Augmentez vos données avec les algorithmes de graphes …

Estelle Scifo
PyConFR 2019, Bordeaux

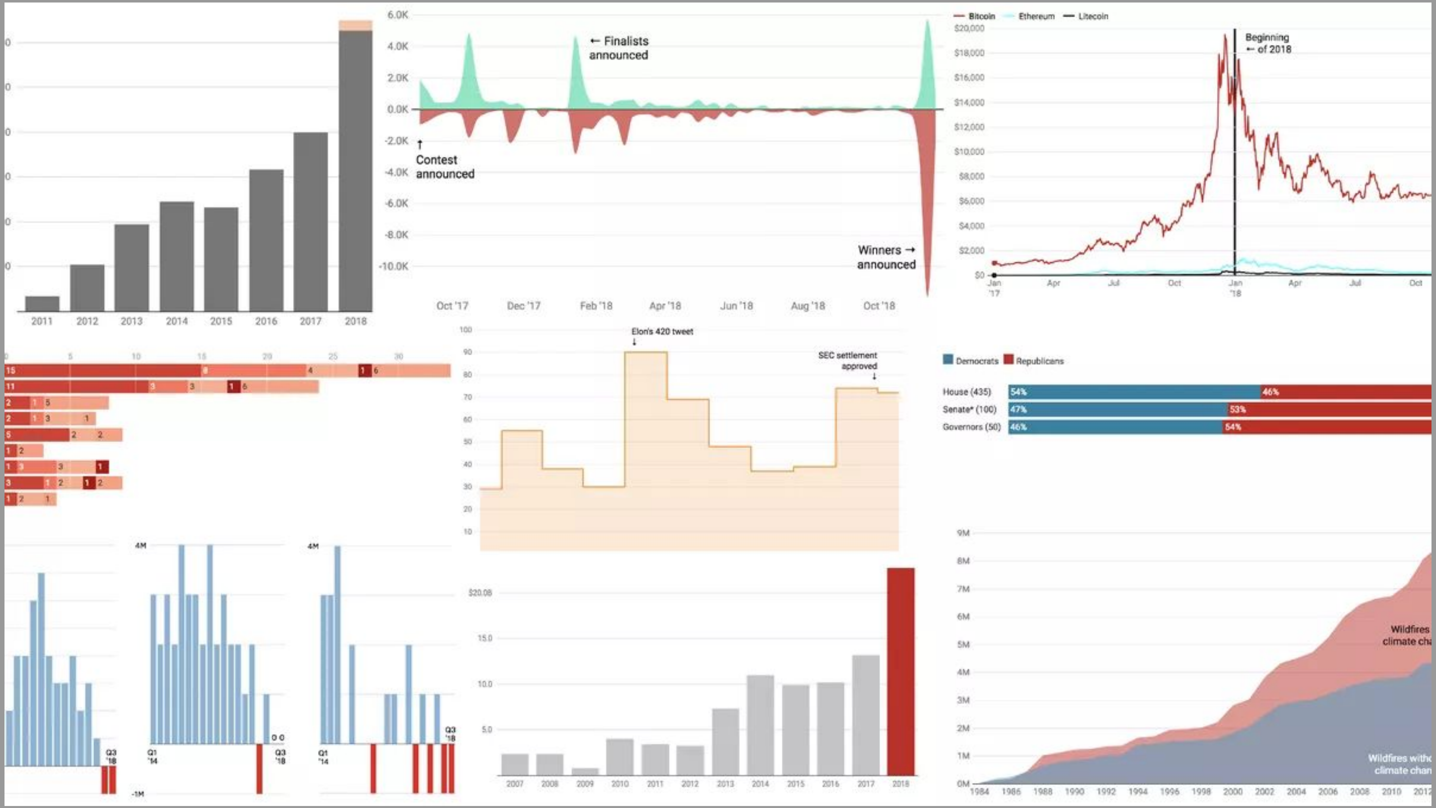
About me

- Physicist & Data scientist (Luxembourg/World/Remote)
- Graph enthousiast
  - Last project: neomap, visualization tool for Neo4j (written in React)
- Slides and code samples available on my github

  github.com/stellasia/pyconfr19
- Get in touch via twitter/linkedin!

twitter: @st3llasia
github: stellasia
linkedin: estellescifo

Graphes



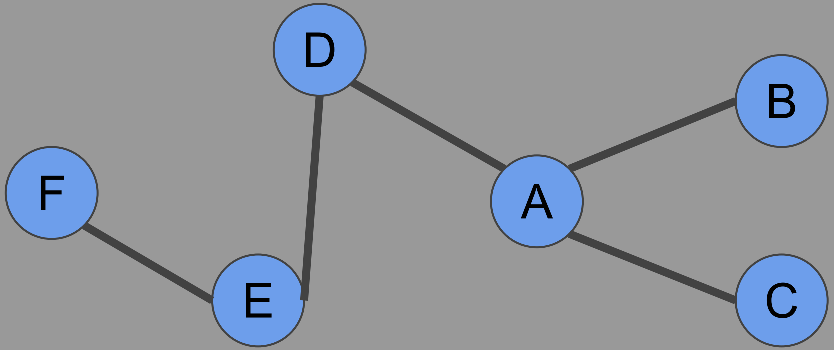
https://www.vox.com/2018/12/13/18106455/best-of-2018-data-charts-tech-end-year-list-amazon-facebook-juul-moviepass-elon-musk

Graphes

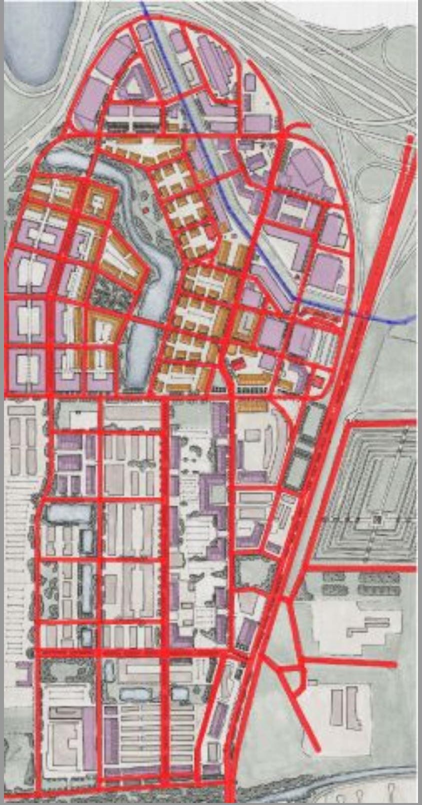
6.0K
2011
2012
2013
2014
2015
2016
Ethereum
Litecoin
Beginning
← of 2018
Oct
Jan
'18
Apr
Jul
Oct
46%
53%
54%
4M
Q1
'14
Q3
'18
-1M
10.0
5.0
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
Wildfires
climate cha
Wildfires witho
climate chan
4M
3M
2M
1M
0M
1984
1986
1988
1990
1992
1994
1996
1998
2000
2002
2004
2006
2008
2010
2012

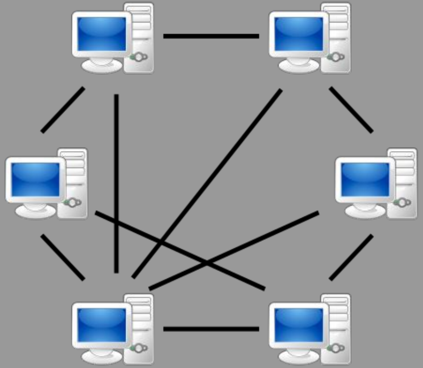
https://www.vox.com/2018/12/13/18106455/best-of-2018-data-charts-tech-end-year-list-amazon-facebook-juul-moviepass-elon-musk

Graphes

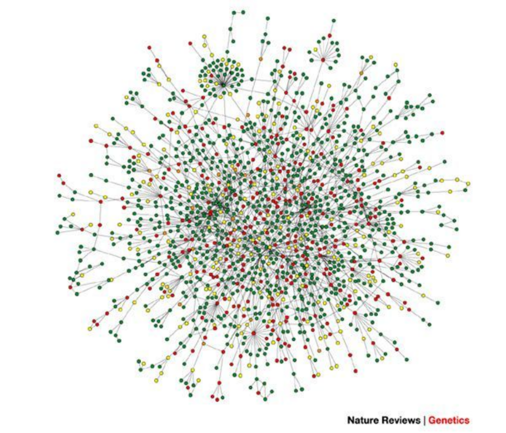


Des réseaux…





Mais aussi…



Proteins interaction network
https://www.nature.com/articles/nrg1272

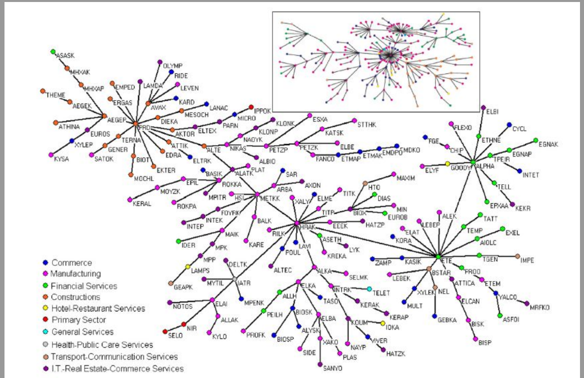
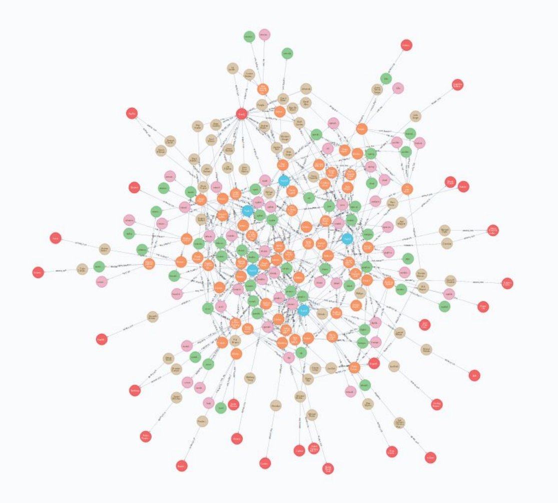

Commerce
Manufacturing
Financial Services
Constructions
Hotel-Restaurant Services
Primary Sector
General Services
Health-Public Care Services
Transport-Communication Services
I.T.-Real Estate-Commerce Services

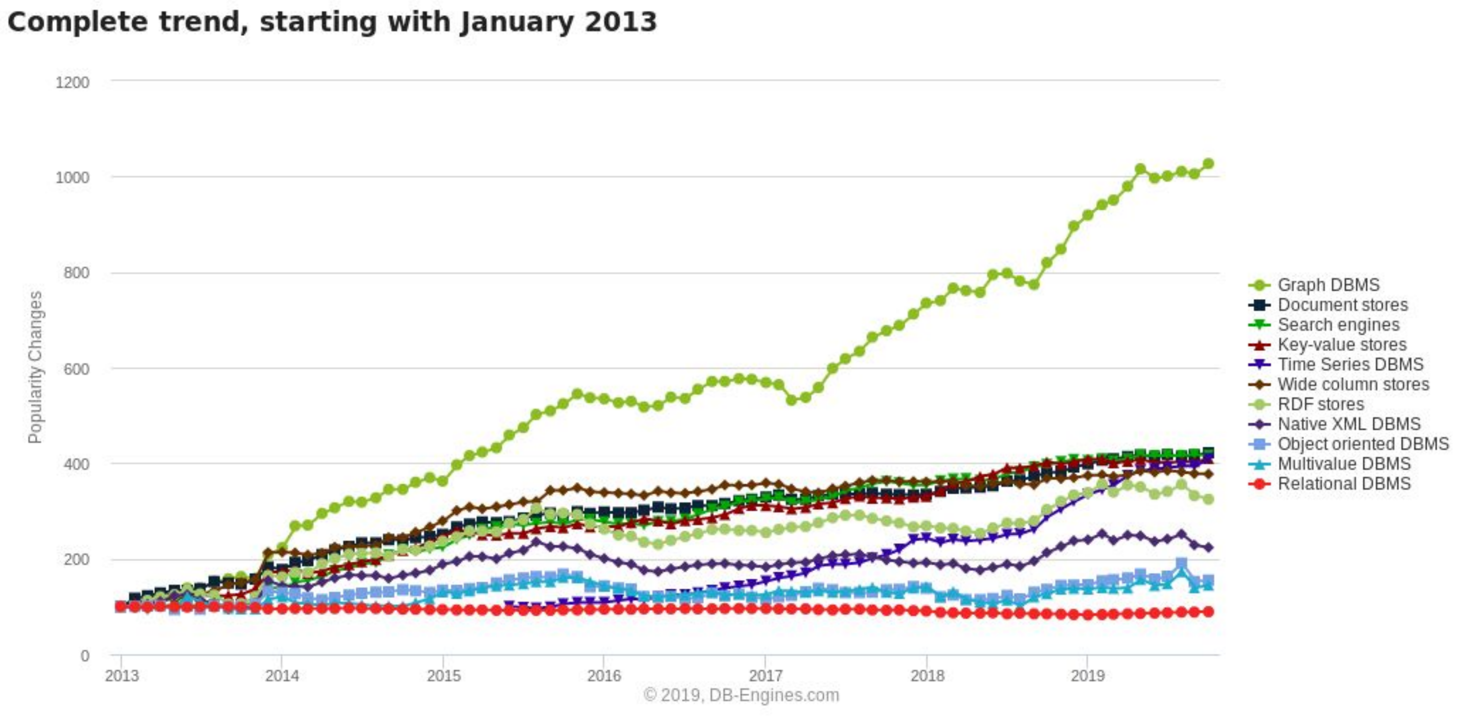
Fig. 1. Minimum spanning tree (MST) obtained by considering the daily logarithmic return time series of all stocks traded in the Greek stock market (Athens Stock Exchange, ASE) in year 1997. The different colours represent the different sectors of economic activity. The existence of stocks that behave as hubs, having more connections than the average, is evident. Inset: MST obtained using all stocks traded in ASE for the year 2000, which is in the middle of the 1999–2001 crisis period.

Stock market
http://kelifos.physics.auth.gr/publications/pdf/p133.pdf



Nodes2019 conference
@WilliamLyon

Pourquoi maintenant ?



Utilisations

- Extraction d’information (description)

- Extraction de “**graphy**” features pour ML

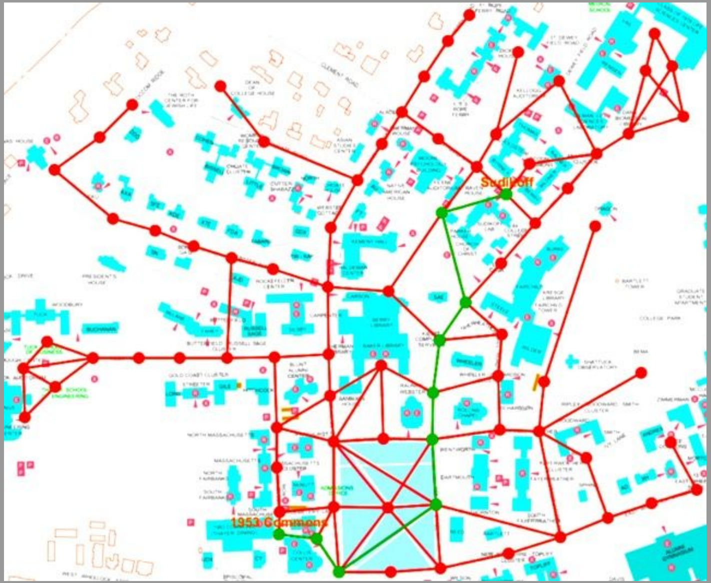
- Node/Graph embedding

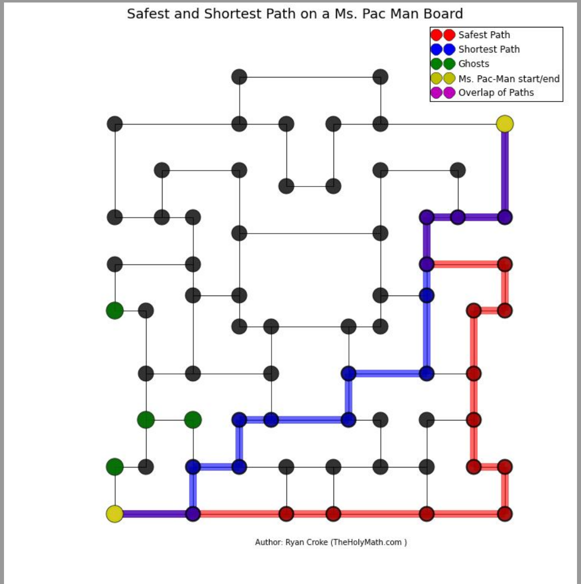
- Prédiction de liens

1. Algorithmes

Plus court chemin
(Shortest Path)

Shortest Path



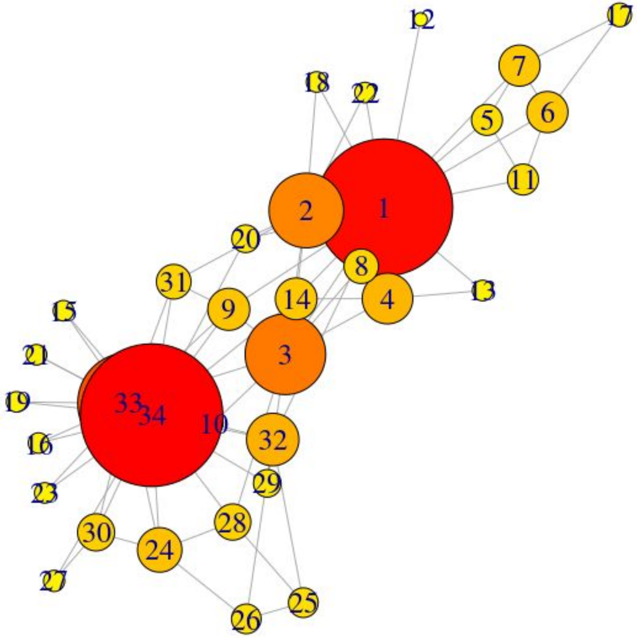

Safest and Shortest Path on a Ms. Pac Man Board
Safest Path
Shortest Path
Ghosts
Ms. Pac-Man start/end
Overlap of Paths
Author: Ryan Croke (TheHolyMath.com)

Importance (Centrality)

PageRank

“Nombre et importance des noeuds liés à `A`”

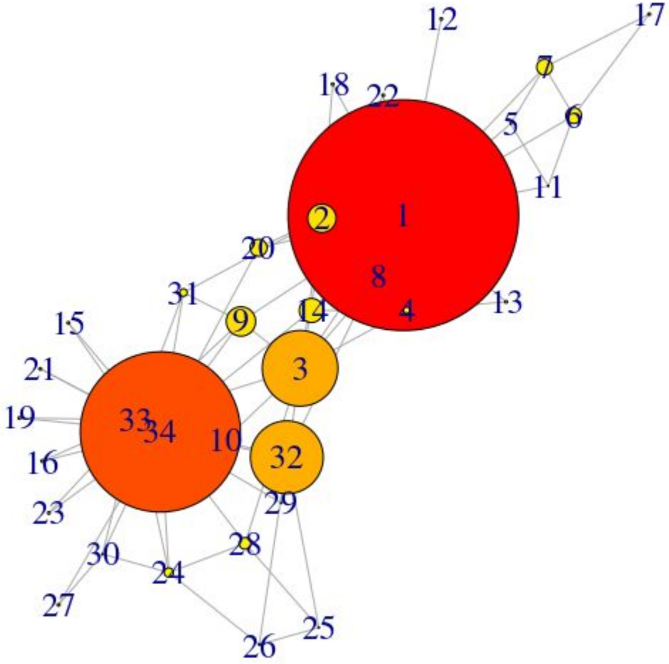
```
PR(A) =
(1-d) + d Σ (PR(i) / Ni)
```



Betweenness

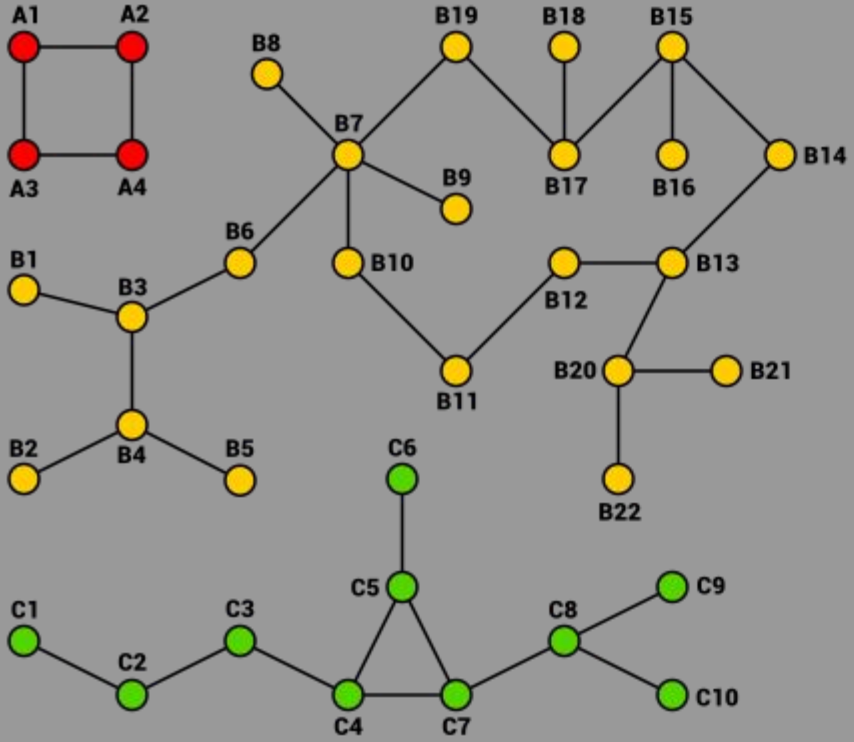
“Nombre de plus courts chemins entre deux noeuds i et j qui passent par `A`”

⇒ Noeud “critique”

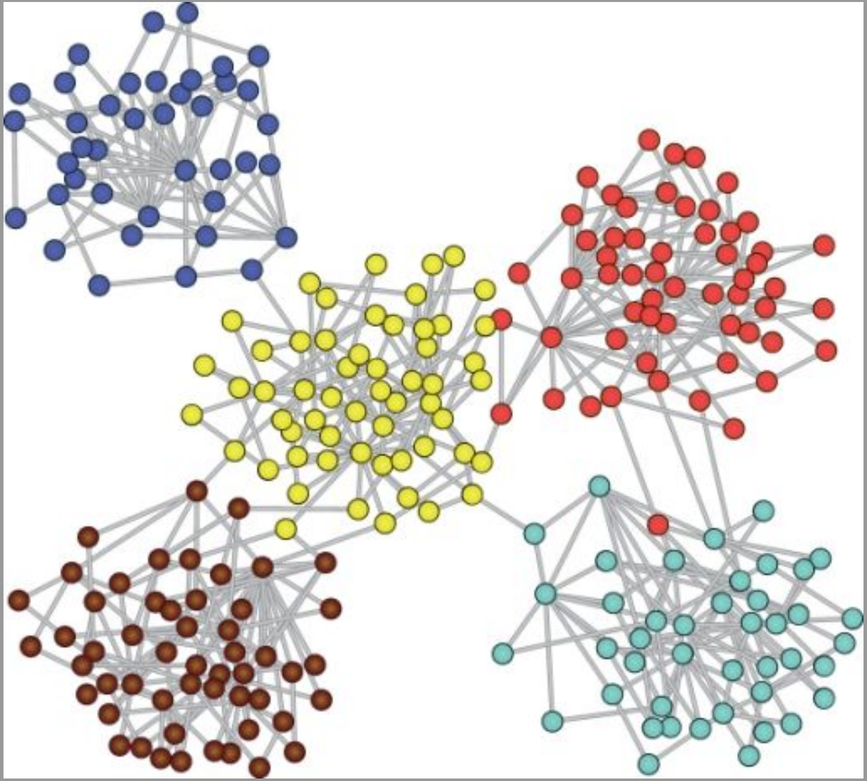


Communauté (Clustering)

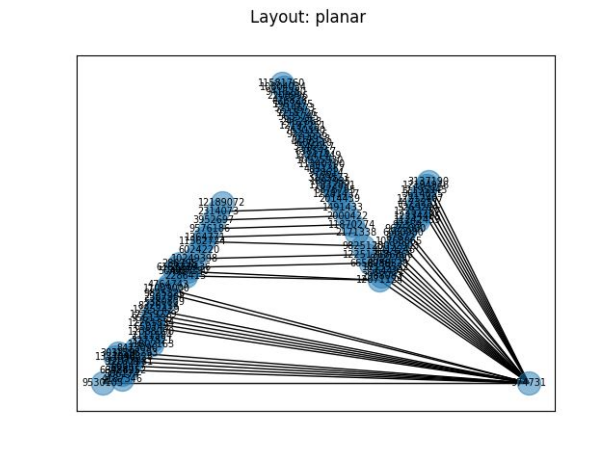
Communauté - Connected components

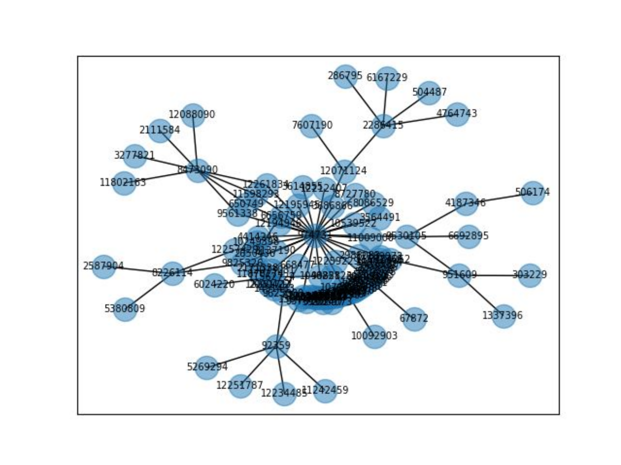


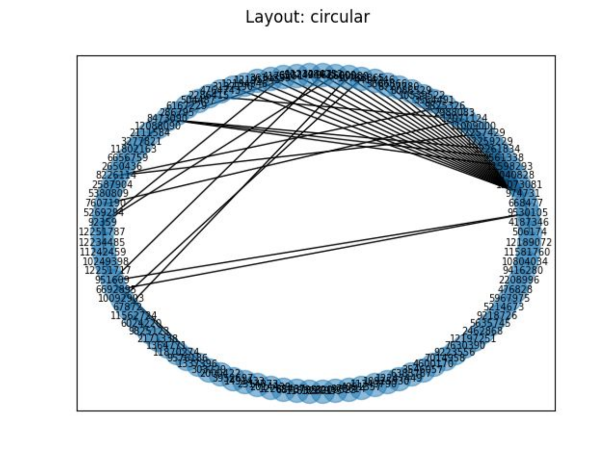
Communauté

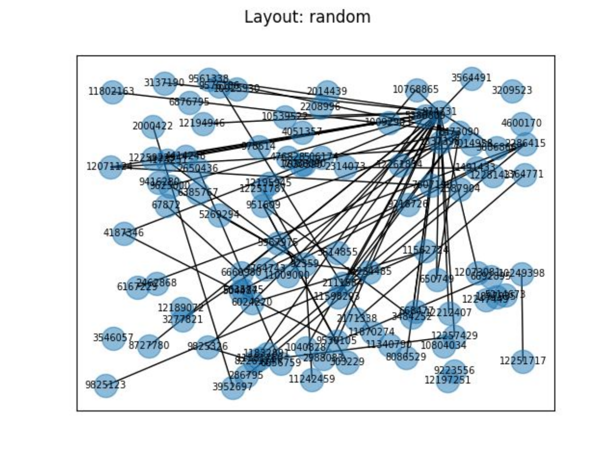


Visualisation


Layout: planar



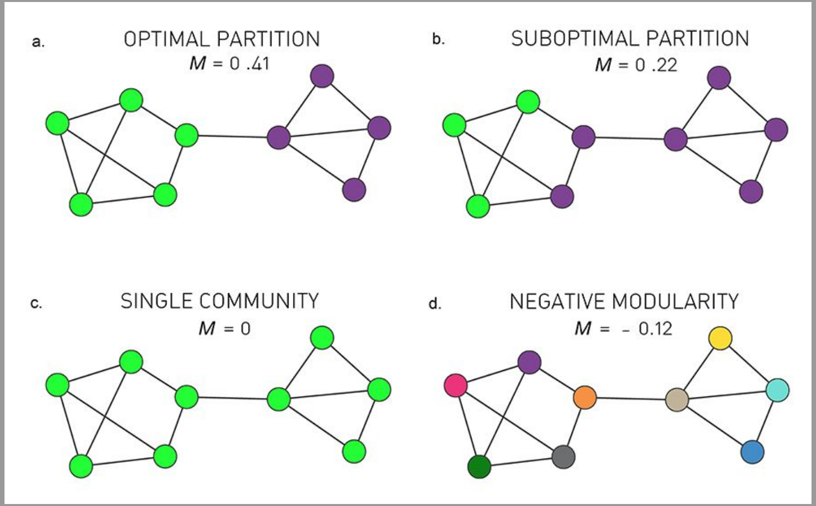

Layout: circular


Layout: random

Communauté - Louvain

- Modularité :
  Mesure la densité de lien à l'intérieur d'une communauté
  VS liens entre communautés

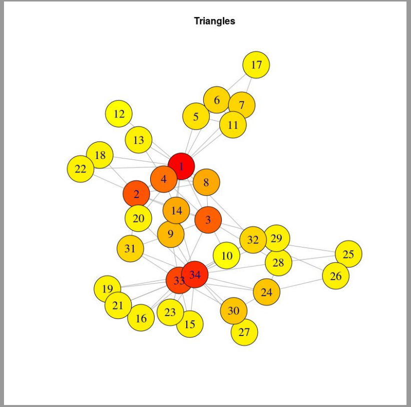
$$Q = \frac{1}{2m} \sum_{ij} \left[A_{ij} - \frac{k_i k_j}{2m} \right] \delta(c_i, c_j),$$



- ~ KMeans

Et d’autres…

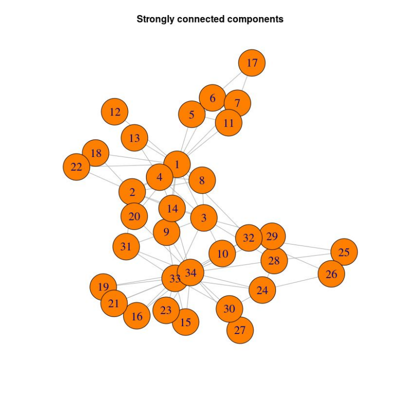
Triangles



Number of triangles using this node

Probability that two neighbours are also connected

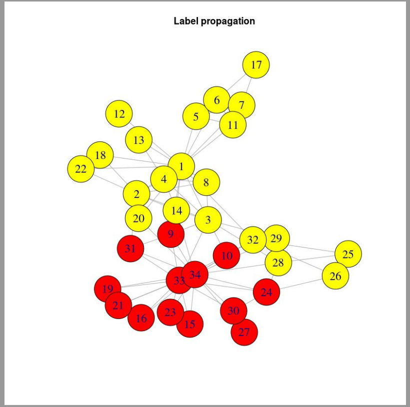
Components



Paths going through the node

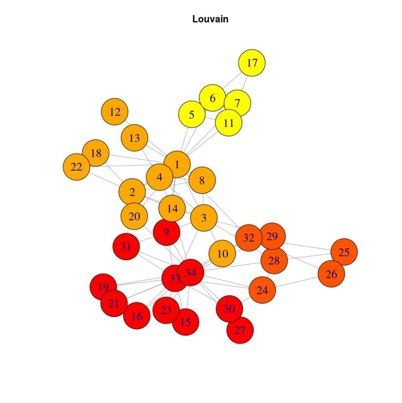
(weak: one direction
strong: both directions)

Label propagation



Label propagation to neighbours using the “majority vote” rule

Louvain



More relationships among nodes within a cluster than with nodes outside the cluster

Prédiction de lien
(Link Prediction)

Prédiction de lien

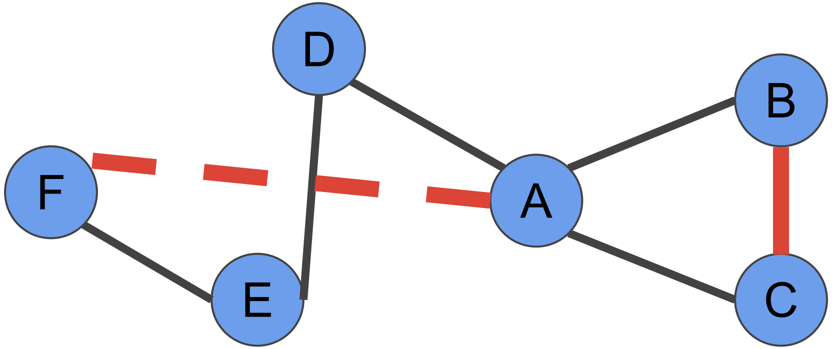
- Combler un manque d'information :
    - Étude de réseaux criminels

- Prédire des liens futurs :
    - Recommendation

Metrics for link prediction

- Common neighbours
  ⇒ ***Two people who have a friend in common are more likely to be introduced than those who don't have any friends in common.***
  ⇒ Size of the set of common friends between u and v:

$$C(u,v) = |\mathcal{N}(u) \cap \mathcal{N}(v)|$$

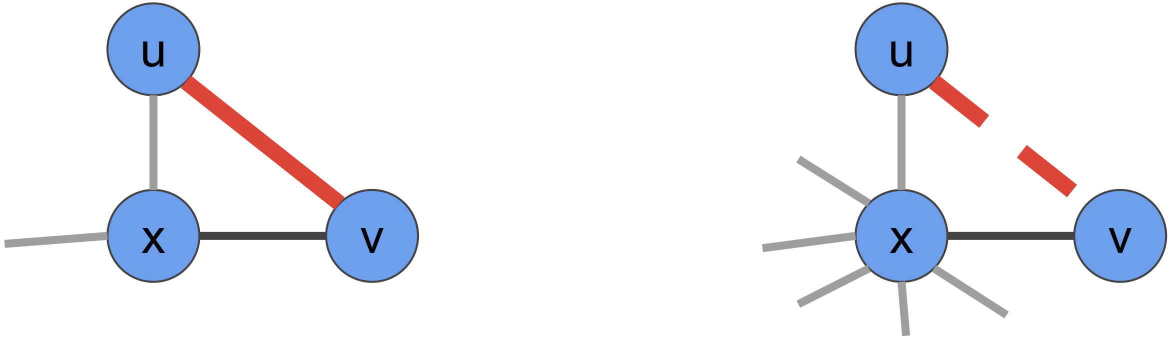


Metrics for link prediction

- Common neighbours
- Adamic-Adar

⇒ Rare connections are giving more informations than common ones

$$A(u,v) = \sum_{x \in \mathcal{N}(u) \cap \mathcal{N}(v)} \frac{1}{log|\mathcal{N}(x)|}$$

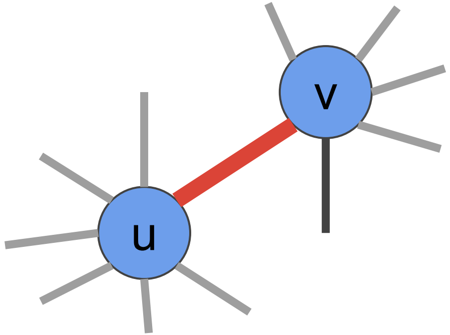


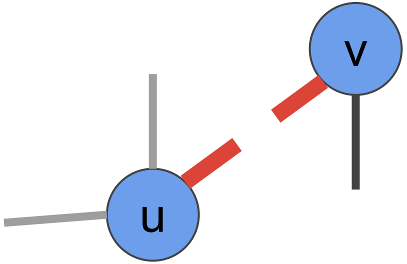
Metrics for link prediction

- Common neighbours
- Adamic-Adar
- Total neighbours

  ⇒ ***the more connected a node is, the more social it is, the more likely it is to receive new links***

$$T(u, v) = |\mathcal{N}(u) \cup \mathcal{N}(v)|$$



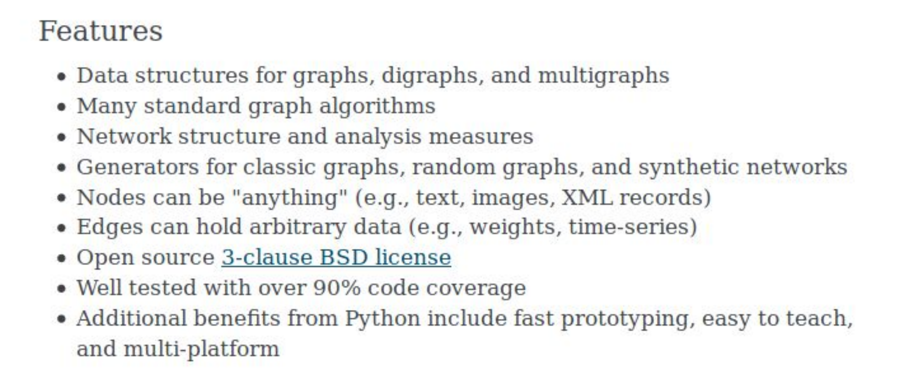


2. Exemple d’utilisation en python

Comment ?

- Base de données graphes :
  - Incluant des implémentations des principaux algorithmes (Neo4j)

- Package python : `networkx`



Features

- Data structures for graphs, digraphs, and multigraphs
- Many standard graph algorithms
- Network structure and analysis measures
- Generators for classic graphs, random graphs, and synthetic networks
- Nodes can be "anything" (e.g., text, images, XML records)
- Edges can hold arbitrary data (e.g., weights, time-series)
- Open source 3-clause BSD license
- Well tested with over 90% code coverage
- Additional benefits from Python include fast prototyping, easy to teach, and multi-platform

Export des données

- Créer un graphe :
  - A partir de fichiers csv

<u>nodes.csv:</u>

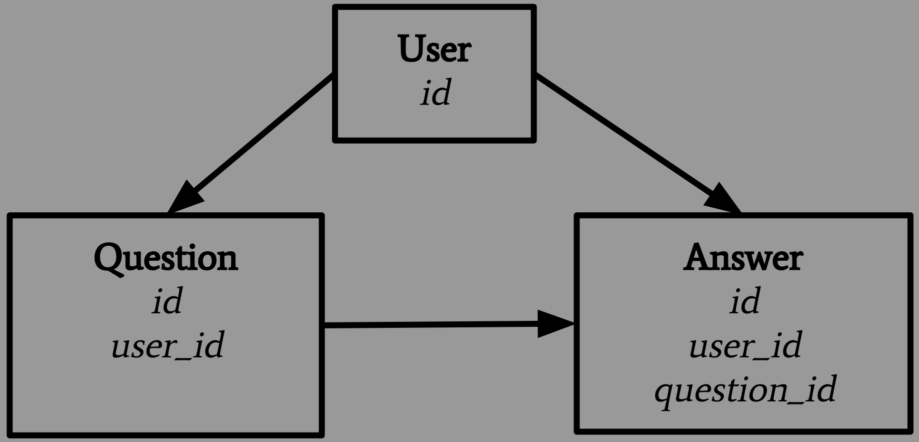
```
node
2014439
12281437
6876795
506174
11832891
12212407
668477
4764743
```

<u>edges.csv:</u>

```
node1, node2
506174,4187346
668477,974731
4764743,2286415
12073081,974731
1040828,974731
11598293,8473090
11598293,974731
12088090,8473090
```

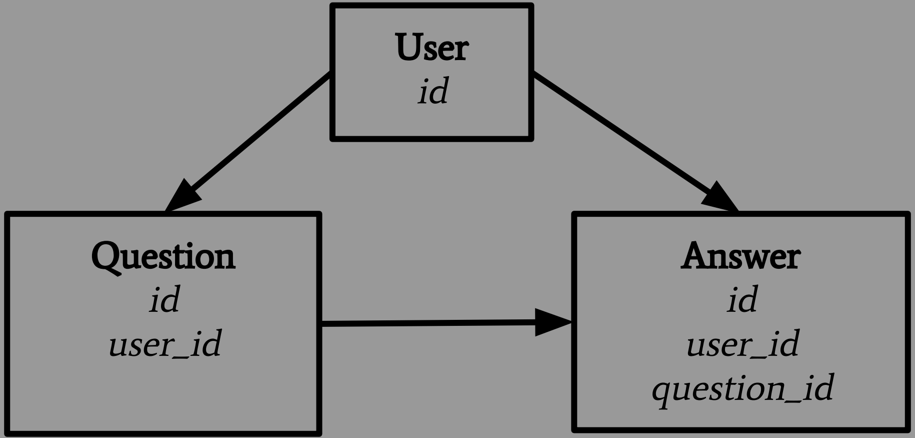
Export des données

- Créer un graphe :
  - Ex: base de données type StackOverflow:
    - **Liens entre les utilisateurs ?**



Export des données

- Créer un graphe :
  - Ex: base de données type StackOverflow:
    - User; Question: Answer
    - **Liens entre les utilisateurs ?**



User
id

Question
id
user_id

Answer
id
user_id
question_id

nodes.csv:

```
SELECT u.id
FROM users
```

edges.csv:

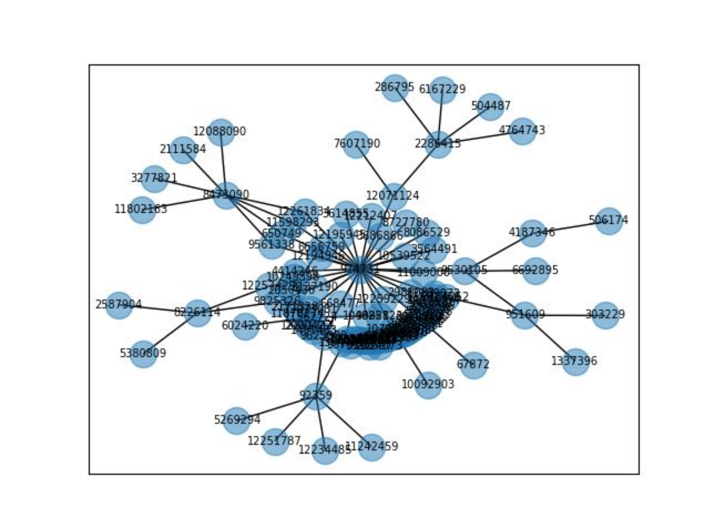
```
SELECT DISTINCT u1.id, u2.id
FROM user u1
JOIN question q ON q.user_id = u1.id
JOIN answer a ON a.question_id = q.id
JOIN user u2 ON a.user_id = u2.id
```

Import dans networkx

```
import networkx as nx

G = nx.Graph()

edges = nx.read_edgelist('edges.csv', delimiter=",")
nodes = nx.read_adjlist("nodes.csv")

G.add_edges_from(edges.edges())
G.add_nodes_from(nodes)
```

Visualisation

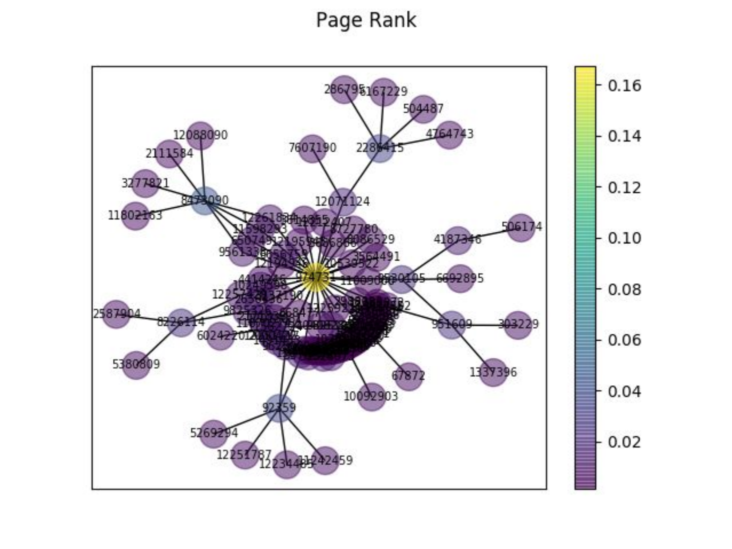
```
import networkx as nx
import matplotlib.pyplot as plt

pos = nx.kamada_kawai_layout(G)
f = plt.figure()

plt_nodes = nx.draw_networkx_nodes(
    G,
    pos,
    nodelist=G.nodes,
    alpha=0.5
)
nx.draw_networkx_edges(G, pos)
nx.draw_networkx_labels(G, pos, font_size=7)
```



2.1 Description des données

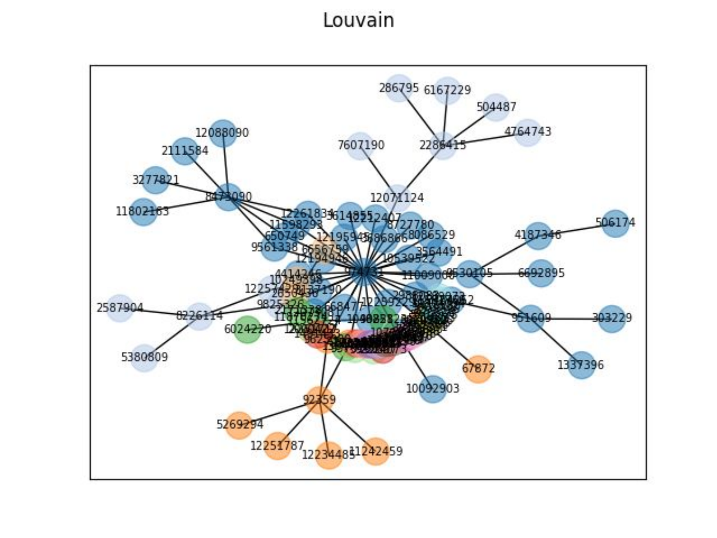
Algorithme : importance (PageRank)

```
res = nx.pagerank(G, alpha=0.85)
res_nodes = list(res.keys())
res_values = list(res.values())

f = plt.figure()
plt_nodes = nx.draw_networkx_nodes(
    G,
    pos,
    nodelist=res_nodes,
    node_color=res_values,
)
nx.draw_networkx_edges(G, pos)
nx.draw_networkx_labels(G, pos, font_size=7)
```



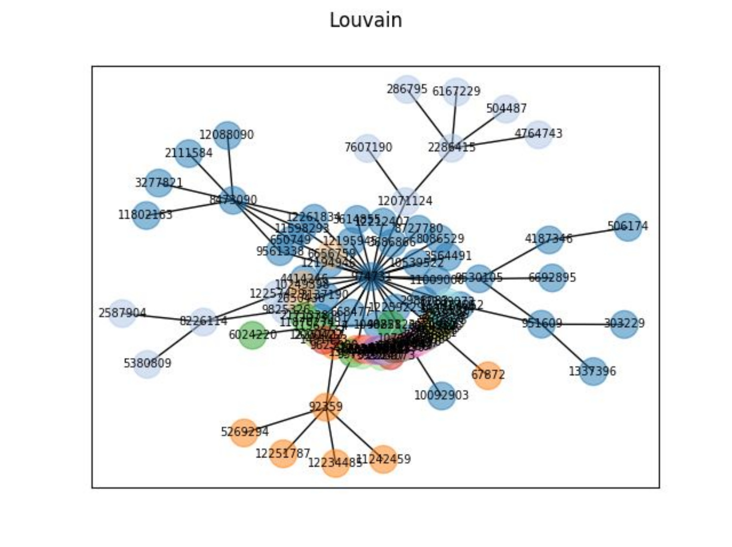
Algorithme : communauté (Louvain)

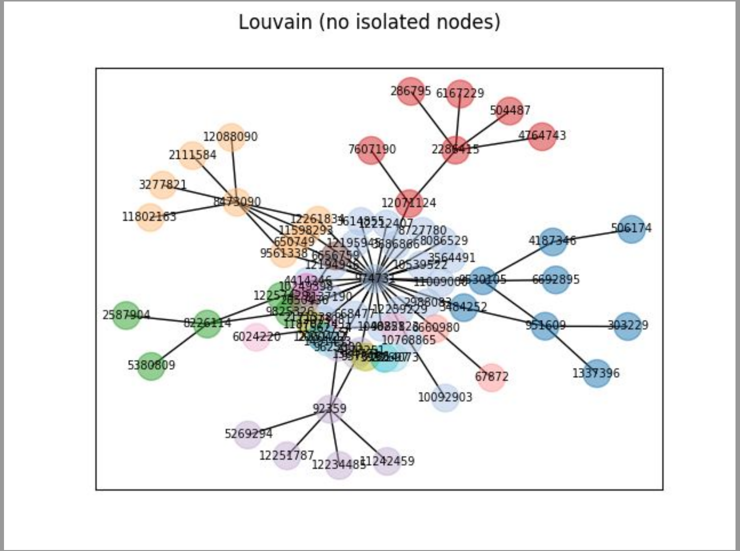
```
import community

res = community.best_partition(G)
res_nodes = list(res.keys())
res_values = list(res.values())

f = plt.figure()
plt_nodes = nx.draw_networkx_nodes(
    G,
    pos,
    nodelist=res_nodes,
    node_color=res_values,
)
nx.draw_networkx_edges(G, pos)
nx.draw_networkx_labels(G, pos, font_size=7)
```



Noeuds isolés

```
import networkx as nx

G.remove_nodes_from(list(nx.isolates(G)))
```





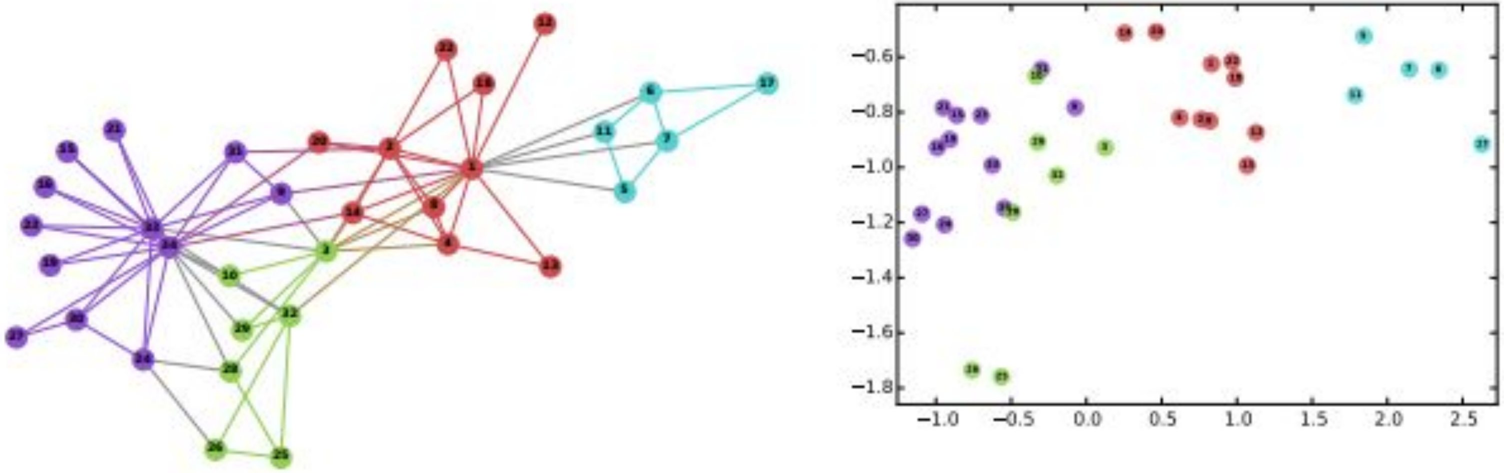
2.2 Prédiction

“Graphy” features

- Problème :
  - Est-ce qu’une personne donnée va venir à la prochaine PyConFR en 2020 ?

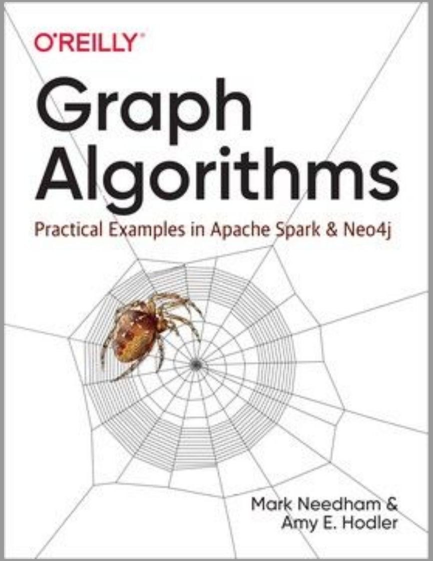
name *(str)*	age *(int)*	is_afpy_member *(bool)*	was_present_in_2019 *(bool)*	...	**Importance** ***(float)***	**Community** ***(int)***
Estelle	30	false	true	...	**0.01**	**4**
Pierre	22	true	true	...	**1.4**	**1**
….	….	….	….	...	**...**	**...**

Node embeddings



(a) Input: Karate Graph (b) Output: Representation

Conclusion

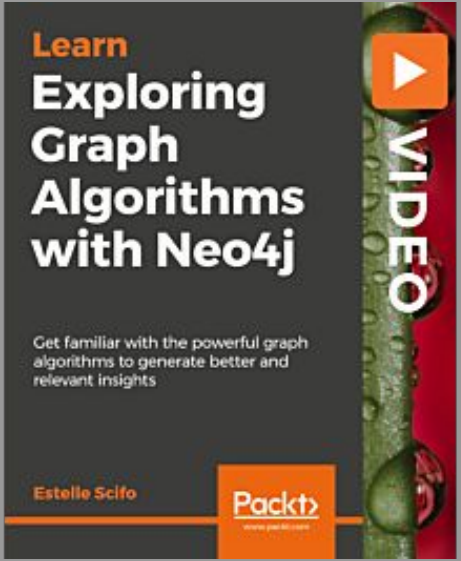


O'REILLY

Graph Algorithms

Practical Examples in Apache Spark & Neo4j

Mark Needham & Amy E. Hodler



Learn

Exploring Graph Algorithms with Neo4j

Get familiar with the powerful graph algorithms to generate better and relevant insights

Estelle Scifo

Packt>

VIDEO

twitter: @st3llasia
github: stellasia
linkedin: estellescifo