Deep Learning in Python (An Example in NLP)

Yuan Luo, PhD, FAMIA

Chief AI Officer

Northwestern University Clinical and Translational Sciences Institute

Associate Professor

Department of Preventive Medicine

Departments of IEMS and EECS (Courtesy)

Northwestern University

yuan.luo@northwestern.edu

@yuanhypnosluo

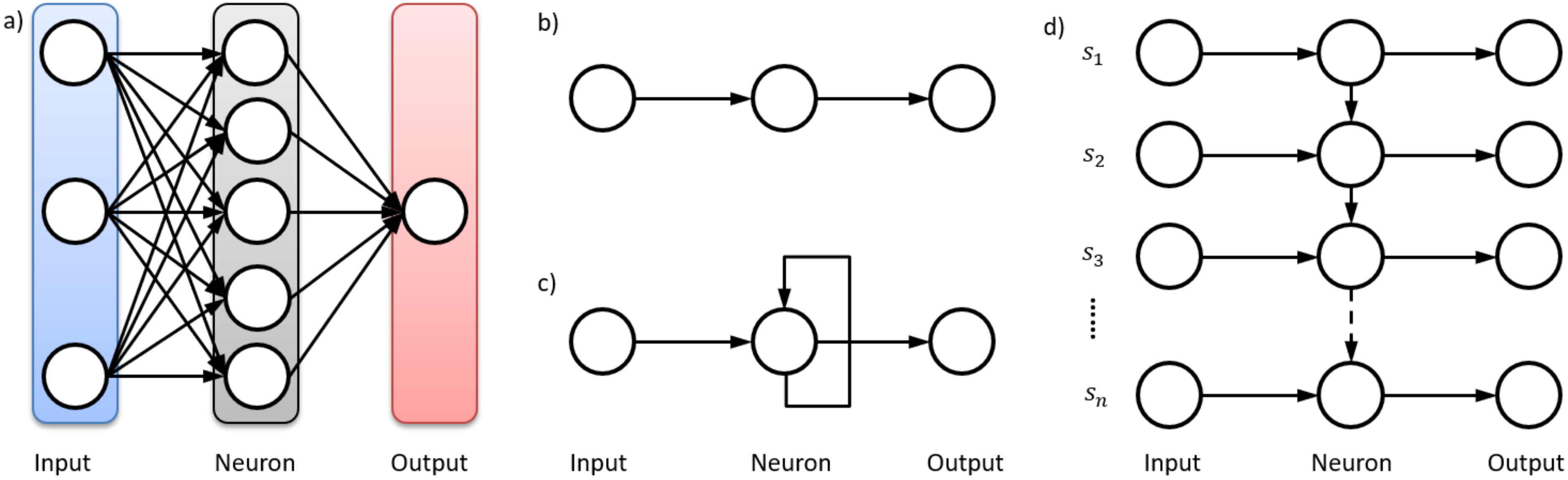
3/5/2020

NORTHWESTERN UNIVERSITY FEINBERG SCHOOL OF MEDICINE

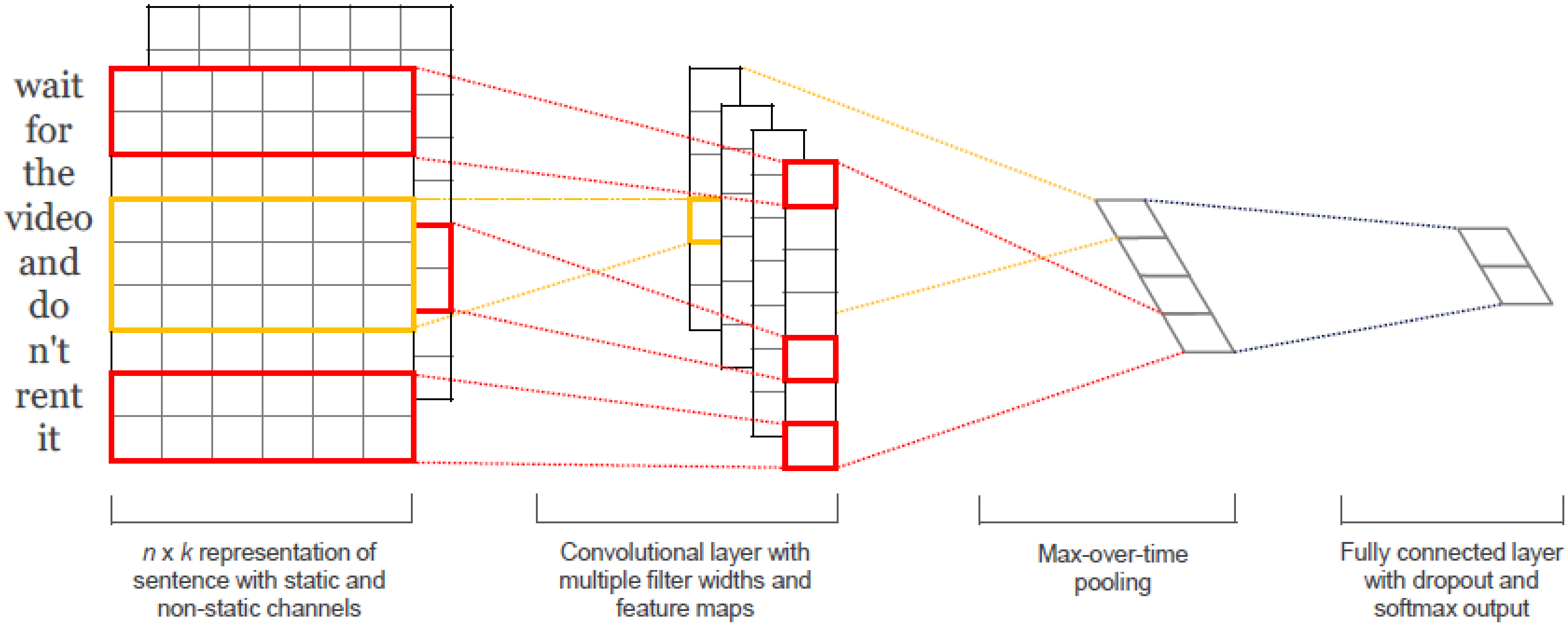
Northwestern University

Basics on Neural Networks

- From multi-layer neural networks to recurrent neural networks



Convolutional Neural Networks


wait
for
the
video
and
do
n't
rent
it
n x *k* representation of sentence with static and non-static channels
Convolutional layer with multiple filter widths and feature maps
Max-over-time pooling
Fully connected layer with dropout and softmax output

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

Using GCN for Long Text Understanding

- Challenges of Long Text Understanding
  - 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

- GCN can
  - 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)

Graph Convolution

- Denote $A \in \mathbb{R}^{n \times n}$ as the dependency graph G's adjacency matrix
- Graph Laplacian is $L = D - A$, where $D_{ii} = \sum_j A_{ij}$
- Let $U \in \mathbb{R}^{n \times n}$ be the matrix of eigenvectors of the normalized graph Laplacian $L_n = I - D^{-\frac{1}{2}} A D^{-\frac{1}{2}} = U \Lambda U^T$
- Let $g_w \in \mathbb{R}^{n \times n}$ be a Fourier domain filter matrix parametrized with a scalar w as its diagonal elements (recall signal processing theory)
- The graph convolution for the 1-dimensional embedding $x \in \mathbb{R}^n$ (for n words) is

$$h = U g_w U^T x = U diag([w, \ldots, w]) U^T x = U U^T x \, diag([w, \ldots, w])$$

- Simplify using Chebyshev polynomial approximation
- $h = \left(I + D^{-\frac{1}{2}} A D^{-\frac{1}{2}}\right) x w$, after renormalization $h = \widetilde{D}^{-\frac{1}{2}} \tilde{A} \widetilde{D}^{-\frac{1}{2}} x w$
- Extend the embedding and convolved signal to d-dimensional $X \in \mathbb{R}^{n \times d}$ and $H \in \mathbb{R}^{n \times d}$

$$H = \rho\left(\widetilde{D}^{-\frac{1}{2}} \tilde{A} \widetilde{D}^{-\frac{1}{2}} X W\right)$$

Graph Convolutional Networks (GCN)

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

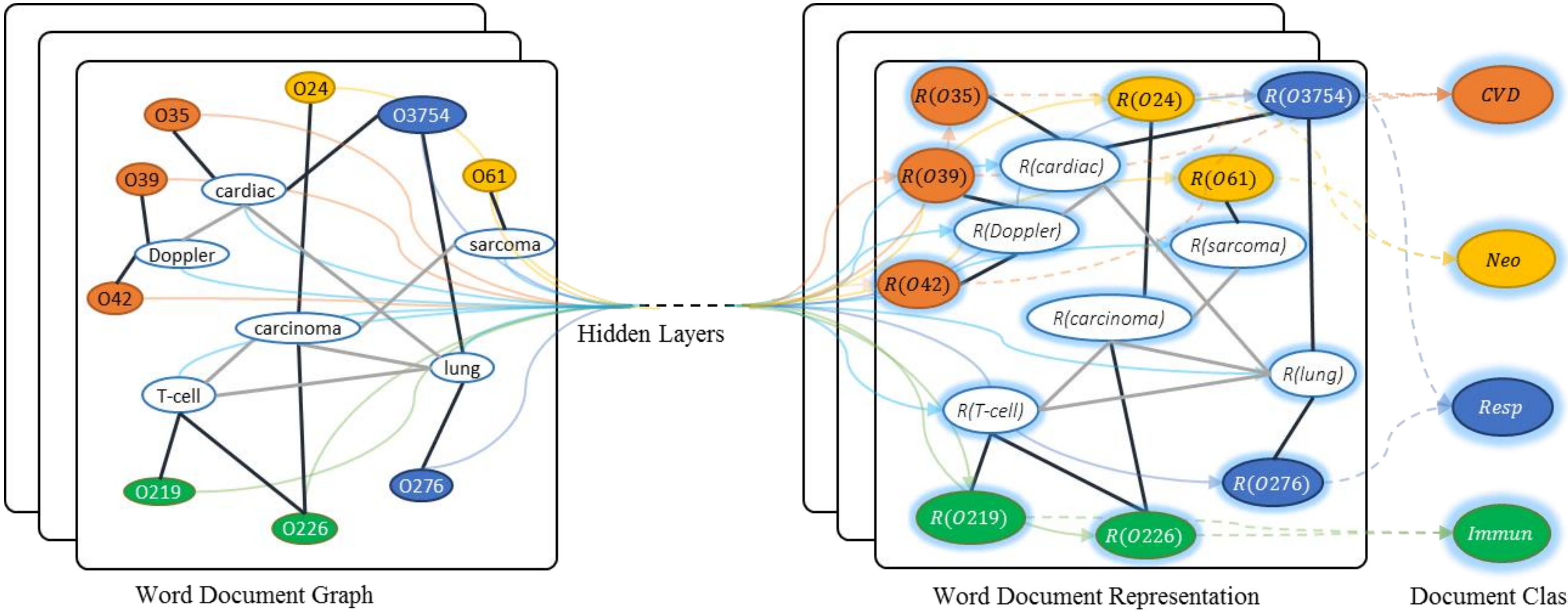
- One layer GCN:
- $L^{(1)} = \rho(\tilde{A} X W_0)$

- Stacking multiple GCN layers:
- $L^{(j+1)} = \rho(\tilde{A} L^{(j)} W_j)$

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 neighbors (second order neighbors for a two-layer GCN).

Text Graph Convolutional Networks (Text GCN)



L Yao, C Mao, **Y Luo***. Graph Convolutional Networks for Text Classification. *Proceedings of AAAI Conference on Artificial Intelligence 2019 7370-7377.*

Text Graph Convolutional Networks (Text GCN)

- Document content and global word co-occurrence
  - Document-word edges: TF-IDF
  - Word-word edges: point-wise mutual information (PMI)

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

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

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

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

Text Graph Convolutional Networks (Text GCN)

- A simple two-layer GCN:
  - one-hot feature matrix for words and documents: $X = I$
  - 1st layer document and word embeddings: $\tilde{A}XW_0$
  - 2nd layer document and word embeddings: $\tilde{A}ReLU(\tilde{A}XW_0)W_1$
  - $\mathcal{Y}_D$ is the set of document indices that have labels and F is the dimension of the output features, which is equal to the number of classes, Y is the label indicator matrix
  - Loss function

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

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

Dataset

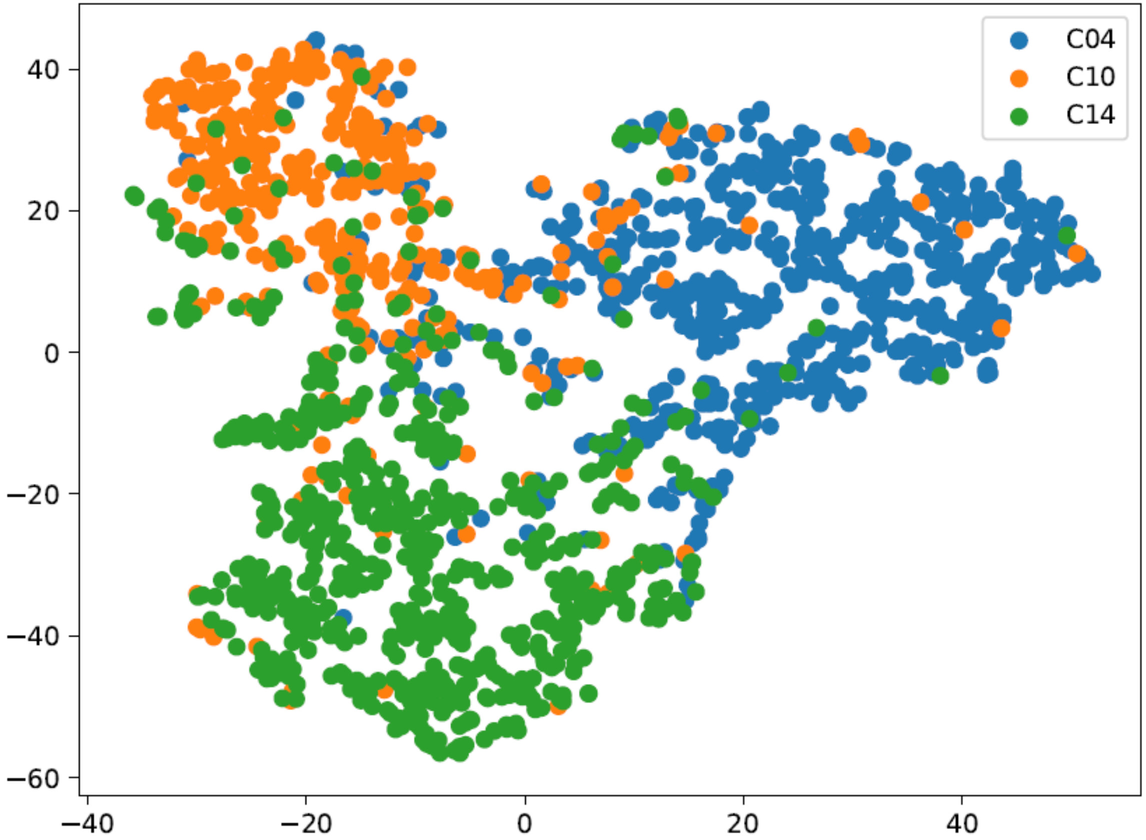
- The Ohsumed corpus is from the MEDLINE database, which is a bibliographic database of important medical literature maintained by the National Library of Medicine
- In this tutorial, we created a subsample of the 2,762 unique diseases abstracts from 3 categories
  - C04: Neoplasms
  - C10: Nervous System Diseases
  - C14: Cardiovascular Diseases
- As we focus on single-label text classification, the documents belonging to multiple categories are excluded
- 1230 train (use 10% as validation), 1532 test

Now let’s look at some code

- https://github.com/yuanluo/text_gcn_tutorial
- Run `python remove_words.py ohsumed_3`
- Run `python build_graph.py ohsumed_3`
- Run `python train.py ohsumed_3`
- Run `python tsne.py`

Results

- C04: Neoplasms
- C10: Nervous System Diseases
- C14: Cardiovascular Diseases



● C04
● C10
● C14
40
20
0
−20
−40
−60
−40
−20
0
20
40