CS247: ADVANCED DATA MINING

Graph and Network: Graph Embedding

Instructor: Yizhou Sun

yzsun@ccs.neu.edu

May 18, 2020

Methods to Learn

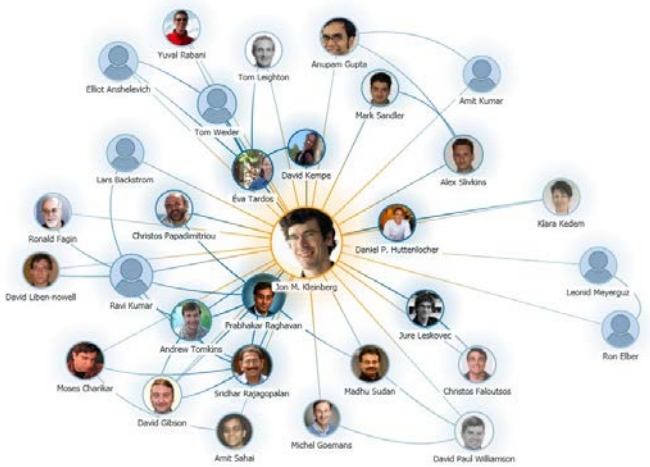
	Vector Data	Text Data	Graph & Network	Recommender Systems
Classification	**Naïve Bayes; Logistic Regression; NN**		**Label Propagation**	
Clustering	**K-means; kernel k-means; Mixture Models**	**PLSA; LDA**	**Spectral Clustering**	Matrix Factorization
Prediction	**NN**			Collaborative Filtering; Factorization machine; Hybrid CF; Recommendation with graph regularization
Ranking			**PageRank**	
Similarity Search			**P-PageRank**	
Representation Learning		**Word embedding**	**Network embedding**	Deep collaborative learning

Graph Embedding

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

How to represent nodes?

- A naïve solution

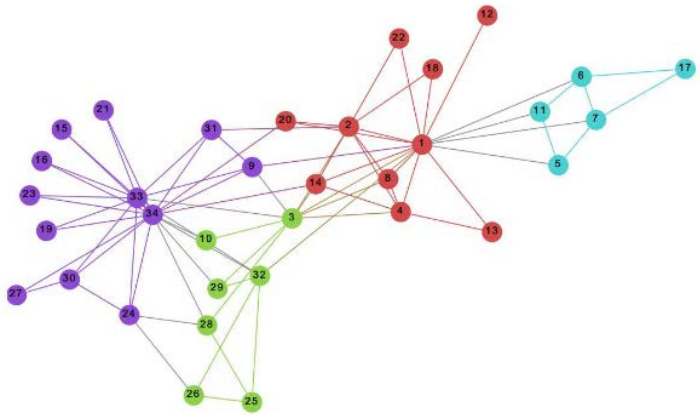


	A	B	C	D	E	F
A	0	1	1	1	0	0
B	1	0	0	0	1	1
C	1	0	0	0	0	1
D	1	0	0	0	0	0
E	0	1	0	0	0	0
F	0	1	1	0	0	0

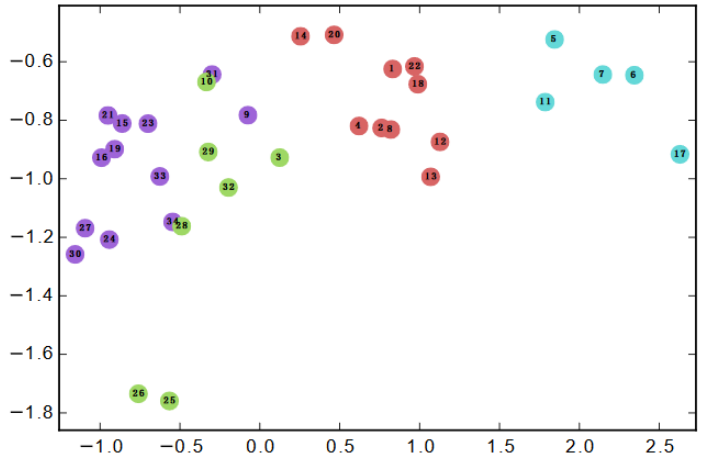
- Limitations:
  - Extremely High-dimensional
  - No global structure information integrated
  - Permutation-variant

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

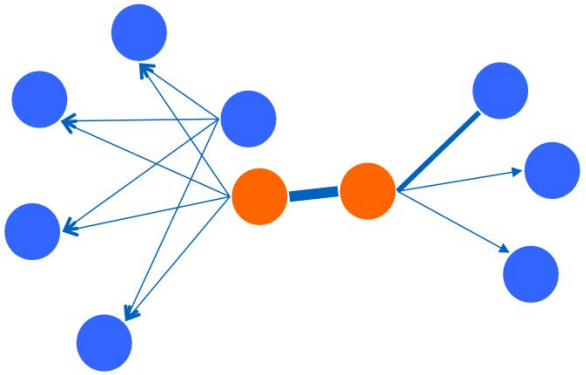
- What is Graph Embedding
- Shallow Network Embedding 
- Graph Convolution Network
- Knowledge Graph Embedding
- Summary

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

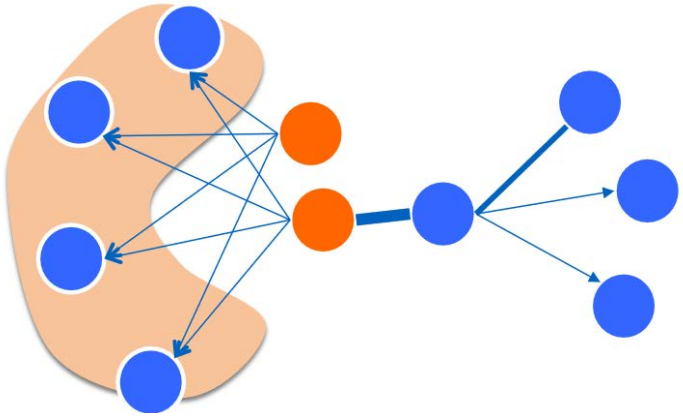
$$\hat{p}_1(v_i, v_j) = \frac{w_{ij}}{\sum_{(m,n)\in E} w_{mn}}$$

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

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

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

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

Graph Embedding

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

Limitation of Shallow Network Embedding

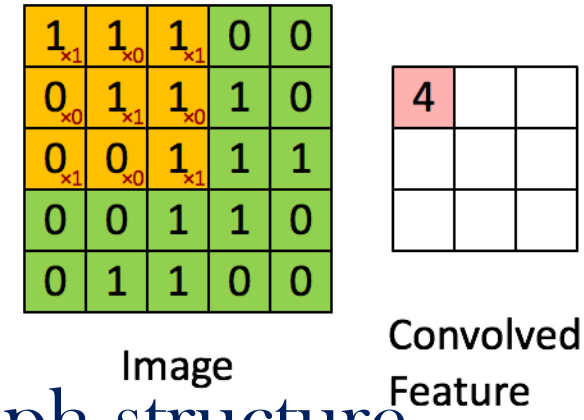
- Too many parameters
  - Each node is associated with an embedding vector, which are parameters
- Not inductive
  - Cannot handle new nodes
- Cannot handle node attributes

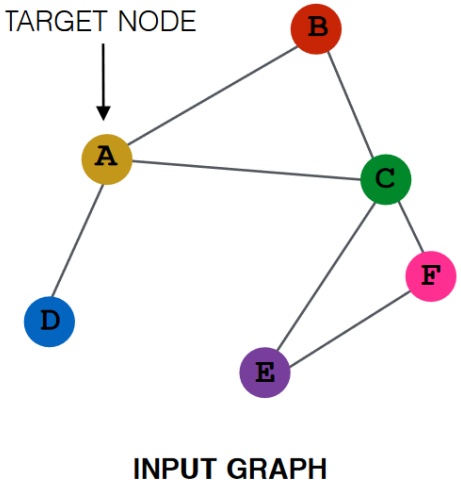
From shallow embedding to Graph Neural Networks

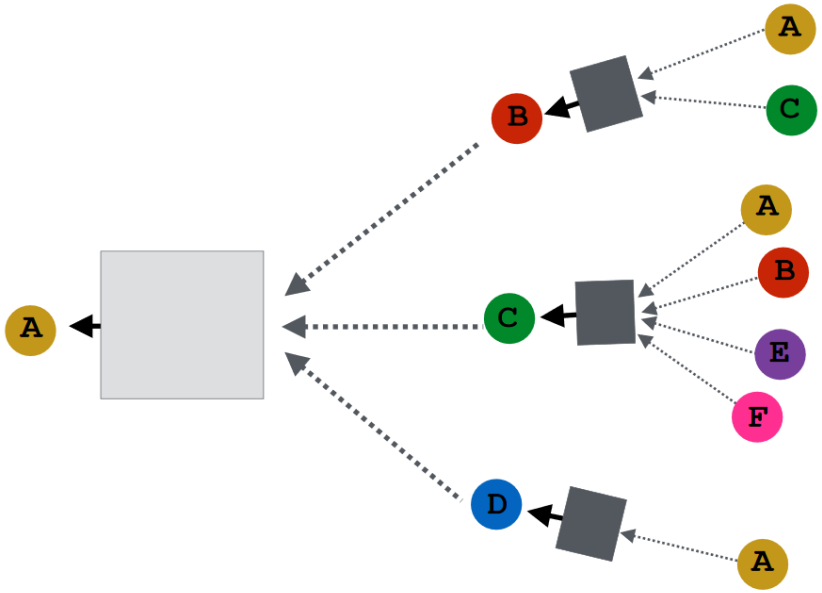
- The embedding function (encoder) is more complicated
  - Shallow embedding
    - $\phi(v) = U^T x_v$, where U is the embedding matrix and x_v is the one-hot encoding vector
  - Graph neural networks
    - $\phi(v)$ is a neural network depending on the graph structure

Graph Convolutional Network

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







Set up

- An attributed graph G=(V,E)
  - A: adjacency matrix
  - X: feature matrix for all the nodes
  - N(v): degree of node v
- Representation vector at Layer l
  - h_v^l

Each GCN Layer

- Neighbor Aggregation
  - Aggregate neighbors' representations
- Nonlinear transformation
  - Transform the representation

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

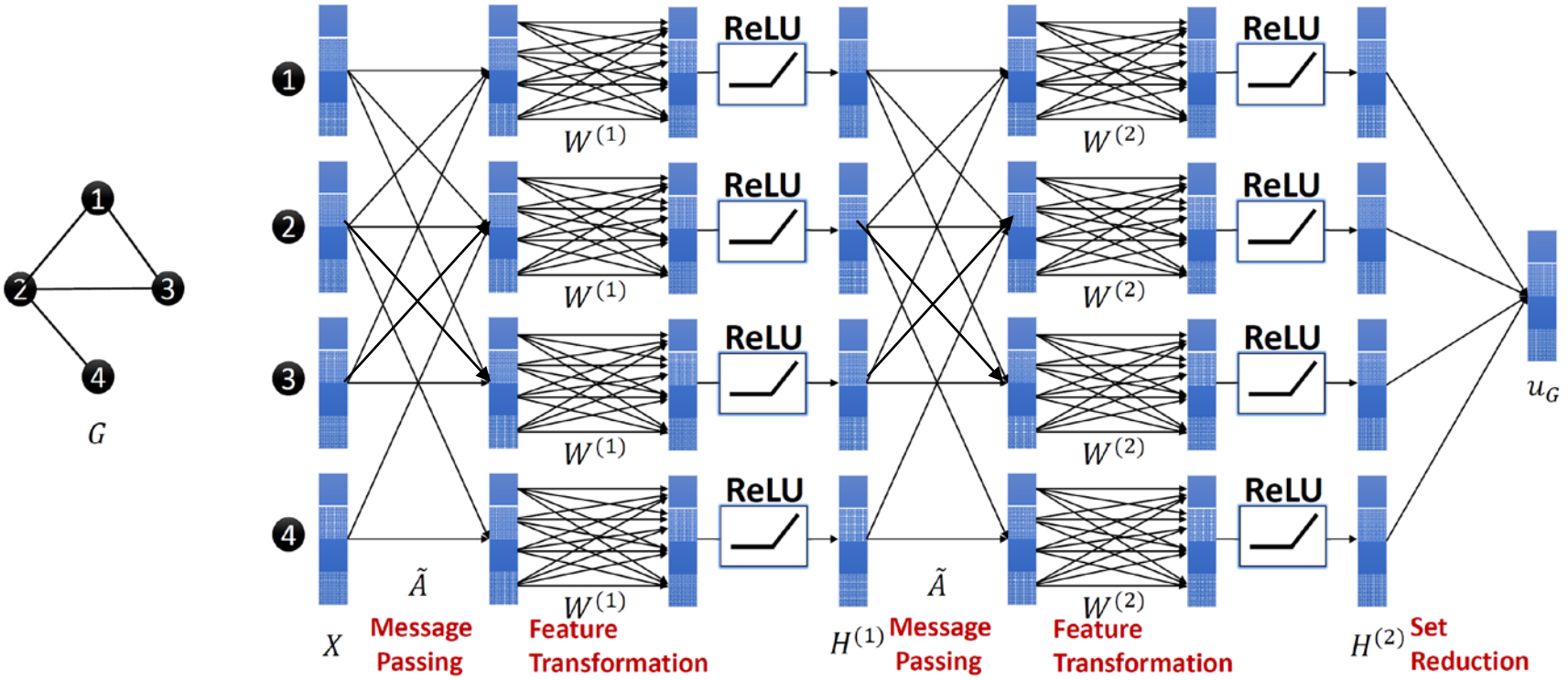
W_k: *weight matrix at Layer k, shared across different nodes*

Illustration

- Matrix form $\quad \hat{A} = A + I, \hat{D}: diagonal\ matrix\ of\ \hat{A}$

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

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



How to train a GCN?

- Let z_v be the final layer output for node v
- Define a task and the determine the loss
  - Node classification
    - Softmax(z_v), then cross entropy loss
  - Link prediction
    - $\sigma(z_u^T z_v)$ for link (u,v), then binary cross entropy loss
  - Graph classification
    - Softmax($aggregation\ (\{z_v\}_{\{v \in V\}})$)

Question

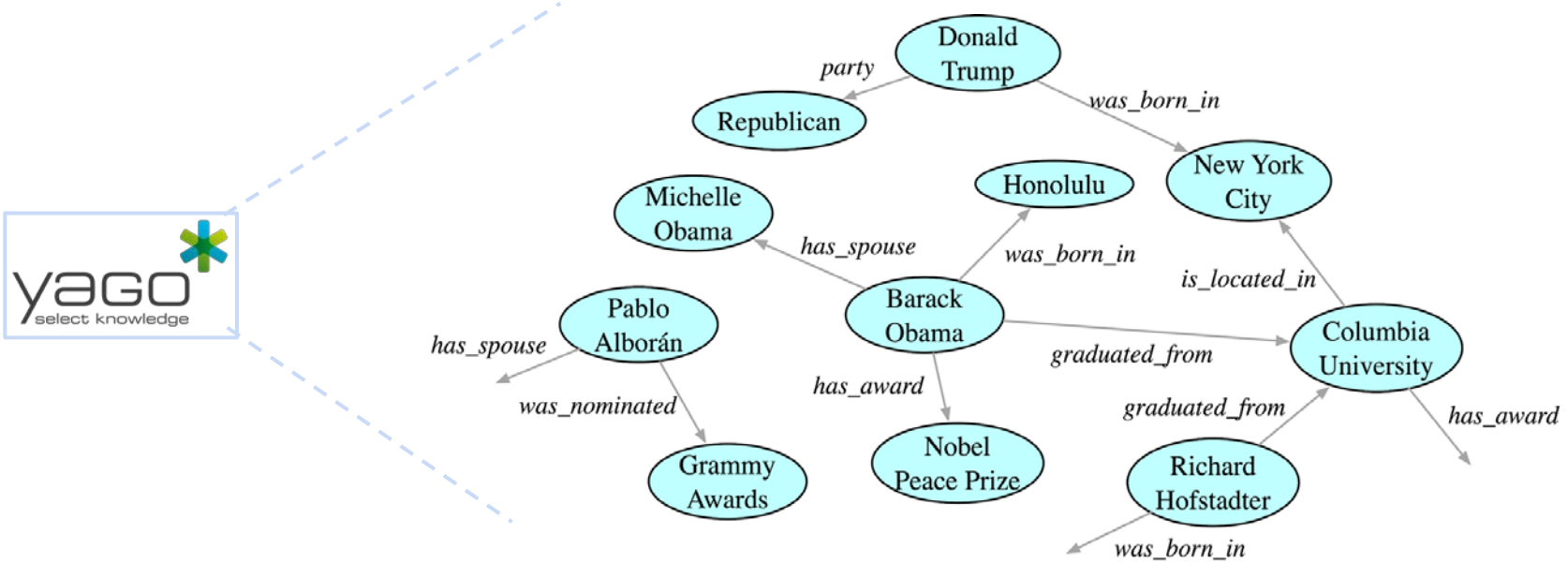
- How many parameters are there in a GCN?
  - Assuming initial features are with d_0 dimensions
  - Representation in later layers are with d dimensions

Graph Embedding

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

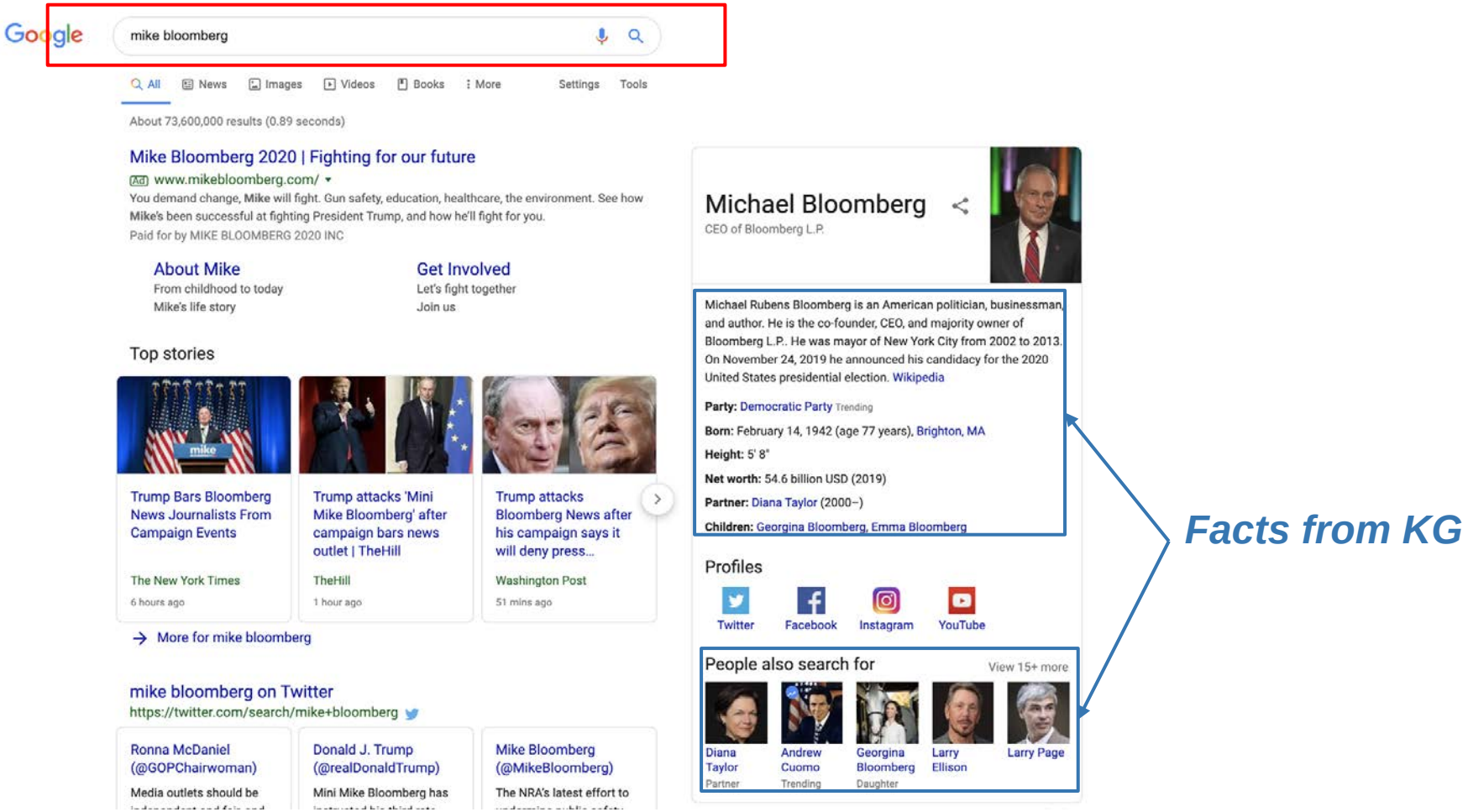
An example of Knowledge Graph

- A set of triples (h, r, t), which form a graph



Knowledge Graph Application

- When you search in Google



KGs are everywhere

General-purpose KGs

Bio & Medical KGs

Product Graphs & E-commerce

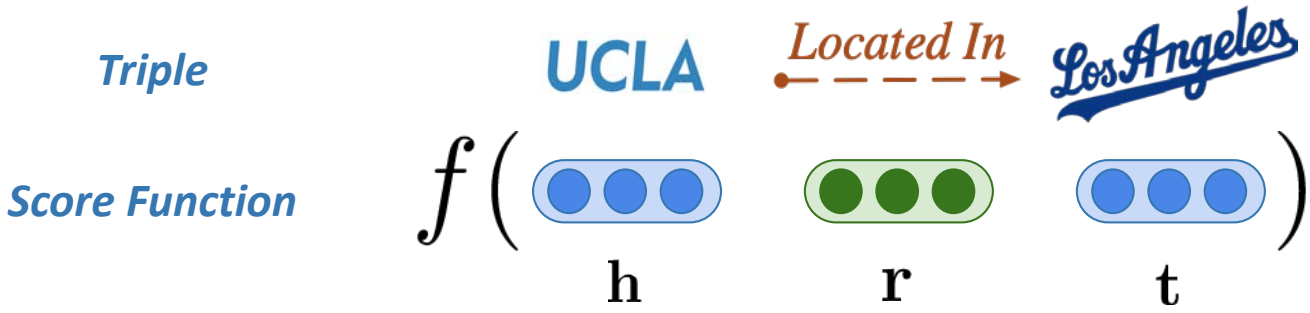
Common-sense KGs & NLP

Knowledge Graph Embedding

- Goal: represent entities and relations as latent vectors or matrices and support effective relation learning and inference
  - Input: Relation facts (triples)
  - Output: Embedding representations of objects and relations

KG embedding algorithms

- Key idea
  - Design a score function for each triple using embeddings



  - The score for a positive triple should be higher than a negative triple
    - Define loss accordingly

Popular approaches

- Differ in score function design

Model	Score Function	Embeddings
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)	$(\mathbf{h}\circ\mathbf{t})\cdot\mathbf{r}$	$\mathbf{h},\mathbf{r},\mathbf{t}\in\mathbb{R}^k$
HolE (Nickel et al., 2016)	$(\mathbf{h}\star\mathbf{t})\cdot\mathbf{r}$	$\mathbf{h},\mathbf{r},\mathbf{t}\in\mathbb{R}^k$
ComplEx (Trouillon et al., 2016)	$\mathrm{Re}\langle\mathbf{r},\mathbf{h},\bar{\mathbf{t}}\rangle$	$\mathbf{h},\mathbf{r},\mathbf{t}\in\mathbb{C}^k$
ConvE (Dettmers et al., 2017)	$\langle\sigma(\mathrm{vec}(\sigma([\mathbf{r},\mathbf{h}]*\Omega))\mathbf{W}),\mathbf{t}\rangle$	$\mathbf{h},\mathbf{r},\mathbf{t}\in\mathbb{R}^k$
RotatE (Sun et al., 2019)	$-\|\|\mathbf{h}\circ\mathbf{r}-\mathbf{t}\|\|^2$	$\mathbf{h},\mathbf{r},\mathbf{t}\in\mathbb{C}^k, \|r_i\|=1$

More Details on TransE

- Objective Function
  - Margin-based ranking loss

$$\mathcal{L} = \sum_{(h,\ell,t)\in S} \sum_{(h',\ell,t')\in S'_{(h,\ell,t)}} \left[\gamma + d(\boldsymbol{h}+\boldsymbol{\ell},\boldsymbol{t}) - d(\boldsymbol{h'}+\boldsymbol{\ell},\boldsymbol{t'})\right]_+$$

    - $[x]_+$ 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

Graph Embedding

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

Summary

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

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