CS247: ADVANCED DATA MINING

09: Graph and Network: Graph Embedding

Instructor: Yizhou Sun

yzsun@ccs.neu.edu

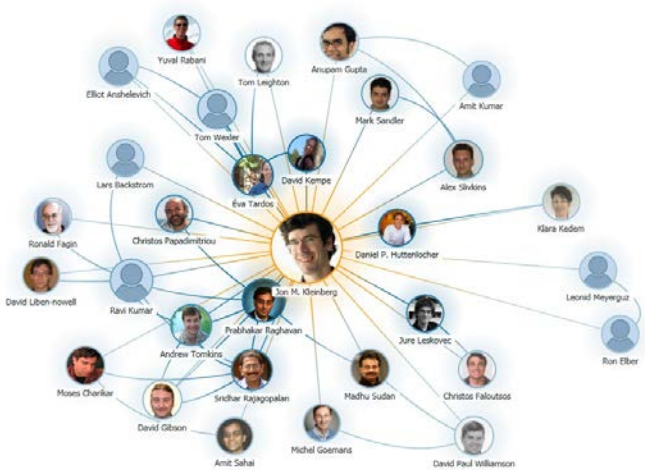
February 8, 2022

Graph Embedding

- What is Graph Embedding
- Shallow Network Embedding
- Knowledge Graph Embedding
- Graph Neural Network
- Summary

How to represent nodes?

- A naïve solution

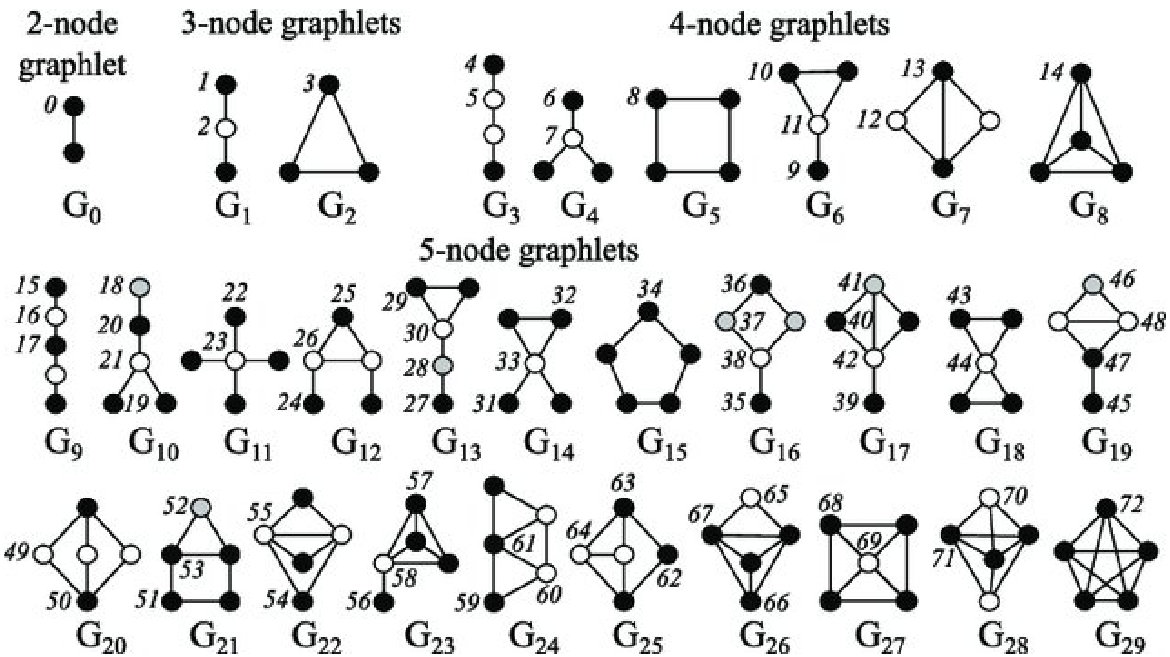


	A	B	C	D	E	F
A	0	1	1	1	0	0
B	1	0	0	0	1	1
C	1	0	0	0	0	1
D	1	0	0	0	0	0
E	0	1	0	0	0	0
F	0	1	1	0	0	0

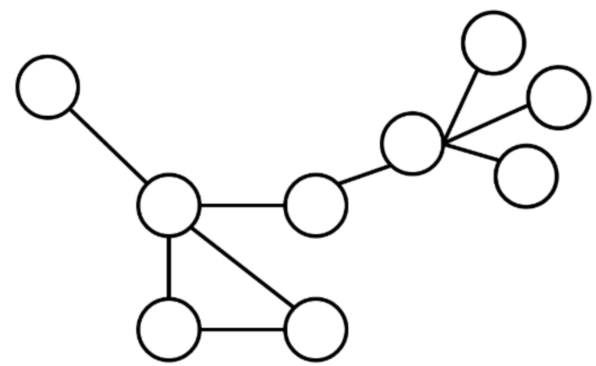
- Limitations:
  - Extremely High-dimensional
  - No global structure information integrated
  - Permutation-variant

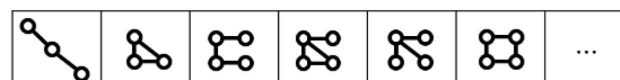
Even more challenging for graph representation

- Ex. Graphlet-based feature vector



Source: DOI: 10.1093/bioinformatics/btv130





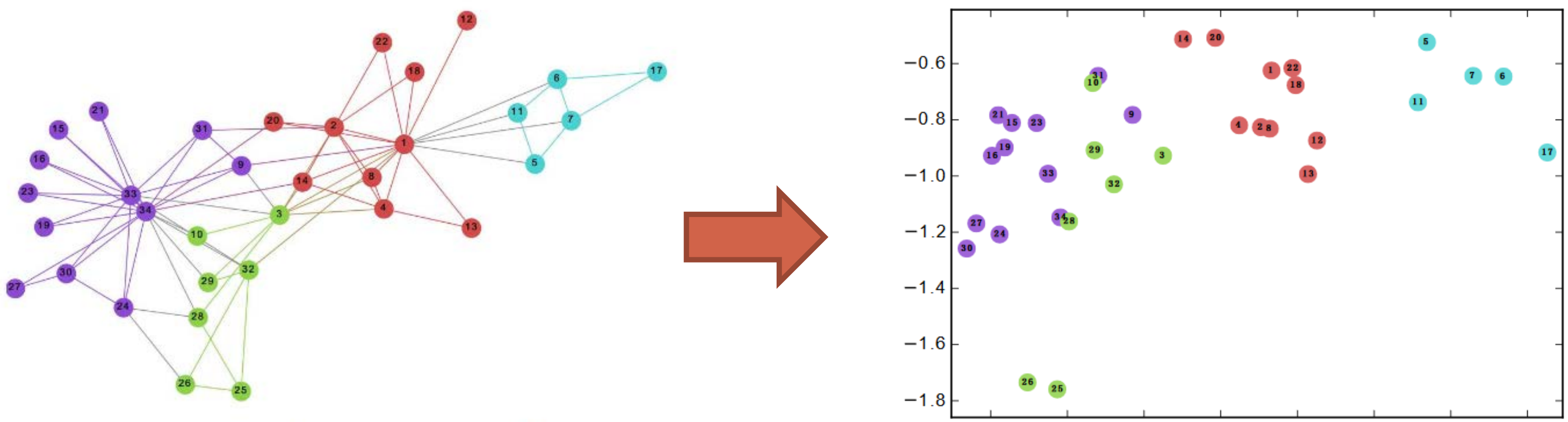
12	1	4	1	6	0	...

Requires subgraph isomorphism test: NP-hard

Source: https://haotang1995.github.io/projects/robust_graph_level_representation_learning_using_graph_based_structural_attentional_learning

A Better Solution

- Map each node into a low dimensional vector
  - $\phi: V \rightarrow R^d$



(a) Input: karate network

(b) Output: representations

Source: DeepWalk

Graph Embedding

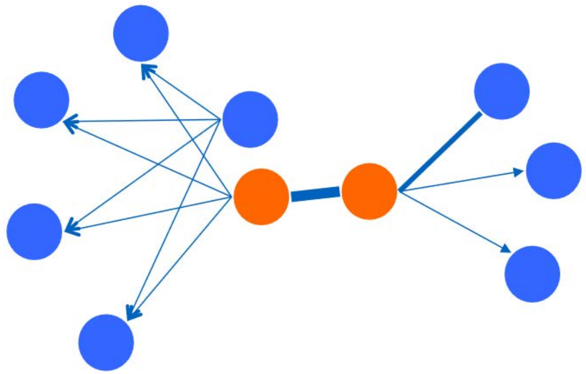
- What is Graph Embedding
- Shallow Network Embedding
- Knowledge Graph Embedding
- Graph Neural Network
- Summary

Shallow Network Embedding Approaches

- Inspired by word embedding
  - A node's embedding is determined by its context
- How to define the local context of a node?
  - DeepWalk [Perozzi, KDD'14]
  - LINE [Tang, WWW'15]
  - Node2Vec [Grover, KDD'16]

LINE: Large-scale Information Network Embedding

- First-order proximity



- Assumption: Two nodes are similar if they are connected

$$p_1(v_i, v_j) = \frac{\exp(\vec{u}_i^T \vec{u}_j)}{\sum_{(m,n)\in V\times V} \exp(\vec{u}_m^T \vec{u}_n)}$$

u_i: *embedding vector for node i*

- Limitation: links are sparse, not sufficient

Objective function for first-order proximity

- Minimize the KL divergence between empirical link distribution and modeled link distribution

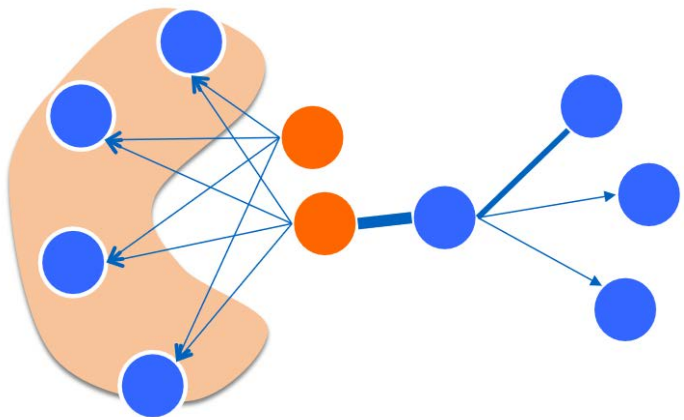
$$\hat{p}_1(v_i, v_j) = \frac{w_{ij}}{\sum_{(m,n)\in E} w_{mn}}$$

$$O_1 = KL(\hat{p}_1, p_1) = -\sum_{(i,j)\in E} w_{ij} \log p_1(v_i, v_j)$$

$w_{ij}: weight\ over\ edge(i,j)$

Second-Order Proximity

- Assumption:
  - Two nodes are similar if their neighbors are similar



$$p_2(v_j|v_i) = \frac{\exp(\vec{u}_j'^T \cdot \vec{u}_i)}{\sum_{k=1}^{|V|} \exp(\vec{u}_k'^T \cdot \vec{u}_i)}$$

u_i: *target embedding vector for node i*
u_j': *context embedding vector for node j*

Objective function for second-order proximity

- Minimize the KL divergence between empirical link distribution and modeled link distribution
  - Empirical distribution $\hat{p}_2(v_j \mid v_i) = \frac{w_{ij}}{\sum_{k \in V} w_{ik}}$
  - Objective function

$$O_2 = \sum_i d_i KL(\hat{p}_2(\cdot \mid v_i), p_2(\cdot \mid v_i)) = -\sum_{(i,j) \in E} w_{ij} \log p_2(v_j \mid v_i)$$

$$d_i = \sum_k w_{ik}$$

Negative Sampling for Optimization

- For second-order proximity derived objective function
  - For each positive link (i, j), sample K negative links (i, n)
    - An edge with weight w can be considered as w binary edges

$$\log \sigma(\vec{u}_j'^{\,T} \cdot \vec{u}_i) + \sum_{i=1}^{K} E_{v_n \sim P_n(v)}[\log \sigma(-\vec{u}_n'^{\,T} \cdot \vec{u}_i)]$$

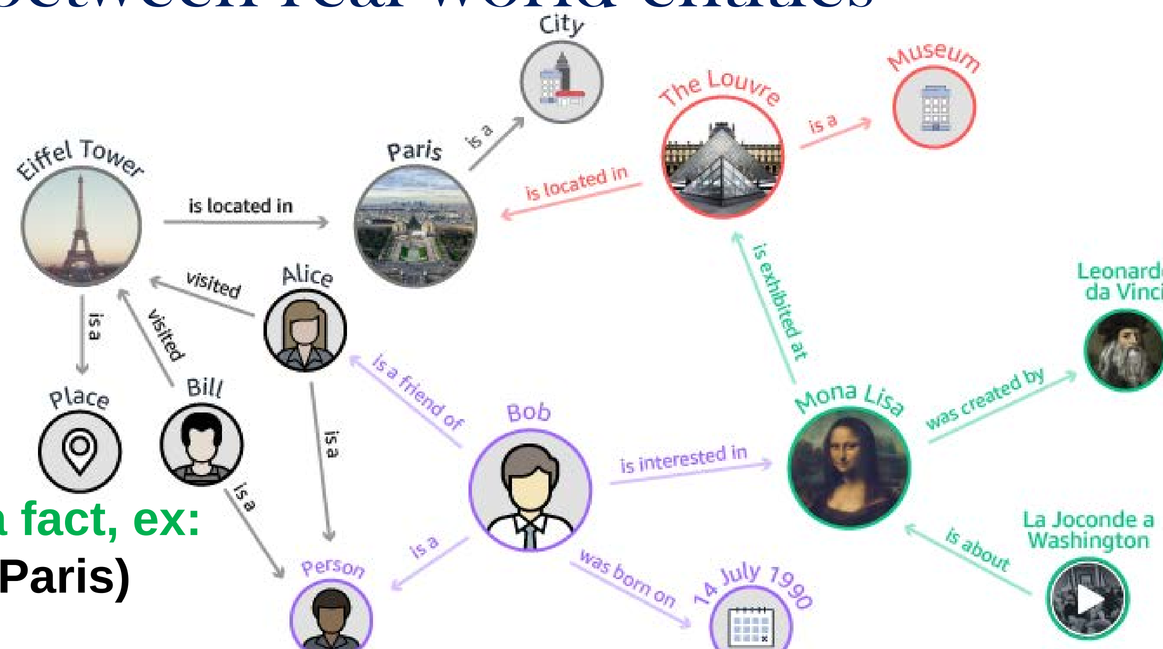
negative distribution: $P_n(v) \propto d_v^{3/4}$

Graph Embedding

- What is Graph Embedding
- Shallow Network Embedding
- Knowledge Graph Embedding
- Graph Neural Networks
- Summary

Knowledge Graph

- What are knowledge graphs?
  - Multi-relational graph data
    - (heterogeneous information network)
  - Provide structured representation for semantic relationships between real-world entities



A triple (h, r, t) represents a fact, ex:
(Eiffel Tower, is located in, Paris)

KGs are everywhere

General-purpose KGs

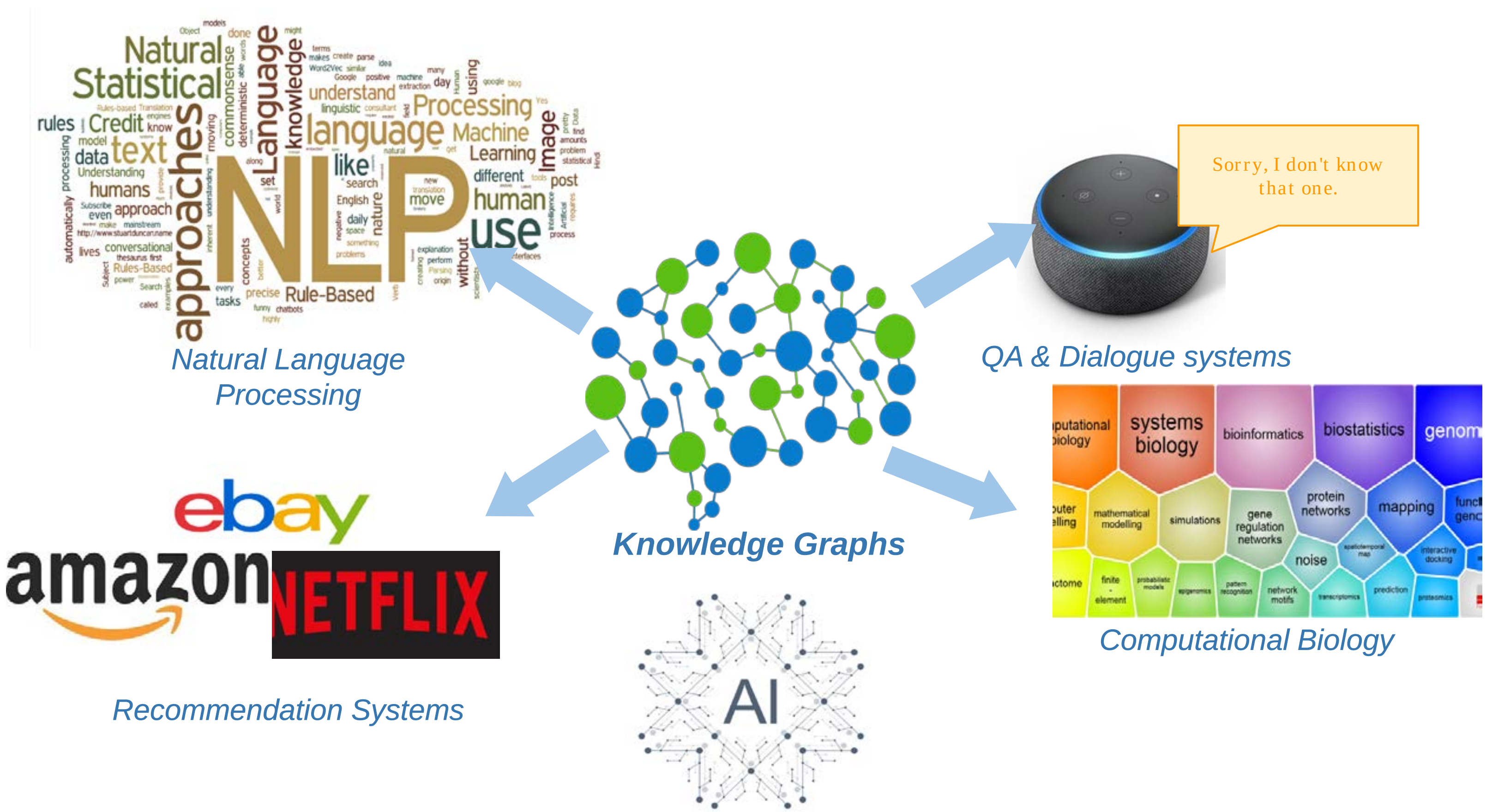
Bio & Medical KGs

Product Graphs & E-commerce

Common-sense KGs & NLP

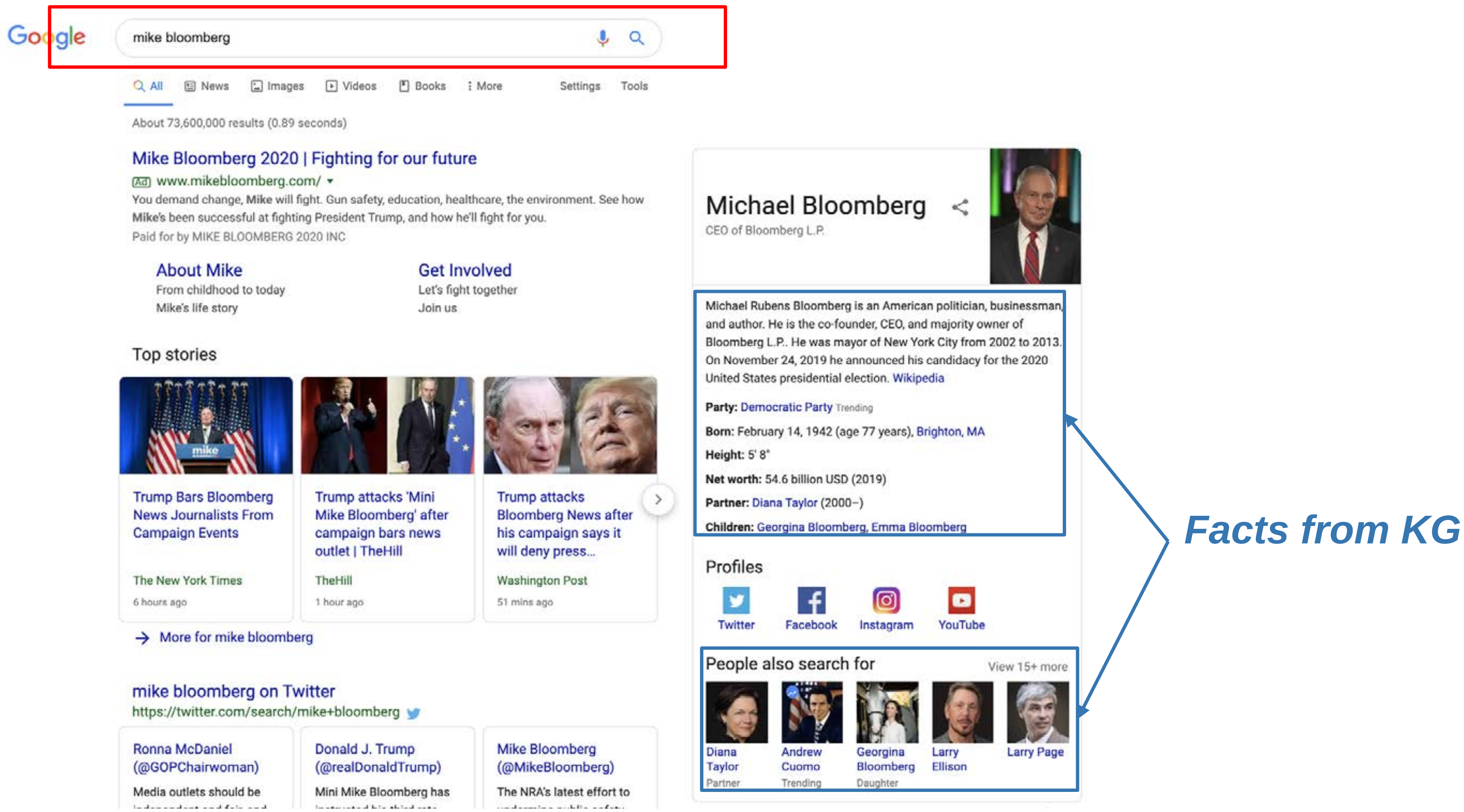
Applications of KGs

- Foundational to knowledge-driven AI systems
- Enable many downstream applications (NLP tasks, QA systems, etc)



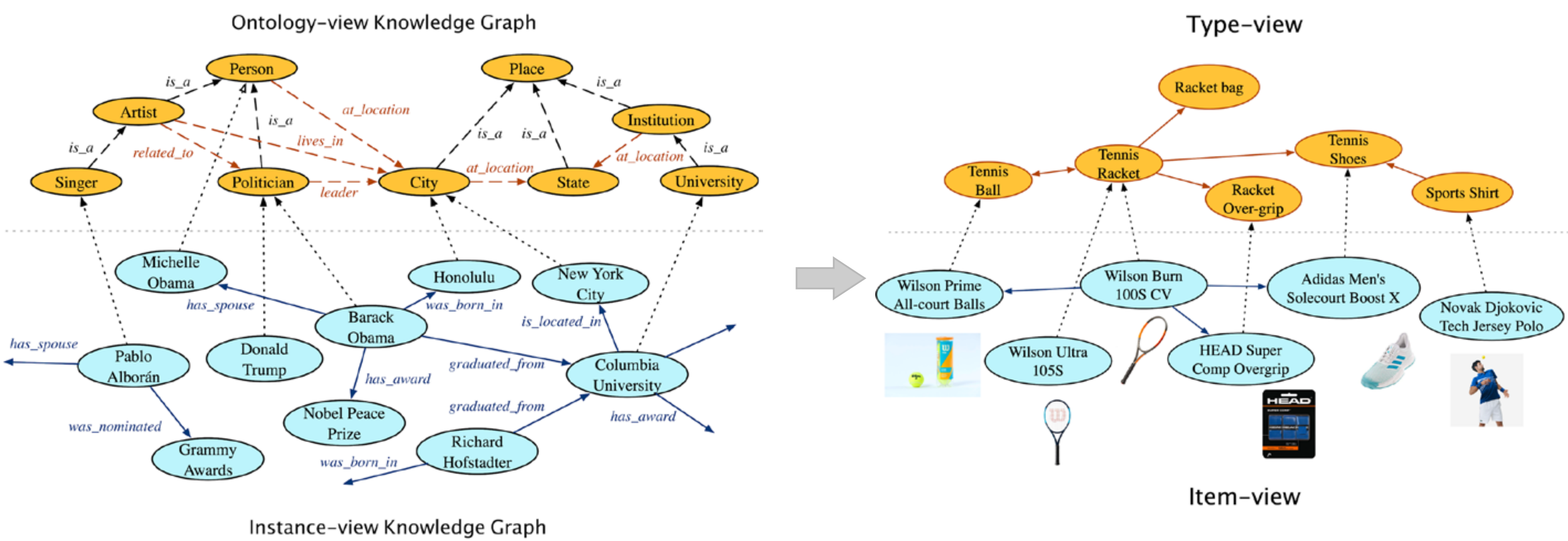
Application in Search Engine

- When you search in Google



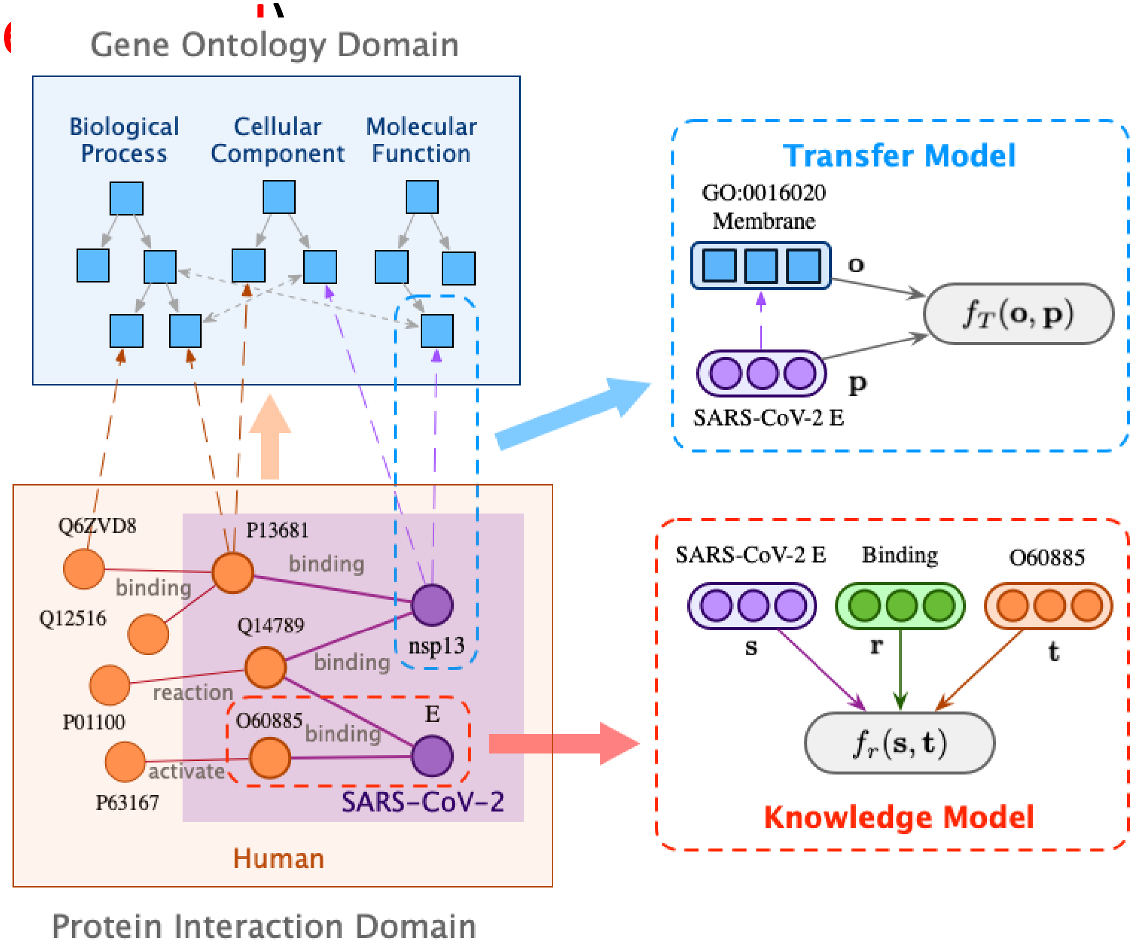
Applications to Product KG

- Hao et al., "P-Companion: A Principled Framework for Diversified Complementary Product Recommendation", CIKM'20



Application to Biological KG

- Hao et al., "Bio-JOIE: Joint Representation Learning of Biological Knowledge Bases", ACM BCB'20 (best student pape

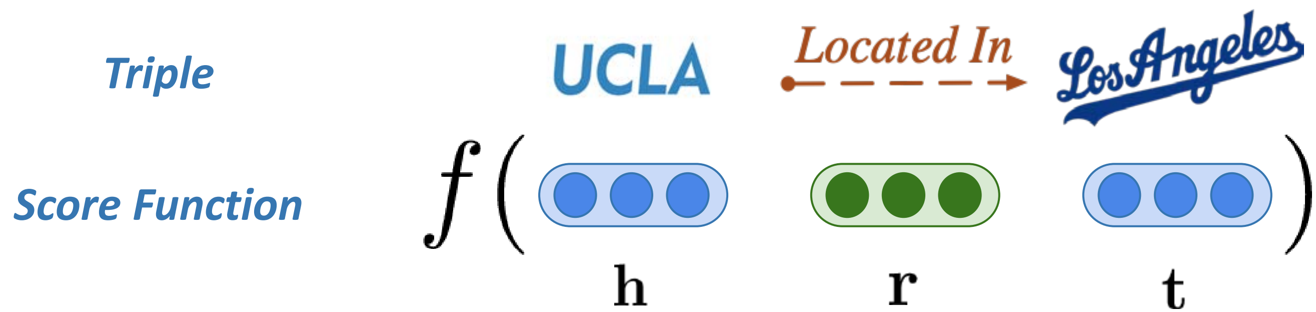


Knowledge Graph Embedding

- Goal: Encode entities as low-dimensional vectors and relations as parametric algebraic operations
  - Input: Relation facts (triples)
  - Output: representations of objects and relations

Key Idea of KG embedding algorithms

- Define a score function for a triple: $f_r(\boldsymbol{h}, \boldsymbol{t})$
  - According to entity and relation representation



- Define a loss function to guide the training
  - E.g., an observed triple scores higher than a negative one

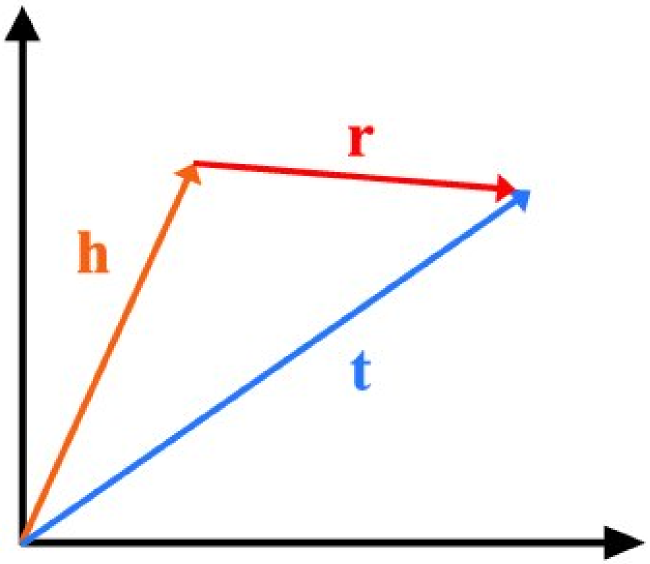
Summary of Existing Approaches

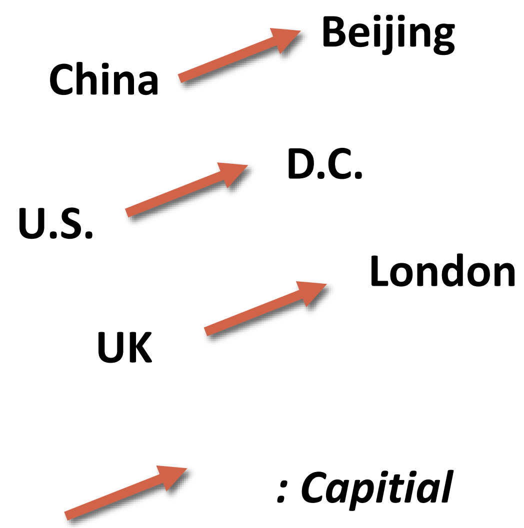
Model	Score Function	
SE (Bordes et al., 2011)	$-\|\boldsymbol{W}_{r,1}\mathbf{h} - \boldsymbol{W}_{r,2}\mathbf{t}\|$	$\mathbf{h}, \mathbf{t} \in \mathbb{R}^k, \boldsymbol{W}_{r,\cdot} \in \mathbb{R}^{k \times k}$
TransE (Bordes et al., 2013)	$-\|\mathbf{h} + \mathbf{r} - \mathbf{t}\|$	$\mathbf{h}, \mathbf{r}, \mathbf{t} \in \mathbb{R}^k$
TransX	$-\|g_{r,1}(\mathbf{h}) + \mathbf{r} - g_{r,2}(\mathbf{t})\|$	$\mathbf{h}, \mathbf{r}, \mathbf{t} \in \mathbb{R}^k$
DistMult (Yang et al., 2014)	$\langle \mathbf{r}, \mathbf{h}, \mathbf{t} \rangle$	$\mathbf{h}, \mathbf{r}, \mathbf{t} \in \mathbb{R}^k$
ComplEx (Trouillon et al., 2016)	$\mathrm{Re}(\langle \mathbf{r}, \mathbf{h}, \overline{\mathbf{t}} \rangle)$	$\mathbf{h}, \mathbf{r}, \mathbf{t} \in \mathbb{C}^k$
HolE (Nickel et al., 2016)	$\langle \mathbf{r}, \mathbf{h} \otimes \mathbf{t} \rangle$	$\mathbf{h}, \mathbf{r}, \mathbf{t} \in \mathbb{R}^k$
ConvE (Dettmers et al., 2017)	$\langle \sigma(\mathrm{vec}(\sigma([\overline{\mathbf{r}}, \overline{\mathbf{h}}] * \boldsymbol{\Omega}))\boldsymbol{W}), \mathbf{t} \rangle$	$\mathbf{h}, \mathbf{r}, \mathbf{t} \in \mathbb{R}^k$
RotatE	$-\|\mathbf{h} \circ \mathbf{r} - \mathbf{t}\|^2$	$\mathbf{h}, \mathbf{r}, \mathbf{t} \in \mathbb{C}^k, \|r_i\| = 1$

Source: Sun et al., RotatE: Knowledge Graph Embedding by Relational Rotation in Complex Space (ICLR’19)

TransE: Score Function

- Relation: translating embedding





- Score function
  - $f_r(\boldsymbol{h}, \boldsymbol{t}) = -||\boldsymbol{h} + \boldsymbol{r} - \boldsymbol{t}|| = -d(\boldsymbol{h} + \boldsymbol{r}, \boldsymbol{t})$

Bordes et al., Translating embeddings for modeling multi-relational data, NeurIPS 2013

TransE: Objective Function

- Objective Function
  - Margin-based ranking loss
  - $L = \sum_{(h,r,t)\in S} \sum_{(h',r,t')\in S'_{(h,r,t)}} [\gamma + d\,(\boldsymbol{h} + \boldsymbol{r}, \boldsymbol{t}) - d(\boldsymbol{h}' + \boldsymbol{r}, \boldsymbol{t}')]_+$
    - $[x]_+$ denotes the positive part of x, i.e., $\max(0, x)$
    - $\gamma > 0$ denotes the margin hyperparameter
      - The higher the bigger difference between positive triple and negative one
    - S: positive triple set; S': corrupted triple set (negative triples)
- Optimization: stochastic gradient descent

TransE: Limitations

- One-one mapping: $t = \phi_r(h)$
  - Given (h,r), t is unique
  - Given (r,t), h is unique
- Anti-symmetric
  - If r(h,t) then r(t,h) is not true
  - Cannot model symmetric relation, e.g., friendship
- Anti-reflexive
  - r(h,h) is not true
  - Cannot model reflexive relations, e.g., synonym

DistMult

- Bilinear score function
  - $f_r(\boldsymbol{h}, \boldsymbol{t}) = \boldsymbol{h}^T \boldsymbol{M}_r \boldsymbol{t}$
    - Where $\boldsymbol{M}_r$ is a diagonal matrix with diagonal vector $\boldsymbol{r}$
  - A simplification to neural tensor network (NTN)
- Objective function
  - $L = \sum_{(h,r,t)\in S} \sum_{(h',r,t')\in S'_{(h,r,t)}} [\gamma - f_r(\boldsymbol{h}, \boldsymbol{t}) + f_r(\boldsymbol{h}', \boldsymbol{t}')]_+$
- Limitation
  - Can only model symmetric relation
    - $f_r(\boldsymbol{h}, \boldsymbol{t}) = f_r(\boldsymbol{t}, \boldsymbol{h})$

Yang et al., Embedding entities and relations for learning and inference in knowledge bases, ICLR 2015

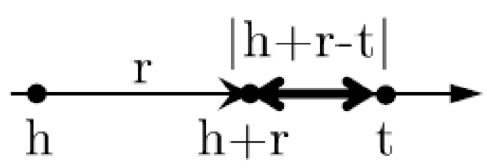
RotatE: Score Function

- Relation: rotation operation in complex space
  - head and tail entities in complex vector space, i.e., $\mathbf{h}, \mathbf{t} \in \mathbb{C}^k$
  - each relation r as an element-wise rotation from the head entity **h** to the tail entity **t**, i.e.,
    - $\boldsymbol{t} = \boldsymbol{h} \circ \boldsymbol{r},\ i.e., t_i = h_i r_i, where\ |r_i| = 1$
    - Equivalently, $r_i = e^{i\theta_{r,i}}, i.e., rotate$ $\mathrm{h_i}$ with $\theta_{r,i}$
- Score function:
  - $f_r(h,t) = -||\boldsymbol{h} \circ \boldsymbol{r} - \boldsymbol{t}||$

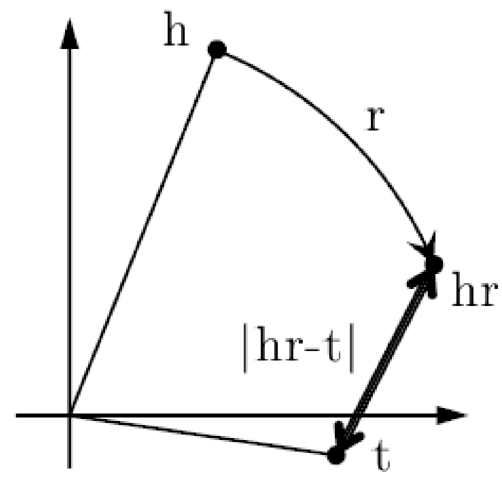
Zhiqing Sun, Zhihong Deng, Jian-Yun Nie, and Jian Tang. "RotatE: Knowledge Graph Embedding by Relational Rotation in Complex Space." ICLR'19.

RotatE: Geometric Interpretation

- Consider 1-d case



(a) TransE models r as translation in real line.



(b) RotatE models r as rotation in complex plane.

RotatE: Objective function

- Smarter negative sampling
  - The negative triple with higher score is more likely to be sampled

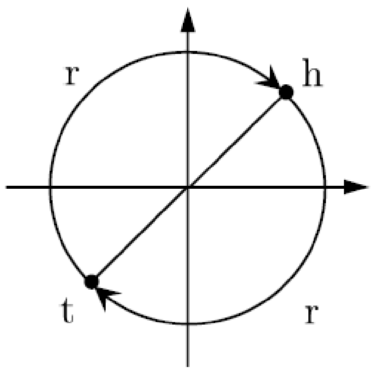
$$p(h'_j, r, t'_j | \{(h_i, r_i, t_i)\}) = \frac{\exp \alpha f_r(\mathbf{h}'_j, \mathbf{t}'_j)}{\sum_i \exp \alpha f_r(\mathbf{h}'_i, \mathbf{t}'_i)}$$

- Cross-entropy loss

$$L = -\log \sigma(\gamma - d_r(\mathbf{h}, \mathbf{t})) - \sum_{i=1}^{n} p(h'_i, r, t'_i) \log \sigma(d_r(\mathbf{h}'_i, \mathbf{t}'_i) - \gamma)$$

RotatE: Pros and Cons

- Pros:
  - Can model relations with different properties
  - Symmetric: $r_i = +1\ or\ -1$
  - Anti-symmetric: $\boldsymbol{r} \circ \boldsymbol{r} \neq \mathbf{1}$
  - Inverse relations: $\mathbf{r}_2 = \overline{\mathbf{r}_1}$
    - E.g., hypernym is the **inverse** relation of hyponym
  - Composition relations: $\boldsymbol{r_3} = \boldsymbol{r_1} \circ \boldsymbol{r_2}$, i.e., $\boldsymbol{\theta_3} = \boldsymbol{\theta_1} + \boldsymbol{\theta_2}$, if $\boldsymbol{r}_j = e^{i\boldsymbol{\theta}_j}$ for j = 1,2,3



- Cons:
  - One-one mapping
  - Relations are commutative: i.e., $\boldsymbol{r_1} \circ \boldsymbol{r_2} = \boldsymbol{r_2} \circ \boldsymbol{r_1}$
    - Which is not always true, e.g., father's wife ≠ wife's father

Graph Embedding

- What is Graph Embedding
- Shallow Network Embedding
- Knowledge Graph Embedding
- Graph Neural Networks
- Summary

Limitation of Shallow Network Embedding

- Too many parameters
  - Each node is associated with an embedding vector, which are parameters
- Not inductive
  - Cannot handle new nodes
- Cannot handle node attributes

From shallow embedding to Graph Neural Networks

- The embedding function (encoder) is more complicated
  - Shallow embedding
    - $\phi(v) = U^T x_v$, where U is the embedding matrix and x_v is the one-hot encoding vector
  - Graph neural networks
    - $\phi(v)$ is a neural network depending on the graph structure

Notations

- An attributed graph $G = (V, E)$
  - V: vertex set
  - E: edge set
  - A: adjacency matrix
  - $X \in R^{d_0 \times |V|}$: feature matrix for all the nodes
  - $N(v)$: neighbors of node v
  - h_v^l: Representation vector of node v at Layer l
    - $Note\, h_v^0 = x_v$
  - $H^l \in R^{d_l \times |V|}$: representation matrix

The General Architecture of GNNs

- For a node v at layer t

$$h_v^{(t)} = f\left(\underline{h_v^{(t-1)}}, \left\{\underline{h_u^{(t-1)} | u \in \mathcal{N}(v)}\right\}\right)$$

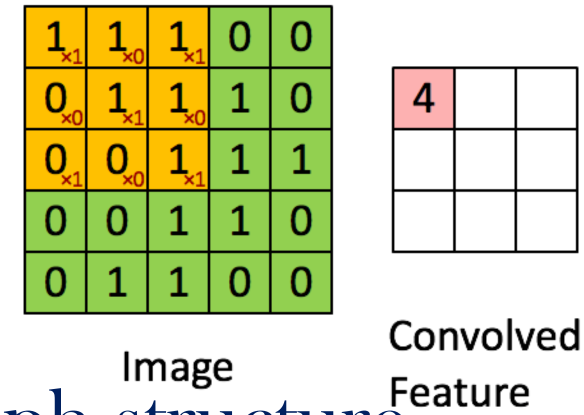
representation vector from previous layer for node v

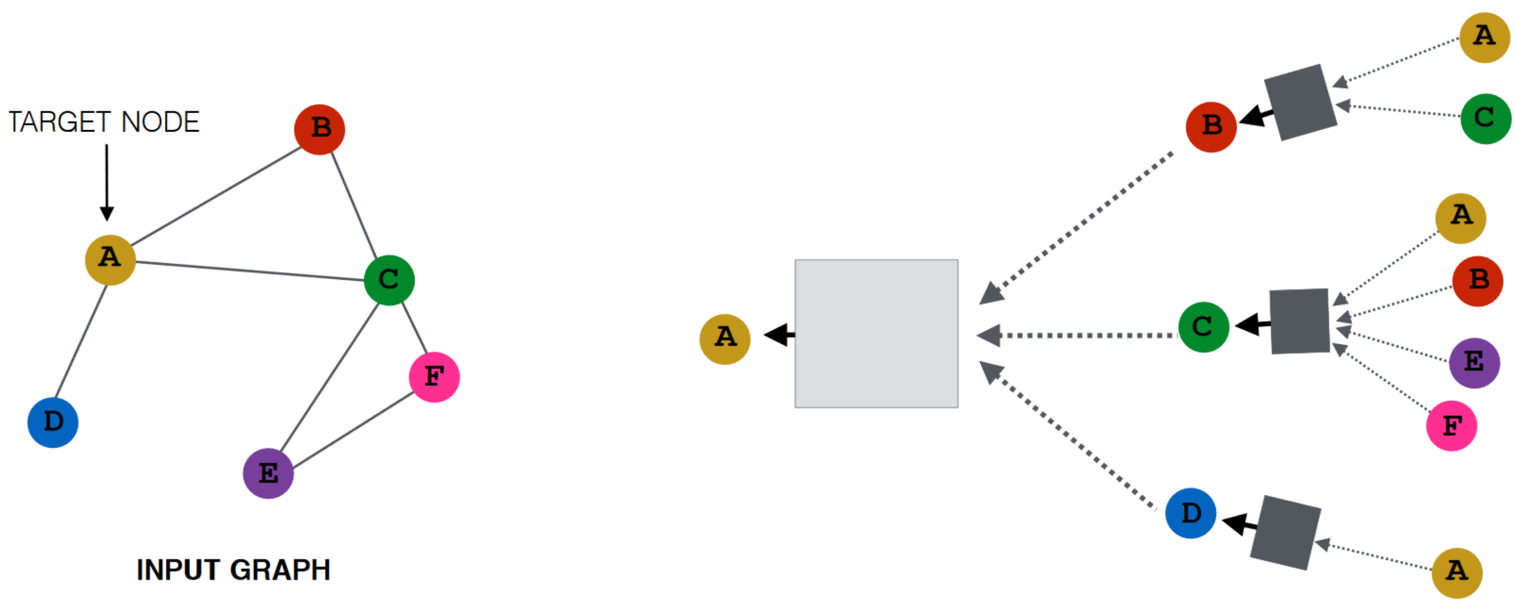
representation vectors from previous layer for node v's neighbors

- A function of representations of neighbors and itself from previous layers
  - **Aggregation** of neighbors
  - **Transformation** to a different space
  - **Combination** of neighbors and the node itself

Compare with CNN

- Recall CNN
  - Regular graph
- GNN
  - Extend to irregular graph structure





Graph Convolutional Network (GCN)

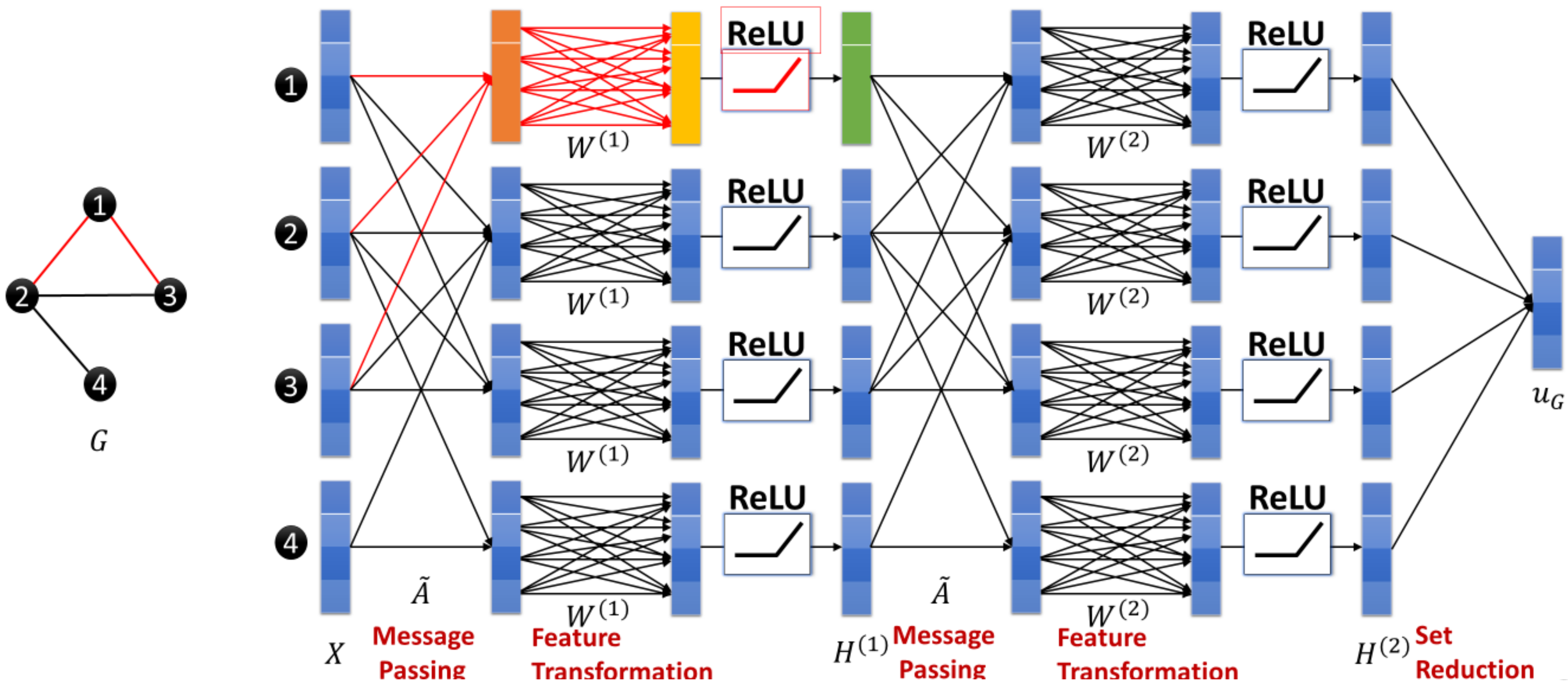
- Kipf and Welling, ICLR'17
  - $f(H^{(l)}, A) = \sigma\left(\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}\right)$, $\hat{A} = A + I$
  - f: graph filter
- From a node v's perspective

$$\mathbf{h}_v^k = \sigma\left(\mathbf{W}_k \sum_{u \in N(v) \cup v} \frac{\mathbf{h}_u^{k-1}}{\sqrt{|N(u)||N(v)|}}\right)$$

W_k: *weight matrix at Layer k, shared across different nodes*

A toy example of 2-layer GCN on a 4-node graph

- Computation graph



Question

- How many parameters are there in a GCN?
  - Assuming initial features are with d_0 dimensions
  - Representation in later layers are with d dimensions

GraphSAGE

- [Inductive Representation Learning on Large Graphs](#)
  William L. Hamilton*, Rex Ying*, Jure Leskovec, NeurIPS'17

$$\mathbf{h}^k_{\mathcal{N}(v)} \leftarrow \text{AGGREGATE}_k(\{\mathbf{h}^{k-1}_u, \forall u \in \mathcal{N}(v)\})$$

$$\mathbf{h}^k_v \leftarrow \sigma\left(\mathbf{W}^k \cdot \text{CONCAT}(\mathbf{h}^{k-1}_v, \mathbf{h}^k_{\mathcal{N}(v)})\right)$$

A more general form

$$\mathbf{h}^k_v = \sigma\left(\left[\mathbf{W}_k \cdot \text{AGG}\left(\{\mathbf{h}^{k-1}_u, \forall u \in N(v)\}\right), \mathbf{B}_k \mathbf{h}^{k-1}_v\right]\right)$$

More about AGG

- Mean $$\text{AGG} = \sum_{u \in N(v)} \frac{\mathbf{h}_u^{k-1}}{|N(v)|}$$

- LSTM $$\text{AGG} = \text{LSTM}\left(\left[\mathbf{h}_u^{k-1}, \forall u \in \pi(N(v))\right]\right)$$
  - $\pi(\cdot)$: *a random permutation*

- Pool $$\text{AGG} = \gamma\left\{\mathbf{Q}\mathbf{h}_u^{k-1}, \forall u \in N(v)\right\}$$
  - $\gamma(\cdot)$: Element-wise mean/max pooling of neighbor set

Message-Passing Neural Network

- Gilmer et al., 2017. Neural Message Passing for Quantum Chemistry. *ICML.*
- *A general framework that subsumes most GNNs*
  - Can also include edge information
- Two steps
  - Get messages from neighbors at step k

$$\mathbf{m}_v^k = \sum_{u \in N(v)} M(\mathbf{h}_u^{k-1}, \mathbf{h}_v^{k-1}, \mathbf{e}_{u,v})$$

e.g., Sum or MLP

  - Update the node latent represent based on the msg

$$\mathbf{h}_v^k = U(\mathbf{h}_v^{k-1}, \mathbf{m}_v^k)$$

e.g., LSTM, GRU

A special case: ***GGNN***, **Li et al., Gated graph sequence neural networks, ICLR 2015**

Graph Attention Network (GAN)

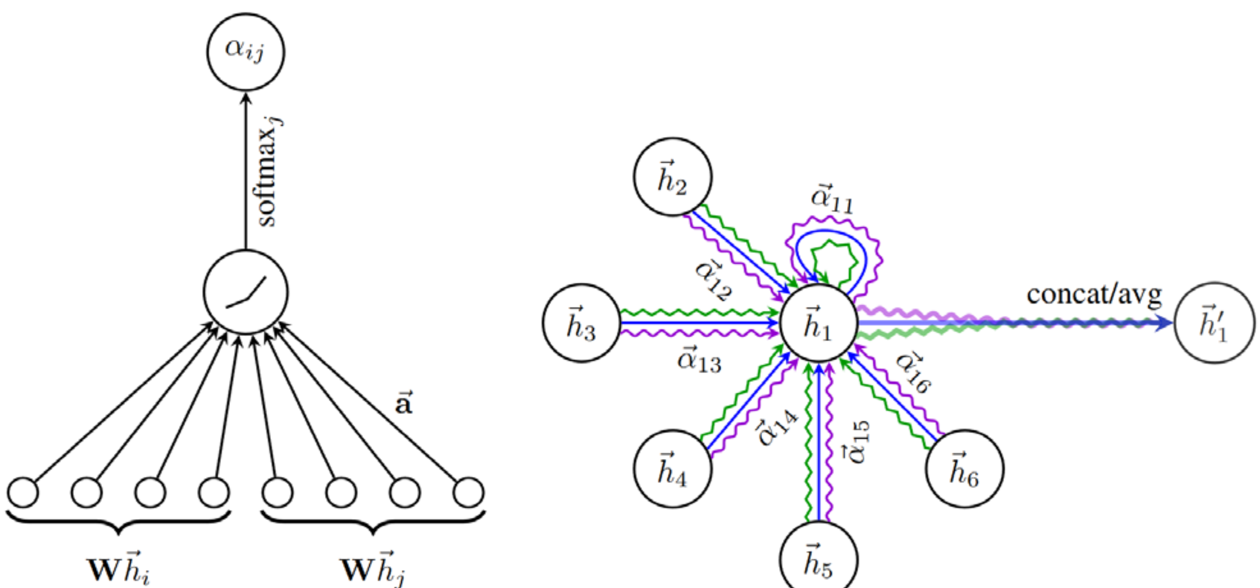
- How to decide the importance of neighbors?
  - GCN: a predefined weight
  - Others: no differentiation
- GAN: decide the weights using learnable attention
  - Velickovic et al., 2018. Graph Attention Networks. *ICLR.*

$$\vec{h}'_i = \sigma\left(\sum_{j \in \mathcal{N}_i} \boxed{\alpha_{ij}} \mathbf{W} \vec{h}_j\right)$$

The attention mechanism

- Potentially many possible designs

$$\alpha_{ij} = \frac{\exp\left(\text{LeakyReLU}\left(\vec{\mathbf{a}}^T[\mathbf{W}\vec{h}_i \| \mathbf{W}\vec{h}_j]\right)\right)}{\sum_{k\in\mathcal{N}_i}\exp\left(\text{LeakyReLU}\left(\vec{\mathbf{a}}^T[\mathbf{W}\vec{h}_i \| \mathbf{W}\vec{h}_k]\right)\right)}$$



Downstream Tasks for Graphs

Typical Graph Functions

- Node level
  - Similarity search
  - **Link prediction**
  - **Classification**
  - Community detection
  - Ranking

- Graph level
  - **Similarity search**
  - Frequent pattern mining
  - Graph isomorphism test
  - Graph matching
  - **Classification**
  - Clustering
  - Graph generation

1. Semi-supervised Node Classification

- Decoder using $z_v = h_v^L$
  - Feed into another fully connected layer
  - $\hat{y}_v = \sigma(\theta^T z_v)$
- Loss function
  - Cross entropy loss
  - In a binary classification case
    - $l_v = -y_v log \hat{y}_v - (1 - y_v)\log(1 - \hat{y}_v)$

Applications of Node Classification

- Social network
  - An account is bot or not
- Citation network
  - A paper's research field
- A program-derived graph
  - The type of a variable

2. Link Prediction

- Decoder using $z_v = h_v^L$
  - Given a node pair (u, v)
  - Determine its probability $p_{uv} = \sigma(z_u^T R z_v)$
  - R could be different for different relation type
- Loss function
  - Cross entropy loss
    - $l_{uv} = -y_{uv} log p_{uv} - (1 - y_{uv})\log(1 - p_{uv})$

Link Prediction Applications

- Social network
  - Friend recommendation
- Citation network
  - Citation recommendation
- Medical network
  - Drug and target binding or not
- A program-derived graph
  - Code autocomplete

3. Graph Classification

- Decoder using $h_G = g(\{z_v\}_{v \in V})$
  - $g(\cdot)$: $a\ read\ out\ function$, e.g., sum
  - Feed h_G into another fully connected layer
  - $\hat{y}_G = \sigma(\theta^T h_G)$
- Loss function
  - Cross entropy loss
  - In a binary classification case
    - $l_G = -y_G log \hat{y}_G - (1 - y_G)\log(1 - \hat{y}_G)$

Graph Classification Applications

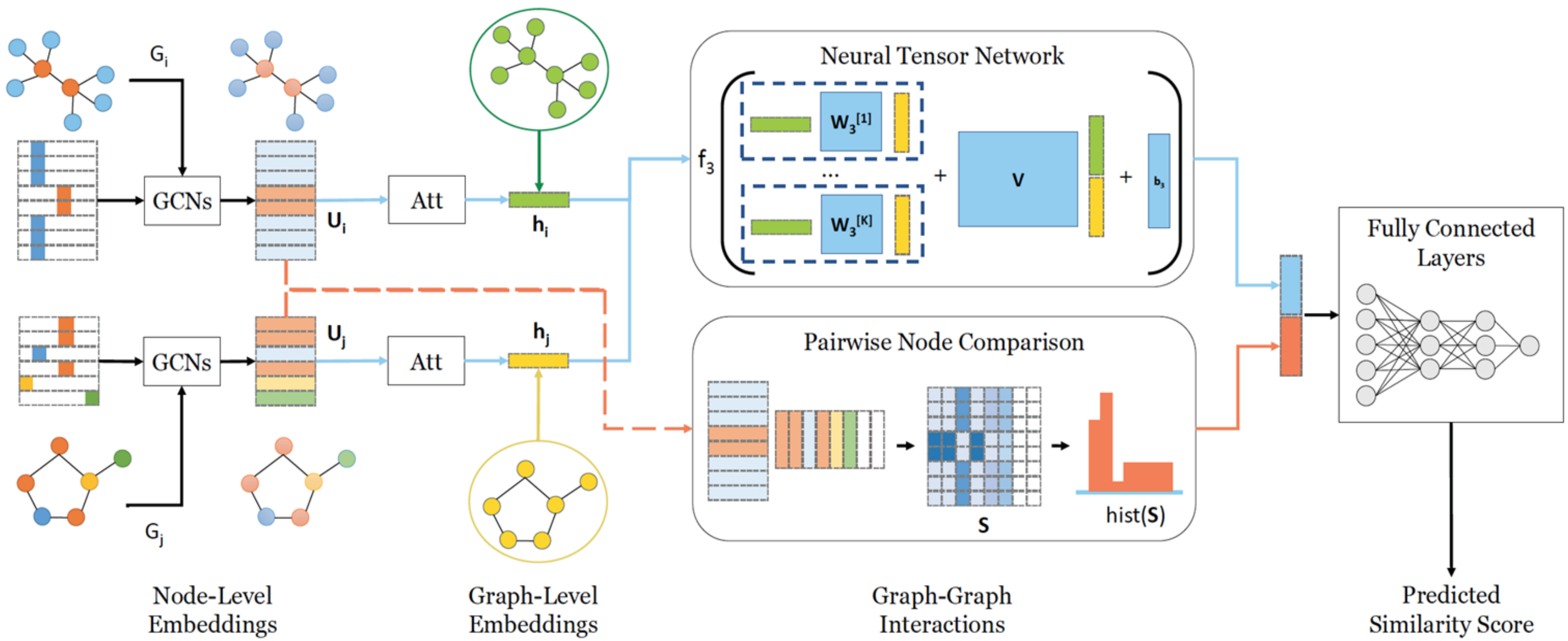
- Chemical compounds
  - Toxic or not
- Proteins
  - Has certain function or not
- Program-derived graphs
  - Contains bugs or not

4. Graph Similarity Computation

- Decoder using $h_G = g(\{z_v\}_{v \in V})$
  - Given a graph pair (G_1, G_2)
  - Determine its score $s_{G_1 G_2} = h_{G_1}^T R h_{G_2}$
- Loss function
  - E.g., Square loss
    - $l_{G_1 G_2} = (y_{G_1 G_2} - s_{G_1 G_2})$^2

A Concrete solution by SimGNN [Bai et al., AAAI 2019]

- Goal: learn a GNN $\phi: \mathcal{G} \times \mathcal{G} \rightarrow R^{+}$ to approximate Graph Edit Distance between two graphs



1. **Attention**-based graph-level embedding
2. **Histogram** features from pairwise node similarities

Graph Similarity Computation Applications

- Drug database
  - Drug similarity search
- Program database
  - Code recommendation
    - Search ninja code for novice code
    - Search java code for COBOL code

CNN BUSINESS

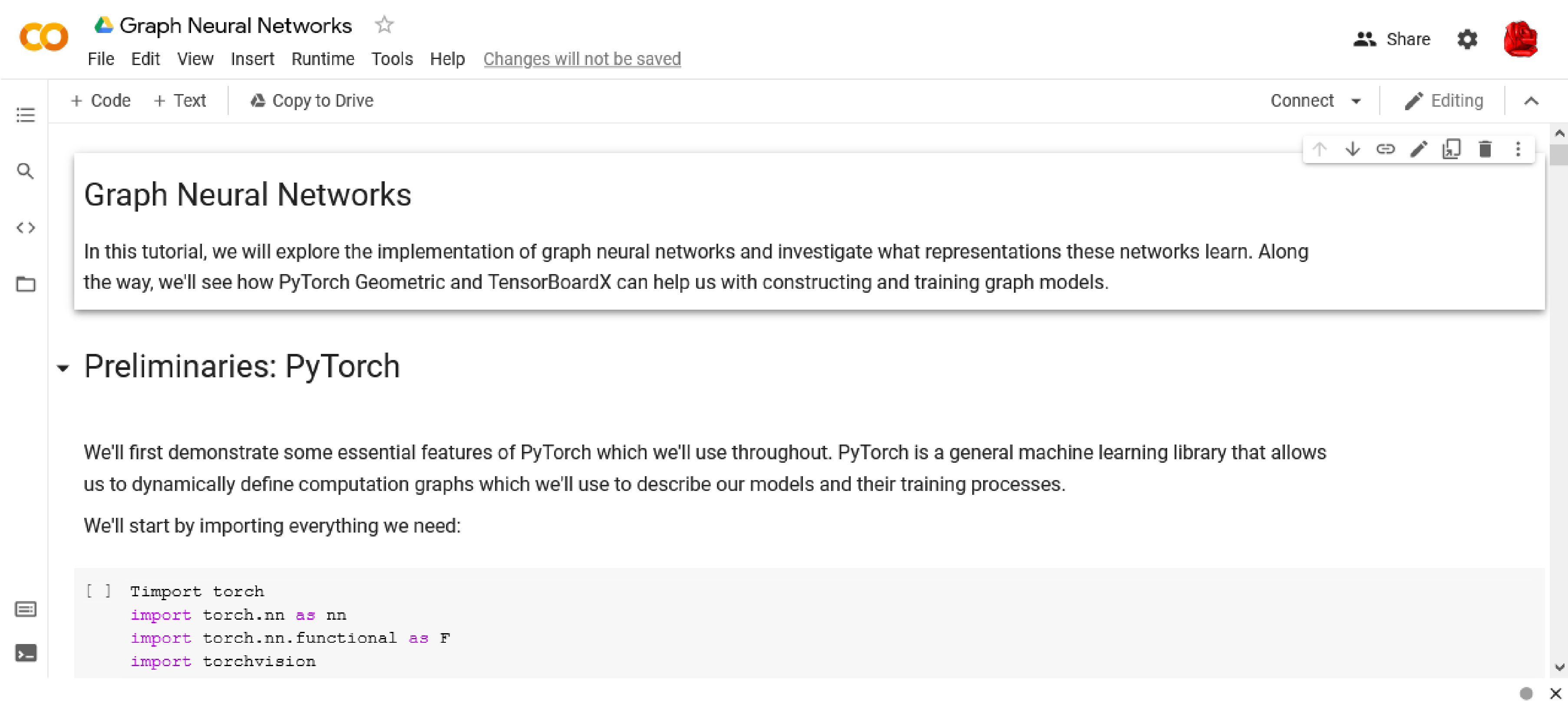
Wanted urgently: People who know a half century-old computer language so states can process unemployment claims

By Alicia Lee, CNN

Updated 4:00 PM ET, Wed April 8, 2020

Tutorial on GNN coding

- https://colab.research.google.com/drive/1DIQm9rOx2mT1bZETEeVUThxcrP1RKqAn



Graph Embedding

- What is Graph Embedding
- Shallow Network Embedding
- Knowledge Graph Embedding
- Graph Neural Networks
- Summary

Summary

- Graph embedding
- Shallow embedding
  - E.g., LINE
- Knowledge graph embedding
  - E.g., TransE
- Graph neural networks
  - E.g., GCN

References

- Bryan Perozzi, Rami Al-Rfou, Steven Skiena, DeepWalk: Online Learning of Social Representations, KDD'14
- Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, Qiaozhu Mei, LINE: Large-scale Information Network Embedding, WWW'15
- Aditya Grover, Jure Leskovec, node2vec: Scalable Feature Learning for Networks, KDD'16
- Kipf & Welling, Semi-Supervised Classification with Graph Convolutional Networks, ICLR 2017
- Tutorial: http://snap.stanford.edu/proj/embeddings-www/files/nrltutorial-part2-gnns.pdf
- Tutorial: http://tkipf.github.io/misc/SlidesCambridge.pdf
- Bordes et al., Translating Embeddings for Modeling Multi-relational Data, NIPS 2013