Dauphine | PSL
UNIVERSITÉ PARIS

Data Wrangling - Project
Implementing a program of graph study over Neo4j

Maximilien Wemaere

31/03/2023

Introduction

This Project based on the article [2] aims to implement a little program to study the structure of a graph database on Neo4J. Neo4j come with an efficient query language called Cypher which permits to study graphs thanks to the module GDS fro Graph Data Science.

However the functions and possibility of Cypher and GDS are relatively restricted and there is no efficient method to compute mathematics formulae. This is why it could be very useful to implement a structure to extract some information of the Neo4j graph to another language in order to do scientific study on it.

Section I will present briefly the work of the reference paper. In section II we will implement the same structure but on another language. We propose a way to go further in section IV with advanced graph study. Finally, an application of this work will be presented in section V.

1 About the article

The article *Higher order graph centrality measures for Neo4j* by Georgios Drakopoulos, Aikaterini Baroutiadi, and Vasileios Megalooikonomou starts by highlighting the importance of graph database today and then recalls the importance of using centrality measures for graph studies.
During the paper they present two centrality measures: eigen centrality and power centrality. They also recall the power method to compute the spectral radius of a matrix and the associated eigenvector. Then they present results of centrality thanks to a program they coded on Java which take information from Neo4j.
However they don't talk at all about this program. The dataset used is Facebook from SNAP [4].

The objective of our project is to implement a similar structure to study centrality on a graph database. We will use the same dataset.
However as they give no information on they Java program, we will propose the implementation in Python as it is a very popular language for scientific computing and has many libraries for graph studies.

2 Implementation of a python program over Neo4j

To start we first launch a Neo4j cloud database thanks to Neo4j Aura. On this instance we load the dataset Facebook.

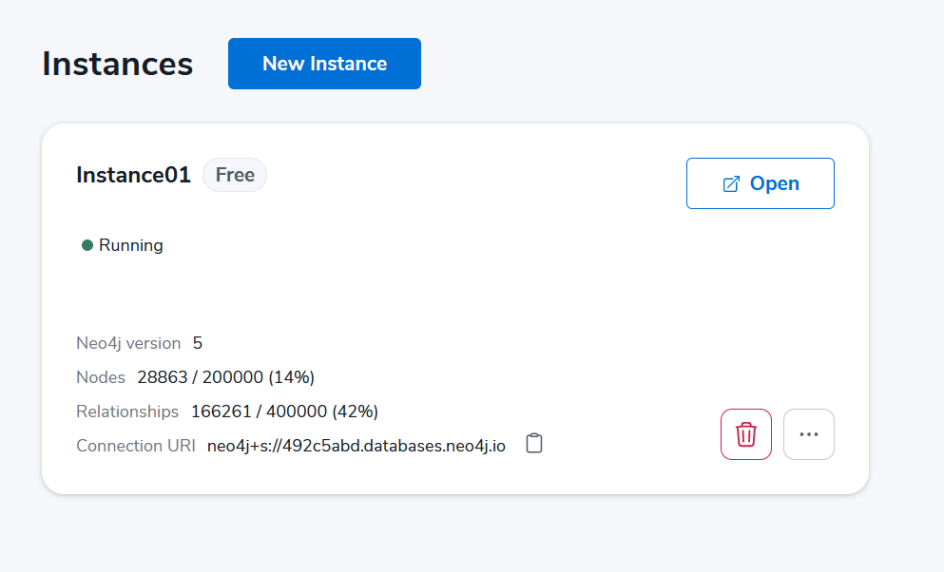

Figure 1: Neo4j cloud database

Now we can connect a python program to query the interesting information.

2.1 Querying the adjacency matrix on Neo4j

To compute the centrality presented in the paper, all that we need is the adjacency matrix of the graph.
Indeed spectral centrality as the name suggests come from the spectral study of the adjacency matrix.

So we use the Neo4j python libraries in order to connect to the Neo4j cloud database.

Once we are connected, we can send a Cypher query to the db and get the result in the python environment.

We first request all the id of the nodes in the graph with the following Cypher query :

```
MATCH(n) RETURN ID(n)
```

Then for each node, we check the existence and if so, the number of edges between the different nodes, thanks to the Cypher query:

```
MATCH(n1)-[s]->(n2) WHERE ID(n1)=id1 AND ID(n2)=id2 RETURN count(s)
```

The result of this query is the element of the adjacency matrix in the row corresponding to node1 and the column corresponding to node 2.
We have now the adjacency matrix of the graph from the Neo4j database.

2.2 Computing the centrality metrics

It is now easy to compute the different centrality with the adjacency matrix in a numpy array.
Degree centrality come directly with the following line:

```
A.dot(np.ones(len(A)))
```

The Neuman metric is also computable in one line:

```
np.diagonal(np.linalg.inv(np.eye(N)-A))
```

And finally the Estrada index:

```
np.diagonal(np.exp(A))
```

For the eigen centrality, we compute the eigenvector associated to the spectral radius with the power method, also called Von Mises method.
This is just the iteration of three calculus:

```
# calculate the matrix-by-vector product Ab
b_k1 = A.dot(b_k)

# calculate the norm
b_k1_norm = np.linalg.norm(b_k1,ord=2)

# re normalize the vector
b_k = 1/(b_k1_norm+0.001)*b_k1
```

The eigenvector correspond to the measures of centrality of each node.

Here we print the distribution of centrality in the graph:

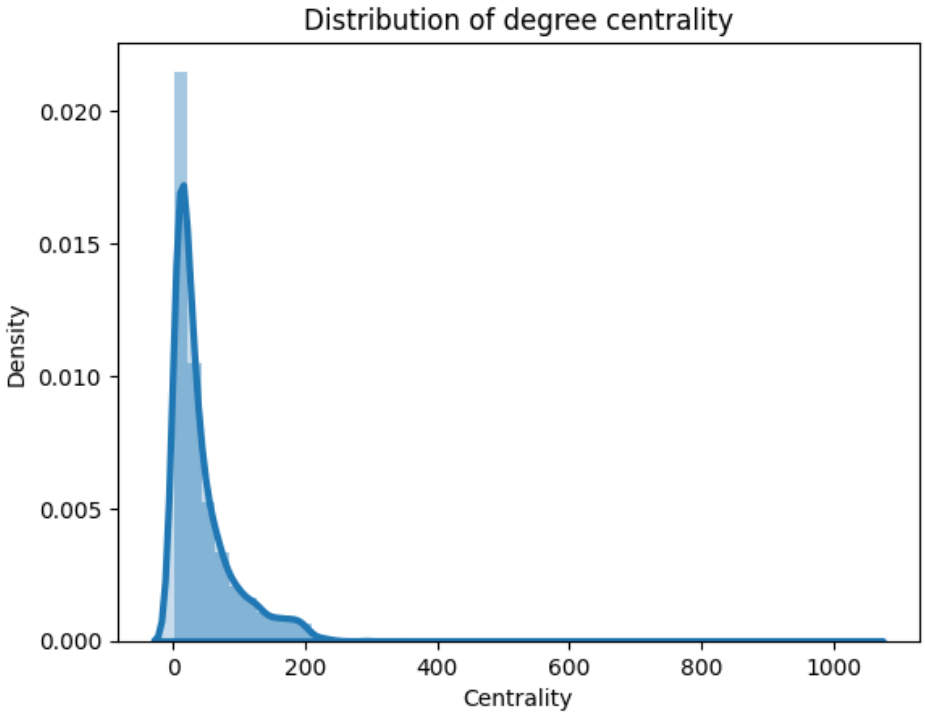

Figure 2: degree centrality distribution

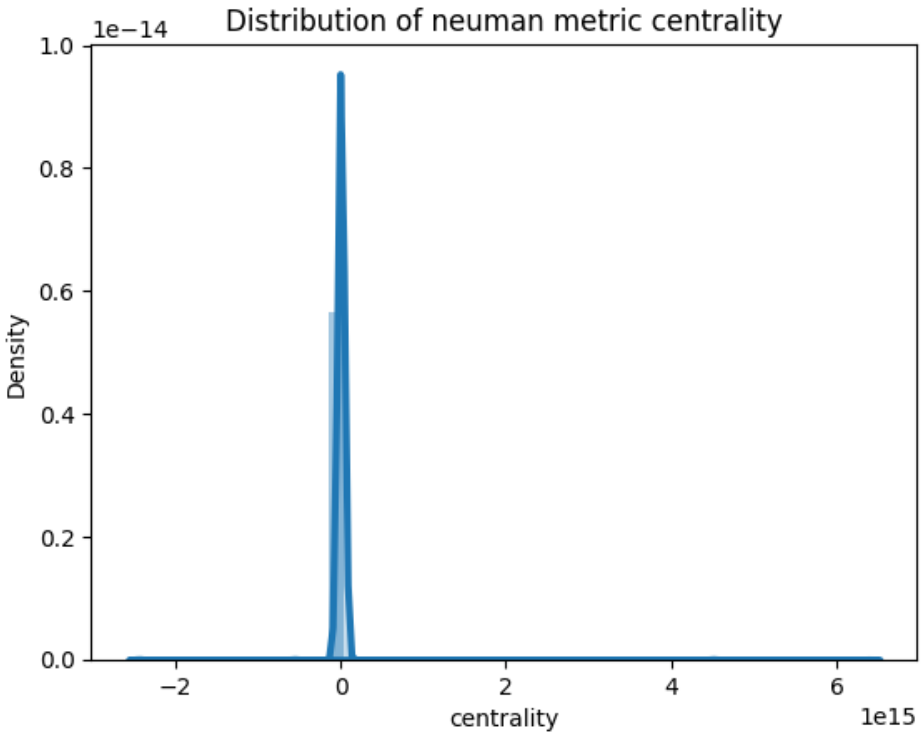

Figure 3: Neuman metric centrality distribution

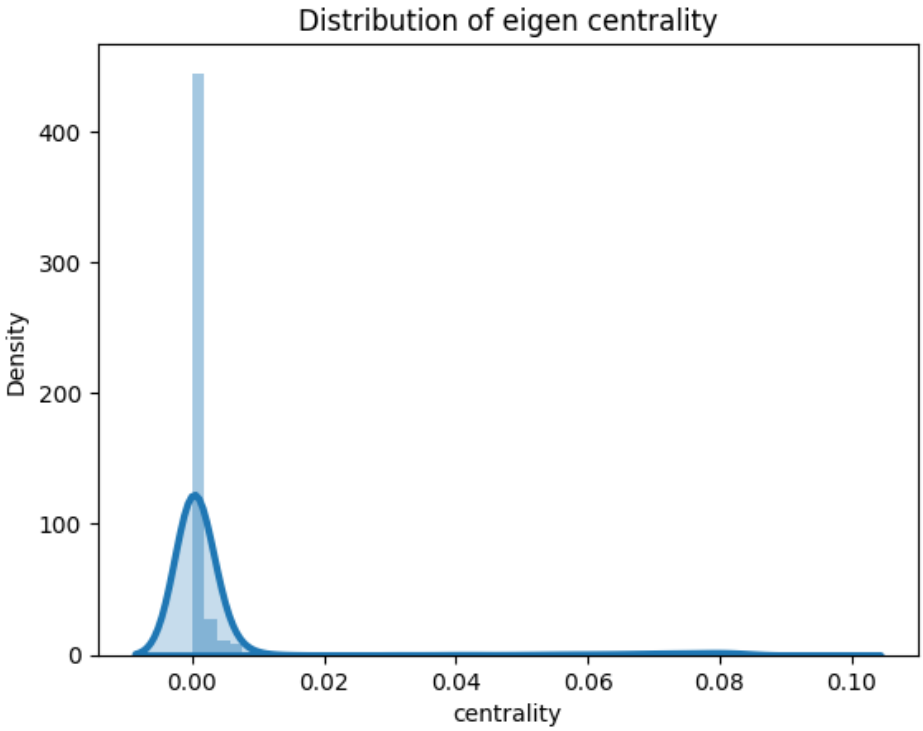

Figure 4: eigen centrality distribution

3 Go further: Advanced graph study over Neo4j

We can go further than just compute the centrality and perform more complex calculus on the graph. For this we use the library NetworkX on python [3] and create a graph model from the adjacency matrix.

```
G = nx.from_numpy_array(A)
```

Now that we have a twin graph of the Neo4j database. However we don't really have the data on the vertex and edges but only the structure of the graph.

With this graph model we can use powerful tool already implemented in python like the computation of the betweenness centrality. In order to be efficient in computation time we use the library RetworkX [7] based on NetworkX but coded in Rust and presenting high performance calculus, very useful for big graph.

```
G_r = rx.networkx_converter(G)
betw_centrality = rx.betweenness_centrality(G_r)
```

Another advantage of having the graph model in a python environment is to be able to implement complex scientific computation on the graph structure. As an example two programs from my internship in the Mathematics Institute of Neuchatel, Switzerland last summer.

The first program GA, try to compute, thanks to a genetic algorithm the node of graph that make the graph vulnerable to infection (spreading of information in the network). We will see this in the next section.
The second program Netshield from [1] has the same objective.

These programs are not so complex but it is impossible to implement them directly in Cypher. This is why a pipeline like we did toward another language environment is necessary to perform advanced data study.

4 Application of the work: Containing epidemics

4.1 SIR model

Now we will see an application on this work inspired from my internship in Switzerland.
The internship was about finding the node to vaccine in a network representing a population, if there was an epidemic spreading in it.
The nodes are the individuals and the edges are their social links. However, this modelisation can be useful for many other application than epidemics. For example it can be the spreading of a rumor or a fake news in a social network. Here, our dataset Facebook is coherent.

We use a simple and famous model of compartmental epidemiology called SIR [6] for Susceptible (to be infected), Infected and Recovered.
In a graph the model is sequential. At each iteration a S node can be infected with a probability τ if it has an infected neighbor. An infected node recovered after a fixed number of iteration with a probability γ. There is lot of more accurate modeling for spreading of disease or rumor like SIS, SIRV, SIRD ... but this model is simple and is used here as a case study.

In order to simulate a SIR infection, we use the library Epidemics on Network [5] based on NetworkX.
The objective of this application is to find the best node to immunize in order to stop the spreading. The way to assess the choice of nodes to vaccine is the number of people which have been infected at the end of the epidemic.
A common way to proceed is to compute centrality of the nodes and vaccine the nodes with the highest

centrality. Indeed the centrality of a node is a measure of importance of it in the graph. So the more important nodes are removed from the graph (immunized) with:

```
G_i.remove_nodes_from(vaccinated)
```

4.2 Results

We first simulate the epidemic on the network without immunization:

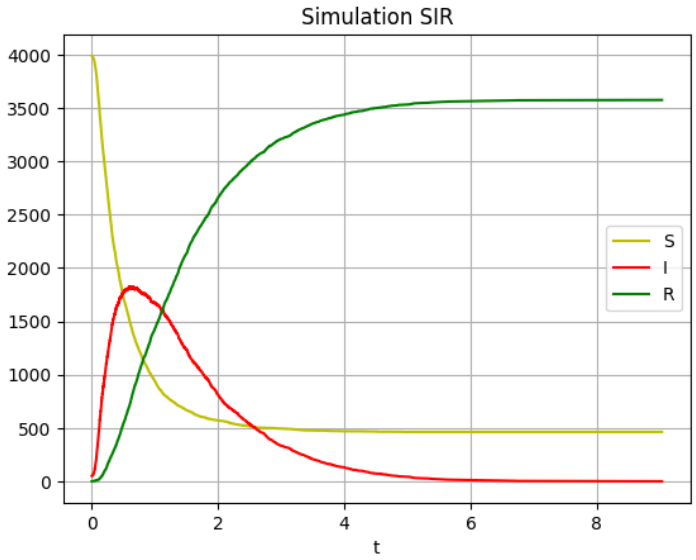

Figure 5: SIR epidemic without immunization

There is a final number of 3649 infected people.

Then we select randomly the nodes to immunize.

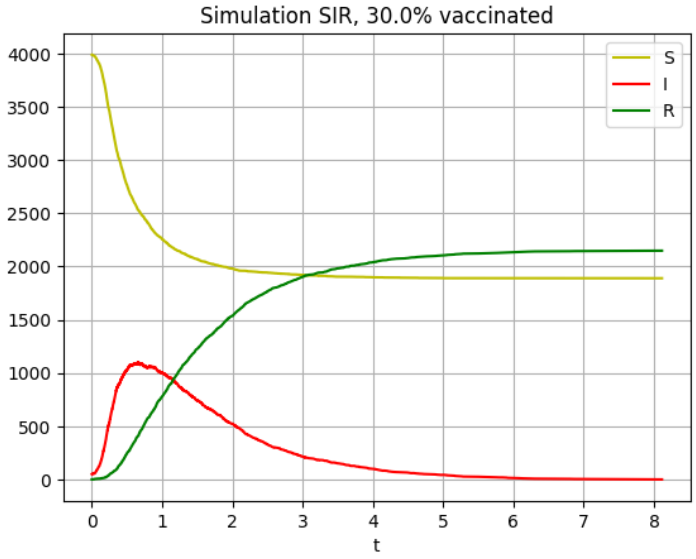

Figure 6: SIR epidemic without immunization

There is a final number of 2420 infected people.

We select the node to immunize with the eigen centrality

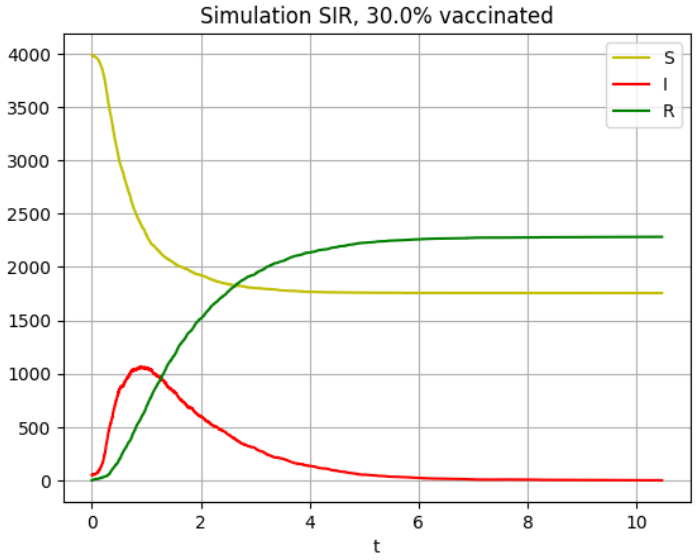

Figure 7: SIR epidemic without immunization

There is a final number of 2283 infected people.
This centrality is not really useful for this kind of epidemic. However, algorithms like pagerank use a enhanced version of eigen centrality to propose an efficient solution to selection of important node in a network.

Now we use the degree centrality

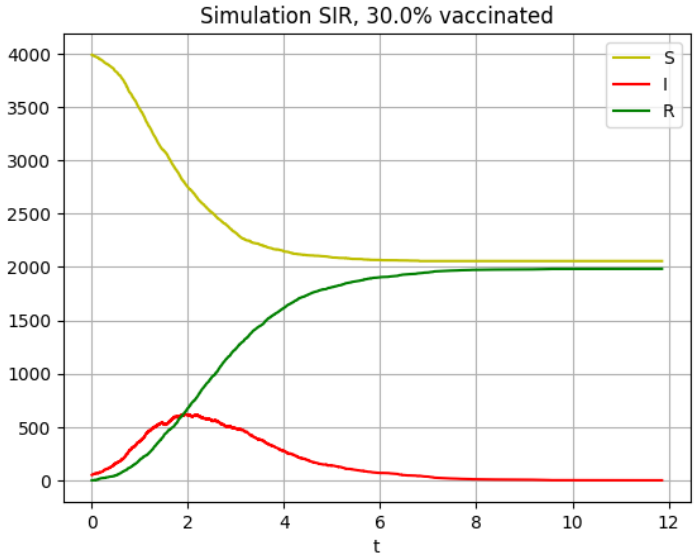

Figure 8: SIR epidemic without immunization

There is a final number of 1983 infected people.
This centrality is more effective and used in real case as it is very simple an very effective for graph with hub (node with very high degree) like scale-free graph.

Now we use the betweenness centrality

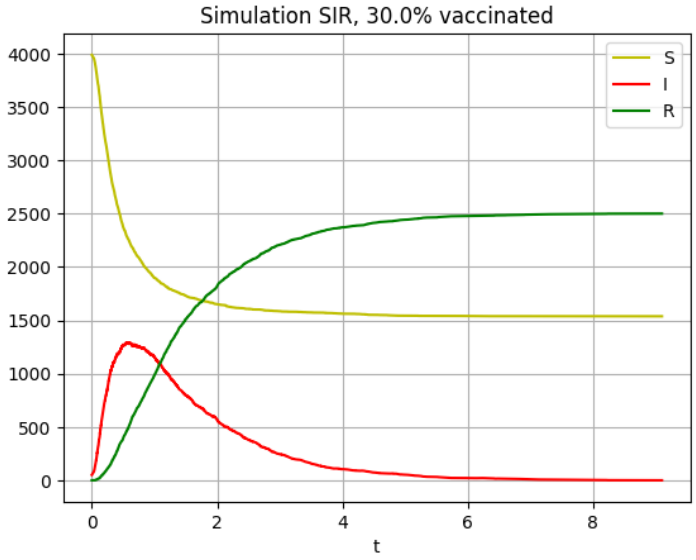

Figure 9: SIR epidemic without immunization

There is a final number of 1636 infected people.
This centrality is really effective, and now even complex algorithm struggle with this metric to propose a better solution to this problem.

We don't present the result of the power centrality (Neuman metric) as this centrality is also not really useful for this problem. Genetic algorithm and Netshield presented earlier are state of the art algorithm for this problem however they demand bigger graph and a fine tuning so they do not perform better than centrality vaccination here.

4.3 immunization of the graph on Neo4j

Now that we have find good nodes to immunize thanks the computation of betweenness centrality we can imagine a practical case where we have to remove the nodes from the Neo4j database. For this we come back to our python environment connected to the Neo4j cloud database and we can send a Cypher query to delete the nodes, one by one by their id:

```
MATCH(n) WHERE ID(n)=id DETACH DELETE n
```

4.4 Applications of this problem

This problem is more generally the identification of the nodes who spread the most an information in a graph.
The problem come from epidemiology for studying the disease in population. But nodes can be individuals or city or even airport to take concrete choice of closing airport in a country. It can be used in cybersecurity to stop a virus to spread or identify the most interesting computer to attack or defend in a network.
In environment science, it can be used to study the forest fire, where the trees are the nodes.
In social science we can measure the influence of a people in a society, on the social media, we can study the spreading of rumors and fake news.
And there is plenty of other application of this problem.

5 Conclusion

We have seen that it can be interesting to study deeply the structure of a graph database in order to understand, or get more information about the data. For this we implemented a python structure to query the adjacency of the graph and it is all that we need to perform mathematical study on the graph. This implementation is really necessary when we need more complex calculus to do than those proposed by the Neo4j module GDS.
Repository: Github link

References

[1] Chen Chen, Hanghang Tong, B. Aditya Prakash, Charalampos E. Tsourakakis, Tina Eliassi-Rad, Christos Faloutsos, and Duen Horng Chau. Node immunization on large graphs: Theory and algorithms. *IEEE Transactions on Knowledge and Data Engineering*, 28(1):113–126, 2016.

[2] Georgios Drakopoulos, Aikaterini Baroutiadi, and Vasileios Megalooikonomou. Higher order graph centrality measures for neo4j. pages 1–6, 2015.

[3] Aric Hagberg, Pieter Swart, and Daniel S Chult. Exploring network structure, dynamics, and function using networkx. Technical report, Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2008.

[4] Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection. `http://snap.stanford.edu/data`, June 2014.

[5] Joel C. Miller and Tony Ting. Eon (epidemics on networks): a fast, flexible python package for simulation, analytic approximation, and analysis of epidemics on networks. *Journal of Open Source Software*, 4(44):1731, 2019.

[6] Romualdo Pastor-Satorras and Alessandro Vespignani. Immunization of complex networks. *Physical Review E*, 65(3), feb 2002.

[7] Matthew Treinish, Ivan Carvalho, Georgios Tsilimigkounakis, and Nahum Sá. retworkx: A high-performance graph library for python. *CoRR*, abs/2110.15221, 2021.