Graph Convolutional Networks for Text Classification

Liang Yao

Liang Yao, Chengsheng Mao and Yuan Luo
Northwestern University
{*liang.yao, chengsheng.mao, yuan.luo*}*@northwestern.edu*

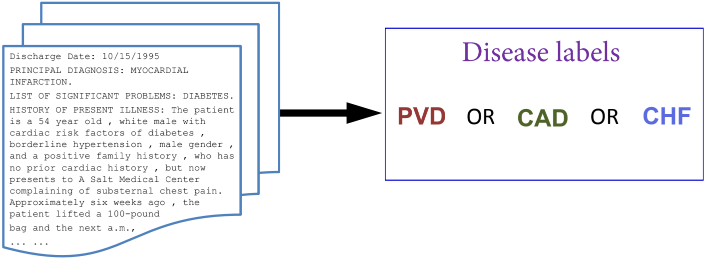
November 16, 2018

Overview

Problem

Text Classification

- A fundamental problem in Natural Language Processing (NLP)
- Many applications:
  - News filtering
  - Spam detection
  - Opinion mining
  - Computational phenotyping
- Essential Intermediate Step: Text Representation



Traditional Methods

Feature Engineering

- bag-of-words
- n-grams
- entities in ontology
- text classification as graph classification
  - frequent subgraphs mining [ACL'15]
  - graph-of-words as regularization [EMNLP'16]

- Could not learn text representations automatically

Deep learning for Text Classification

Word embedding based methods

- aggregated word embeddings as document embeddings
  - PV-DBOW, PV-DM [ICML'14]
  - fastText [EACL'17]
  - SWEM [ACL'18]
- jointly learned word/document and label embeddings
  - PTE [KDD'15]
  - LEAM [ACL'18]

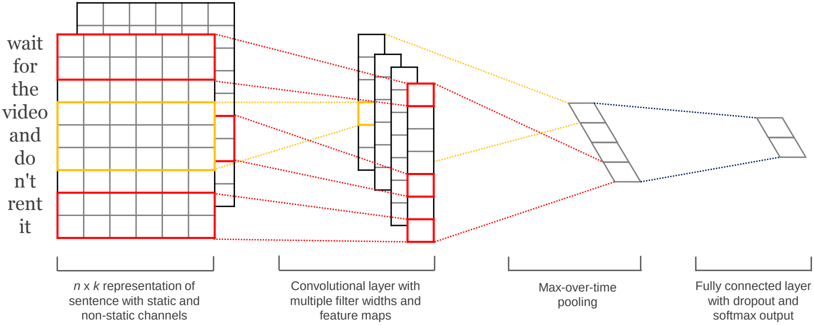
- Building document representations after learning word embeddings
- Can we learn word & document embeddings simultaneously?

Deep learning for Text Classification

Deep neural network

- Convolutional Neural Networks (CNN)
  - Word CNN [EMNLP'14]
  - Char CNN [NIPS'15]
  - Very Deep CNN [EACL'17]
- Recurrent Neural Networks (RNN)
  - LSTM
  - Bi-LSTM
  - GRU
- Attention mechanisms
  - HAN [NAACL'16]
  - Attention-based LSTM [EMNLP'16]

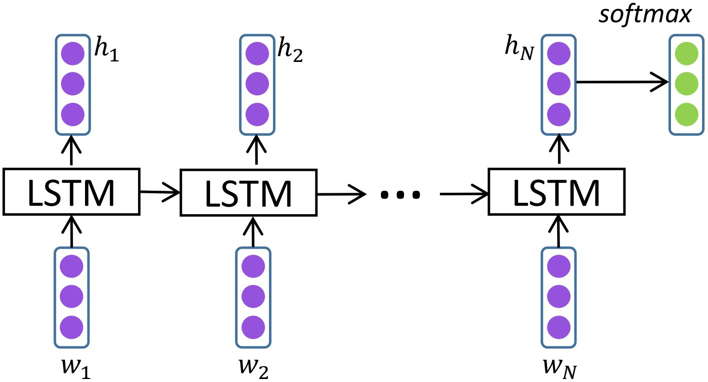
Word CNN



Reference

Kim, Y. 2014. Convolutional neural networks for sentence classification. In *EMNLP*, 1746–1751.

LSTM



Inductive Bias

Component	Entities	Relations	Rel. inductive bias	Invariance
Fully connected	Units	All-to-all	Weak	-
Convolutional	Grid elements	Local	Locality	Spatial translation
Recurrent	Timesteps	Sequential	Sequentiality	Time translation
Graph network	Nodes	Edges	Arbitrary	Node, edge permutations

- CNN and RNN prioritize *locality* and *sequentiality*.
- They can model local consecutive word sequences well.
- They may ignore **global word co-occurrence** in a corpus.

Reference

Battaglia, P. W. et al. 2018. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261.*

Graph Neural Networks

- Generalizing well-established neural network models like CNN that apply to regular grid structure (2-d mesh or 1-d sequence) to work on arbitrarily structured graphs.
- Can preserve global structure information of a graph in graph embeddings (node, edge, subgraph and whole graph embeddings).

Reference

Cai, H.; Zheng, V. W.; and Chang, K. 2018. A comprehensive survey of graph embedding: problems, techniques and applications. *IEEE TKDE* 30(9):1616–1637.

Graph Convolutional Networks (GCN)

- A graph $G = (V, E)$:
  - $(v, v) \in E$ for any v
  - $X \in \mathbb{R}^{n \times m}$: node features matrix
  - A: adjacency matrix, degree matrix $D_{ii} = \sum_j A_{ij}$
  - $\tilde{A} = D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$: normalized symmetric adjacency matrix:
  - W_j: weight matrix, trained via SGD
- One layer GCN:

$$L^{(1)} = \rho(\tilde{A} X W_0) \tag{1}$$

- Stacking multiple GCN layers:

$$L^{(j+1)} = \rho(\tilde{A} L^{(j)} W_j) \tag{2}$$

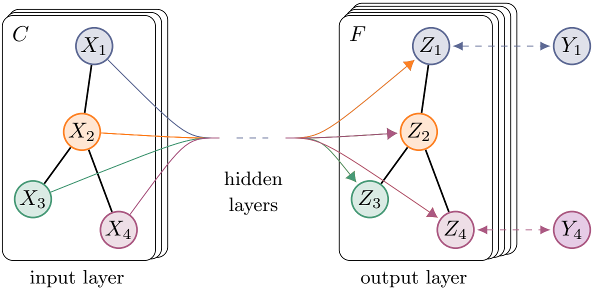
Graph Convolutional Networks (GCN)

- GCN can capture information only about immediate neighbors with one layer.
- When multiple GCN layers are stacked, one can incorporate higher order neighborhoods information.
  - e.g., a **two**-layers GCN can allow message passing among nodes that are at maximum **two** steps away.
- A special form of Laplacian smoothing:
  - Computes the new features of a node as the weighted average of itself and its second order neighbors

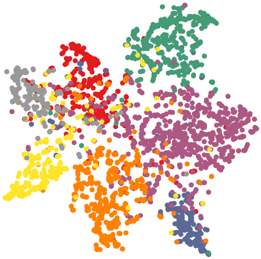
Reference

Li, Q.; Han, Z.; and Wu, X. 2018. Deeper insights into graph convolutional networks for semisupervised learning. In *AAAI*.

Graph Convolutional Networks (GCN)



(a) Graph Convolutional Network

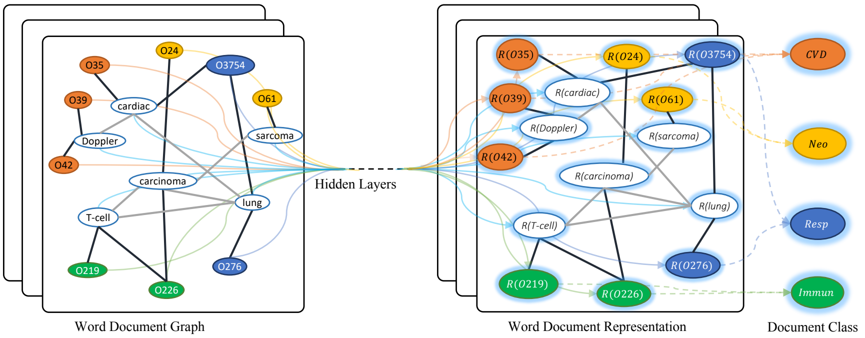


(b) Hidden layer activations

Reference

Kipf, T. N., and Welling, M. 2017. Semi-supervised classification with graph convolutional networks. In *ICLR*.

Text Graph Convolutional Networks (Text GCN)



Text GCN

$$\mathrm{PMI}(i,j) = \log \frac{p(i,j)}{p(i)p(j)} \tag{3}$$

$$A_{ij} = \begin{cases} \mathrm{PMI}(i,j) & i, j \text{ are words, } \mathrm{PMI}(i,j) > 0 \\ \text{TF-IDF}_{ij} & i \text{ is document, } j \text{ is word} \\ 1 & i = j \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

$$Z = \mathrm{softmax}(\tilde{A}\ \mathrm{ReLU}(\tilde{A}XW_0)W_1) \tag{5}$$

$$\mathcal{L} = -\sum_{d \in \mathcal{Y}_D} \sum_{f=1}^{F} Y_{df} \ln Z_{df} \tag{6}$$

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

Implementation Details

- Parameters:
  - first embedding size: 200
  - window size: 20
  - dropout rate: 0.5
  - learning rate: 0.02
  - validation set: 10% of training set
  - number of epochs: 200
- We also tried other parameters but do not find much difference.
- Adam algorithm as the optimizer.

Results

Table 2: Test Accuracy on document classification task. We run all models 10 times and report mean $\pm$ standard deviation. Text GCN significantly outperforms baselines on 20NG, R8, R52 and Ohsumed based on student t-test ($p < 0.05$).

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

Results

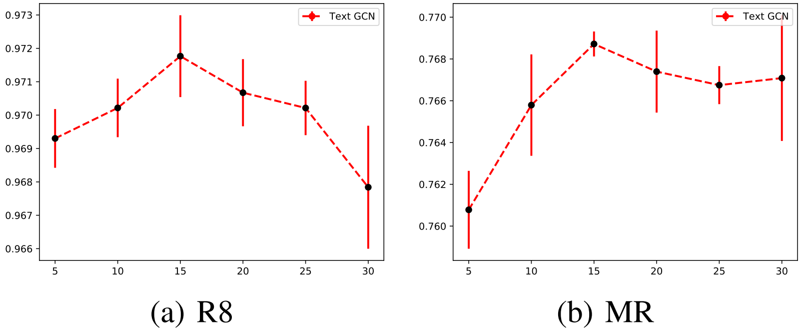
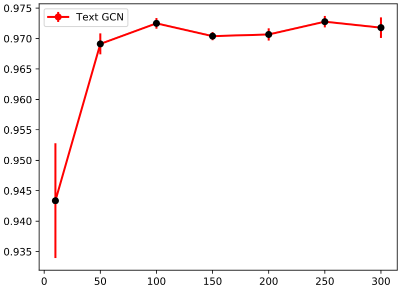
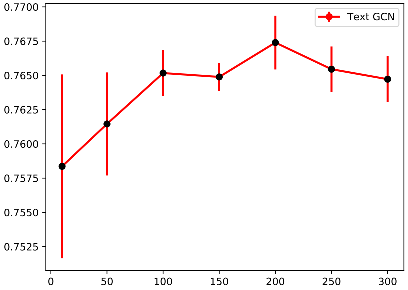


Figure 2: Test accuracy with different sliding window sizes.

Results



(a) R8



(b) MR

Figure 3: Test accuracy by varying embedding dimensions.

Results

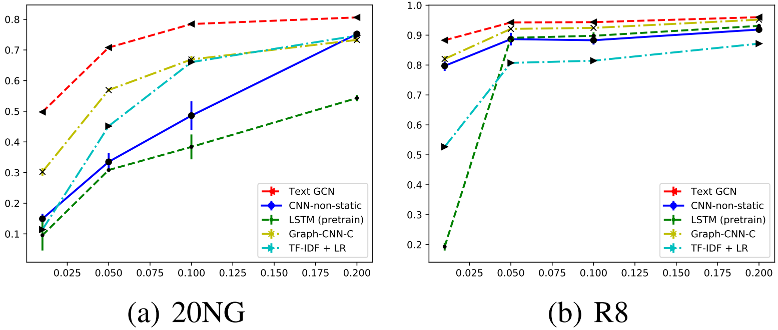
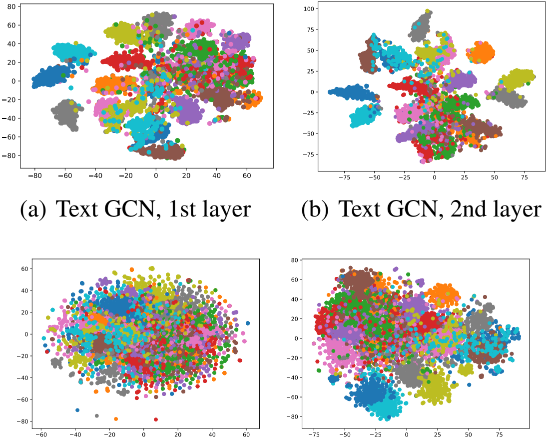


Figure 4: Test accuracy by varying training data proportions.

Results



(a) Text GCN, 1st layer (b) Text GCN, 2nd layer

(c) PV-DBOW (d) PTE

Figure 5: The t-SNE visualization of test set document embeddings in 20NG.

Results

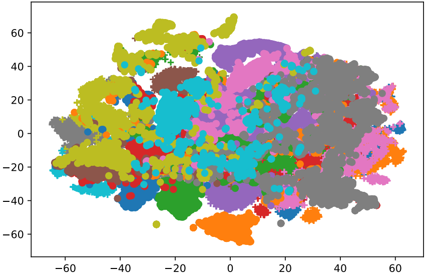


Figure 6: The t-SNE visualization of the second layer word embeddings (20 dimensional) learned from 20NG. We set the dimension with the largest value as a word's label.

Results

Table 3: Words with highest values for several classes in 20NG. Second layer word embeddings are used. We show top 10 words for each class.

comp.graphics	sci.space	sci.med	rec.autos
jpeg	space	candida	car
graphics	orbit	geb	cars
image	shuttle	disease	v12
gif	launch	patients	callison
3d	moon	yeast	engine
images	prb	msg	toyota
rayshade	spacecraft	vitamin	nissan
polygon	solar	syndrome	v8
pov	mission	infection	mustang
viewer	alaska	gordon	eliot

Results

- We released our code.
- `https://github.com/yao8839836/text_gcn`

Future Work

- *Transducitve* to *Inductive*

Reference

Hamilton, W.; Ying, Z.; and Leskovec, J. 2017. Inductive representation learning on large graphs. In *NIPS*, 1024–1034.

- Fast Text GCN

Reference

Chen, J.; Ma, T.; and Xiao, C. 2018. Fastgcn: Fast learning with graph convolutional networks via importance sampling. In *ICLR*

- Using attention mechanisms

Reference

Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; and Bengio, Y. 2018. Graph attention networks. In ICLR.

Future Work

- Unsupervised text representation learning
- Combining knowledge graphs
- ...

Thank You!