Why networks?

Pretty... so what?

I query! Cluster network data... wow

What kind of clusters can be found based on the similarity of packages?

"Bipartite network"

Newman Projection

Hairball

Backzone Filtering

Let's find communities

Small enough to visualize in Python

But still better with another program
Communities and more

Györgyi Vasarhelyi
Center For Network Science, CEU, Budapest
About me

PhD student at CEU’s Center for Network Science, Budapest, Hungary

Former data analyst

Python enthusiastic

Datapussyz, wanna be journalist

Research interest: temporal networks and team dynamics
Network = Graph + Data

directed

Who follows whom on Twitter

udirected

Friendship network on Facebook
Why networks?

a picture is worth a thousand words

understanding complexity

hidden correlations
#migrantinvasion

Reference:


Image credit: Irish Defence Forces
Pretty... so what?

Community detection

*built-in function:*

**Communities**

**K-Clique**

http://perso.crans.org/aynaud/communities/

http://igraph.org/python/doc/igraph.Graph-class.html

Visualization

http://matplotlib.org/

http://pygraphviz.github.io/

https://github.com/erocarrera/pydot

http://lightning-viz.org/
I suppose I have network data... so?

- Get data
- Figure out where the connections are
- Build up the network --- networkx, igraph
- Think again what is ur aim
- Do some preliminary analysis -- stats
- Visualize (if u can)
- Analyze the results
Data - Python Libraries Dependency Network

Source: @ogirardot
https://github.com/ogirardot/meta-deps/tree/a61d2c0ef8d246c3e666f2a191df303ea416da7

```python
import networkx as nx, json
from base64 import b64decode

data = []
with open('pypi-deps.csv', 'r') as file:
    for line in file:
        name, version, deps = line.split(' 
')
        deps = json.loads(b64decode(deps))
data += [(name, version, deps)]

G = nx.Graph()
edges_dict = {}
for ex in data:
    name, version, deps = ex
    G.add_node(ex[0])
    edges_dict[ex[0]] = (ex[0], ex[1])
    if not ' ' in deps:
        G.add_edge(ex[0], ex[1], dep.replace('', ' '))

nx.write_edgelist(G, 'Edges.txt')

G.edges()[:10]
```

Super easy to have a graph

Two nodes are connected if one is the dependency of the other
So, let's do some stats

Based on matplotlib u can create network analytical charts

We have very few nodes with high degree (setuptools), but most of them has less than 100 connections.

In the similarity of packages?
What kind of clusters can be found based on the similarity of packages?

"Bipartite network"

My network looks like it from inside

But I want to see how similar packages are

Arrows point on the package it depends on

Let's find communities

Newman Projection

original network

projected network based on common neighbors

Small enough to visualize in Python

Hairball :

Projection calculates similarity between all nodes, so we get this...

But based on weights (similarity) we can filter, but how?

Backbone Filtering

Depend enc filter is a network reduction algorithm to extract the backbone structure of undirected weighted network. I create a threshold based on the Bonferroni Correction for each edge and decide if it's significantly important for the overall node or not.

But still better with another program

Two packages are connected if they share significantly high level of dependencies. Network kept mainly web development based packages (note only 1% of original data in it).
"Bipartite network"

My network looks like it from inside

But I want to see how similar packages are

Arrows point on the package it depends on
Newman Projection

original network

Newman Projection (2001)

\[ W_{ij} = \text{sum}(1/N_p - 1) \]

where \( N_p \) is the number of authors on paper \( p \).

In the context of scientific collaboration networks, it means that if two scientists only wrote a single paper together with no other co-authors get a weight of 1. If two scientists have written two papers together without any co-author, the weight of their tie would be 2 and so on. This method finally takes care about the different set sizes.

More about projections: https://github.com/velf/Network_Analytical_Notebooks.git

projected network
based on common neighbors
Hairball :( 

Projection calculates similarity between all nodes, so we get this...

But based on weights (similarity) we can filter, but how?
Backbone Filtering

Disparity filter is a network reduction algorithm to extract the backbone structure of undirected weighted network. I create a threshold based on the Bonferroni Correction for each edge and decide is it significantly important for the certain node or not.
Let's find communities

```python
# The community package is an extension of networkX, the Louvain Modularity is one of the most used algorithm in community detection.
# More info: https://en.wikipedia.org/wiki/Louvain_Modularity
part = community.best_partition(NG) # calculate best partition for each node
values = [part.get(node) for node in NG.nodes()]
counterx=Counter(values)

part orig= community.best_partition(G) # calculate best partition for each node
values_orig = [part_orig.get(node) for node in G.nodes()]
counterx_orig=Counter(values_orig)

print 'Number of communities by the Louvain method: '+'str(len(counterx.keys()))
print 'Modularity: '+'str(community.modularity(part, NG))

Number of communities by the Louvain method: 11837
Modularity: 0.787374600201

print 'Number of communities by the Louvain method: '+'str(len(counterx_orig.keys()))
print 'Modularity: '+'str(community.modularity(part_orig, G))

# Modularity: defined as a value between -1 and 1 that measures the density of links inside communities compared to links
# across communities.

Number of communities by the Louvain method: 128
Modularity: 0.792284486941
```

Size of partitions, Original Data

Size of partitions, Projected Data
Small enough to visualize in Python

```python
pos=nx.fruchterman_reingold_layout(NG, scale=20) # Fruchterman-Reingold force-directed algorithm
# See more: https://en.wikipedia.org/wiki/Force-directed_graph_drawing
nx.draw_networkx_nodes(NG,pos,node_size=[i*6 for i in NG.degree().values()],cmap=plt.get_cmap('jet'),
node_color=values)
nx.draw_networkx_edges(NG,pos,edgelist=weights,width=weights_to_plot,alpha=0.5,edge_color='#A6AFB4')
fig = plt.gcf()
fig.set_size_inches(90, 70)
fig.savefig('Python_Dependency_Network_Clustered_final.png', dpi=120)
plt.close()
```
But still better with an other program

Two packages are connected if they share significantly high level of dependencies. Network kept mainly web development based packages (note only 10% of original data in it).
What kind of clusters can be found based on the similarity of packages?

"Bipartite network"

Let's find communities

Newman Projection

Small enough to visualize in Python

Hairball :(

But still better with another program

Backbone Filtering

Two packages are connected if they share significantly high level of dependencies. Network kept mainly web development based packages (rate on 10% of original data in it).
Interactive graphs with Lightning

```
import os
from lightning import Lightning
from numpy import random, asarray, linspace, corrcoef
from colorsys import hsv_to_rgb
from sklearn import datasets

lgn = Lightning(ipython=True, host='http://public.lightning-viz.org')

```

U can zoom in and play with it!

```
lgn = Lightning(ipython=True, host='http://public.lightning-viz.org')

```

Connected to server at http://public.lightning-viz.org

```
mat = nx.adjacency_matrix(G).todense()
n=nx.number_of_nodes()
c = [list(asarray(hsv_to_rgb(float(y)/4, 0.8, 1.0))*255) for x, y in part.iteritems()]
g = G.degrees().values()
lgn.force(mat, color=c, size=(asarray(g) + 1.5))

```
Summary

NetworkX is a good tool, with huge support from the scientific community

Still lack of the newest results/methods of network science (Come and work on it!)

Not efficient in every case (large and very dense networks, visualization)

Think twice if u want network visualizations (not always worth the effort)

Other good python packages (iGraph, written in C/C++ -faster, but not the best documentation)

For visualization: Gephi, Cytoscape
Thx for ur attention:

Orsi Vasarhelyi
CEU, Center for Network Science