Exercises for

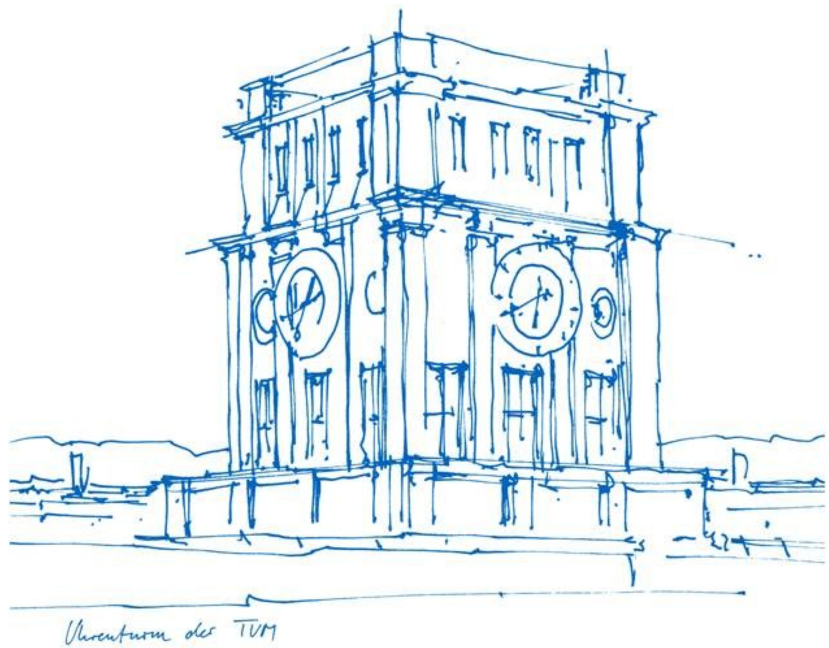
Social Gaming and

Social Computing

(IN2241 + IN0040) – Introduction to

Exercise Sheet 2

Centrality



Exercise Content

Sheet Number	Exercise	Working Time
1	• Introduction to Python and SOMETHING ELSE TODO	TBA
2	• Centrality and PageRank	TBA
3	• Clustering methods	TBA
4	• LINEAR REGRESSION TODO	TBA
5	• Natural Language Processing: Hate Speech detection	TBA
6	• Bi-LSTMs and Explainable AI	TBA

Exercise Sheet 2: Centrality Measures

- goals:
  - experiment with different centrality measures
  - detect important nodes and edges in a graph
  - understand how the Google Engine works
- data: UniversityNetwork.graphml
  - represents the faculty of a university
  - consists of individuals (vertices) and their directed and weighted connections (edges)
  - the edges' weights are a measure of friendship between the persons

Repetition from Lecture: Centrality Indices

- Degree Centrality
  - the more friends a node has, the more central it is

$$C_{deg}(u) = \deg(u)$$

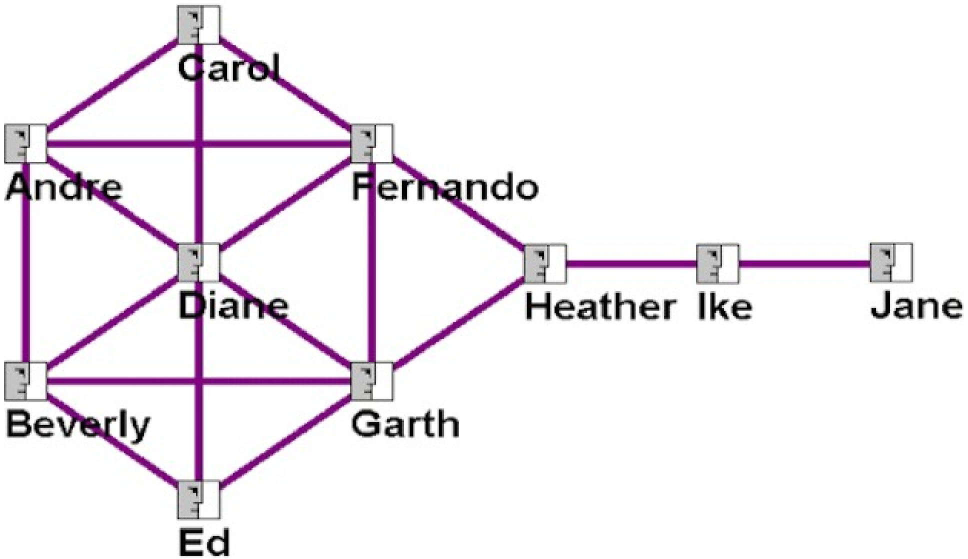
- Closeness Centrality
  - the inverse average length of the shortest path between the node and all other nodes

$$C_{clo}(u) = \frac{1}{\sum_{v \in V} d(u,v)} \text{ or } \frac{|V|-1}{\sum_{v \in V} d(u,v)}$$

- (Shortest Path) Betweenness Centrality
  - probability of a node acting as a bridge along the shortest path between two other nodes

$$C_{btw}(u) = \sum_{s \neq u, t \neq u} \frac{\sigma_{st}(u)}{\sigma_{st}}$$

Degree Centrality

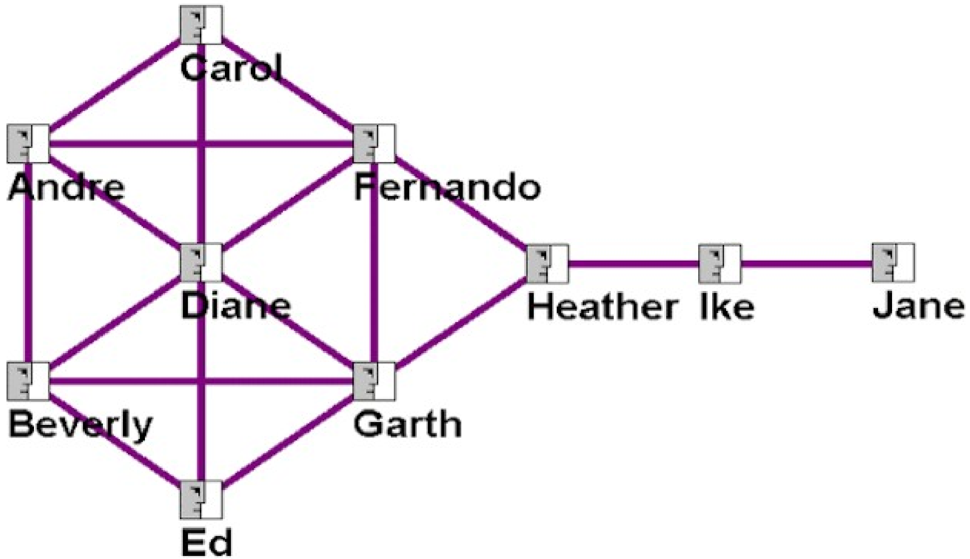


Name	Degree
Andre	4
Beverly	4
Carol	3
Diane	6
Ed	3
Fernando	5
Garth	5
Heather	3
Ike	2
Jane	1

- is the number of connections a node has
- Diane has the most direct connections in the network, making hers the most active node in the network
- however, she only connects people that are already friends with each other

SOURCE: http://www.orgnet.com/sna.html

Closeness Centrality

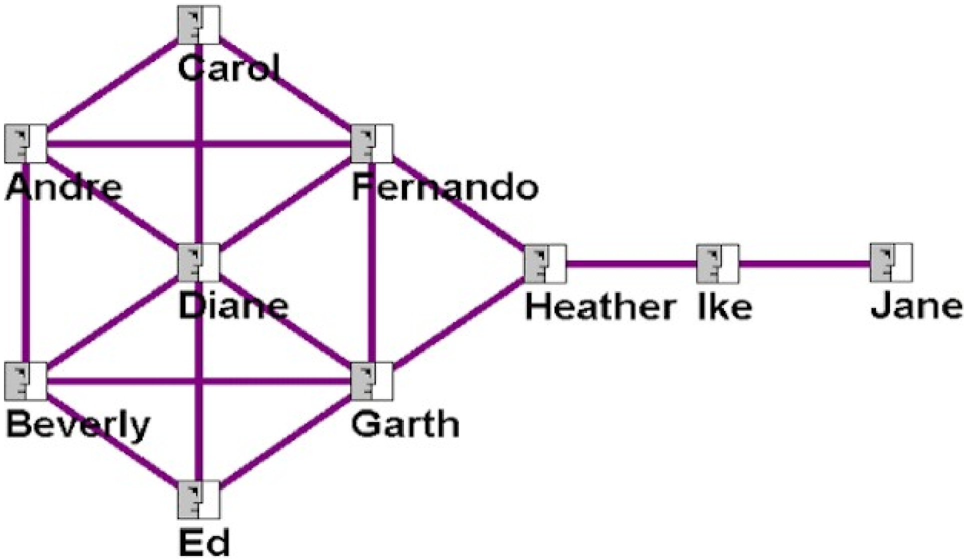


Name	Closeness
Andre	0.529
Beverly	0.529
Carol	0.500
Diane	0.600
Ed	0.500
Fernando	0.643
Garth	0.643
Heather	0.600
Ike	0.429
Jane	0.310

- is the inverse of the average shortest path length to all other nodes in the graph
- Fernando and Garth can access all the nodes in the network more quickly than anyone else
  - they have the shortest average path length to all users

SOURCE: http://www.orgnet.com/sna.html

(Shortest Path) Betweenness Centrality



Name	Betweenness
Andre	0.833
Beverly	0.833
Carol	0
Diane	3.667
Ed	0
Fernando	8.333
Garth	8.333
Heather	14
Ike	8
Jane	0

- measures the number of times the shortest path between two nodes goes through the investigated node, divided by the total number of shortest paths between the two nodes
- Heather has few direct connections, yet she has an important role for Ike and Jane, who wouldn't be connected to the network without her
  - she has high control of information flow

PageRank Centrality

- feedback-centrality named after Larry Page, co-founder of Google
- used for the Google search engine
- basic idea: a node is more central the more central its neighbors are

Task 2.1: The Krackhardt Kite Graph

We will use the Krackhardt Kite for the first exercise. As you know from exercise 1, the Krackhardt Kite is a simple connected, unweighted and undirected graph. This figure illustrates the Krackhardt Kite.

Calculate the degree centrality of the Krackhardt Kite graph - just a list of ten values, one for each node. You can use the pre-defined function of the NetworkX library.

Optional: Look at the graph and the list with the degree centrality values. Can you identify which node has which degree centrality?
Optional: Calculate the closeness and betweeness centrality as well. What information do they give us?

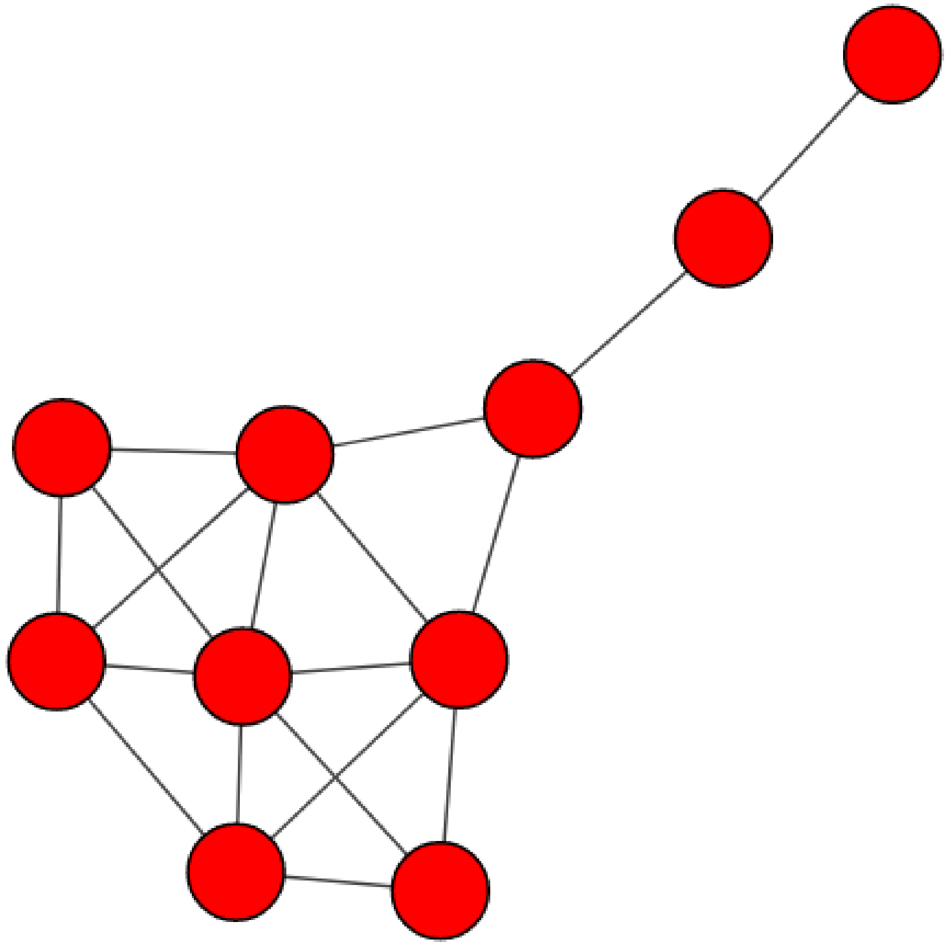
```
# Importing the graph (connected, unweighted, undirected social network)
krackhardt_kite = nx.krackhardt_kite_graph()

# Formatting the graph
nodeColor = "red"
nodeSize = 400
pos = nx.spring_layout(krackhardt_kite)

# TODO: Calculate and print the Kite's degree centrality
```

The Krackhardt Kite

```
Degree Centrality Kite: [4, 4, 3, 6, 3, 5, 5, 3, 2, 1]
```



Task 2.2: Betweenness Centrality

Betweenness centrality also measures centrality based on shortest paths. For every pair of vertices in a graph, there exists a shortest path between the vertices such that either the number of edges that the path passes through (for undirected graphs) or the edges' sum of the weights (for directed graphs) is minimized.
Vertices with high betweenness may have considerable influence within a network by virtue of their control over information passing between others.

a) Write a Python program that computes the betweeness centrality for each node for the given social network. The output should be a list where each item contains the value of the closeness centrality of a node. You are **not allowed** to use the pre-defined function `betweenness_centrality()` from NetworkX , but you can look up its documentation for help.

Notes:

- The program only have to implement the undirected graph version (without edge weights)
- Look up the mathematical expression in the documentation
- Normalize your centrality values
- You are allowed to use pre-defined functions from NetworkX for determining (shortest) paths

Task 2.2: Betweenness Centrality

Now you have implemented a programm for Betweenness Centrality, copy your solution and try to change your code in the following way.

b) Write a Python program that computes the epsilon betweeness centrality for each node for the given social network, when a path is ϵ longer than the shortest path, it is considered a valid path for the computation of the betweenness centrality of a node

Notes:

- All notes from above still apply
- This time only shortest paths are not sufficient to compute the centrality, maybe NetworkX can help you once more?
- Consider only ϵ-longer paths that do not contain the same node more than once

Task 2.3: PageRank

In this task we will introduce you to the **PageRank centrality**. It is a feedback-centrality named after Larry Page, who together with Sergei Brin founded Google. The PageRank algorithm was used in Googles search engine to rank the pages for the search result. Since 2013 PageRank was superseded by the Hummingbird algorithm. Although PageRank remains one of many ingredients in the Hummingbird algorithm. Its basic idea is that a node is more central the more central its neighbors are. In order to understand PageRank the concept of a random walk is required.

The **random walk** model describes a user surfing the web, starting at a web page and following hyperlinks to other web pages. If there is no link to other pages, the surfer jumps to a random web page.

Tasks (cont.)

This web is a (directed) graph with vertices connected by edges. The random walk is an iterative process. Starting from a random vertex, an outgoing edge of the current vertex is selected randomly and the random walker follows it. If no outgoing edges are present, a random vertex is selected and the walker jumps to it ("teleportation"-mechanism), as some vertices may not have any outgoing edges and the process would terminate.

The PageRank measures a stationary distribution of one specific kind of a random walk that starts from a random vertex, with a predefined probability $1 - d$, jumps to a random vertex, and with a probability d follows a random outgoing edge of the current vertex.

a) First create a graph and test out the pre-defined NetworkX PageRank function.

1. Create a graph using `erdos_renyi_graph` function of NetworkX.

Tasks (cont.)

2. Calculate the PageRank values of our `simple_graph`, using the built-in function of NetworkX.

3. Print the first 15 elements of the PageRank.

```
# use this values for the built-in function
ITERATIONS = 100
DAMPING = 0.85



# TODO: calculate PageRank

# TODO: print the results
```

Tasks (cont.)

b) Create a simple PageRank function using **Jacobi power iteration**, which you can find in the lecture slides. To avoid matrix inversion we use an iterative formula for the PageRank algorithm:

$$c_i^{(k+1)} = d \cdot \sum_j P_{ij} c_j^{(k)} + \frac{(1-d)}{N}$$

where the superscript k denotes the iteration index, d the damping, N the number of nodes in our graph (which is left out in the lecture notes and also in the original papers, but is used in the built-in PageRank calculation algorithm of NetworkX).

1. Your first task is to **implement a function** which calculates the transition matrix element P_{ij}.

Note: If the out-degree of a node is 0, the user should make a "random jump"

Tasks (cont.)

2. The second task is to **normalize** a list, so that `sum(list) = 1.0` after every iteration in the Jacobi power iteration algorithm.

```
# TODO: renormalize after every step
def renormalize(pagerank_list):
    '''
    input arbitrary float number list
    return a list where of all elements (sum(list)) equals 1.0
    '''
    # TODO: implement the function
```

Tasks (cont.)

3. The third and last task is to **implement the PageRank** calculation using Jacobi power iteration yourself. **Print** the first 15 elements and make sure you have the same result as in task ***a)***.

Note:

- `summe_j` is the term $\sum_j P_{ij} c_j^k$ in the formula

Tasks (cont.)

c) Personalized PageRank

Now that you have calculated the PageRank centrality you will enhance the PageRank calculation to the personalized PageRank.

Personalized PageRank is a modification of the PageRank algorithm. It is basically the same but biased to a personalized set of the starting vertices, a so-called `personalization` or preferences vector of the user.

Instead of jumping to a random vertex with probability d, the walker jumps to a random vertex from the set of the starting vertices. By varying the damping factor d the algorithm can be adjusted either towards the structure of the network itself, by using a close to 1 value of d, or towards the personal preferences by making d smaller. Personalized PageRank can be used for personalized recommendations.

Copy and modify the `calcPageRank()` function, in order to include personal preferences. You have to modify the starting vector and the formula slightly. In addition to that `pij()` must be corrected for the personal jump too, define `pij_pers()` in order for your personalized PageRank to work!

Tasks (cont.)

d) Describe in 3-4 sentences what happens if you modify the starting vector or the damping factor? How does it influence your recommendation?

Submitting your solution

- work by expanding the .ipynb iPython notebook for the exercise that you downloaded from Moodle
- save your expanded .ipynb iPython notebook in your working directory
- submit your .ipynb iPython notebook via Moodle (nothing else)
- remember: working in groups is not permitted. Each student must submit their own .ipynb notebook!
- we check for plagiarism. Each detected case will be graded with 5.0 for the whole exercise
- deadline: check Moodle

Citations