.9571 ± 0.0052	0.8759 ± 0.0048	0.5844 ± 0.0106	$\mathbf{0.7775 \pm 0.0072}$
LSTM	0.6571 ± 0.0152	0.9368 ± 0.0082	0.8554 ± 0.0113	0.4113 ± 0.0117	0.7506 ± 0.0044
LSTM (pretrain)	0.7543 ± 0.0172	0.9609 ± 0.0019	0.9048 ± 0.0086	0.5110 ± 0.0150	0.7733 ± 0.0089
Bi-LSTM	0.7318 ± 0.0185	0.9631 ± 0.0033	0.9054 ± 0.0091	0.4927 ± 0.0107	0.7768 ± 0.0086
PV-DBOW	0.7436 ± 0.0018	0.8587 ± 0.0010	0.7829 ± 0.0011	0.4665 ± 0.0019	0.6109 ± 0.0010
PV-DM	0.5114 ± 0.0022	0.5207 ± 0.0004	0.4492 ± 0.0005	0.2950 ± 0.0007	0.5947 ± 0.0038
PTE	0.7674 ± 0.0029	0.9669 ± 0.0013	0.9071 ± 0.0014	0.5358 ± 0.0029	0.7023 ± 0.0036
fastText	0.7938 ± 0.0030	0.9613 ± 0.0021	0.9281 ± 0.0009	0.5770 ± 0.0049	0.7514 ± 0.0020
fastText (bigrams)	0.7967 ± 0.0029	0.9474 ± 0.0011	0.9099 ± 0.0005	0.5569 ± 0.0039	0.7624 ± 0.0012
SWEM	0.8516 ± 0.0029	0.9532 ± 0.0026	0.9294 ± 0.0024	0.6312 ± 0.0055	0.7665 ± 0.0063
LEAM	0.8191 ± 0.0024	0.9331 ± 0.0024	0.9184 ± 0.0023	0.5858 ± 0.0079	0.7695 ± 0.0045
Graph-CNN-C	0.8142 ± 0.0032	0.9699 ± 0.0012	0.9275 ± 0.0022	0.6386 ± 0.0053	0.7722 ± 0.0027
Graph-CNN-S	–	0.9680 ± 0.0020	0.9274 ± 0.0024	0.6282 ± 0.0037	0.7699 ± 0.0014
Graph-CNN-F	–	0.9689 ± 0.0006	0.9320 ± 0.0004	0.6304 ± 0.0077	0.7674 ± 0.0021
Text GCN	$\mathbf{0.8634 \pm 0.0009}$	$\mathbf{0.9707 \pm 0.0010}$	$\mathbf{0.9356 \pm 0.0018}$	$\mathbf{0.6836 \pm 0.0056}$	0.7674 ± 0.0020

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)



(a) 20NG

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,xpqiu,xjhuang}@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

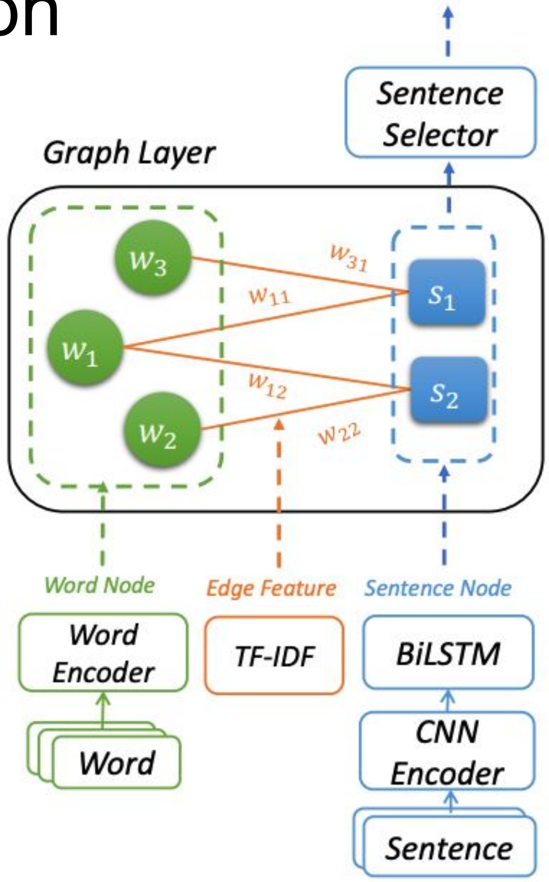
$$D = \{s_1, \cdots, s_n\}$$

$$y_1, \cdots, y_n (y_i \in \{0, 1\})$$

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)

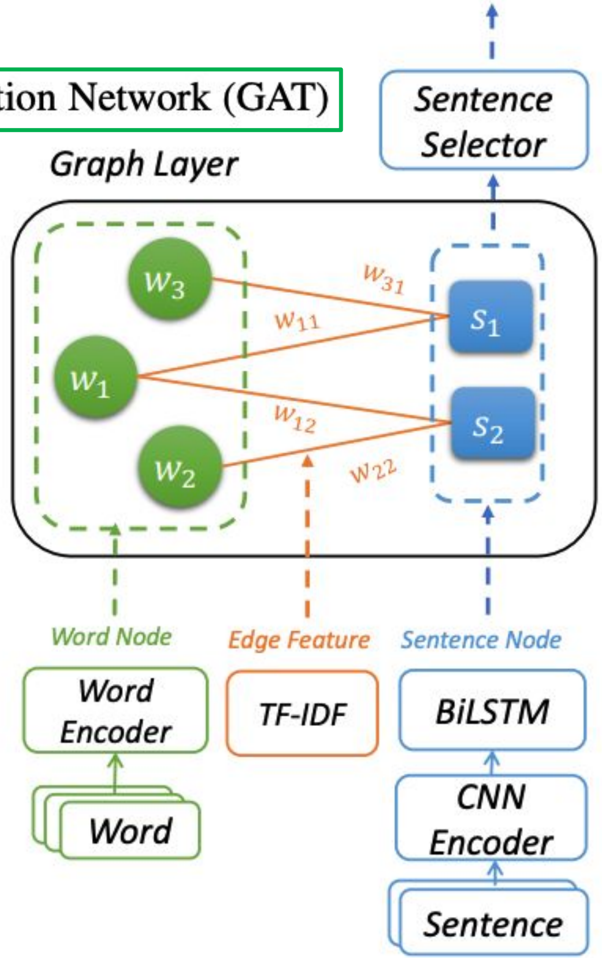
$$z_{ij} = \text{LeakyReLU}\left(\mathbf{W}_a[\mathbf{W}_q \boldsymbol{h}_i; \mathbf{W}_k \boldsymbol{h}_j]\right),$$

$$\alpha_{ij} = \frac{\exp(z_{ij})}{\sum_{l \in \mathcal{N}_i} \exp(z_{il})},$$

$$\boldsymbol{u}_i = \sigma\left(\sum_{j \in \mathcal{N}_i} \alpha_{ij} \mathbf{W}_v \boldsymbol{h}_j\right),$$

$$\boldsymbol{u}_i = \|_{k=1}^{K} \sigma\left(\sum_{j \in \mathcal{N}_i} \alpha_{ij}^k \mathbf{W}^k \boldsymbol{h}_i\right)$$

Graph Attention Network (GAT)

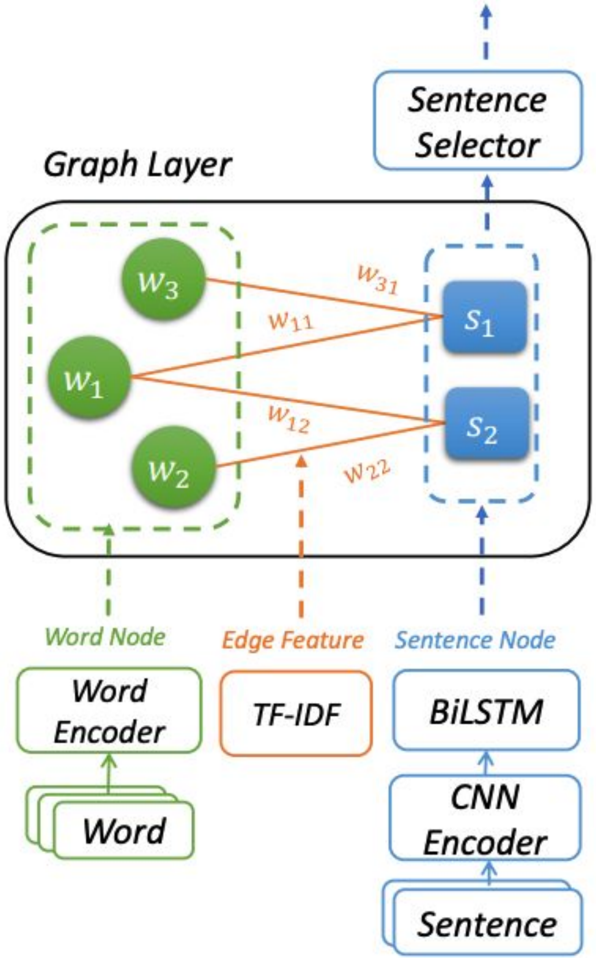


Sentence selector

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.

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

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.

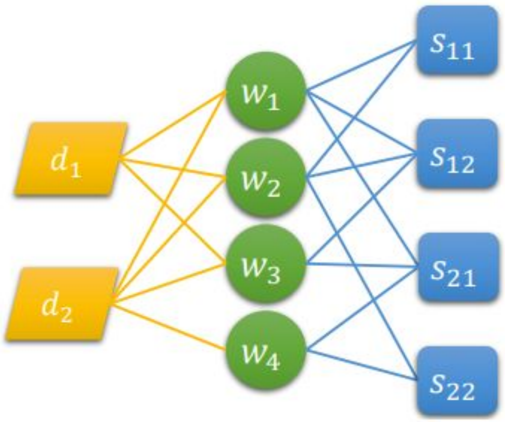


Figure 3: Graph structure of HETERDOCSUMGRAPH for multi-document summarization (corresponding to the *Graph Layer* part of Figure 1). Green, blue and orange boxes represent word, sentence and document nodes respectively. d_1 consists of s_{11} and s_{12} while d_2 contains s_{21} and s_{22}. As a relay 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

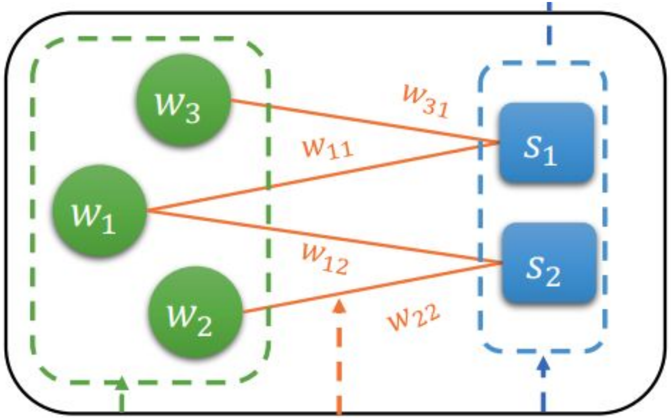
Multi-News (Fabbri, ACL 2019)
2-10 sources; each limits to 500 tokens
44,972/5,622/5,622 for training, validation and test

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

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?



Paper 3: A Graph-based Coarse-to-fine Method for Unsupervised Bilingual Lexicon Induction (ACL, 2020)

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:

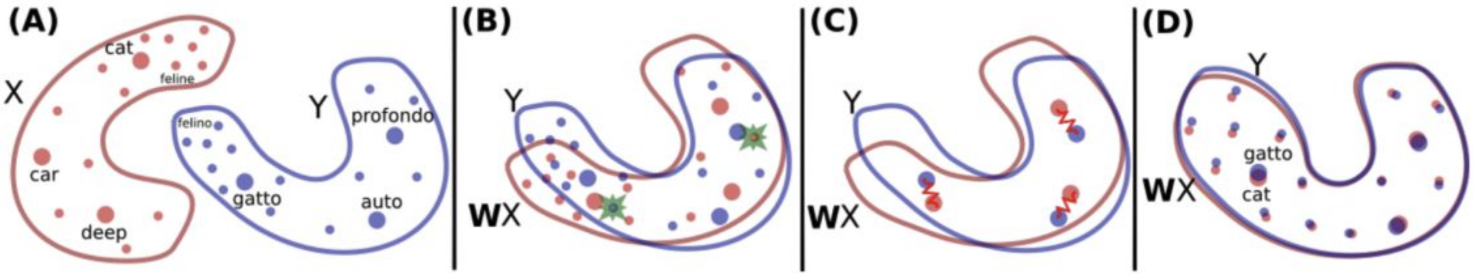
Becomes an essential part of recent unsupervised machine translation approaches (Lample et al., 2018; Artetxe et al., 2018c; Marie and Fujita, 2018; Ren et al., 2019; Artetxe et al., 2019)

Source: https://slideslive.com/38928855/a-graphbased-coarsetofine-method-for-unsupervised-bilingual-lexicon-induction

Related Work for BLI

Unsupervised cross-lingual word embeddings.

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

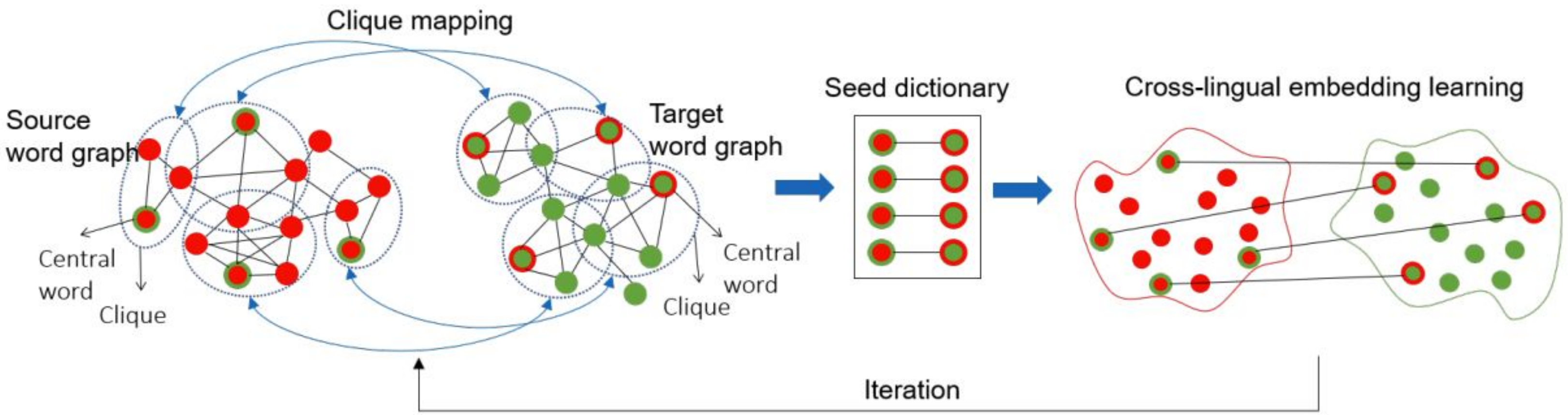


$$W^{\star} = \underset{W \in M_d(\mathbb{R})}{\operatorname{argmin}} \|WX - Y\|_{\mathrm{F}}$$

[WORD TRANSLATION WITHOUT PARALLEL DATA (ICLR, 2018)]

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

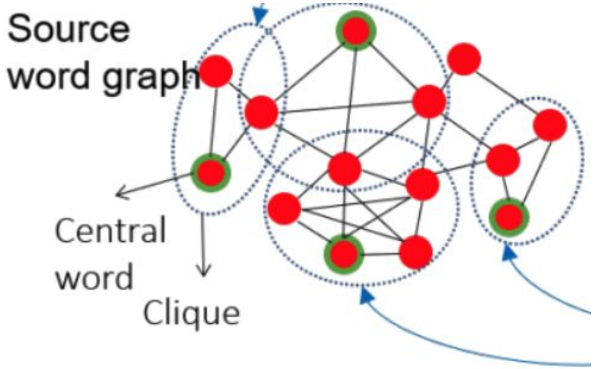
Construct a “word graph” for each language

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

Edges:

Cos similarities based on embeddings.

No self-loops.



CSLS similarity: [Word Translation Without Parallel Data](), ICLR 2018

Clique Extraction

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

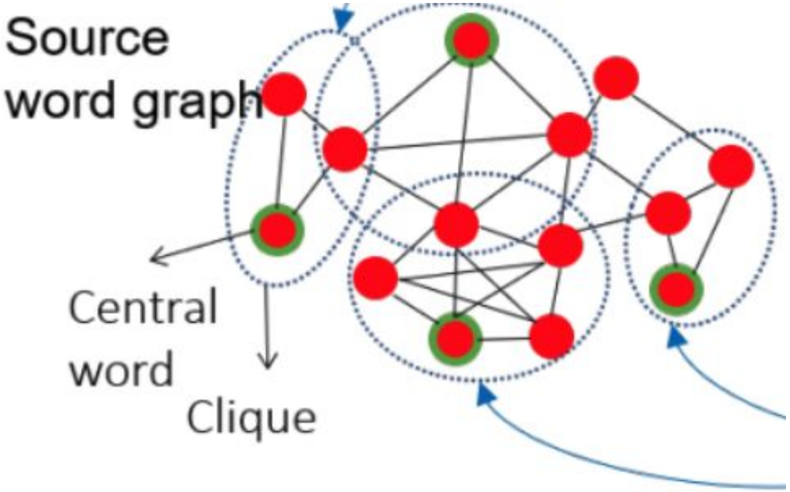
Bron-Kerbosch (BK) algorithm is able to recognize cliques in graphs: **more efficient than clustering.**

Find **central word** of each clique:

Average all embeddings;

Find the closest word.

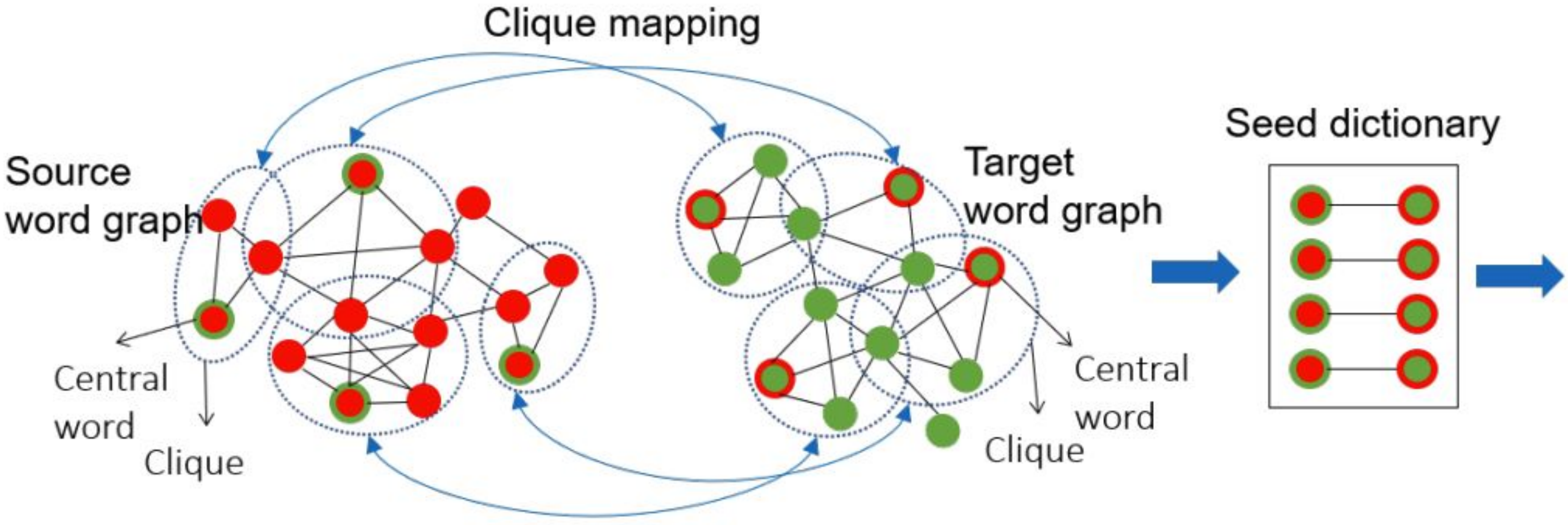
Match Cliques on the two languages.



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

$$\mathbf{W}* = \underset{\mathbf{W}}{\operatorname{argmin}} \|\mathbf{W}\mathbf{X} - \mathbf{Y}\|_F, \mathbf{W}^\top \mathbf{W} = \mathbf{I}$$

Group mapping: learn an individual mapping Mi for each clique i:

$$\mathbf{W}_\mathrm{i}* = \underset{\mathbf{W}_\mathrm{i}}{\operatorname{argmin}} \|\mathbf{W_i}\mathbf{X_i} - \mathbf{Y_i}\|_{\boldsymbol{F}}, \mathbf{W_i}^\top \mathbf{W_i} = \mathbf{I}$$

Given a new word: find its embedding x, find clique, calculate y using Wi.

Experiments

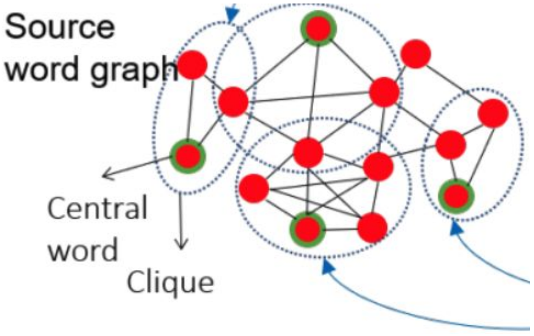
	Method	en-fr		en-de		en-es		en-it		en-ru		en-zh	
		→	←	→	←	→	←	→	←	→	←	→	←
	Supervised												
	(Smith et al., 2017)	81.1	82.4	73.5	72.4	81.4	82.9	43.1	38.0	51.7	63.7	42.7	36.7
	(Artetxe et al., 2018a)	80.5	83.1	73.5	73.5	80.5	83.8	**61.3**	**39.6**	50.5	67.3	32.3	43.4
	(Joulin et al., 2018)	**83.3**	84.1	**79.1**	76.3	**84.1**	**86.3**	-	-	**57.9**	67.2	45.9	**46.4**
	(Jawanpuria et al., 2019)	82.1	**84.2**	74.9	**76.7**	81.9	85.5	-	-	52.8	**67.6**	**49.1**	45.3
	Unsupervised												
MUSE	(Conneau et al., 2017)	82.3	81.1	74.0	72.2	81.7	83.3	77.4	76.1	44.0	59.1	32.5	31.4
	(Xu et al., 2018)	77.9	75.5	69.3	67.0	79.5	77.8	72.6	73.4	-	-	-	-
	(Alvarez-Melis and Jaakkola, 2018)	81.3	78.9	71.9	72.8	81.7	80.4	78.9	75.2	45.1	43.7	-	-
VecMap	(Artetxe et al., 2018b)	82.3	83.6	75.1	74.3	82.3	84.7	78.8	79.5	49.2	**65.6**	-	-
	(Hoshen and Wolf, 2018)	82.3	**84.1**	74.7	73.0	82.1	84.1	77.9	77.5	47.5	61.8	-	-
	Ours (without GM)	82.7	83.4	**75.5**	75.7	82.6	84.8	78.6	79.5	48.9	63.9	38.1	35.2
	Ours (with GM)	**82.9**	83.9	75.3	**76.1**	**82.9**	**85.3**	**79.1**	**79.9**	**49.7**	64.7	**38.9**	**35.9**

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

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

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

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

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!

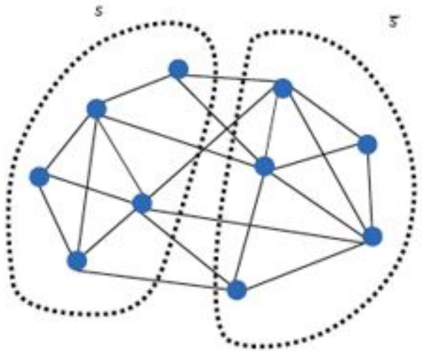
Heterogeneous Graph Attention Network

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

How could this help in NLP applications?

Discussion (3)

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

How do you choose them?

Open Question

Related resources

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

https://www.inference.vc/how-powerful-are-graph-convolutions-review-of-kipf-welling-2016-2/

CS224W: Machine Learning with Graphs (Fall 2019)

Thanks

Q&A