Graph Convolutional Networks for Text Classification

Liang Yao, Chengsheng Mao, Yuan Luo

Outline

Power of Graph

ML with Graph

Evolution of Graph CNN

Text Classification with graph CNN

Helpful Blog

https://towardsdatascience.com/how-to-do-deep-learning-on-graphs-with-graph-convolutional-networks-7d2250723780

Why?

Shows many basic graph - matrix operations that are used heavily in graph CNN.

Power of Graph

Let’s start with a not so fun example.

"Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city and returns to the origin city?"

Power of Graph (not so funny) Contd.

Given a connected, edge-weighted undirected graph G, find a subset of the edges of G that connects all the vertices together, without any cycles and with the minimum possible total edge weight.

What about Social Networking Graphs!!

A social network graph is a graph where the nodes represent people and the lines between nodes, called edges, represent social connections between them, such as friendship or working together on a project.

Facebook Friendship graph => bidirectional

What about Twitter!!

Fun Fact

Six degrees of separation is the idea that all people are six or fewer social connections away from each other so that a chain of a friend of a friend.

More Examples

Pages on the internet and the links between them form a social network in much the same way as people form networks with other people.

Counter-intelligence agencies have used cell-phone data and calls to map out terrorist cells.

More Examples

Ever wonder how Facebook suggests "People you may know"? One of the first (and simplest) algorithms that is used, is to look for strangers with whom you have many mutual friends.

For example, if you and Colin are not friends on Facebook yet, but both of you are friends with Belinda, then {you, Belinda, Colin} form a Friend Suggestion Triangle (FST). The more FST's that exist, the more strangers Facebook can suggest with greater confidence that you might know them.

Pictorial Representation

Facebook Social Graph

Representation of information in Facebook

1. Nodes
2. Edges
3. Fields

Facebook Social Graph

Every thing, such as a user, photo, image, event, comment, page... all of them are considered as nodes.

Connection between these things are edges.

Friends, activities, photos, Tags, Likes, Interests etc. are treated as edges.

Information about these things are fields.

User has age, birth day. Page has category, description etc.

Influence Propagation

Social Networking => Influence Propagation

A graph structure is used to represent with what probability node i can influence node j.

ML in Graph- Challenges

Graph Structure => not a structured structure

Compared to images, audio or text, graph has a very irregular structure.

Graphs instead are non-Euclidean: the number of nodes are arbitrary and their connections are too.

Graph Convolution

Basic operations

Refer to
https://towardsdatascience.com/how-to-do-deep-learning-on-graphs-with-graph-convolutional-networks-7d2250723780

Graph Convolutional Networks for Text Classification

Create a graphical structure from a corpus of documents

Nodes in the graph => nodes for each document + nodes for each unique word

Edges in the graph => document-word edge + word-word edge

Edge weight between document-word edge => based on TF-IDF value

TF-IDF value => frequency of a word in a document, inverse fraction of documents that contain the word

Graph Creation

Word-word edge weight => PMI value of word(i,j) pair

PMI(i,j) = log[p(i,j)/ p(i). p(j)]

p(i,j) = #W(i,j) / #W

p(i) = #W(i) / #W

#W(i,j) = number of sliding windows that contain word i and j

#W = total number of sliding windows in the corpus

Create edges only when PMI is positive.

GCN Operations

$L(1) = f(AXW_0)$

$A = D^{-1/2}AD^{-½}$

F is relu at layer 1, Softmax at layer 2.

$L(j+1) = (AL(j)W_j)$

Initial Feature Matrix => Identity matrix

GCN Operations

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

Loss function => cross entropy over all labeled documents

$L = \sum_{d \text{ in } Y} \sum_{f = 1 \text{ -> } F} Y_{df} \ln Z_{df}$

Document indices => d

Set of labeled documents => Y

Number of output dimension => F

Observations and Parameter Settings

- Two-layer GCN performs better than a one layer GCN, while more layers did not improve the performances
- Embedding size of the first convolution layer as 200 and set the window size as 20.
- small changes did not change the results much.
- Set the learning rate as 0.02, dropout rate as 0.5, L2 loss weight as 0.
- Selected 10% of training set as validation set.
- Trained Text GCN for a maximum of 200 epochs using Adam (Kingma and Ba 2015) and stop training if the validation loss does not decrease for 10 consecutive epochs.

Baseline Models

TF-IDF + LR : bag-of-words model with TF-IDF. Logistic Regression is used as the classifier.

CNN, LSTM, BiLSTM,

PV-DBOW: a paragraph vector model proposed by (Le and Mikolov 2014), the orders of words in text are ignored. LR is the classifier.

PV-DM: a paragraph vector model proposed by (Le and Mikolov 2014), which considers the word order. LR is the classifier.

Baseline Models

PTE: predictive text embedding (Tang, Qu, and Mei 2015), which firstly learns word embedding based on heterogeneous text network containing words, documents and labels as nodes, then averages word embeddings as document embeddings for text classification.

fastText: a simple and efficient text classification method (Joulin et al. 2017), which treats the average of word/n-grams embeddings as document embeddings, then feeds document embeddings into a linear classifier. We evaluated it with and without bigrams.

SWEM: simple word embedding models (Shen et al. 2018), which employs simple pooling strategies operated over word embeddings.

Baseline Models

LEAM: label-embedding attentive models (Wang et al. 2018), which embeds the words and labels in the same joint space for text classification. It utilizes label descriptions.

Graph-CNN-C: a graph CNN model that operates convolutions over word embedding similarity graphs (Defferrard, Bresson, and Vandergheynst 2016), in which Chebyshev filter is used.

Graph-CNN-S: use Graph-CNN-C with Spline filter (Bruna et al. 2014).

Graph-CNN-F: the same as Graph-CNN-C but using Fourier filter (Henaff, Bruna, and LeCun 2015).

Details of the dataset

The Ohsumed corpus2 is from the MEDLINE database, which is a bibliographic database of important medical literature maintained by the National Library of Medicine.

MR is a movie review dataset for binary sentiment classification, in which each review only contains one sentence.

The 20NG dataset1 (bydate version) contains 18,846 documents evenly categorized into 20 different categories.

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

Table 2: Test Accuracy on document classification task. We run all models 10 times and report mean ± 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	**0.7775 ± 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	**0.8634 ± 0.0009**	**0.9707 ± 0.0010**	**0.9356 ± 0.0018**	**0.6836 ± 0.0056**	0.7674 ± 0.0020

More Observations

Text GCN did not outperform CNN and LSTM-based models on MR. This is because GCN ignores word orders that are very useful in sentiment classification, while CNN and LSTM model consecutive word sequences explicitly.

Another reason is that the edges in MR text graph are fewer than other text graphs, which limits the message passing among the nodes. There are only few document-word edges because the documents are very short. The number of word-word edges is also limited due to the small number of sliding windows.

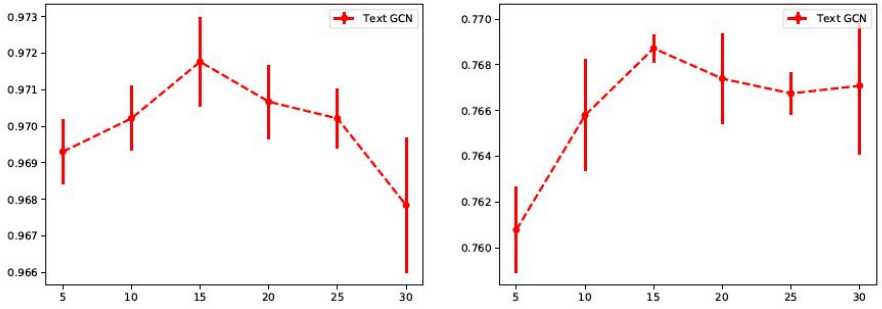
Nevertheless, CNN and LSTM rely on pre-trained word embeddings from external

corpora while Text GCN only uses information in the target

input corpus.

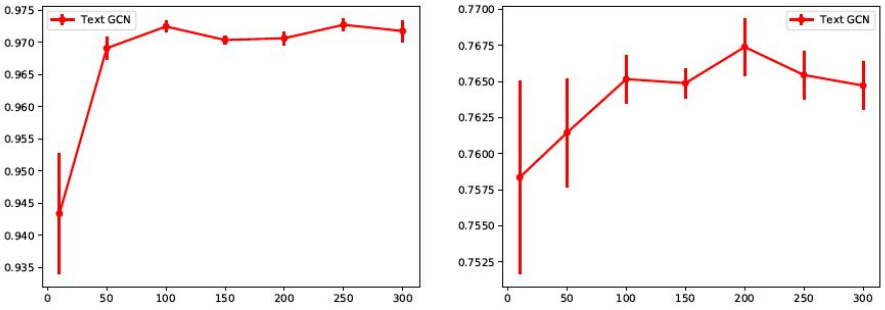
Transductive vs Inductive Learning

GCN model is inherently transductive, in which test document nodes (without labels) are included in GCN training. Thus Text GCN could not quickly generate embeddings and make prediction for unseen test documents. Possible solutions to the problem are introducing inductive (



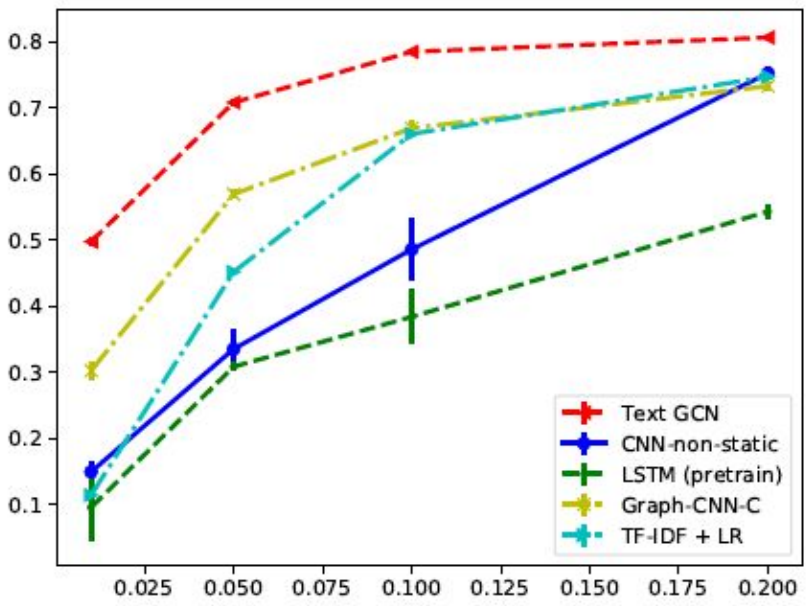
(a) R8 (b) MR

Figure 2: Test accuracy with different sliding window sizes.

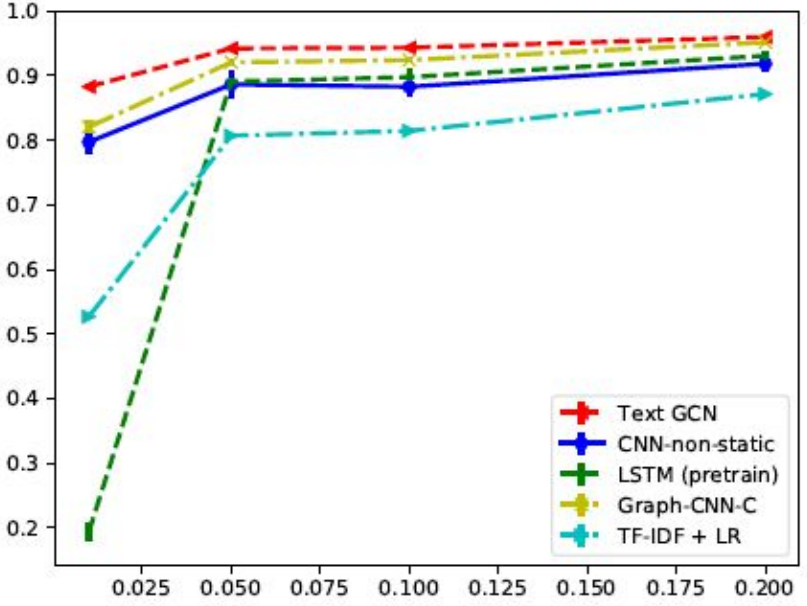


(a) R8 (b) MR

Figure 3: Test accuracy by varying embedding dimensions.

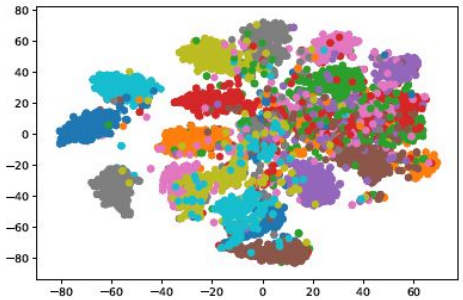


(a) 20NG

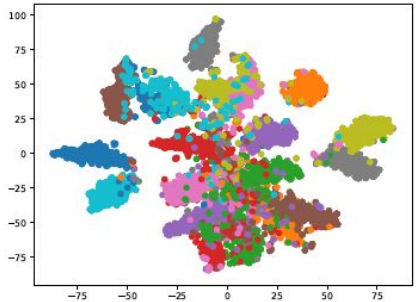


(b) R8

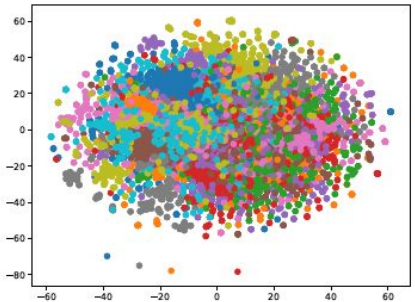
Figure 4: Test accuracy by varying training data proportions.



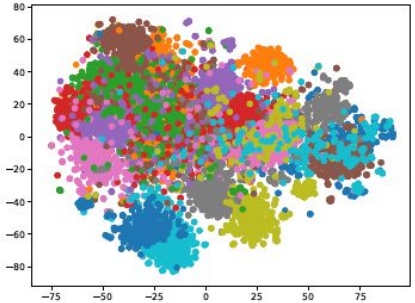
(a) Text GCN, 1st layer



(b) Text GCN, 2nd layer



(c) PV-DBOW



(d) PTE

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

Thank you