COMP4222 Machine Learning with Structured Data

Introduction

Yangqiu Song

Slides credits: Jure Leskovec @Stanford

Logistics

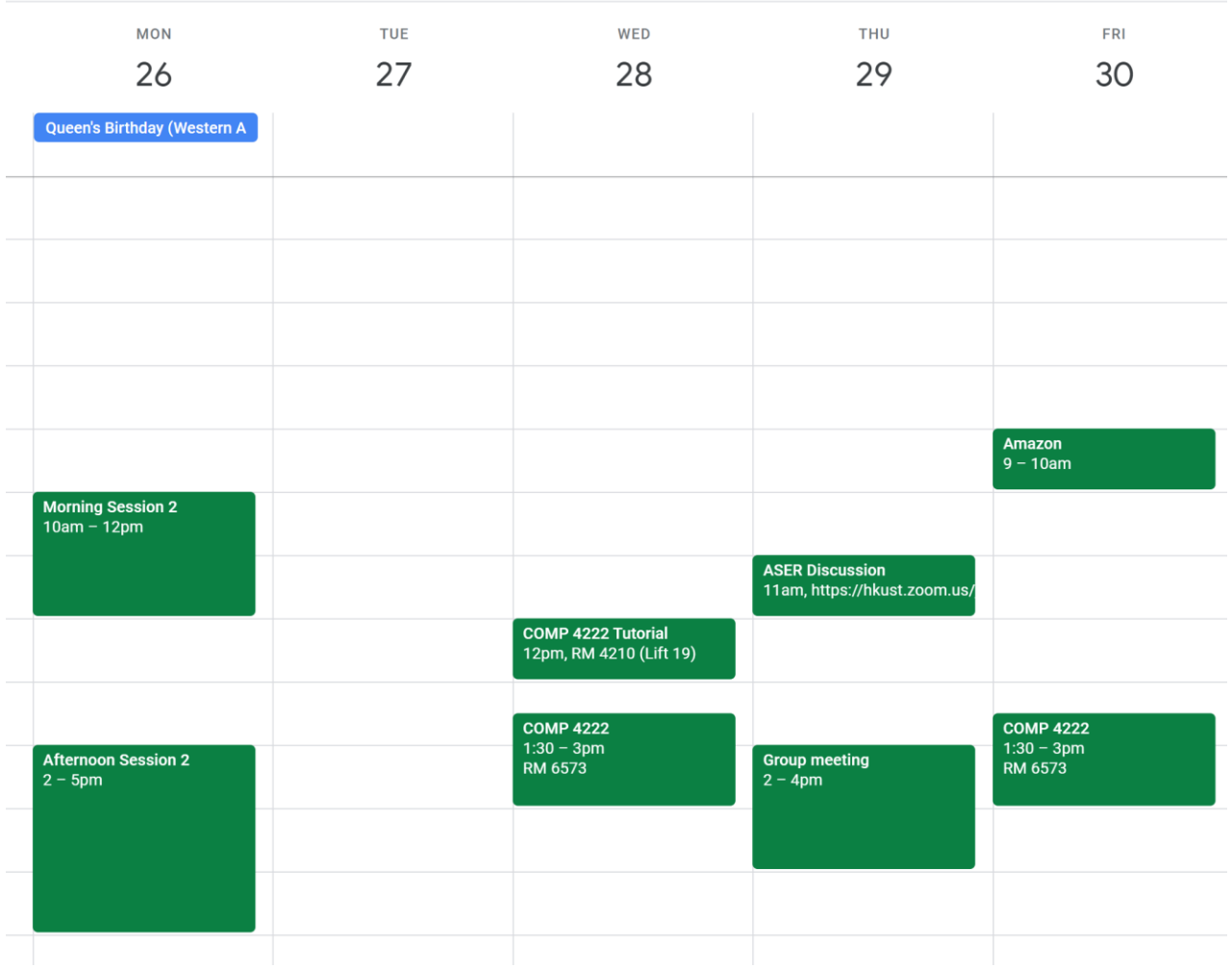
- Instructor: Yangqiu Song yqsong@cse.ust.hk
- TAs:
  - Jiayang Cheng jchegaj@connect.ust.hk
  - Qi Hu qhuaf@connect.ust.hk
- Canvas (https://canvas.ust.hk)
  - Lecture notes
  - Assignments and Projects

Background of this Course

- Originally, I taught COMP4901K: Machine Learning for NLP
- However
  - It's overlapped with COMP4221: NLP
  - My personal interests changed to knowledge graphs
- So this version will be focused on structured data such as graphs

Office Hours

- My typical week day meetings
- You need to book a meeting
  - All OHs will be zoom based



Course Grading

- Reading notes: 15%
  - Read three papers in different topics
  - Point out the pros and cons of the methods proposed in the paper
- Group based project: 35
  - Each group can have up to three students
  - You need to submit the code before the first deadline
  - Then you have some time to finish your final report after presentations
- Project presentation: 10%
  - Share your insights to all other students (plan in the last 2-3 weeks)
- Final Exam: 40%
  - Open book, but you cannot access the Internet

Canvas

- We will setup deadlines for assignments and project submission
  - Late submission will receive a penalty
- For the project, you can leverage the code from github, but you have to clearly show what is new you introduced to the project
- Plagiarism
  - Canvas will check the duplicated content, so please complete by yourselves

Tutorials

- All tutorials will be zoom based

- Our TAs will introduce
  - Some fundamental machine learning algorithms that will be essential for our course
  - Some sampled code

Prerequisites

- We will cover many topics and this is what makes the course hard
- It’s better to have good knowledge about
  - Machine learning
  - Algorithms and graph theory
  - Probability and statistics
- Programming
  - Familiar with Python
  - Being able to use PyTorch or other deep learning packages will be a plus

Existing Graph Learning Tools

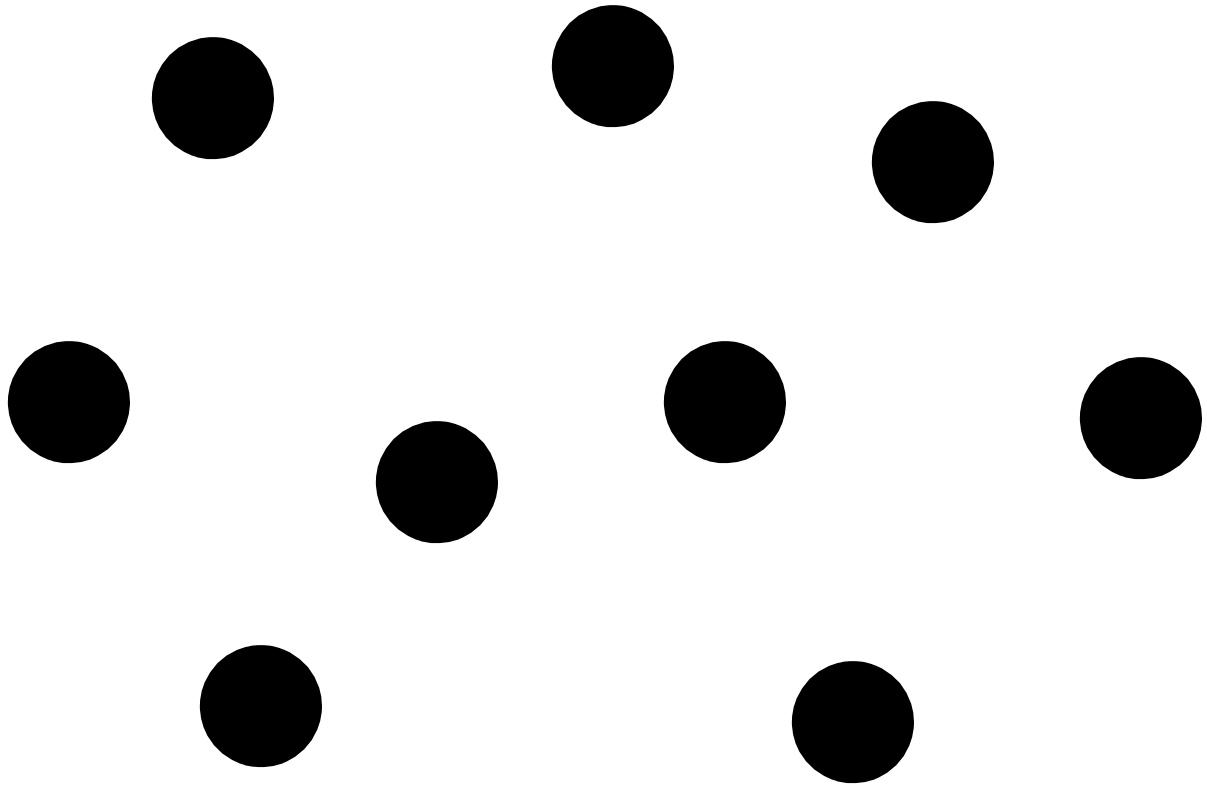
- PyG (PyTorch Geometric):
  - The ultimate library for Graph Neural Networks
- GraphGym: Platform for designing Graph Neural Networks
  - Modularized GNN implementation, simple hyperparameter tuning, flexible user customization
- DGL:
  - Deep graph learning library developed by Amazon

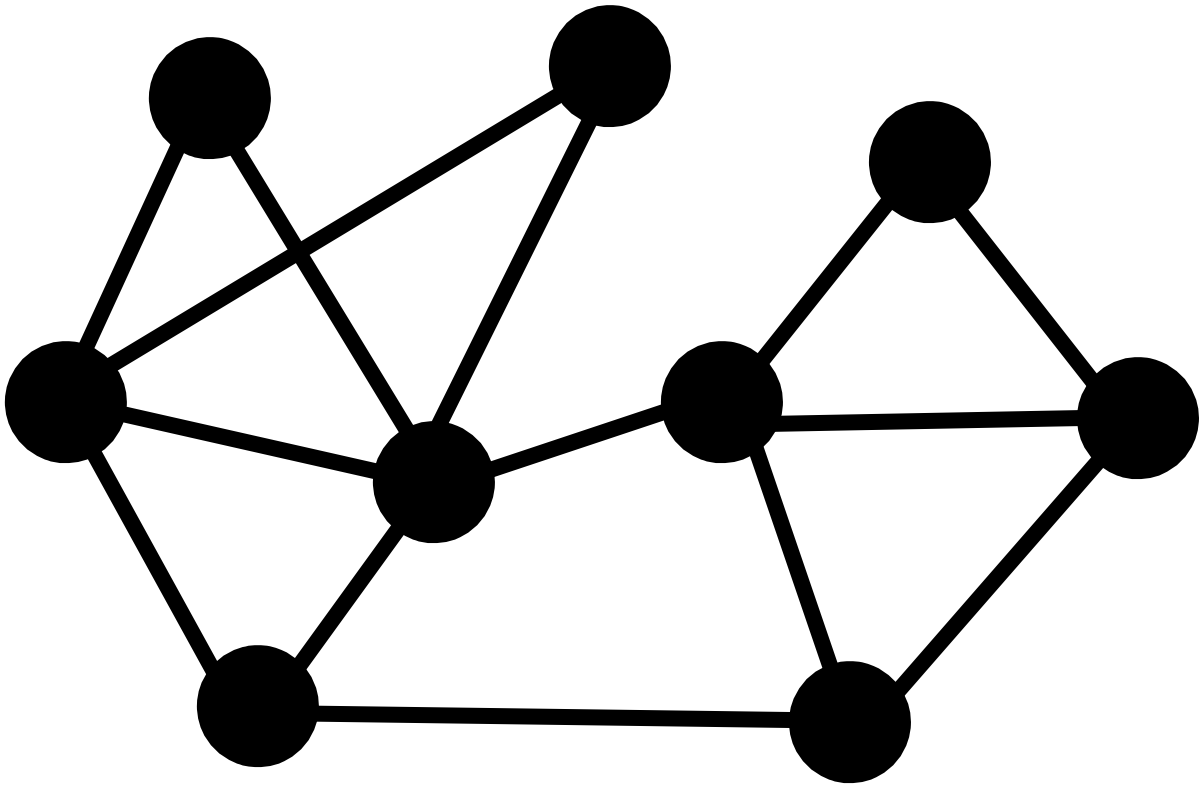
Computational Resources

- COMP students cluster for teaching
  - Will be announced later
- Kaggle notebook:
  - 12 hours execution time for CPU and GPU notebook sessions and 9 hours for TPU notebook sessions
  - https://www.kaggle.com/docs/notebooks
- Google Cloud
  - 90-day, $300 Free Trial
  - https://cloud.google.com/free/docs/free-cloud-features

Why Graphs

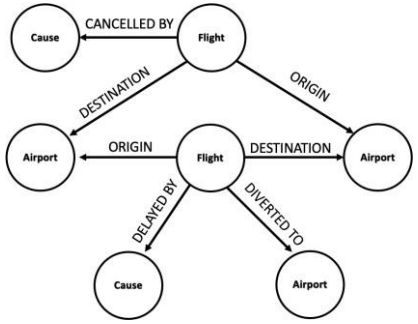
- Graphs are a general language for describing and analyzing entities with relations/interactions





Graph

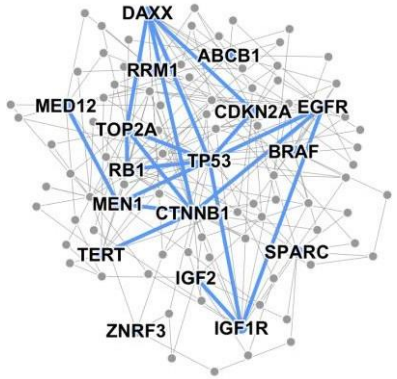
Many Types of Data are Graphs



Event Graphs

Image credit: SalientNetworks

Computer Networks



Disease Pathways

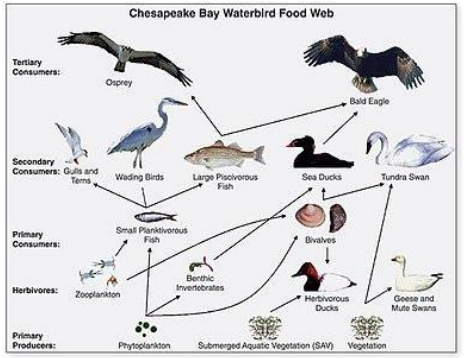


Image credit: Wikipedia

Food Webs

Image credit: Pinterest

Particle Networks

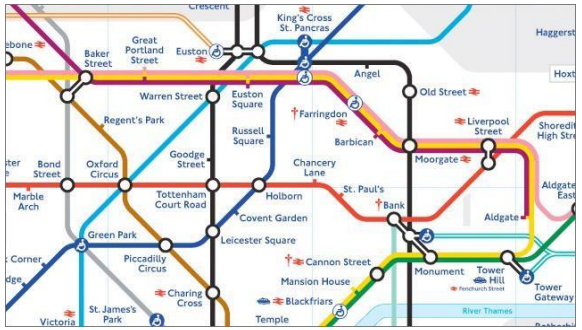


Image credit: visitlondon.com

Underground Networks

Many Types of Data are Graphs

Image credit: Medium

Social Networks

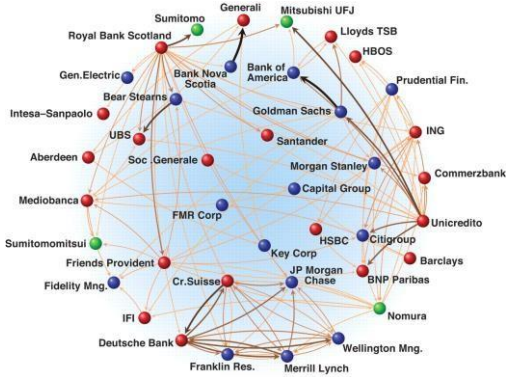
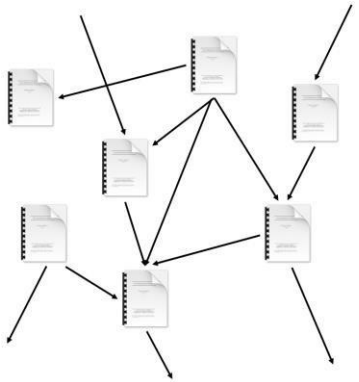


Image credit: Science

Economic Networks

Image credit: Lumen Learning

Communication Networks



Citation Networks

Image credit: Missoula Current News

Internet

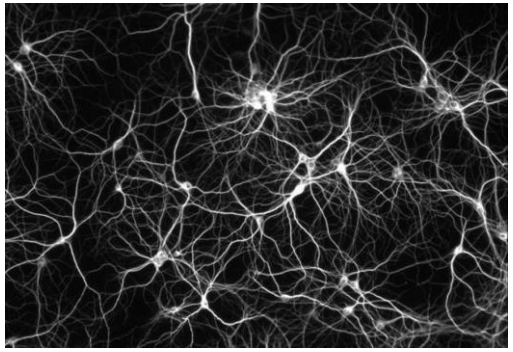


Image credit: The Conversation

Networks of Neurons

Many Types of Data are Graphs

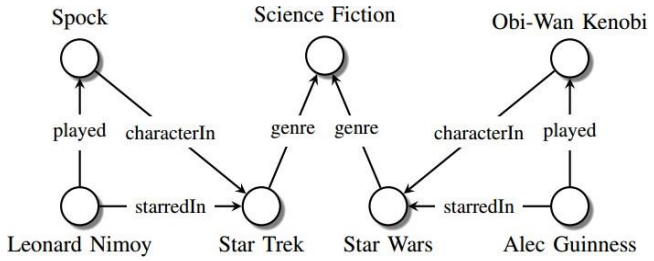


Image credit: Maximilian Nickel et al

Knowledge Graphs

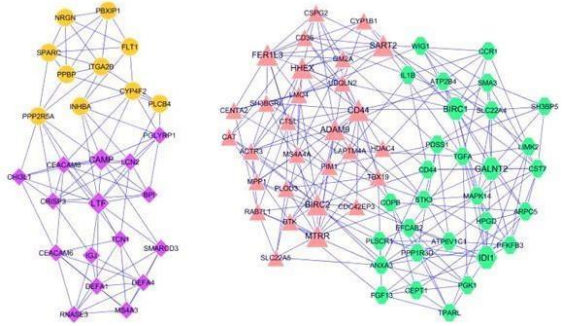


Image credit: ese.wustl.edu

Regulatory Networks

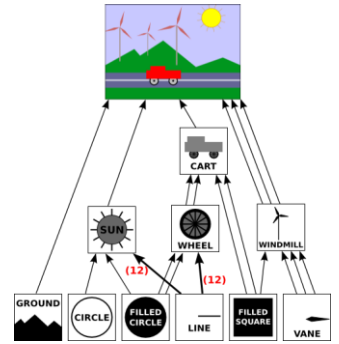


Image credit: math.hws.edu

Scene Graphs

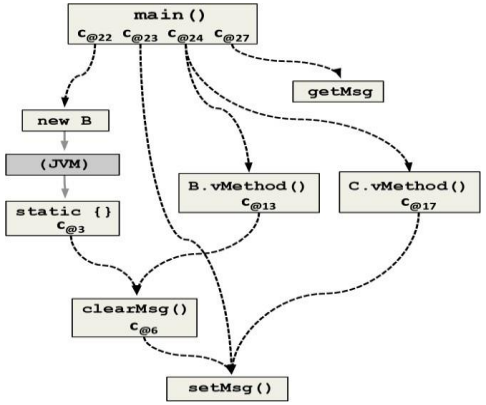


Image credit: ResearchGate

Code Graphs

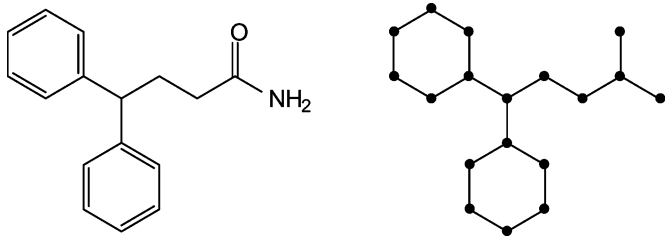


Image credit: MDPI

Molecules

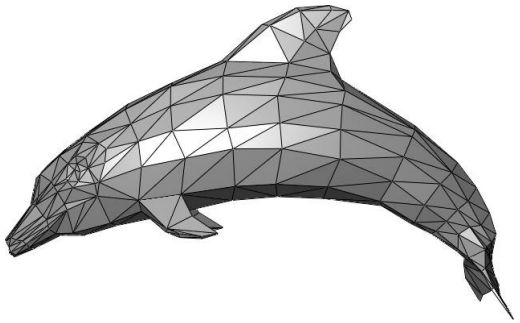


Image credit: Wikipedia

3D Shapes

Main Question

- How do we take advantageof relational structure for better prediction?

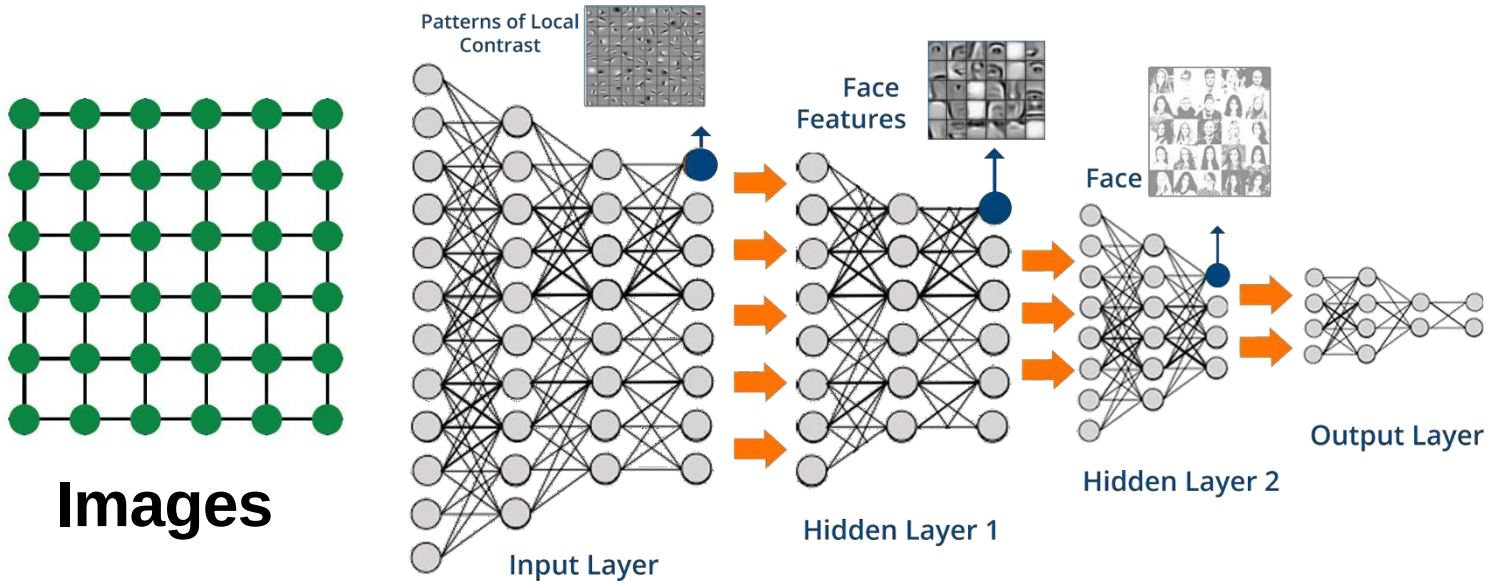
Machine Learning from/for Graphs

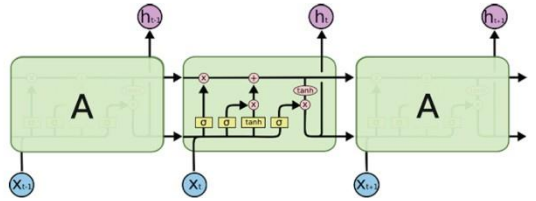
- Complex domains have a rich relational structure, which can be represented as a **relational graph**

- Many tasks on graphs can benefit from learning
  - By explicitly modeling relationships we achieve better performance!

Modern Machine Learning

- Modern deep learning toolbox is designed for simple sequences & grids

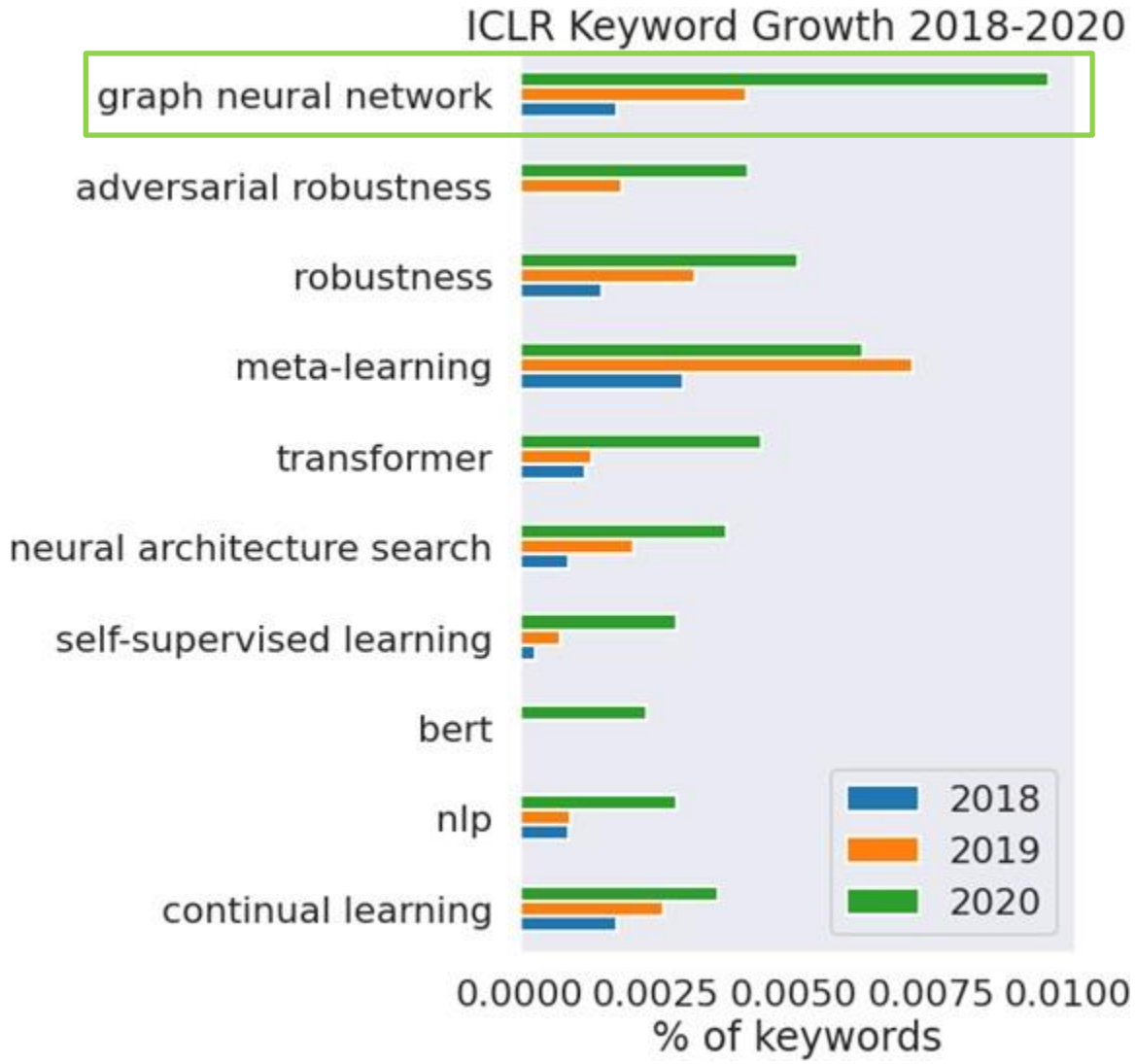




- Not everything can be represented as a sequence or a grid

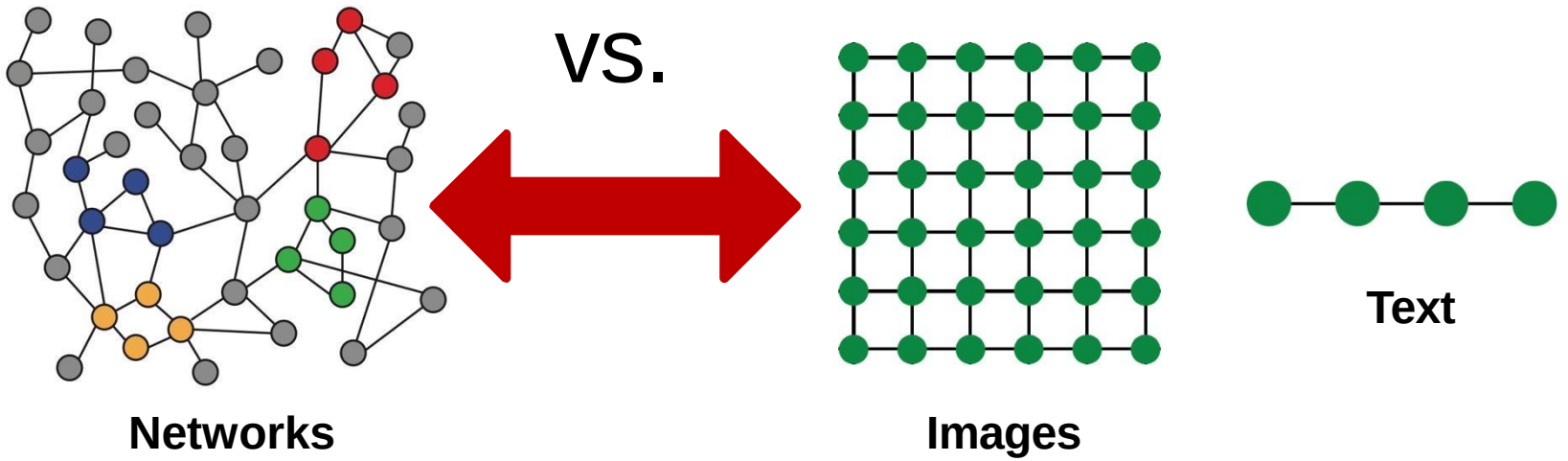
- How can we develop neural networks that are much more broadly applicable?

The Hottest Subfield in Machine Learning



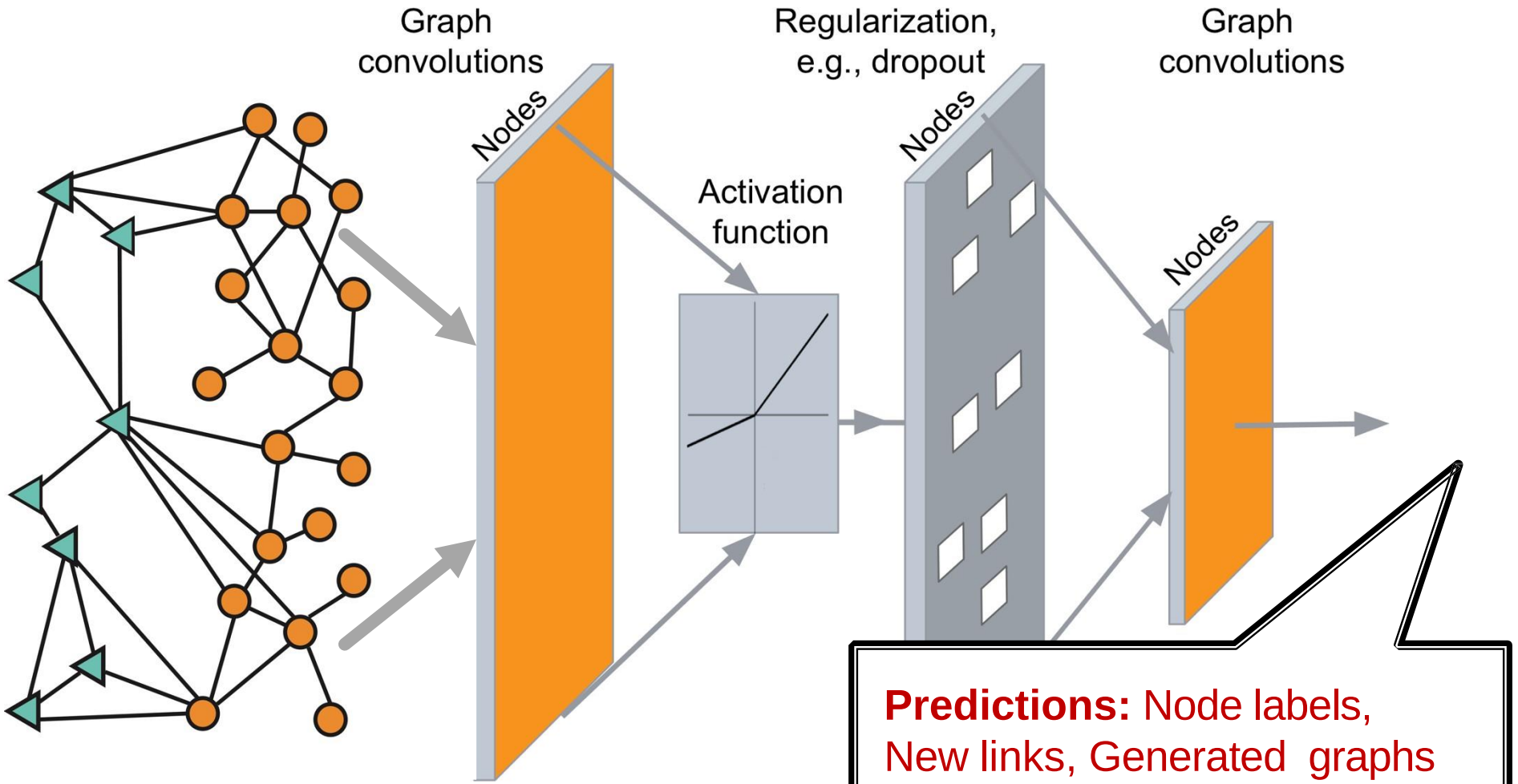
Why is Graph Deep Learning Hard?

- Networks are complex
  - Arbitrary size and complex topological structure (*i.e.*, no spatial locality like grids)



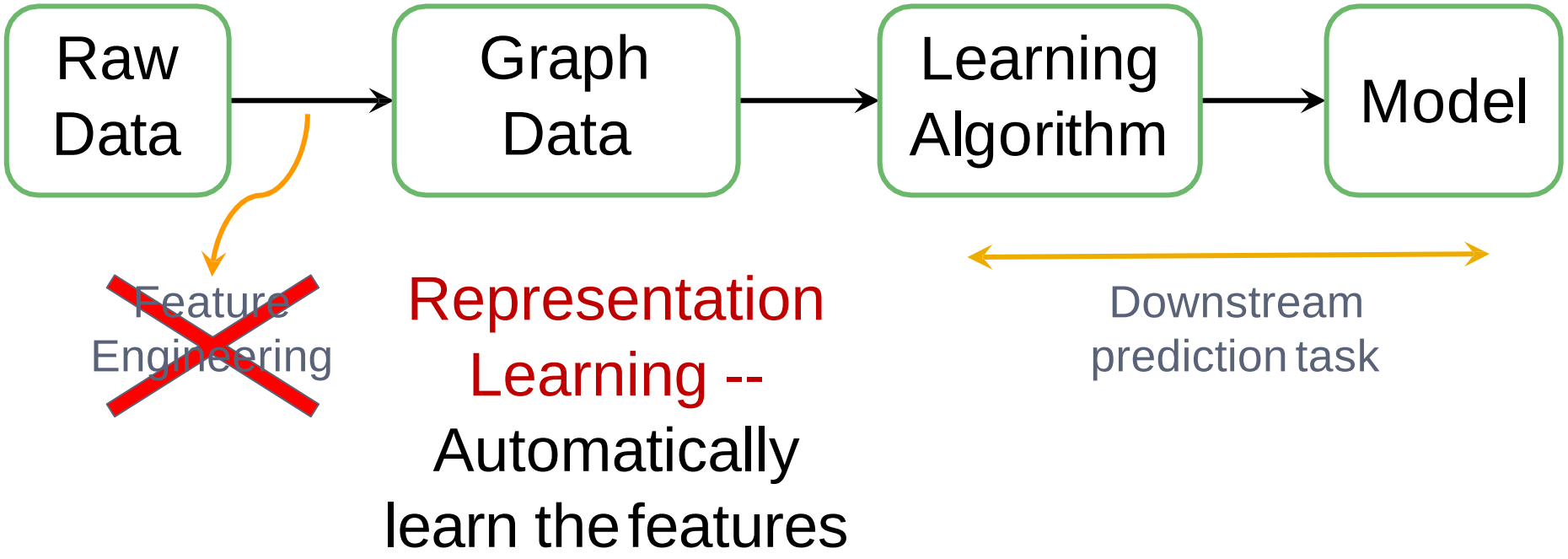
  - No fixed node ordering or reference point
  - Often dynamic and have multimodal features

Deep Learning on Graphs



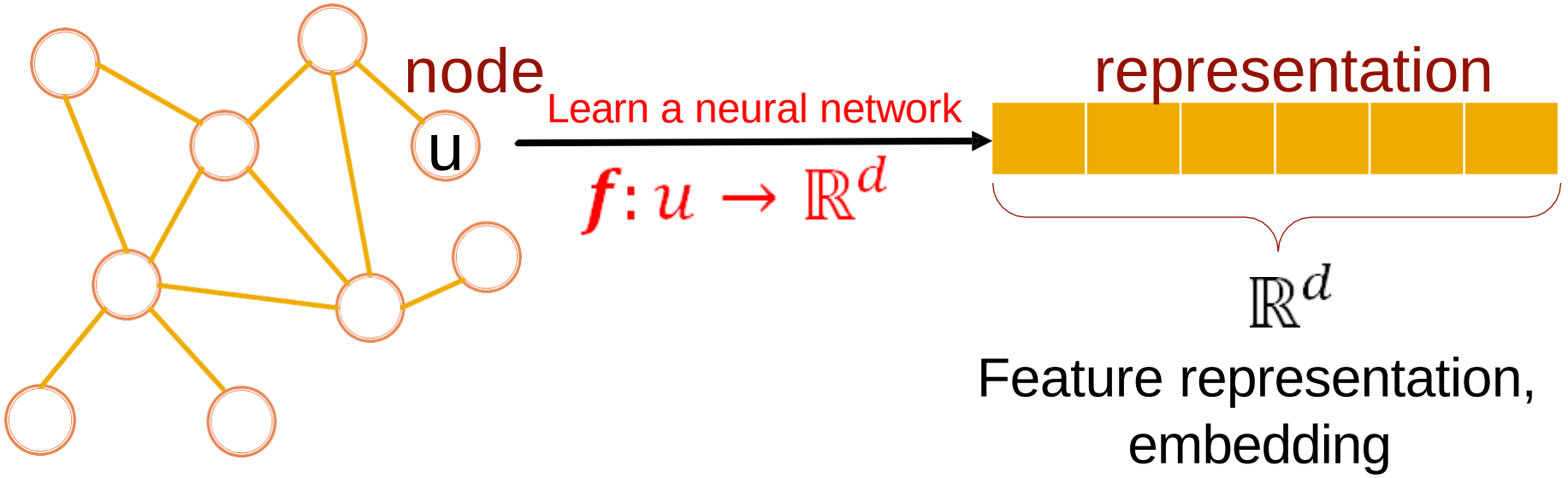
Essentially, Representation Learning

- (Supervised) Machine Learning Lifecycle:
  - This feature, that feature. Every single time!



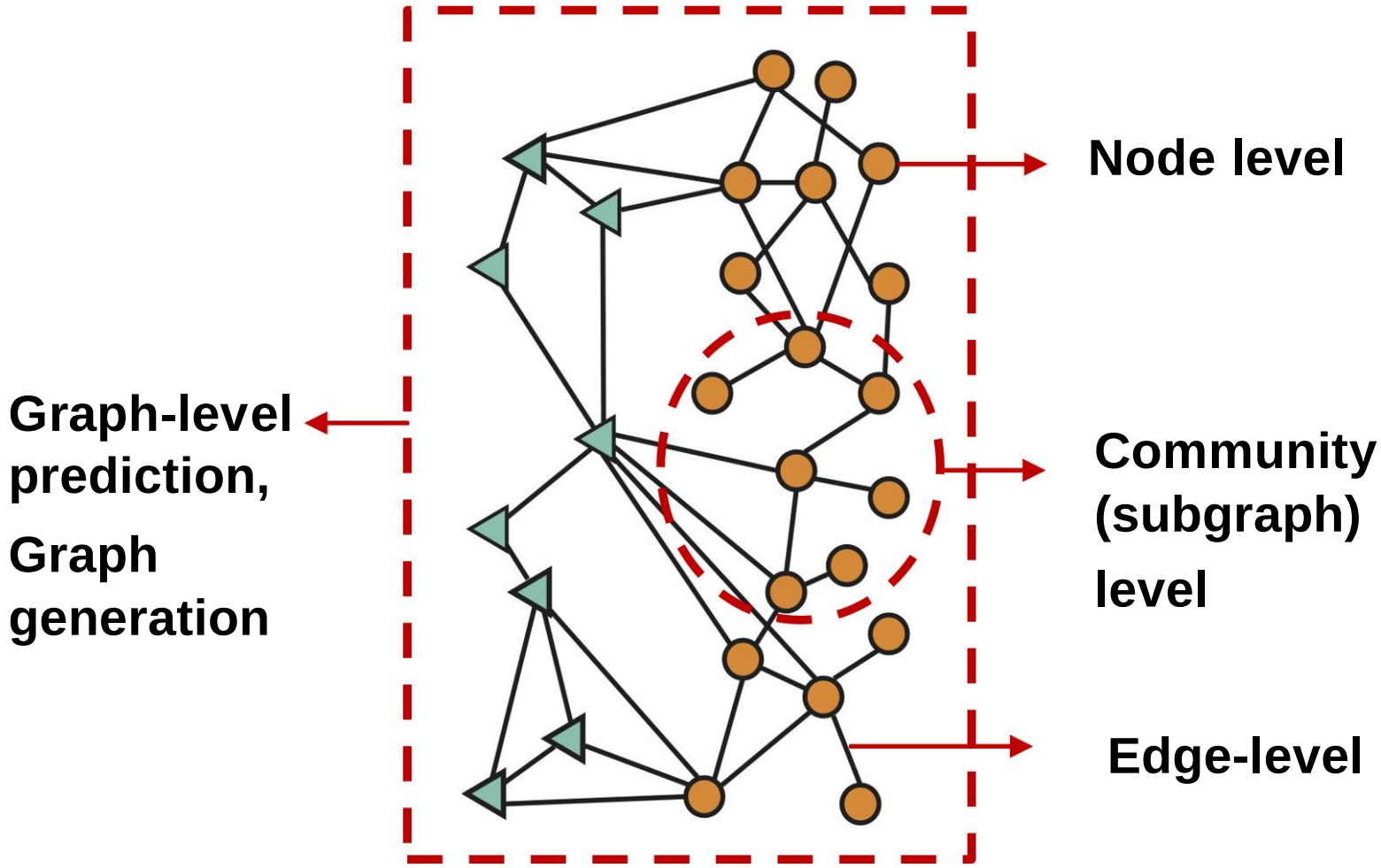
Essentially, Representation Learning

- Map nodes to d-dimensional **embeddings** such that similar nodes in the network are embedded close together



Applications of Graph ML

Different Types of Tasks



Classic Graph ML Tasks

- Node classification: Predict a property of a node
  - Example: Categorize online users / items
- Link prediction: Predict whether there are missing links between two nodes
  - Example: Knowledge graph completion
- Graph classification: Categorize different graphs
  - Example: Molecule property prediction
- Clustering: Detect if nodes form a community
  - Example: Social circle detection
- Other tasks:
  - Graph generation: Drug discovery
  - Graph evolution: Physical simulation

Example 1 (Node Level): Protein Folding

- A protein chain acquires its native 3D structure

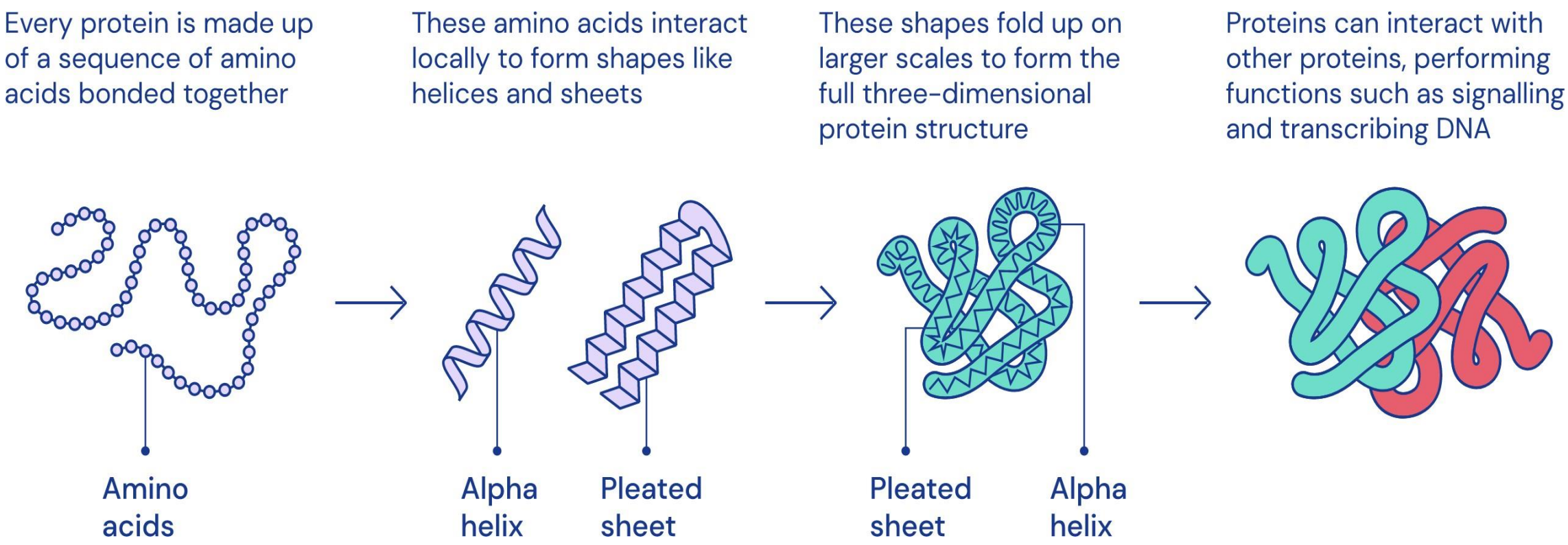


Image credit: DeepMind

The Protein Folding Problem

- Computationally predict a protein's 3D structure based solely on its amino acid sequence

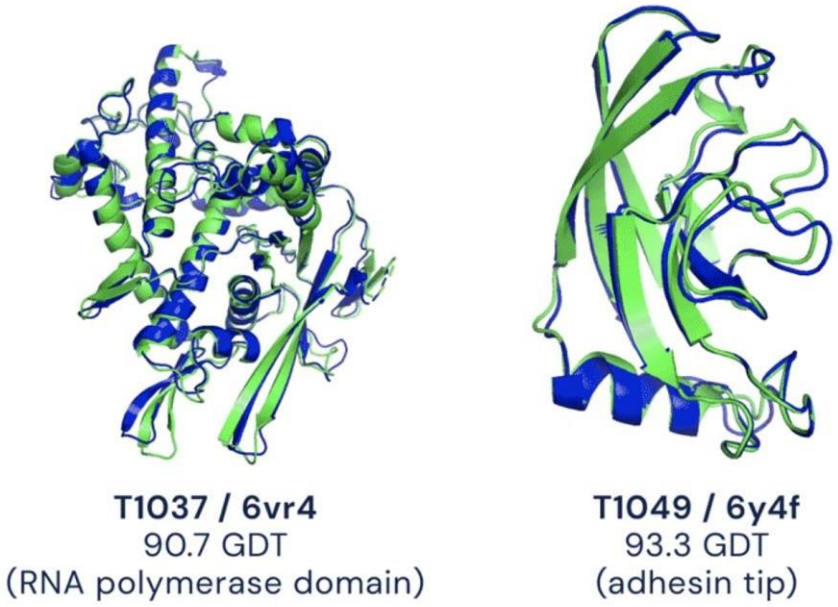


Image credit: DeepMind

AlphaFold by DeepMind

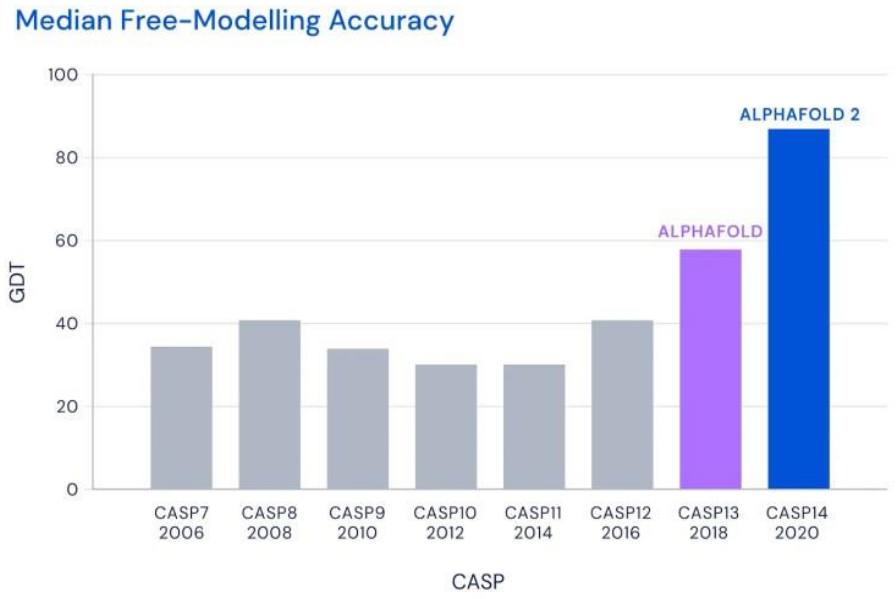


Image credit: DeepMind

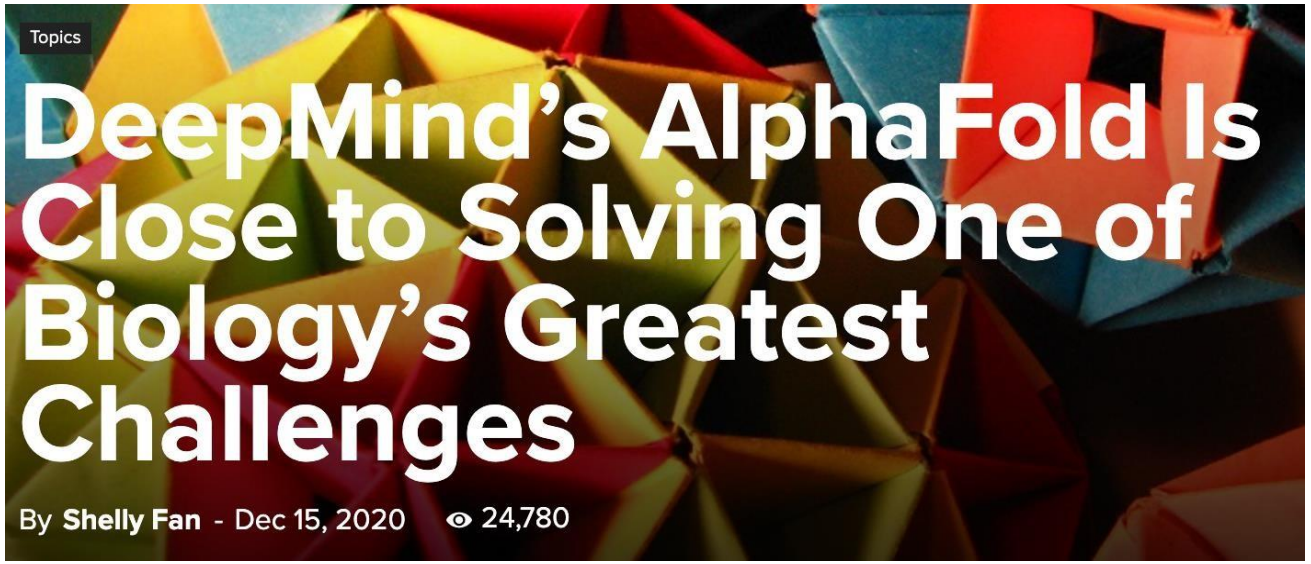


Image credit: SingularityHub

DeepMind's latest AI breakthrough can accurately predict the way proteins fold

AlphaFold's AI could change the world of biological science as we know it

12-14-20

DeepMind's latest AI breakthrough could turbocharge drug discovery

Has Artificial Intelligence 'Solved' Biology's Protein-Folding Problem?

AlphaFold: Solving Protein Folding

- Key idea: “Spatial graph”
  - Nodes: Amino acids in a protein sequence
  - Edges: Proximity between amino acids (residues)

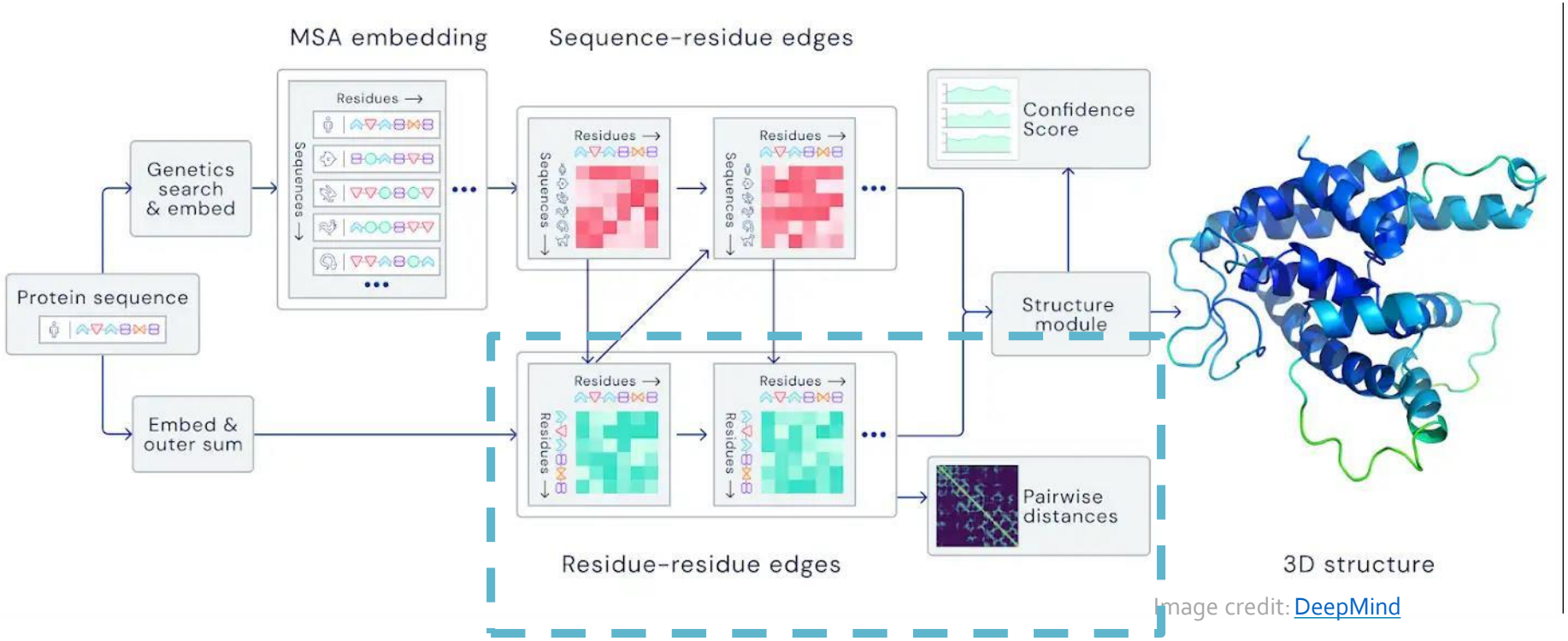
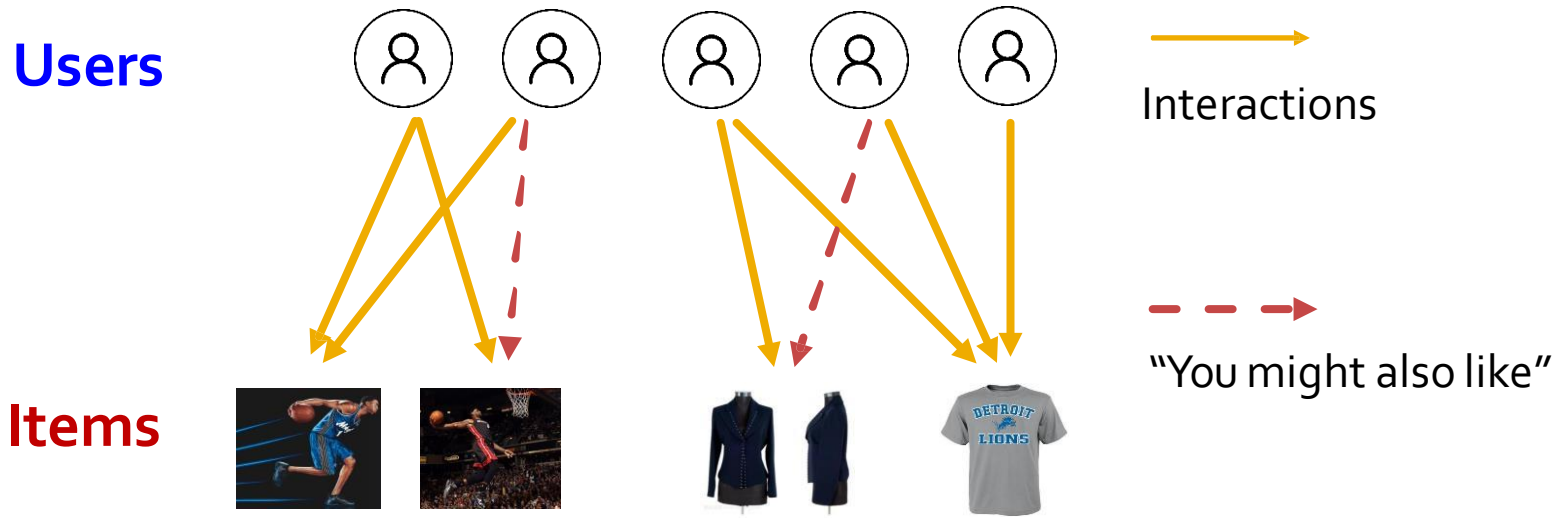


Image credit: DeepMind

Example 2 (Edge Level): Recommender Systems

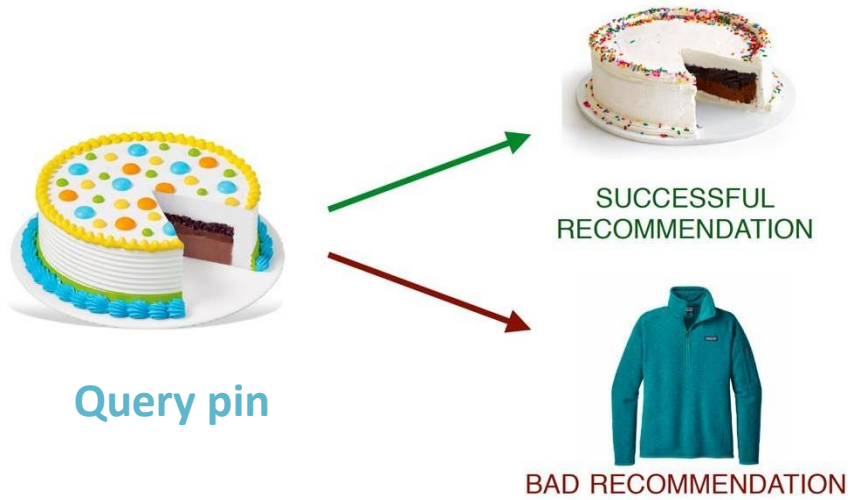
- Users interacts with items
  - Watch movies, buy merchandise, listen to music
  - Nodes: Users and items
  - Edges: User-item interactions
- Goal: Recommend items users might like

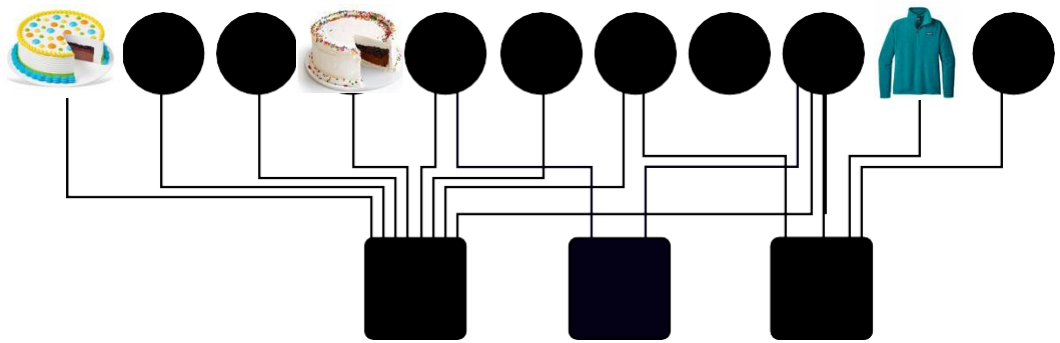


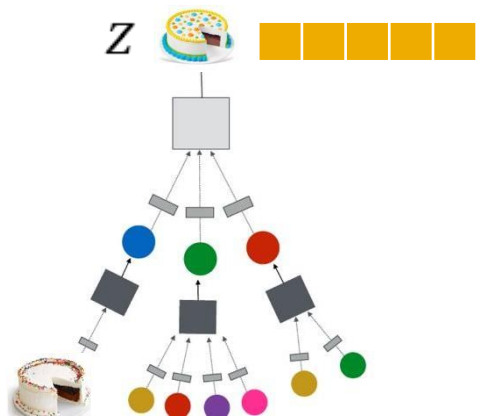
PinSage: Graph-based Recommendation

- Task: Recommend related pins to users

Task: Learn node embeddings such that $(z_{cake1}, z_{cake2}) < d(z_{cake1}, z_{sweater})$



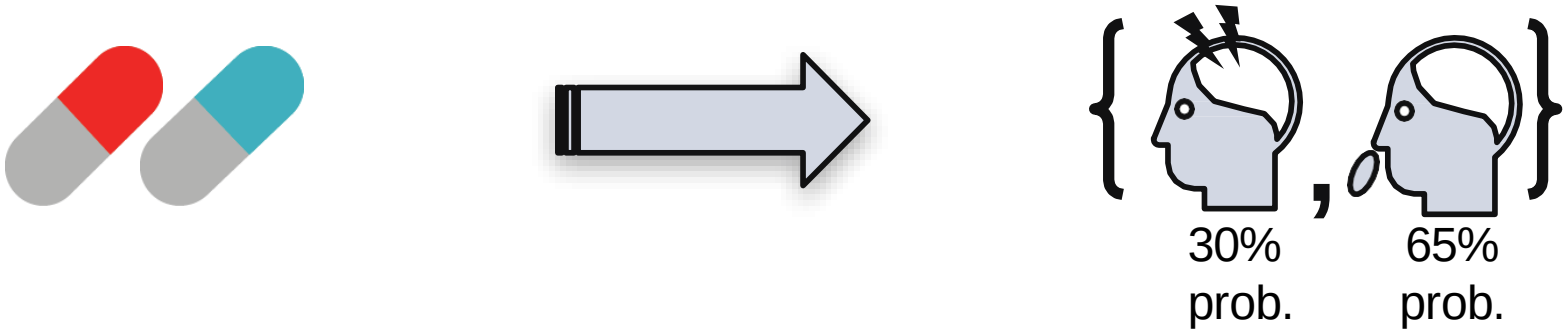




Ying et al., Graph Convolutional Neural Networks for Web-Scale Recommender Systems, KDD 2018

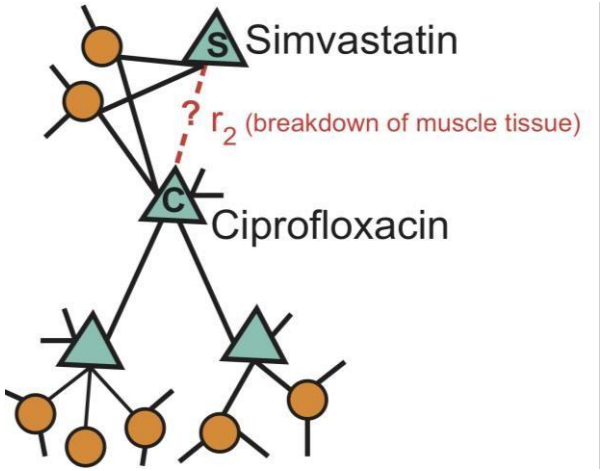
Example 3 (Edge Level): Drug Side Effects

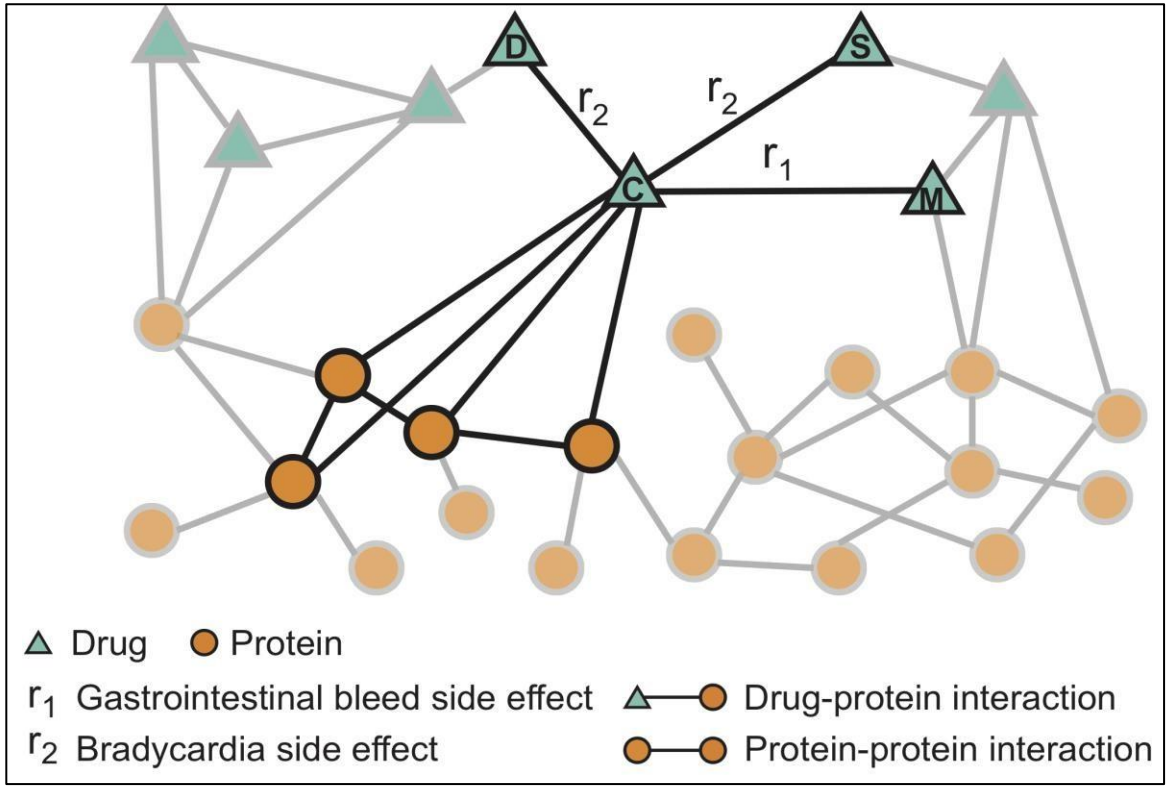
- Many patients take multiple drugs to treat complex or co-existing diseases:
  - 46% of people ages 70-79 take more than 5 drugs
  - Many patients take more than 20 drugs to treat heart disease, depression, insomnia, etc.
- Task: Given a pair of drugs predict adverse side effects



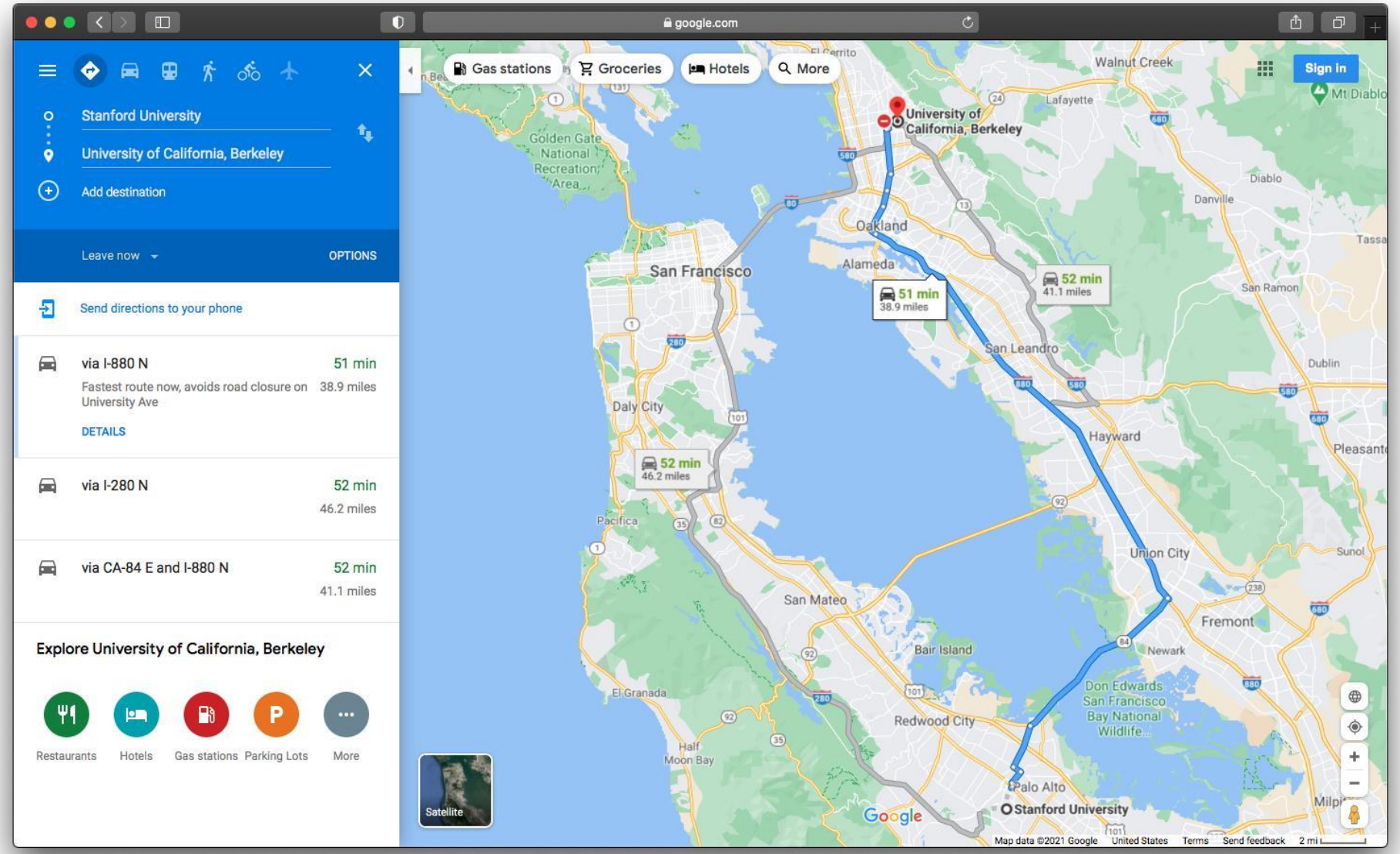
Formulate as a Linking Prediction Problem

- Nodes: Drugs & Proteins
- Edges: Interactions
- Query: How likely will Simvastatin and Ciprofloxacin, when taken together, break down muscle tissue?



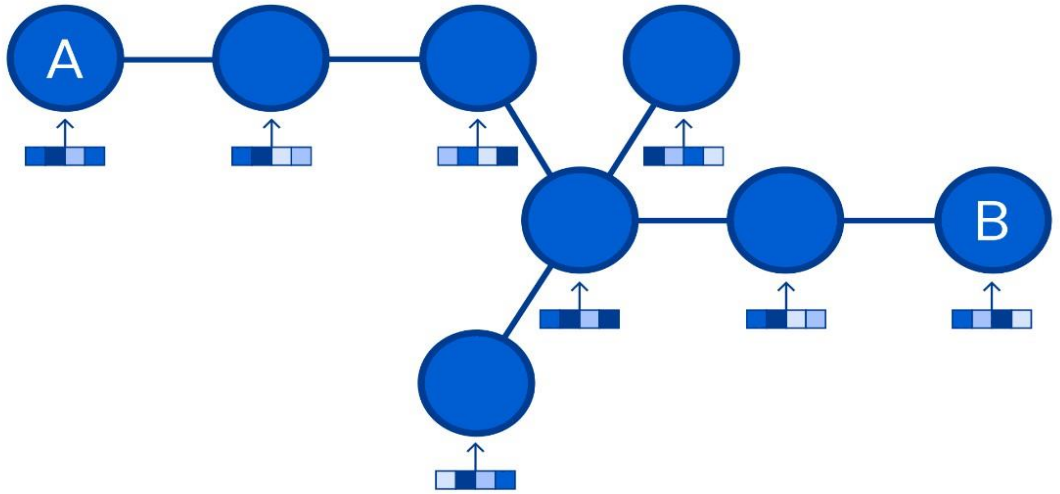


Example 4 (Subgraph Level): Traffic Prediction



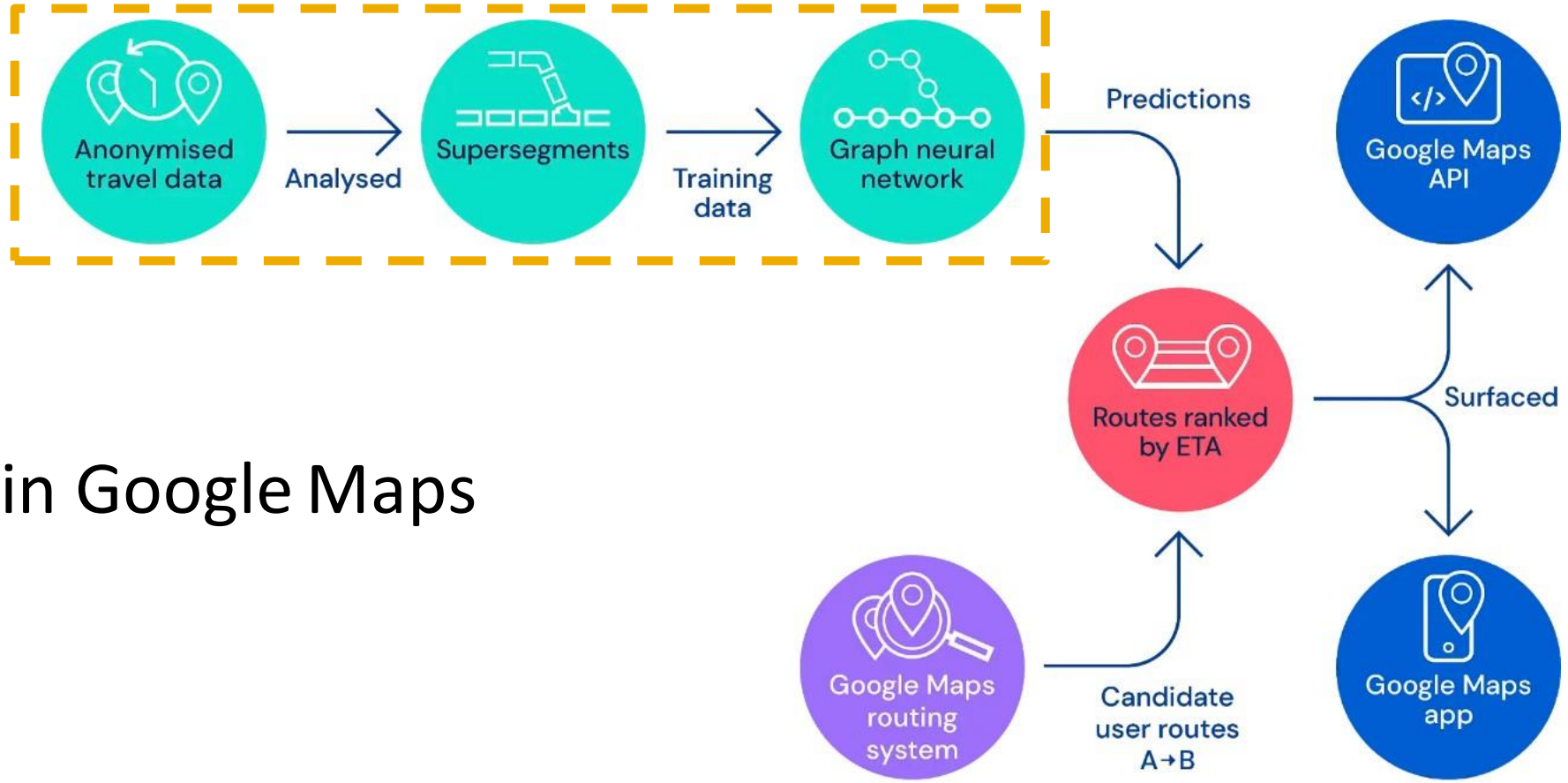
Road Network as a Graph

- Nodes: Road segments
- Edges: Connectivity between road segments
- Prediction: Time of Arrival (ETA)



Traffic Prediction via GNN

- Predicting Time of Arrival with Graph Neural Networks



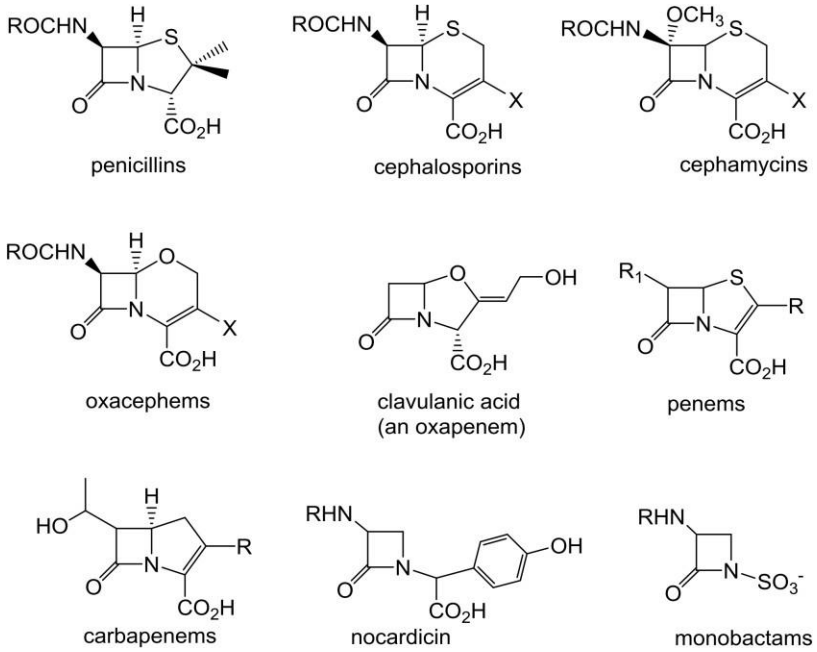
THE MODEL ARCHITECTURE FOR DETERMINING OPTIMAL ROUTES AND THEIR TRAVEL TIME.

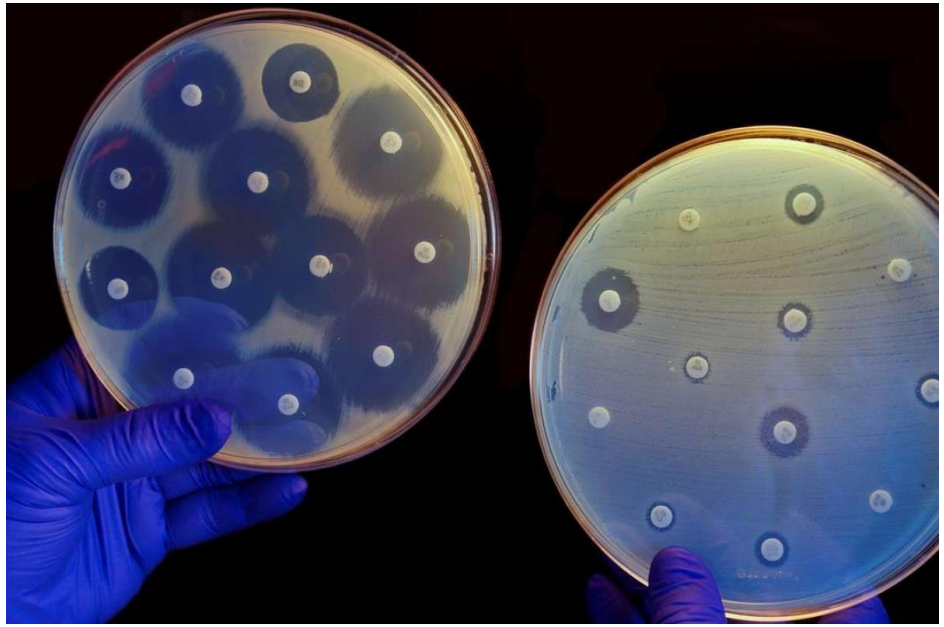
Used in Google Maps

Image credit: DeepMind

Example 5 (Graph Level): Drug Discovery

- Antibiotics are small molecular graphs
  - Nodes: Atoms
  - Edges: Chemical bonds



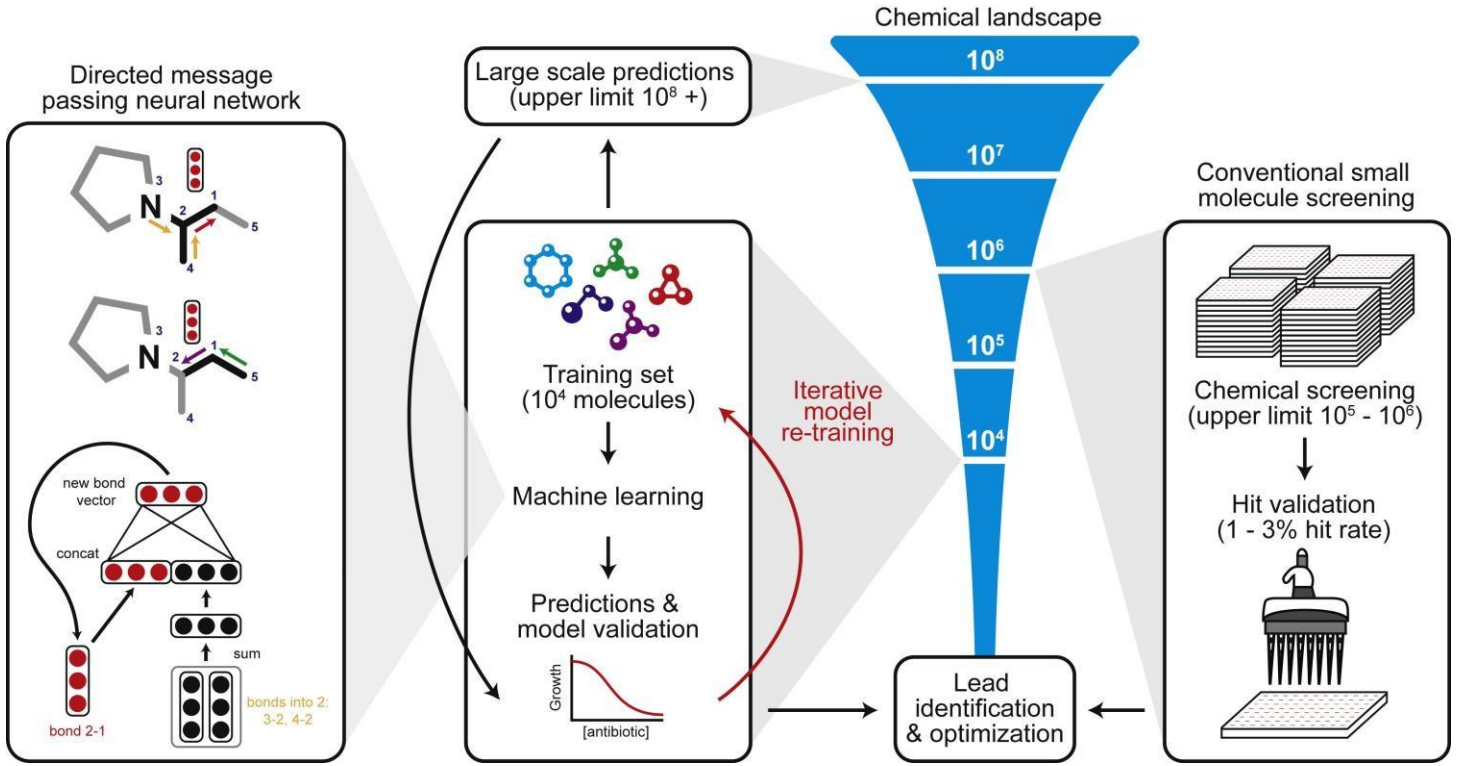


Konaklieva, Monika I. "Molecular targets of β-lactam-based antimicrobials: beyond the usual suspects." Antibiotics 3.2 (2014): 128-142.

Image credit: CNN

Deep Learning for Antibiotic Discovery

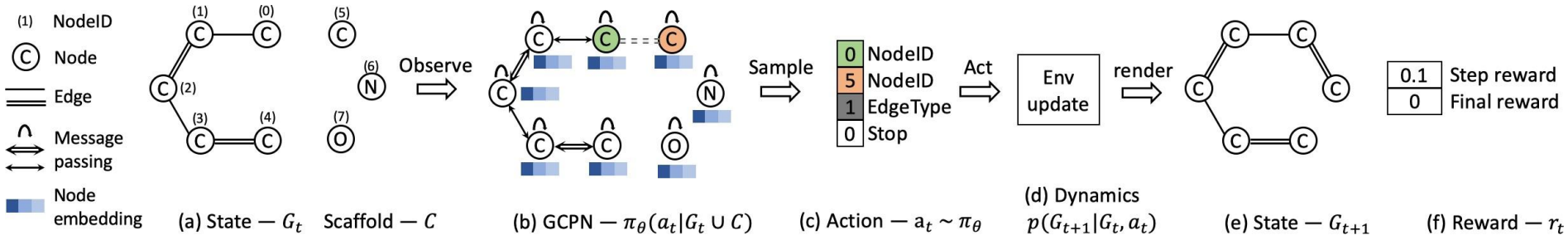
- A Graph Neural Network graph classification model
- Predict promising molecules from a pool of candidates



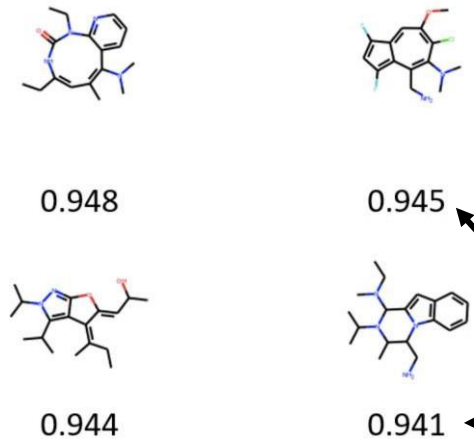
Stokeset al., A Deep Learning Approach to Antibiotic Discovery, Cell 2020

Molecule Generation/Optimization

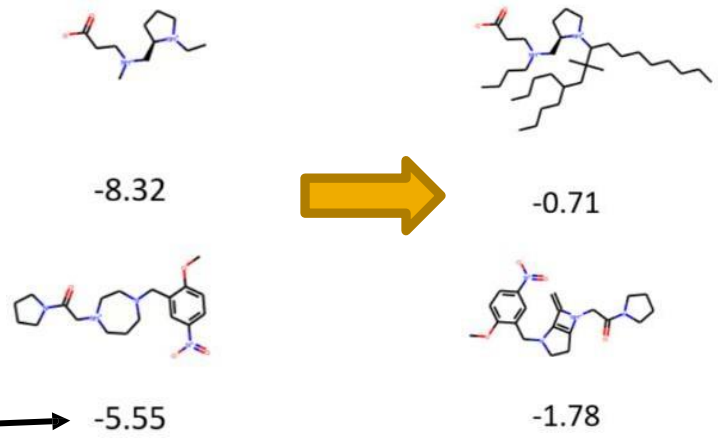
- Graph generation: Generating novel molecules



Use case 1: Generate novel molecules with high Drug likeness value



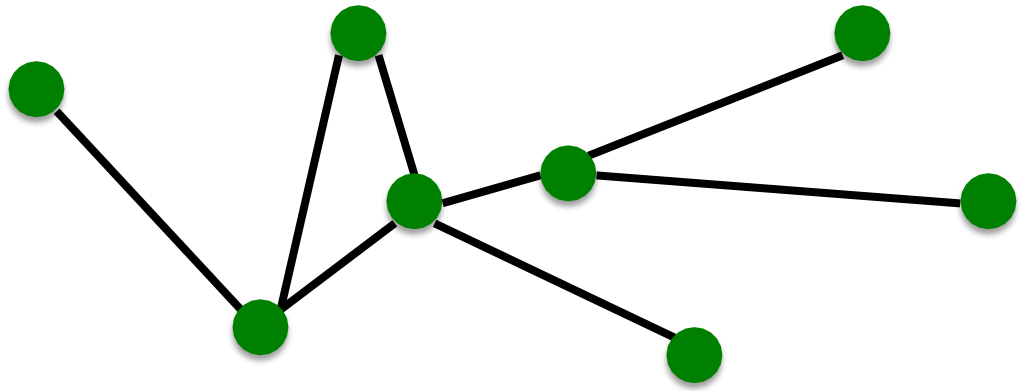
Use case 2: Optimize existing molecules to have desirable properties



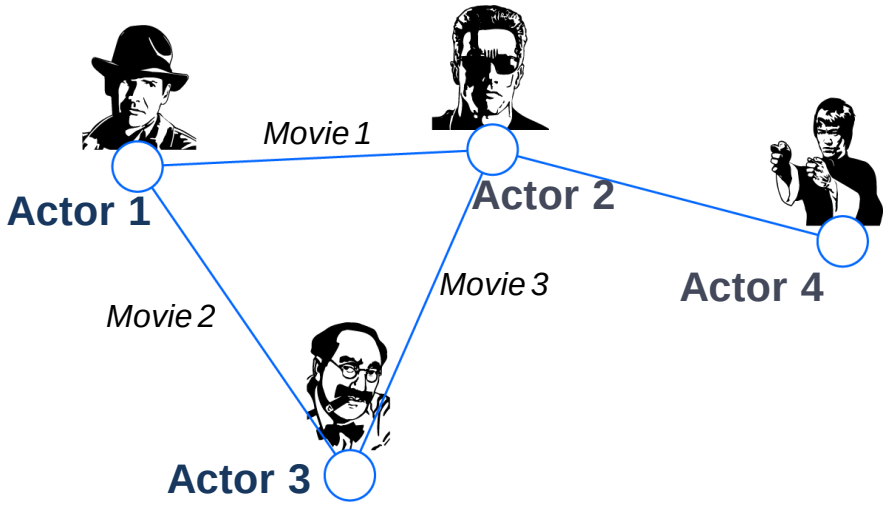
Choice of Graph Representation

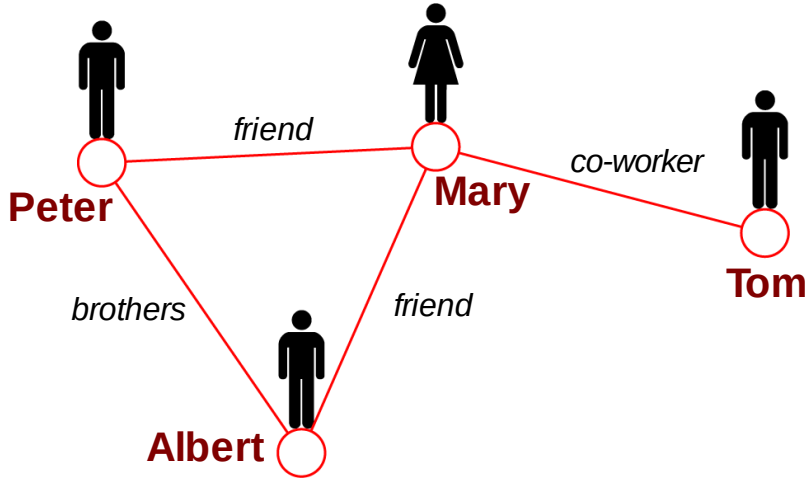
Components of a Network

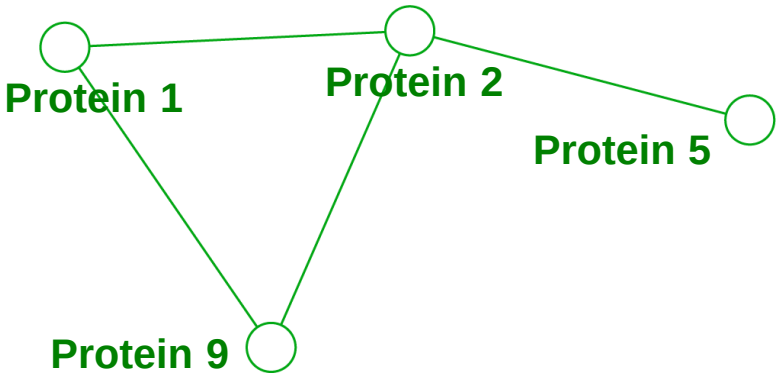
- Objects: nodes, vertices $\boldsymbol{N}$
- Interactions: links, edges $\boldsymbol{E}$
- System: network, graph $\boldsymbol{G(N,E)}$

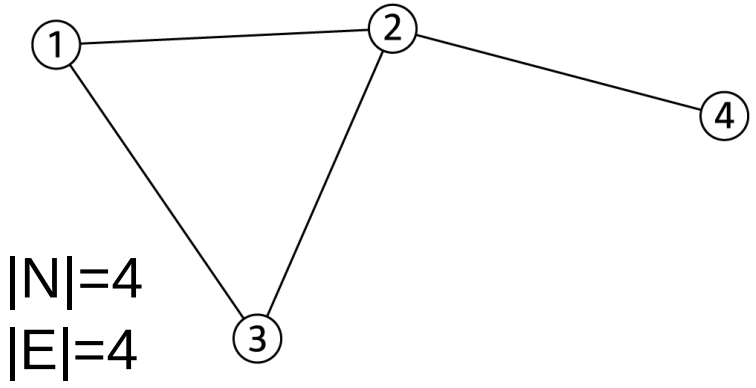


Graphs: A Common Language









Choosing a Proper Representation

- If you connect individuals that work with each other, you will explore a professional network
- If you connect those that have a sexual relationship, you will be exploring sexual networks
- If you connect scientific papers
- that cite each other, you will be studying the citation network
- **If you connect all papers with the same word in the title, what will you be exploring?** It is a network, nevertheless

Image credit: EuroScientists

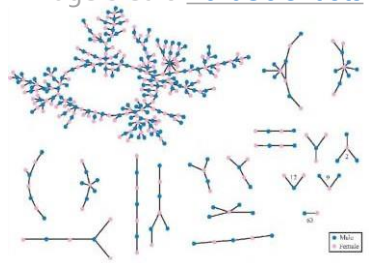
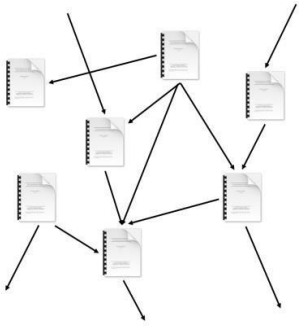


Image credit: ResearchGate



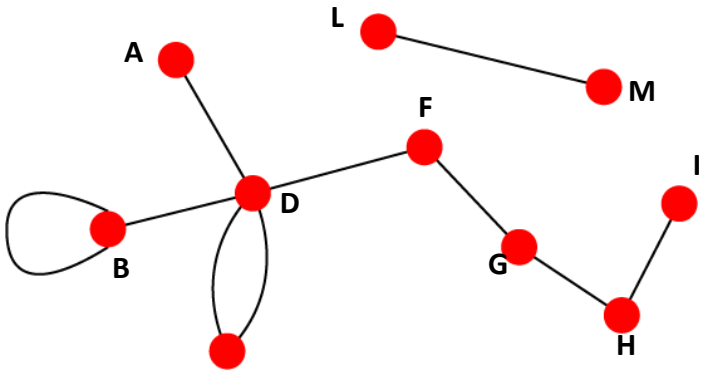
How to Define a Graph?

- How to build a graph:
  - What are nodes?
  - What are edges?
- Choice of the proper network representation of a given domain/problem determines our ability to use networks successfully:
  - In some cases, there is a unique, unambiguous representation
  - In other cases, the representation is by no means unique
  - The way you assign links will determine the nature of the question you can study

Directed vs. Undirected Graphs

Undirected

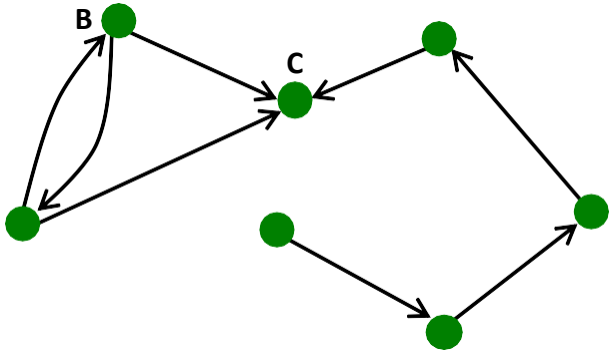
- **Links:** undirected (symmetrical, reciprocal)



- **Examples:**
  - Collaborations
  - Friendship on Facebook

Directed

- **Links:** directed (arcs)

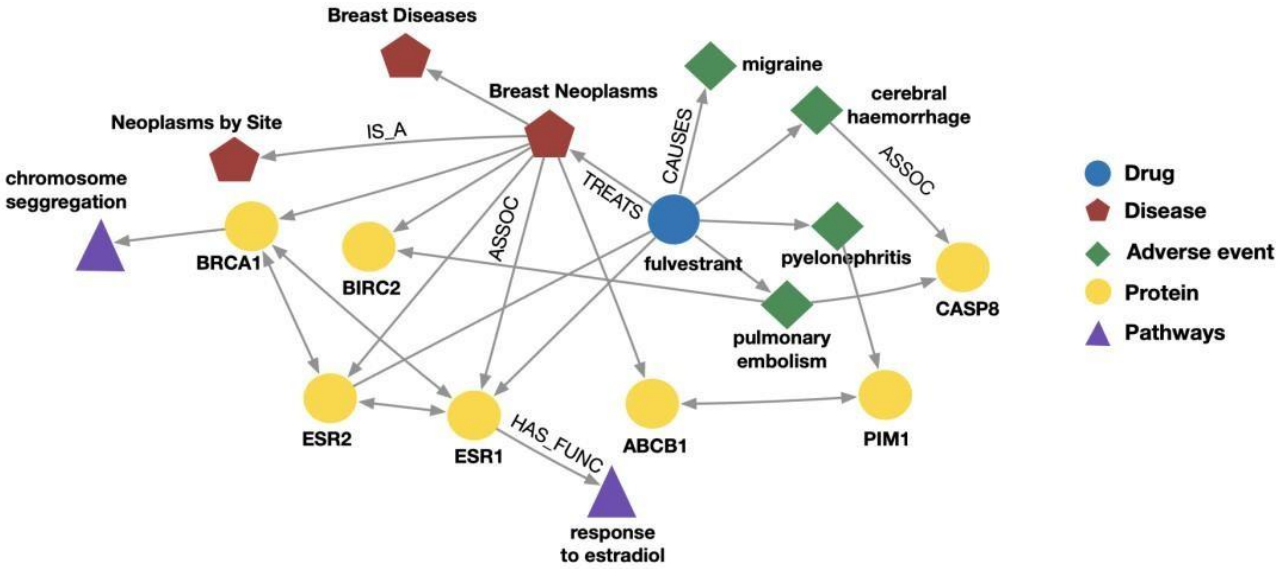


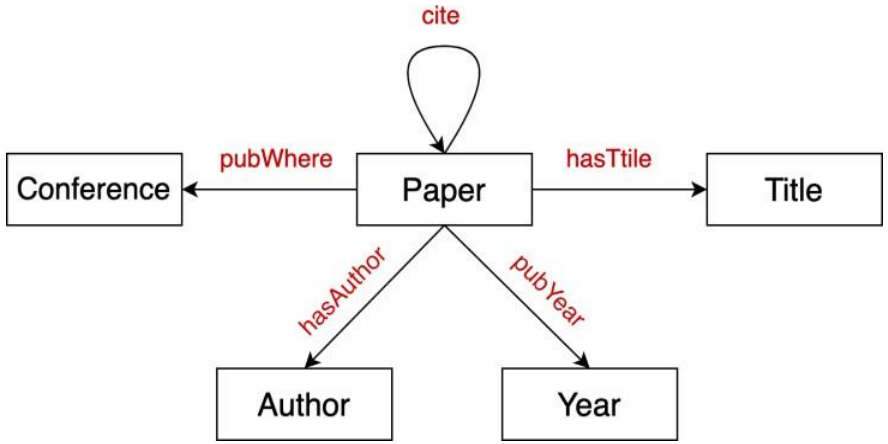
- **Examples:**
  - Phone calls
  - Following on Twitter

Heterogeneous Graphs

- A heterogeneous graph is defined as $\boldsymbol{G = (V, E, R, T)}$
  - Nodes with node types $v_i \in V$
  - Edges with relation types $(v_i, r, v_j) \in E$
  - Node type $T(v_i)$
  - Relation type $r \in R$

Many Graphs are Heterogeneous Graphs





Biomedical Knowledge Graphs

Example node: Migraine
Example edge: (fulvestrant, Treats, Breast Neoplasms)
Example node type: Protein
Example edge type (relation): Causes

Academic Graphs

Example node: ICML
Example edge: (GraphSAGE, NeurIPS)
Example node type: Author
Example edge type (relation): pubYear

Node Degrees

- Node degree, k_i: the number of edges adjacent to node *I*, $k_A=4$

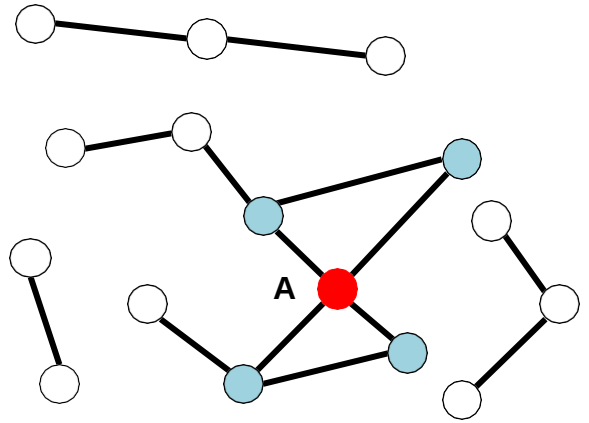
- In directed networks we define an in-degree and out-degree.

- The (total) degree of a node is the sum of in- and out-degrees.

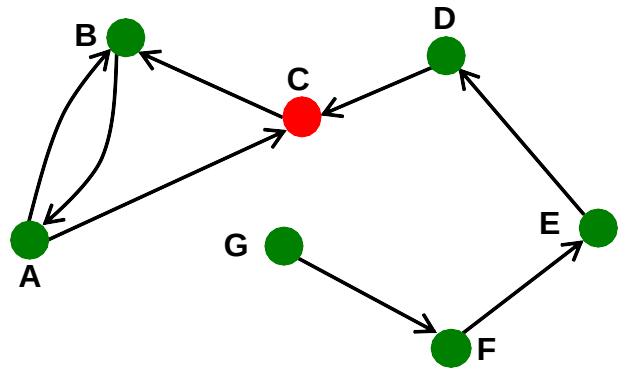
$$k_C^{in}=2 \qquad k_C^{out}=1 \qquad k_C=3$$

$$\overline{k}=\frac{E}{N} \qquad \overline{k^{in}}=\overline{k^{out}}$$

Undirected



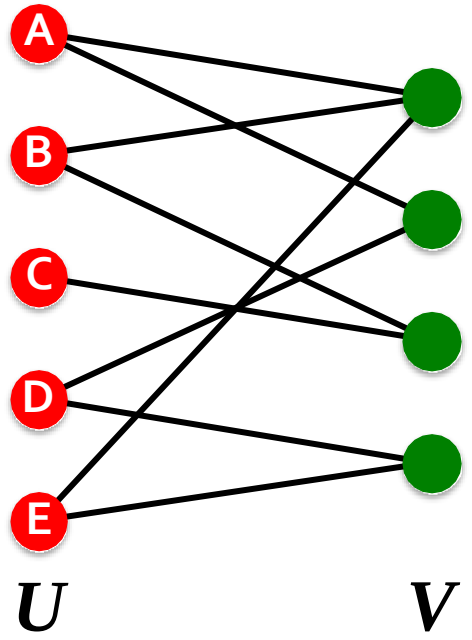
Directed



Source: Node with $k^{in}=0$
Sink: Node with $k^{out}=0$

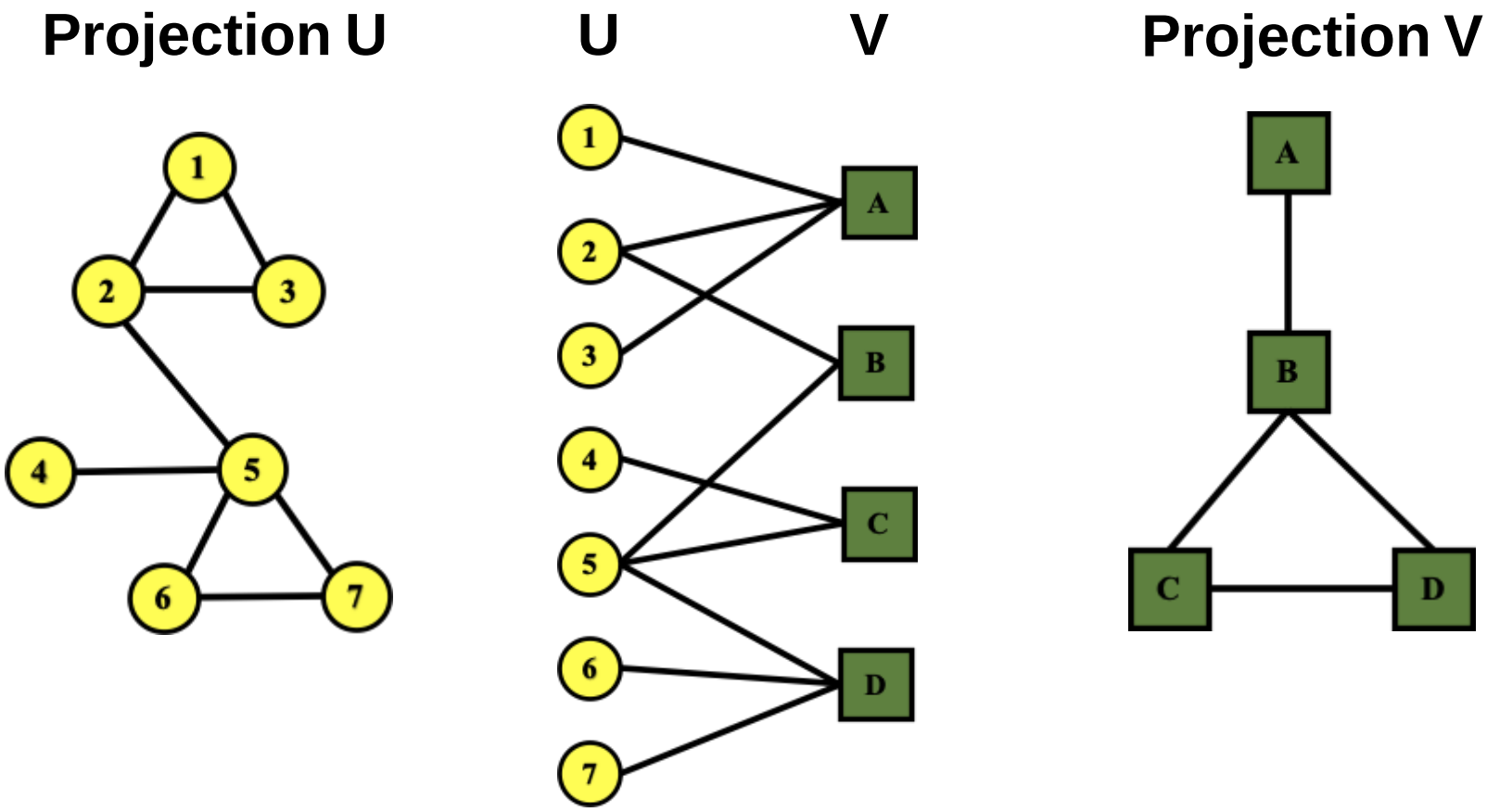
Bipartite Graph

- Bipartite graph is a graph whose nodes can be divided into two disjoint sets U and V such that every link connects a node in U to one in V; that is, U and V are independent sets
- Examples:
  - Authors-to-Papers (they authored)
  - Actors-to-Movies (they appeared in)
  - Users-to-Movies (they rated)
  - Recipes-to-Ingredients (they contain)

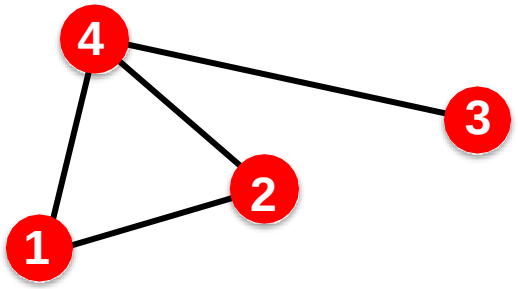


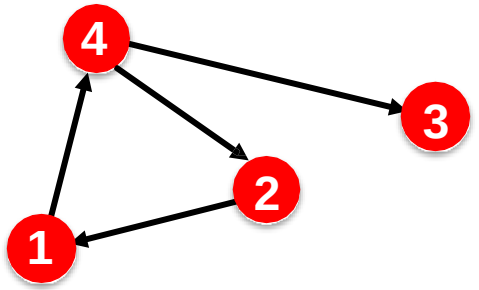
Folded/projected Networks as Bipartite Graphs

- "Folded" networks:
  - Author collaboration networks
  - Movie co-rating networks



Representing Graphs: Adjacency Matrix





$A_{ij} = 1$ if there is a link from node *i* to node *j*

$A_{ij} = 0$ otherwise

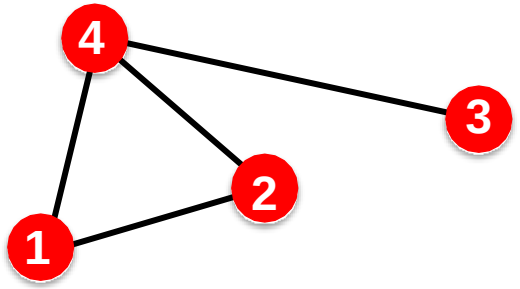
$$A = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}$$

$$A = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

Note that for a directed graph (right) the matrix is not symmetric.

Adjacency Matrix

Undirected



$$A = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}$$

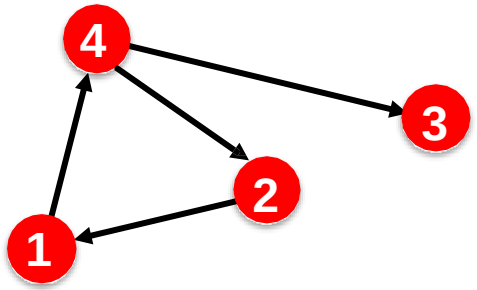
$$A_{ij} = A_{ji}$$

$$k_i = \sum_{j=1}^{N} A_{ij}$$

$$k_j = \sum_{i=1}^{N} A_{ij}$$

$$L = \frac{1}{2}\sum_{i=1}^{N} k_i = \frac{1}{2}\sum_{ij} A_{ij}$$

Directed



$$A = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

$$A_{ij} \neq A_{ji}$$

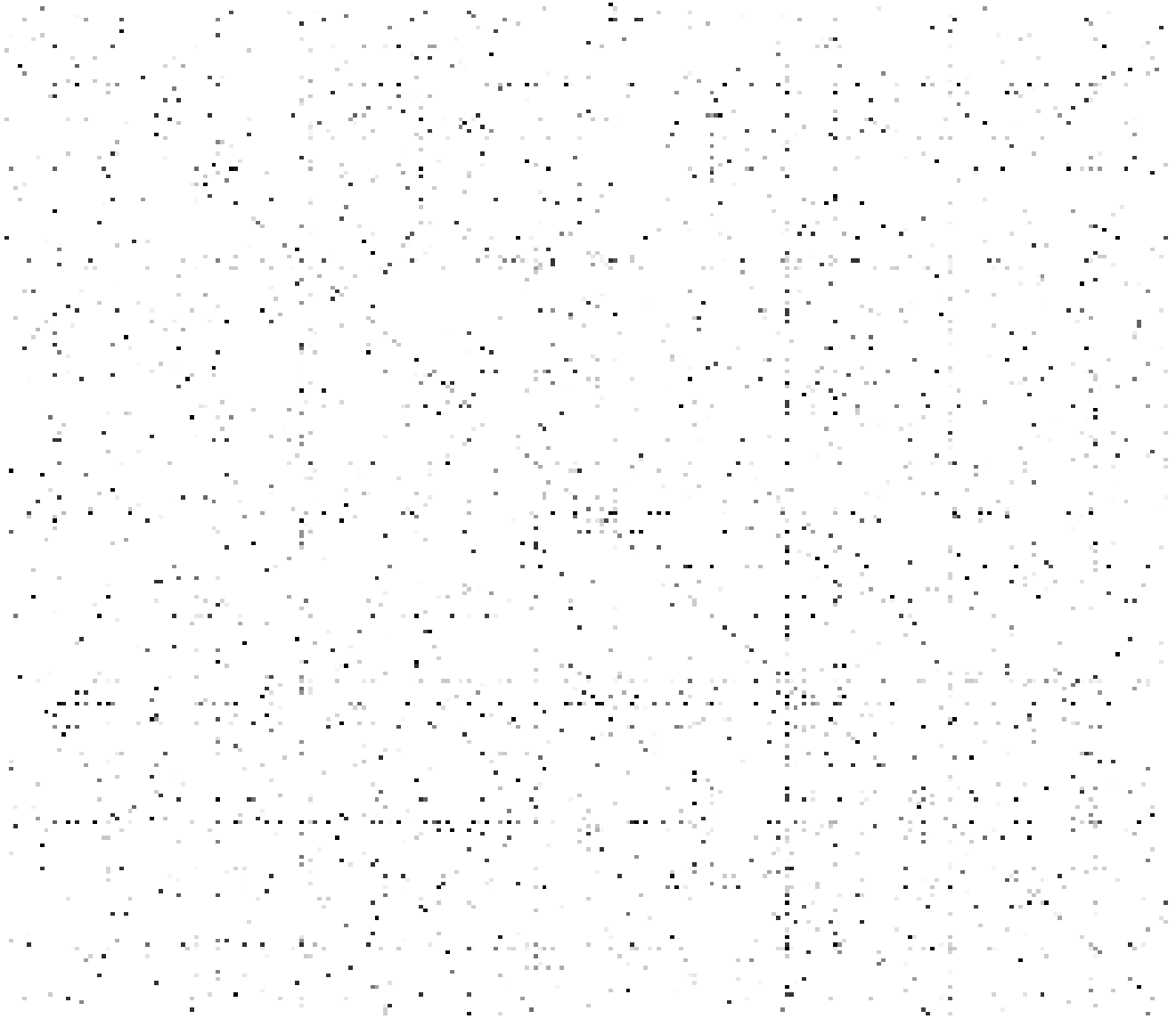
$$A_{ii} = 0$$

$$k_i^{out} = \sum_{j=1}^{N} A_{ij}$$

$$k_j^{in} = \sum_{i=1}^{N} A_{ij}$$

$$L = \sum_{i=1}^{N} k_i^{in} = \sum_{j=1}^{N} k_j^{out} = \sum_{i,j} A_{ij}$$

Adjacency Matrices are Sparse



Networks are Sparse Graphs

- Most real-world networks are sparse
  - $E << E_{max}$ (or $k << N-1$)

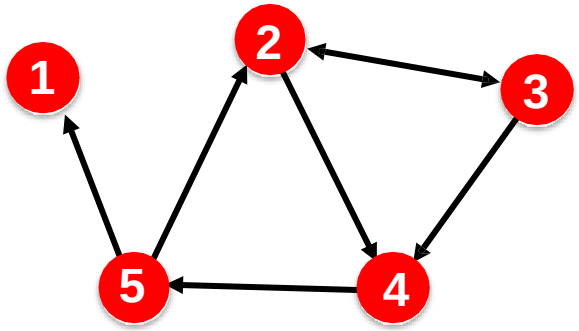
NETWORK	NODES	LINKS	DIRECTED/ UNDIRECTED	N	L	<k>
Internet	Routers	Internet connections	Undirected	192,244	609,066	6.33
WWW	Webpages	Links	Directed	325,729	1,497,134	4.60
Power Grid	Power plants, transformers	Cables	Undirected	4,941	6,594	2.67
Phone Calls	Subscribers	Calls	Directed	36,595	91,826	2.51
Email	Email Addresses	Emails	Directed	57,194	103,731	1.81
Science Collaboration	Scientists	Co-authorship	Undirected	23,133	93,439	8.08
Actor Network	Actors	Co-acting	Undirected	702,388	29,397,908	83.71
Citation Network	Paper	Citations	Directed	449,673	4,689,479	10.43
E. Coli Metabolism	Metabolites	Chemical reactions	Directed	1,039	5,802	5.58
Protein Interactions	Proteins	Binding interactions	Undirected	2,018	2,930	2.90

Consequence: Adjacency matrix is filled with zeros!

(**Density of the matrix (E/N^2):** WWW=1.51×10^{-5}, MSN IM = 2.27×10^{-8})

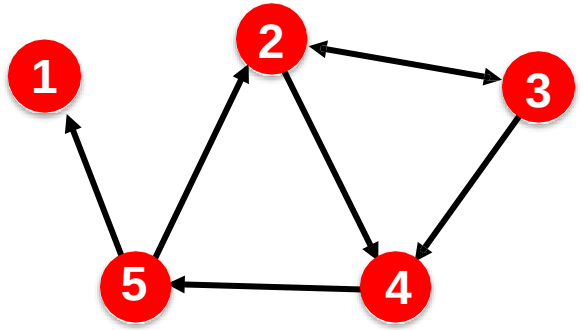
Representing Graphs: Edge List

- Represent graph as a list of edges:
  - (2, 3)
  - (2, 4)
  - (3, 2)
  - (3, 4)
  - (4, 5)
  - (5, 2)
  - (5, 1)



Representing Graphs: Adjacency List

- Adjacency list:
  - Easier to work with if network is
    - Large
    - Sparse
  - Allows us to quickly retrieve all neighbors of a given node
    - 1:
    - 2: 3, 4
    - 3: 2, 4
    - 4: 5
    - 5: 1, 2



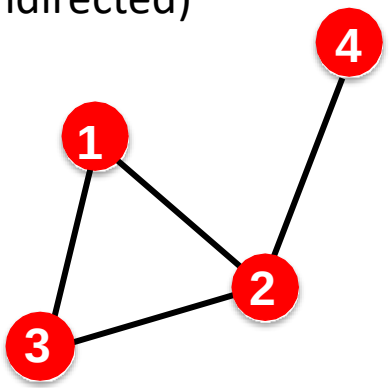
Node and Edge Attributes

- Possible Options:
  - Weight (*e.g.*, frequency of communication)
  - Ranking (best friend, second best friend…)
  - Type (friend, relative, co-worker)
  - Sign: Friend vs. Foe, Trust vs Distrust
  - Properties depending on the structure of the rest of the graph: Number of common friends

More Types of Graphs

- **Unweighted**

(undirected)



$$A = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

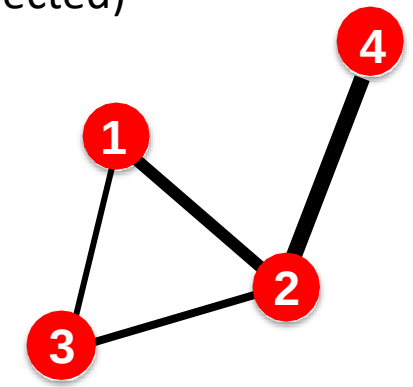
$$A_{ii} = 0 \qquad A_{ij} = A_{ji}$$

$$E = \frac{1}{2}\sum_{i,j=1}^{N} A_{ij} \qquad \bar{k} = \frac{2E}{N}$$

Examples: Friendship, Hyperlink

- **Weighted**

(undirected)



$$A = \begin{bmatrix} 0 & 2 & 0.5 & 0 \\ 2 & 0 & 1 & 4 \\ 0.5 & 1 & 0 & 0 \\ 0 & 4 & 0 & 0 \end{bmatrix}$$

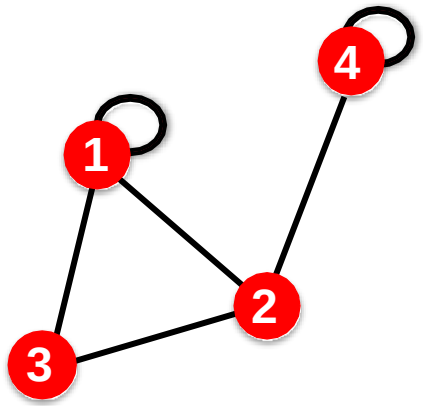
$$A_{ii} = 0 \qquad A_{ij} = A_{ji}$$

$$E = \frac{1}{2}\sum_{i,j=1}^{N} nonzero(A_{ij}) \qquad \bar{k} = \frac{2E}{N}$$

Examples: Collaboration, Internet, Roads

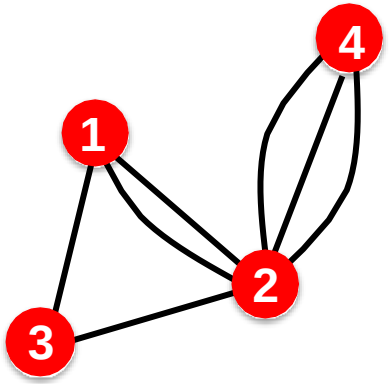
More Types of Graphs

- **Self-edges (self-loops)**
  (undirected)



Examples: Proteins, Hyperlinks

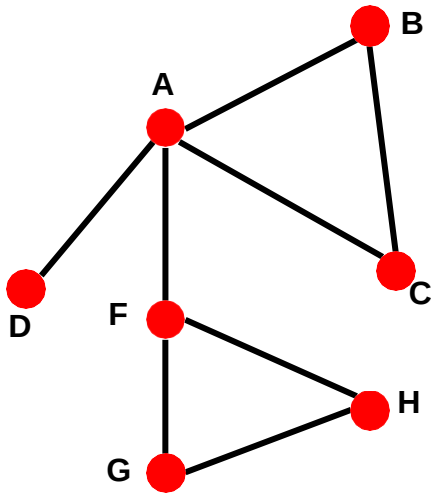
- **Multigraph**
  (undirected)

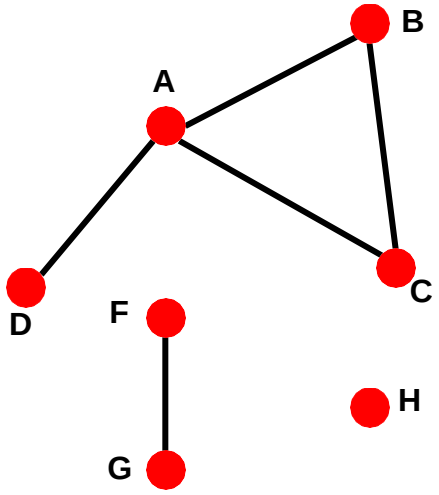


Examples: Communication, Collaboration

Connectivity of Undirected Graphs

- Connected (undirected) graph:
  - Any two vertices can be joined by a path
- A disconnected graph is made up by two or more connected components





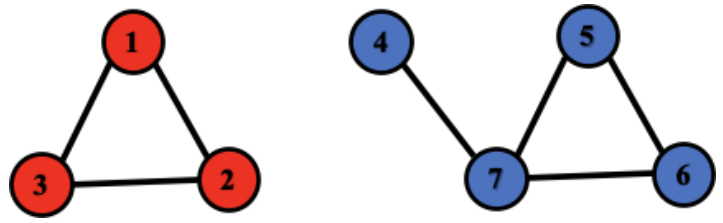
Largest Component:
Giant Component

Isolated node (node H)

Connectivity: Example

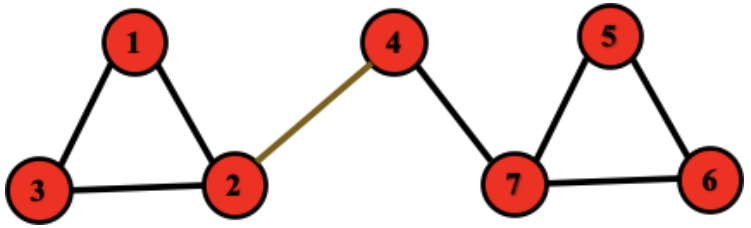
- The adjacency matrix of a network with several components can be written in a block- diagonal form, so that nonzero elements are confinedto squares, with all other elements beingzero:

Disconnected



$$\begin{pmatrix} 0&1&1&0&0&0&0\\ 1&0&1&0&0&0&0\\ 1&1&0&0&0&0&0\\ 0&0&0&0&0&0&1\\ 0&0&0&0&0&1&1\\ 0&0&0&0&1&0&1\\ 0&0&0&1&1&1&0 \end{pmatrix}$$

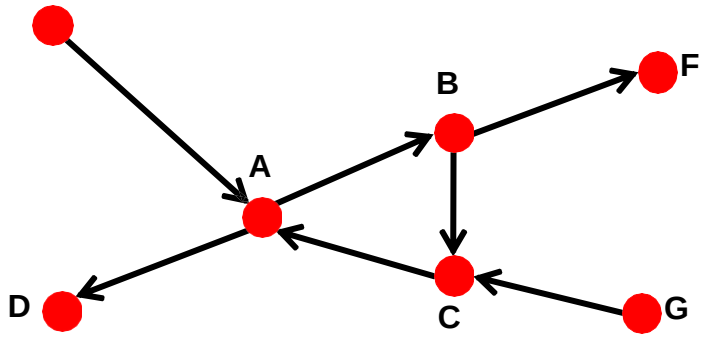
Connected



$$\begin{pmatrix} 0&1&1&0&0&0&0\\ 1&0&1&1&0&0&0\\ 1&1&0&0&0&0&0\\ 0&1&0&0&0&0&1\\ 0&0&0&0&0&1&1\\ 0&0&0&0&1&0&1\\ 0&0&0&1&1&1&0 \end{pmatrix}$$

Connectivity of Directed Graphs

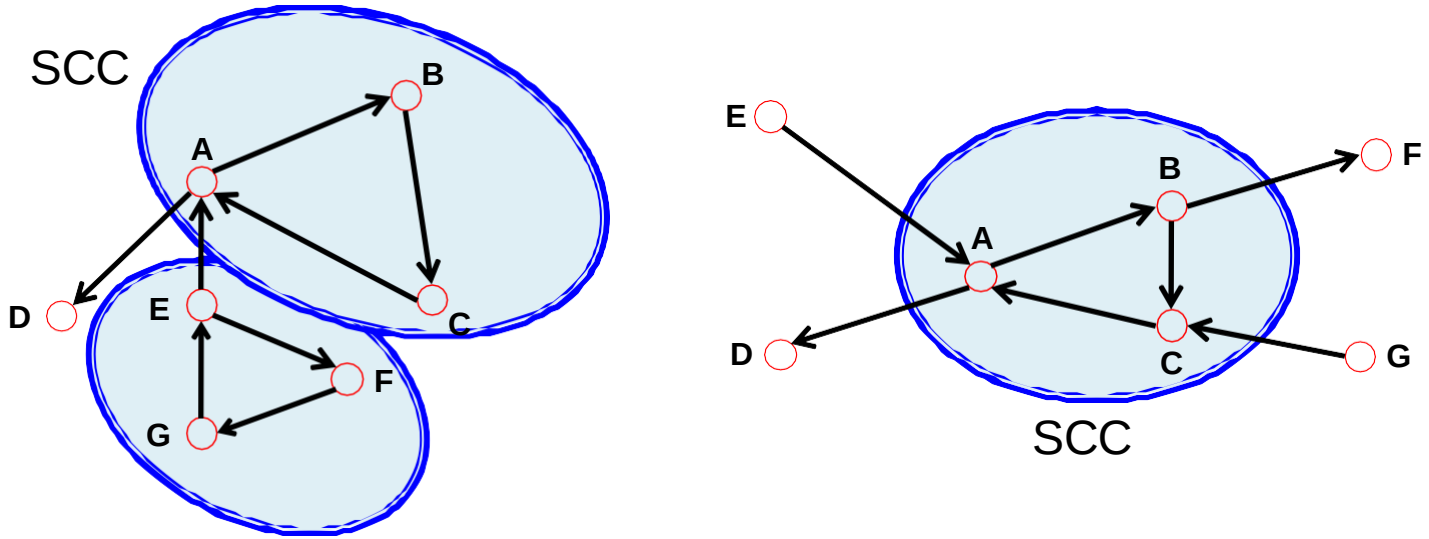
- Strongly connected directed graph
  - has a path from each node to every other node and vice versa (e.g., A-B path and B-A path)
- Weakly connected directed graph
  - is connected if we disregard the edge directions



Graph on the left is connected
but not strongly connected (e.g., there is no way to get from F to G by following the edge directions).

Connectivity of Directed Graphs

- **Strongly connected components (SCCs)** canbe identified, but not every node is part of a nontrivial strongly connected component.



- In-component: nodes that can reach the SCC,
- Out-component: nodes that can be reached from the SCC.

Summary

- **Machine learning with Graphs**
  - Applications and use cases
- **Different types of tasks:**
  - Node level
  - Edge level
  - Graph level
- **Choice of a graph representation:**
  - Directed, undirected, bipartite, weighted, adjacency matrix