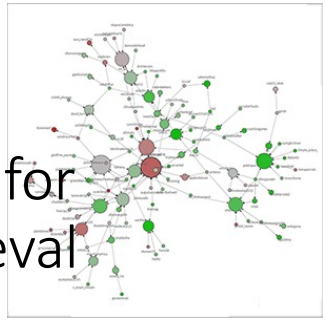
Network analysis for information retrieval



Julien Velcin
Master MALIA-MIASHS
2024-2025

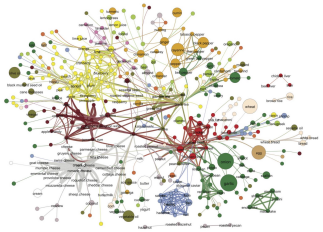
(part 4/4)

Outline

- Motivation
  - ubiquity of information networks
  - applications (in particular to IR)
  - importance of indexing
- Representation of documents
  - sparse representations
  - dense representations
  - topic models
- Network analysis
  - spectral clustering, modularity
  - representation learning for graphs
- Analyzing information networks
  - Graph Neural Networks

Network analysis

Network science



- New discipline (early 2000s)
- Heavily relies on (many) previous works, in particular using **graph theory**, but not only
- Various network structures sharing similar properties (metabolic networks, WWW, social networks…)
- Closely related to the field of **complex systems** (Barabási, 2016)

https://scienceandfooducla.wordpress.com/2013/02/26/the-flavor-network/

Different networks, same graph

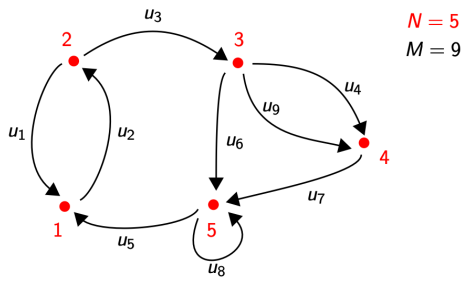
Network	Nodes	Links	Directed / Undirected	N	L	‹k›
Internet	Routers	Internet connections	Undirected	192,244	609,066	6.34
WWW	Webpages	Links	Directed	325,729	1,497,134	4.60
Power Grid	Power plants, transformers	Cables	Undirected	4,941	6,594	2.67
Mobile-Phone Calls	Subscribers	Calls	Directed	36,595	91,826	2.51
Email	Email addresses	Emails	Directed	57,194	103,731	1.81
Science Collaboration	Scientists	Co-authorships	Undirected	23,133	93,437	8.08
Actor Network	Actors	Co-acting	Undirected	702,388	29,397,908	83.71
Citation Network	Papers	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

from: (Barabási, 2016)

Basics on graphs (1)

- Let **G = (V, E)** be determined by:
  - set V (elements of V are called **vertices** or nodes). If n=|V| is the number of nodes, the order of G is N
  - set E, with e ∈ E (E ⊂ V x V) so that e = (i,j) is an ordered tuple (i,j) called an edge; i is the origin (initial) node and j is the destination (final) node. We note M= |E| the number of arcs.
  - If G is **directed** (digraph), edges have a direction and they are called directed edges (« arcs »).
  - If G is **undirected**, edges have no direction and they are called undirected edges (« arêtes »). Therefore (i,j) ∈ E ⟺ (j,i) ∈ E

Example of directed graph

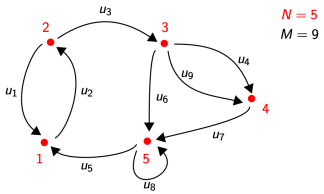


Basics on graphs (2)

- G = is usually coded with its **adjacency matrix A:**

  A[i,j]= 1 if (i,j) ∈ E, 0 otherwise

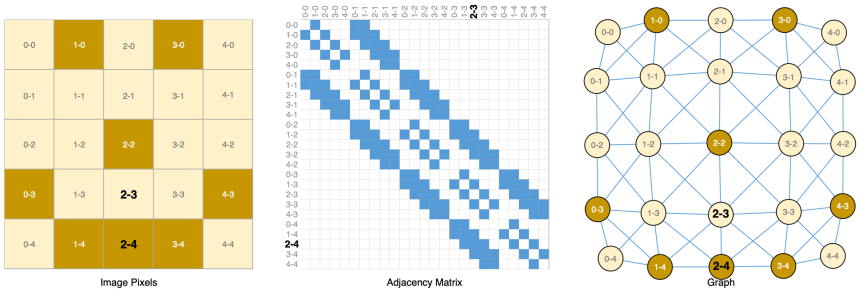
  A[i,j]= wij, wij in ℝ, if G is a weighted graph



Basics on graphs (3)

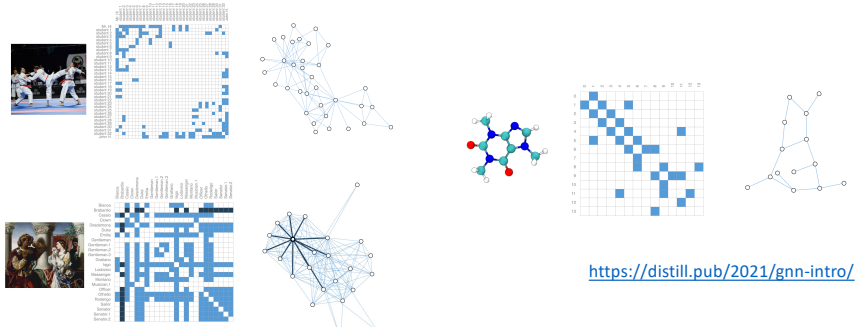
- Degrees:
  - If G is undirected, the **degree of G** is the sum of W of incident nodes (number of neighbors for unweighted G)
  - If G is directed, the **in-degree** of G is the sum of W for incoming edges and **out-degree** of G is the sum of W for outgoing edges (or number if G is unweighted)
- A **path** p in a graph is a sequence of edges which joins a sequence of vertices: p = (v_0, v_1… v_n) s.t. (v_i, v_{i+1}) ∈ E, p is of **length** n
- The powers of A are related to the paths
  - A^k encodes the paths of length k
  - A^k can be approximated by random walks

Because a picture is worth a 1000 words



https://distill.pub/2021/gnn-intro/

Some datasets

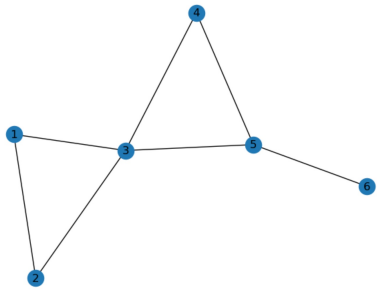


https://distill.pub/2021/gnn-intro/

Proximity measures: from local to global

- One-order proximity: direct links
  - see A
- Two-order proximity: similar neighborhood
  - see A.A
- Higher-order proximity: related to the distance in the graph
  - see A^k
- Approximation by *random walks* using the transition matrix P
  - P[u,v] = probability to go from node u to node v in one step

Illustration (1)



A

```
[[0, 1, 1, 0, 0, 0],
 [1, 0, 1, 0, 0, 0],
 [1, 1, 0, 1, 1, 0],
 [0, 0, 1, 0, 1, 0],
 [0, 0, 1, 1, 0, 1],
 [0, 0, 0, 0, 1, 0]]
```

A²

```
[[2, 1, 1, 1, 1, 0],
 [1, 2, 1, 1, 1, 0],
 [1, 1, 4, 1, 1, 1],
 [1, 1, 1, 2, 1, 1],
 [1, 1, 1, 1, 3, 0],
 [0, 0, 1, 1, 0, 1]]
```

A³

```
[[2, 3, 5, 2, 2, 1],
 [3, 2, 5, 2, 2, 1],
 [5, 5, 4, 5, 6, 1],
 [2, 2, 5, 2, 4, 1],
 [2, 2, 6, 4, 2, 3],
 [1, 1, 1, 1, 3, 0]]
```

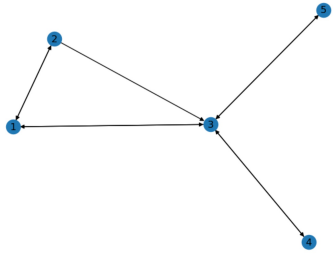
P

0	0.5	0.25	0	0	0
0.5	0	0.25	0	0	0
0.5	0.5	0	0.5	1/3	0
0	0	0.25	0	1/3	0
0	0	0.25	0.5	0	1
0	0	0	0	1/3	0

A + A² + A³

```
[[4, 5, 7, 3, 3, 1],
 [5, 4, 7, 3, 3, 1],
 [7, 7, 8, 7, 8, 2],
 [3, 3, 7, 4, 6, 2],
 [3, 3, 8, 6, 5, 4],
 [1, 1, 2, 2, 4, 1]]
```


Illustration (2)



A

```
[[0, 1, 1, 0, 0],
 [1, 0, 1, 0, 0],
 [1, 0, 0, 1, 1],
 [0, 0, 1, 0, 0],
 [0, 0, 1, 0, 0]]
```

A²

```
[[2, 1, 0, 1, 1],
 [1, 2, 1, 1, 1],
 [0, 1, 3, 0, 0],
 [1, 1, 0, 1, 1],
 [1, 1, 0, 1, 1]]
```

A³

```
[[1, 3, 4, 1, 1],
 [2, 2, 3, 1, 1],
 [4, 3, 0, 3, 3],
 [0, 1, 3, 0, 0],
 [0, 1, 3, 0, 0]]
```

P

0	0.5	1/3	0	0
0.5	0	0	0	0
0.5	0.5	0	1	1
0	0	1/3	0	0
0	0	1/3	0	0

A + A² + A³

```
[[3, 5, 5, 2, 2],
 [4, 4, 5, 2, 2],
 [5, 4, 3, 4, 4],
 [1, 2, 4, 1, 1],
 [1, 2, 4, 1, 1]]
```


Local proximity measures

$$\mathbf{S}[u,v] = |\mathcal{N}(u) \cap \mathcal{N}(v)|$$

$$\mathbf{S}_{\text{Salton}}[u,v] = \frac{2|\mathcal{N}(u) \cap \mathcal{N}(v)|}{\sqrt{d_u d_v}}$$

a way to take the « popularity » bias into account

$$\mathbf{S}_{\text{Sorenson}}[u,v] = \frac{2|\mathcal{N}(u) \cap \mathcal{N}(v)|}{d_u + d_v}$$

$$\mathbf{S}_{\text{Jaccard}}[u,v] = \frac{|\mathcal{N}(u) \cap \mathcal{N}(v)|}{|\mathcal{N}(u) \cup \mathcal{N}(v)|}$$

Local measures can be very effective for **link prediction**

Centrality measures

- Degree centrality: simply based on node degree
- Betweenness centrality: number of times a node lies on the shortest path between other nodes
- Closeness centrality: sum of shortest paths to all other nodes
- EigenCentrality: importance based on eigen decomposition
- PageRank: eigenCentrality + direction + weight

Eigen centrality and PageRank

- Centrality score of vertice v is defined as: $x_v = \frac{1}{\lambda} \sum_{n \in N(v)} x_n$
- It can be rewritten as: $\mathbf{A}.x = \lambda.x$
- Assuming that the centrality values are positive and following the **Perron–Frobenius theorem**, we know we have to choose the vector associated to the *highest eigenvalue*
- This main eigenvector can be computed using power iteration
- Note that the Google PR is a generalization of this measure to a directed graph with a *damping* factor

PageRank (Page et al., 1999)

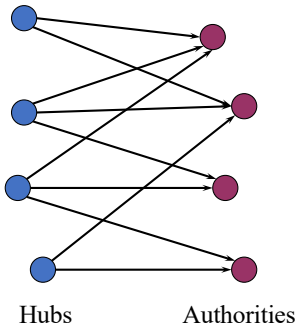
- Centrality score based on the idea of a random walker: $r_j = \sum_{i \to j} \frac{r_i}{d_i}$
- Designed for directed graphs with the ability to do teletransportation
- If π is a given score, we aim to reach the stationary distribution of:
  $\pi = P \cdot \pi$
- The **transition matrix P** is a positive, square matrix, so we can use the Frobenius-Perron theorem to find π by the power iteration method
- To overcome some limitations (e.g., dead ends), the PR integrates the possibility to jump to any node in the graph:
  $R = \beta \cdot P + (1-\beta) \cdot v \cdot e^T$
  with v the vector of ones and e a vector of 1/n

Hubs and Authorities

- Intuition behind HITS is that each web page has two natures:
  - Good **content page** (authority weight)
  - Good **hub** (hub weight)
- Idea behind the algorithm:
  - Good authority page is pointed to by good hub pages
  - Good hub page is pointing to good authority pages

Hubs and Authorities

[Kleinberg, 99]



"Hubs and authorities exhibit what could be called a *mutually reinforcing* relationship"

Iterative relaxation:

$$\text{Hub}(p) = \sum_{q:p \to q} \text{Authority}(q)$$

$$\text{Authority}(p) = \sum_{q:q \to p} \text{Hub}(q)$$

Global proximity measures (1)

$$\mathbf{S}_{\text{Katz}}[u,v] = \sum_{i=1}^{\infty} \beta^i \mathbf{A}^i[u,v]$$

path of lenght i between two nodes

with β a user-defined hyper-parameter
solved by a spectral decomposition: $\mathbf{S}_{\text{Katz}} = (\mathbf{I} - \beta\mathbf{A})^{-1} - \mathbf{I}$

(demonstration)

Global proximity measures (2)

$\mathbf{S}_{\text{RW}}[u,v] = \mathbf{q}_u[v] + \mathbf{q}_v[u]$ (Personalized Page Rank)

with: q_u estimates the probability to visit node v starting from node u

$$\mathbf{q}_u = c\mathbf{P}\mathbf{q}_u + (1-c)\mathbf{e}_u = (1-c)(\mathbf{I} - c\mathbf{P})^{-1}\mathbf{e}_u$$

restart factor

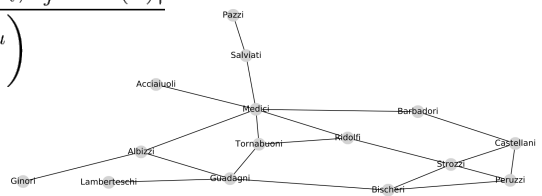
transition matrix AD^{-1}
(for an undirected graph)

OHE of the node

Clustering coefficient

- CC measures the proportion of closed triangles in a node's local neighborhood

$$c_u = \frac{|(v_i, v_j) \in E : v_i, v_j \in N(u)|}{\binom{d_u}{2}}$$



(example from Hamilton, 2020)

Power law

- Power law describes relations between the objects in the network
  - Very characteristic for the networks generated within some kind of social process
  - Describes **scale invariance** found in many natural phenomena (including physics, biology, sociology, economy and linguistics)
- In the analysis of information networks, we usually deal with power law distributed graphs

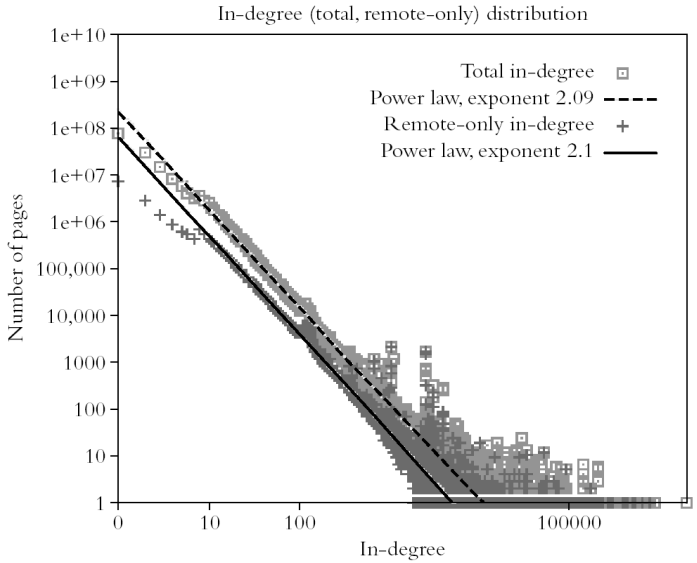
Power law on the Web

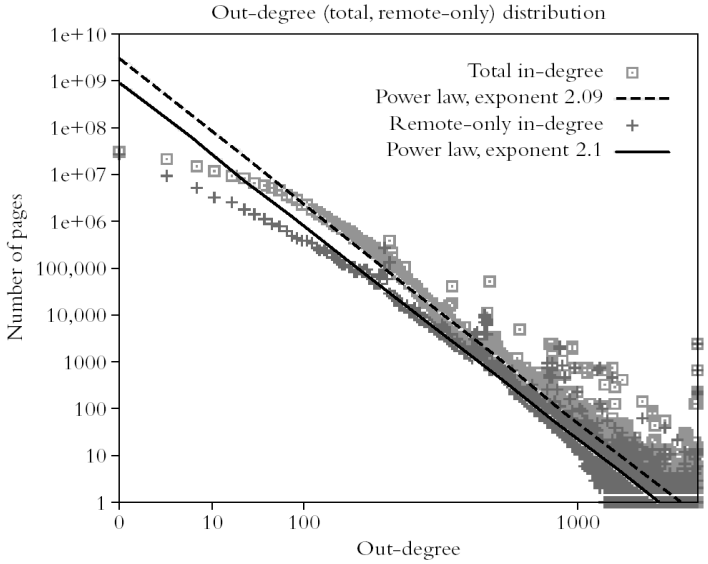
- In the context of Web the power-law appears in many cases:
  - Web pages sizes
  - Web page connectivity
  - Web connected components' size
  - Web page access statistics
  - Web Browsing behavior
- Formally, power law describing web page degrees are:

$$\Pr(\text{out-degree is } k) \propto 1/k^{a_{\text{out}}}$$

$$\Pr(\text{in-degree is } k) \propto 1/k^{a_{\text{in}}}$$

(This property has been preserved as the Web has grown)





Small world theory

- Empirical observation for the Web-Graph is that the diameter of the Web-Graph is small relative to the size of the network
  - This property is called "Small World"
  - Formally, small-world networks have diameter exponentially smaller than the size
- By simulation it was shown that for the Web-size of 1B pages the diameter is approx. 19 steps
  - Empirical studies confirmed the findings

Example of Small World: project collaboration network

- The network represents collaboration between institutions on projects funded by European Union
  - 7886 organizations collaborating on 2786 projects
  - Each node is an organization, two organizations are connected if they collaborate on at least one project
- Small world properties of the collaboration network:
  - **Main connected part** of the network contains 94% of the nodes
  - **Max distance** between any two organizations is 7 steps … meaning that any organization can be reached in up to 7 steps from any other organization
  - **Average distance** between any two organizations is 3.15 steps (with standard deviation 0.38)
  - 38% (2770) of organizations have avg. distance 3 or less

Modeling the Web Growth

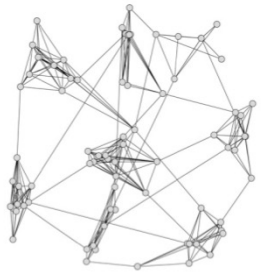
- Links/Edges in the Web-Graph are not created at random
  - Probability that a new page gets attached to one of the more popular pages is higher than to a one of the less popular pages
  - Intuition: “rich gets richer” or “winners takes all”
  - Simple algorithm “Preferential Attachment Model” (Barabasi, Albert) efficiently simulates Web-Growth

Community detection

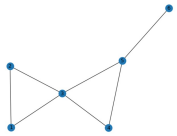
Network analysis for information retrieval, M2 MALIA-MIASHS, Julien Velcin

Clusters and communities

- Identifying underlying structures
- Based on graph properties
- Partitioning vertices means:
  - many edges within
  - few edges between



Laplacian for undirected graphs



- For an undirected weighted G, $L_u = D - W$

  where D is the matrix with degrees in its diagonal

W

```
[[0 1 1 0 0 0]
 [1 0 1 0 0 0]
 [1 1 0 1 1 0]
 [0 0 1 0 1 0]
 [0 0 1 1 0 1]
 [0 0 0 0 1 0]]
```

D

```
[[2 0 0 0 0 0]
 [0 2 0 0 0 0]
 [0 0 4 0 0 0]
 [0 0 0 2 0 0]
 [0 0 0 0 3 0]
 [0 0 0 0 0 1]]
```

L

```
[[ 2 -1 -1  0  0  0]
 [-1  2 -1  0  0  0]
 [-1 -1  4 -1 -1  0]
 [ 0  0 -1  2 -1  0]
 [ 0  0 -1 -1  3 -1]
 [ 0  0  0  0 -1  1]]
```

- L captures interesting features on the G
- New space of the k first eigenvectors can be used for clustering

Features of the Laplacian (1)

- For any $\mathbf{f} \in \mathbb{R}^n$:

$$\mathbf{f}^T\mathbf{L}\mathbf{f} = \frac{1}{2}\sum_{i,i'=1}^{n} w_{i,i'}(f_i - f_{i'})^2$$

- L is **symmetric** and **positive semi-definite** (all $\boldsymbol{\lambda} \geq 0$)
- $0 = \lambda_1 \leq \lambda_2 \ldots \leq \lambda_n$ and $v_1 = \mathbb{1}$

Features of the Laplacian (2)

- Spectral decomposition gives us the number of strongly connected components

- Spectrum : $\boldsymbol{\lambda}$ = [0, 0, 1, 2, 3]

multiplicity linked to # of connected components

Normalized Laplacian

- L have normalized forms, such as:

$$\mathbf{L}_{\mathrm{RW}} = \mathbf{D}^{-1}\mathbf{L}$$ (related to the transition matrix)

$$\mathbf{L}_{\mathrm{sym}} = \mathbf{D}^{-\frac{1}{2}}\mathbf{L}\mathbf{D}^{-\frac{1}{2}}$$ (symmetric version)

with L_{sym} eigenvalues: $0 = \lambda_1 \leq \ldots \leq \lambda_n \leq 2$

Spectral clustering

- Eigen (spectral) decomposition of L is linked to the min-cut problem :

$$Cut(C_1, C_2 \ldots C_s) = \frac{1}{2}\sum_{m=1}^{s} w(C_m, \bar{C_m})$$

- Better optimization criterion can be found with RatioCut and Ncut
- For the RatioCut:

$$\min_{\mathcal{A}\subset\mathcal{V}} \mathbf{a}^\top \mathbf{L}\mathbf{a} \quad s.t.$$

simplification to just two clusters $\mathcal{A}$ and $\bar{\mathcal{A}}$

$$\mathbf{a} \perp \mathbb{1}$$ ← **a** sums to 1

$$\|\mathbf{a}\|^2 = |\mathcal{V}|$$ ← number of vertices

- By relaxing to the continuous domain, the solution to the NCut problem is given by the k smallest eigenvalues of L (λ_2 for the binary case).

Cluster properties

- Density and separability:

$$\delta_{\text{int}}(G/C_1 \ldots, C_k) = \frac{1}{k}\sum_{i=1}^{k} \delta_{\text{int}}(C_i) \qquad \delta_{\text{int}}(C) = \frac{\left|\{\{u,v\}/u \in C, v \in C\}\right|}{|C| \times |C-1|}$$

$$\delta_{ext}(G/C_1 \ldots, C_k) = \frac{\left|\{\{u,v\}/u \in C_i, v \in C_j, i \neq j\}\right|}{n(n-1) - \sum_{l=1}^{k}(|C_l|(|C_l|-1))}$$

- Plenty of proposed "holistic" measures (or *fitness measures*), such as the modularity (Newman, 04)

Modularity (Newman, 06)

- Poor performance of spectral clustering for detecting natural communities
- Avoiding the trivial solution (one cluster!)
- Good division = the number of edges between groups is smaller than *expected*

Q = (number of edges within communities)
- (expected number of such edges)

Optimizing modularity

- "Expected number" => null model
- Modularity Q:

$$Q = \frac{1}{2m}\sum_{C}\sum_{i,j\in C}(A_{ij} - P_{ij}) \qquad Q = \frac{1}{2m}\sum_{C}\sum_{i,j\in C}(A_{ij} - \frac{k_i k_j}{2m})$$

- Spectral optimization of Q
  - For 2 communities: "leading eigenvector" method
  - Generalization for more than 2 communities

Leading eigenvector method

- index vector s : (1,1,1,-1,1…-1,-1,1)
- Q can be rewritten as:

$$Q = \frac{1}{4m} s^T B s \quad \text{with} \quad B_{ij} = A_{ij} - P_{ij}$$

- B = modularity matrix
- Maximizing Q => PCA decomposition of B
- Leading eigenvector method = choosing s closest to the leading eigenvector u (+1 if u_i>=0, -1 otherwise)

Louvain algorithm

- Proposed by Blondel et al., 2008
- Greedy approximation for finding communities, based on modularity
- Basically: initialize with small communities then merge as soon as the modularity is increased
- Complexity in O(n.log n), where n is the number of nodes
- Really efficient, implemented in Gephi https://gephi.org
- Can be found in python, for instance https://scikit-network.readthedocs.io

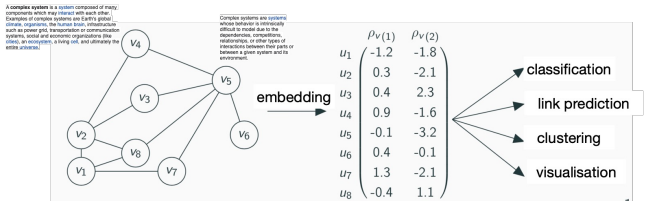
Machine learning with graphs

- Define which object is primarily concerned:
  - vertice / node
  - edge
  - graph as a whole
- What kind of ML do you target?
  - unsupervised
  - supervised
  - semi-supervised
  - self-supervised

Representing graph data

Network analysis for information retrieval, M2 MALIA-MIASHS, Julien Velcin

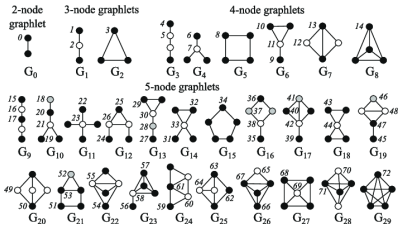
Document networks



- **Document network**: “graph of vertices, where each vertex is associated with a text document” (Tuan et al., 2014), e.g.: scientific articles, newspapers, social media…
- Embedding for building a **joint** space for solving downstream tasks (e.g., link prediction, node classification, community detection)

Using graph features

- For a graph:
  - structural properties (e.g., number connected components, width, density)
  - distributions over individual measures
  - graphlets
- For a node:
  - ego-network and all of the above
  - attributes labelling the node
  - centrality measures
  - clustering coefficient



Graph kernel

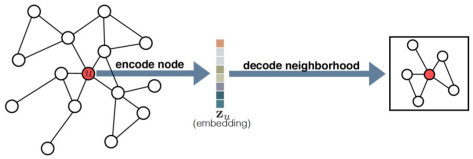
- Useful to calculate if two nodes / graphs are similar
- Some kernels:
  - Node level statistics and features (e.g., centrality, clustering coef)
  - Overlapping measures on the neighborhood (local and global)
  - Patterns, such as closed triangles, graphlets, etc.
  - Graphs as « bag of nodes »
  - Path-based methods
  - Personalized PageRang (PPR)
  - Weisfeiler-Lehman kernel (iterative neighborhood aggregation)

Neighborhood reconstruction

Network analysis for information retrieval, M2 MALIA-MIASHS, Julien Velcin

Encoder-decoder perspective

- We can adopt an **encoder-decoder** perspective for learning how to encode node into a latent space (Hamilton, 2020) :



- Example of a classic decoder: $\mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}^+$

rep of v_i rep of v_j edge (v_i, v_j)

Optimizing an Encoder-Decoder Model

$$\mathcal{L} = \sum_{(u,v)\in\mathcal{D}} \ell\left(\mathrm{DEC}(\mathbf{z}_u, \mathbf{z}_v), \mathbf{S}[u,v]\right)$$

with

- l: loss function
- DEC: pairwise decoder
- S: proximity measure

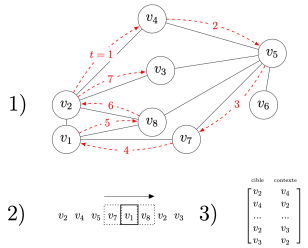
Laplacian eigenmaps (Belkin and Niyogi, 2002)

- Based on spectral clustering
- Decoder: $\mathrm{DEC}(z_u, z_v) = \|z_u - z_v\|_2^2$
- Loss function: $\mathcal{L} = \sum_{(i,j)\in V} \mathrm{DEC}(z_i, z_j).S(i,j)$
- If S = the adjacency matrix, the solution is provided by the matrix of eigenvectors corresponding to the **lowest eigenvalues** of the (generalized) eigenvalue problem:
- Note that other decoders make use of the dot product

$$\mathrm{DEC}(z_i, z_j) = z_i^T.z_j$$

Node2vec and DeepWalk

- Similarity matrix S can be extended to farther nodes (A^2, A^3…) and it can be *approximated* by random walks
- **Main idea:** two nodes are similar if they co-occur on the same paths



Node2vec and DeepWalk

- We try to approximate p(v|u), probability to *visit* v when we *start* at u
- The higher p(v|u), the higher $z_u^T.z_v$
- Several solutions:
  - DeepWalk (Perozzi et al., 2014): hierarchical softmax for approximate

$$\mathcal{L} = \sum_{(u,v)\in D} -\log\left(\frac{e^{z_u^T.z_v}}{\sum_{v_k\in V} e^{z_u^T.z_k}}\right)$$

  - Node2vec (Grover & Leskovec, 2016): noise contrastive approach

$$\mathcal{L} = \sum_{(u,v)\in D} -\log(\sigma(z_u^T.z_v)) - \gamma\mathbb{E}_{v_n\sim P_n(V)}[\log(\sigma(-z_u^T.z_{vn}))]$$

positive examples — negative examples

Link DeepWalk / spectral clustering

- Qiu et al. (2018) shows that the embeddings Z of DeepWalk satisfies:

$$\mathbf{Z}^\top\mathbf{Z} \approx \mathbf{S}_{\mathrm{DW}} \text{ avec } \mathbf{S}_{\mathrm{DW}} = \log\left(\frac{\mathrm{vol}(\mathcal{V})}{T}\left(\sum_{t=1}^{T}\mathbf{P}^t\right)\mathbf{D}^{-1}\right) - \log(b)$$

length of the random walk

$$\mathbf{D}^{-\frac{1}{2}}\left(\mathbf{U}\left(\sum_{t=1}^{T}\Lambda^t\right)\mathbf{U}^\top\right)\mathbf{D}^{-\frac{1}{2}}$$

- $\mathbf{U}\Lambda\mathbf{U}^\top = \mathbf{L}_{\mathrm{sym}}$ is the *eigendecomposition* of the normalized Laplacian
- They show that LINE (Tang et al., 2015) is a special case of DeepWalk (T=1)

GVNR (Brochier et al., 2019)

- Following GloVe (Pennington et al., 2014), GVNR solves a **regression** task on the weighted cooccurrence matrix X where cells with small values are set to 0 (> threshold x_{min})
- We're looking for (U, b^U) and (V, b^V) s.t.:

$$\arg\min_{U,V,b^U,b^V} \sum_i^n \sum_j^n s(x_{ij})(u_i.v_j + b_i^U + b_j^V - log(1+x_{ij}))^2$$

with $s(x_{ij}) = 1$ if $x_{ij} > 0$ and $m_i \sim B(\alpha)$ else

where α is chosen s.t. $m = k$ in average

$$X = \begin{pmatrix} 0 & 4 & 8 & 0 & 0 & 0 \\ 4 & 0 & 0 & 3 & 0 & 0 \\ 8 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 6 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 6 & 1 & 0 \end{pmatrix}$$

Some results with GVNR

Table 3: Accuracy on the citation (1) network.

	% of training data				
	10%	20%	30%	40%	50%
GloVe	57.7	62.4	69.5	72.8	73.8
GVNR (x_{min} = 0)	58.5	62.5	70.7	73.4	75.0
NetMF	65.7	**72.9**	**76.4**	**78.6**	79.4
DeepWalk	67.8	71.6	74.5	75.8	79.2
GVNR (x_{min} = 1)	**69.5**	72.6	75.9	78.1	**80.2**

Table 4: Accuracy on the citation (2) network.

	% of training data				
	10%	20%	30%	40%	50%
GloVe	42.8	53.5	55.3	56.2	56.8
GVNR (x_{min} = 0)	38.7	46.8	49.1	50.4	50.9
NetMF	**51.2**	54.8	55.1	55.0	54.8
DeepWalk	41.3	52.5	54.5	55.5	56.0
GVNR (x_{min} = 1)	45.6	**55.6**	**57.3**	**58.7**	**59.0**

Table 5: Accuracy on the co-authorship network.

	% of training data				
	10%	20%	30%	40%	50%
GloVe	41.0	42.1	43.7	46.4	51.2
GVNR (x_{min} = 0)	60.3	64.6	67.4	67.1	68.2
NetMF	60.7	66.2	70.1	72.1	72.8
DeepWalk	**75.4**	**77.2**	**77.3**	**75.9**	**79.3**
GVNR (x_{min} = 1)	74.7	75.3	76.3	73.8	74.6

Table 6: F_1 score on the language network.

	% of training data				
	10%	20%	30%	40%	50%
GloVe	**34.0**	**44.1**	**46.7**	**47.7**	**48.6**
GVNR (x_{min} = 0)	31.7	40.7	43.2	44.7	45.1
NetMF	27.5	33.5	36.2	37.7	38.7
DeepWalk	33.6	43.6	46.2	47.6	48.2
GVNR (x_{min} = 1)	32.2	41.7	44.0	45.2	46.1

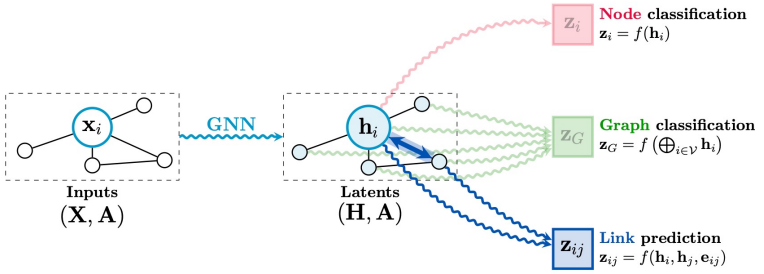
Limits of these approaches

- shallow embedding : one unique vector for each node
- no parameter sharing
- usually, no node features
- mostly transductive

Analyzing information networks

Network analysis for information retrieval, M2 MALIA-MIASHS, Julien Velcin

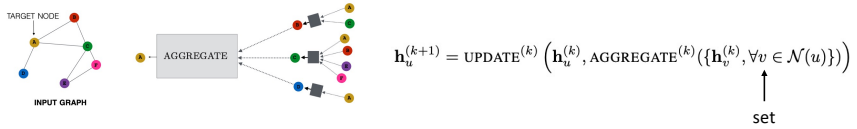
Graph Neural Networks



(Veličković, 2023)

New kind of deep learning architecture

- Permutation **invariance** and **equivariance**:
  - $f(PAP^T) = f(A)$ *permutation invariance*
  - $f(PAP^T) = Pf(A)$ *permutation equivariance*
  - with P a permutation matrix
- Neural **message passing**



GCN (Kipf et al., 2017)

- In the message passing formulation:

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

- It should be noted that all this is grounded in term of an analogy to a spectral formulation of a **convolution** operator on a graph
- One layer of the GCN computes:

$$H^{(n)} = \text{ReLU}(\tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}H^{(n-1)}W^{(n)})$$

with H^0=X (feature vector), $\tilde{A}$ = A + I (normalization trick), $W^{(n)}$ the trainable weights

GAT (Veličković et al., 2018)

- GAT adds an *attention* weight to the neighbors:

$$h_i^{(n)} = \sum_{v_j \in \mathcal{N}(i)} \alpha_{ij}^{(n)} h_j^{(n-1)} W^{(n)}$$

- The attention is calculated on the latent representations:

$$\alpha_{ij}^{(n)} = \frac{\exp\left(\text{LeakyReLU}\left(a \cdot \left[W^{(n)}h_i^{(n-1)}, W^{(n)}h_j^{(n-1)}\right]\right)\right)}{\sum_{k \in \mathcal{N}(i)} \exp\left(\text{LeakyReLU}\left(a \cdot \left[W^{(n)}h_i^{(n-1)}, W^{(n)}h_k^{(n-1)}\right]\right)\right)}$$

a an additional parameter vector
[.,.] = concatenation

- As the GCN, we can **stack** GAT layers to go beyond 1-hop

GATv2 (Brody et al., 2022)

- GAT computes static attention:

linear

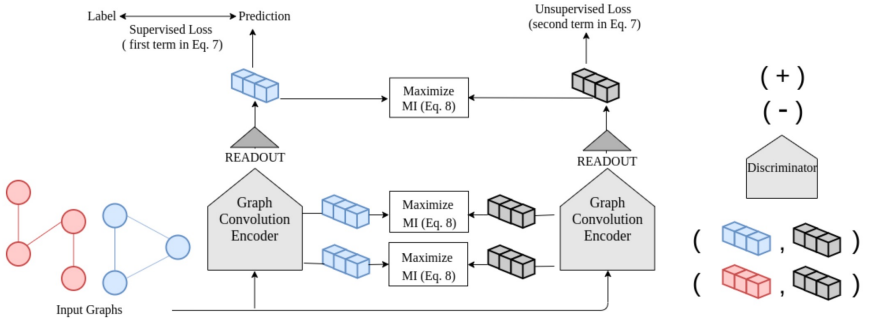
$$\alpha_{ij}^{(n)} = \frac{\exp\left(\text{LeakyReLU}\left(a \cdot \left[W^{(n)}h_i^{(n-1)}, W^{(n)}h_j^{(n-1)}\right]\right)\right)}{\sum_{k \in \mathcal{N}(i)} \exp\left(\text{LeakyReLU}\left(a \cdot \left[W^{(n)}h_i^{(n-1)}, W^{(n)}h_k^{(n-1)}\right]\right)\right)}$$

- GATv2 is more expressive by using dynamic attention, similarly to the early work on (Bahdanau et al., 2014), with a simple trick:

$$e(\boldsymbol{h}_i, \boldsymbol{h}_j) = \text{LeakyReLU}\left(\boldsymbol{a}^\top \cdot [\boldsymbol{W}\boldsymbol{h}_i \| \boldsymbol{W}\boldsymbol{h}_j]\right)$$

$$e(\boldsymbol{h}_i, \boldsymbol{h}_j) = \boldsymbol{a}^\top \text{LeakyReLU}\left(\boldsymbol{W} \cdot [\boldsymbol{h}_i \| \boldsymbol{h}_j]\right)$$

Beyond node representations: Deep Graph InfoMax (Veličković et al., 2019), InfoGraph (Sun et al., 2020)



(Sun et al., 2020)

About Information Retrieval

Network analysis for information retrieval, M2 MALIA-MIASHS, Julien Velcin

Recent developments in IR

- Leveraging LLMs, see ColBERT (Khattab and Zaharia, 2020) and ColBERT v2 (Santhanam et al., 2022)
- Retrieval Augmented Generation (RAG) https://ai.meta.com/blog/retrieval-augmented-generation-streamlining-the-creation-of-intelligent-natural-language-processing-models/
- Natural Language is All a Graph Needs (Ye et al., arXiv 2023)

Some references

- Graph Representation Learning. W. Hamilton. Morgan and Claypool, 2020. https://www.cs.mcgill.ca/~wlh/grl_book/
- A Tutorial on Spectral Clustering. U. von Luxburg. Statistics and Computing, 17 (4), 2007 https://people.csail.mit.edu/dsontag/courses/ml14/notes/Luxburg07_tutorial_spectral_clustering.pdf
- Network Science. Barabási A.L., Cambridge University Press, 2016. http://networksciencebook.com
- Everything is Connected: Graph Neural Networks. Petar Velickovic, 2023. https://paperswithcode.com/paper/everything-is-connected-graph-neural-networks