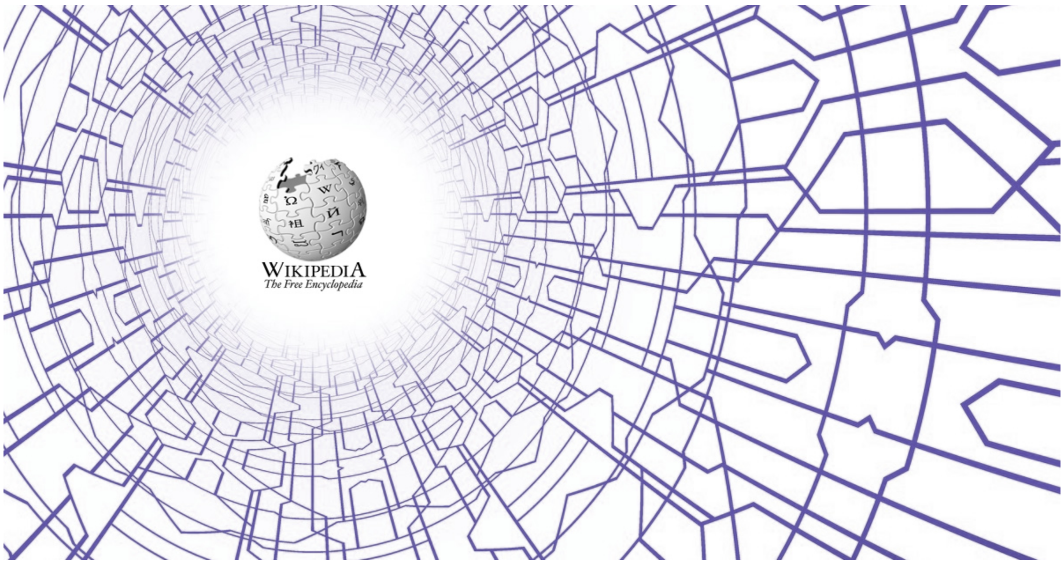
Homework 5 - Explore Wikipedia's hyperlinks network

In this assignment we analyze the Wikipedia's articles network by applying graph algorithms.



!!!Read the entire homework before coding anything!!!

Data and setting

There are many options to collect and build the Wikipedia's underlying network, we rely on the dataset provided here. For the purpose of our exploration, we do not consider the entire dataset. Instead, we focus on the articles belonging to a subset of categories.

1. Download the reduced version of the graph Wikicat hyperlink graph. Every row indicates an edge. In particular, the two elements are the source and the target, respectively.
2. From this page download:
   - `wiki-topcats-categories.txt.gz`: list of pages per category
   - `wiki-topcats-page-names.txt.gz`: page names

Note that in the reduced version of the network we removed the categories whose number of articles in less than 5000 and more than 30000.

General notes

1. You will notice that one article might belong to a single category or multiple ones. In the case of multiple appearance, you break the ties uniformly at random. Please, do it before solving any task in the homework.
2. We assume that all edges in the graphs we will consider have weight equal to 1.
3. **All the algorithms**, unless specified, must be implement from scratch.
4. The algorithms should handle exceptions, e.g. what if there is no path between two nodes?
5. Differently from other homeworks, we will execute your functions.

RQ1

Build the graph G=(V, E), where *V* is the set of articles and *E* the hyperlinks among them. Then, provide its basic information:

- Is the graph directed?
- How many articles are we considering?
- How many hyperlinks between pages exist?
- Compute the average number of links in an arbitrary page. What is the graph density? Do you believe that the graph is dense or sparse? Is the graph dense?
- Visualize the nodes' degree distribution

RQ2

Define a function that takes in input:

- A page *v*
- A number of clicks *d*

and returns the set of all pages that a user can reach within *d* clicks.

RQ3

Define a function that takes in input:

- A category *C*
- A set of pages in *C*, $p = \{p_1, ..., p_n\}$

and returns the minimum number of clicks required to reach all pages in *p*, starting from the page *v*, corresponding to the most central article, according to the *in-degree* centrality, in *C*.

Consider that:

- The algorithm needs to handle the case that the graph is not connected, thus not all the pages in *p* are reachable from *v*. In such scenario, it is enough to let the program give in output the string "Not possible".
- Since we are dealing with graph exploration, you can pass more than once on the same page p_i.
- Since the problem's complexity is high, consider to provide just an approximation/heuristic solution for the problem.
- You can use whatever metrics of centrality.

RQ4

Given in input two categories: C_1 and C_2, we get the subgraph induced by all the articles in the two categories.

- Let *v* and *u* two arbitrary pages in the subgraph. What is the minimum set of hyperlinks one can remove to disconnect *u* and *v*?

RQ5

Write a function that, given an arbitrary category C_0 as input, returns the list of remaning categories sorted by their distance from C_0. In particular, the distance between two categories is defined as

distance(C_0, C_i) = median(ShortestPath(C_0, C_i))

where ShortestPath(C_0, C_i) is the set of shortest paths from each pair of nodes in the two categories.

RQ6

Write a function that sorts the categories in the graph according to their PageRank (PR). For this task you need to model the network of categories such that you can apply the PR algorithm.