Tutorial Random Walk with Restart on networks using NetworkX

Tutorial: Random Walk with Restart on networks using Personalized PageRank with NetworkX

Introduction

A Random Walk with Restart is a link analysis algorithm used to measure the importance of nodes in a graph. Its behaviour is to propagate node weights along the graph's edges until a convergence is reached.

In this tutorial, we will use NetworkX, a Python library for working with graphs, and use the personalized PageRank in order to perform the random walk with restart.

This technique can be useful for ranking nodes in a graph based on their importance to a specific set of seed nodes (called personalized preferences).

Have a look on the internet to understand RWR and the PageRank algorithm. the following links should be a good starting point:

- `https://en.wikipedia.org/wiki/PageRank`
- `medium.com/@chaitanya_bhatia/random-walk-with-restart-and-its-applications-f53d7c98cb9`

Questions:

- What are the general use cases of this technique ?
- Question: What are the use of this technique in the frame of computational biology and/or used with Protein-Protein Interaction networks ?

Prerequisites

Make sure you have the NetworkX library installed in your Python environment. You can install it using `pip`:

```
pip install networkx
```

Step 1: Setting up the Environment

Start by creating a Python script (e.g., `networkx_pagerank.py`) and import the necessary libraries:

```
import networkx as nx
import csv
import os
```

Define an empty function called `rwr` that we will fill in with the random walk with restart code.

```
def rwr(network_file, weights_file, outfile):
    # The personalized PageRank calculation will go here
    pass
```

The algorithm needs two data files: one containing the network structure (i.e. the graph topology), the other containing weights on the nodes: it depicts the seeds of the node, i.e. the personalized preferences.

Set up the basic structure of the script with the `if __name__ == "__main__":` block:

```
if __name__ == "__main__":
    # Define your file paths and other variables here
    rwrd = "."
    network_file = os.path.join(rwrd, "data/PPI_HiUnion_LitBM_APID_gene_names_190123.tsv")
    weights_file = os.path.join(rwrd, "data/tAML_P3_SNV_polyphen_15390_nodes.tsv")
    outfile = os.path.join(rwrd, "output/RW_tAML_P3_nx.tsv")

    # Call the rwr function with the appropriate arguments
    rwr(network_file, weights_file, outfile)
```

The nodes weights come from acute myeloid leukemia patients.

Questions:

- What kind of omics data are SNV?
- What is a Polyphen score? What is the accepted range? What does a high value mean? A low one?

Step 2: Loading Network Data

Inside the `rwr` function, create an empty NetworkX graph (`g`) and read the edge list data from the provided CSV file (`network_file`). Add edges to the graph:

```
def rwr(network_file, weights_file, outfile):
    # Create an empty NetworkX graph
    g = nx.Graph()

    # Read the edge list from the CSV file and add edges to the graph
    with open(network_file, "r") as f:
        csv_reader = csv.reader(f, delimiter="\t")
        for row in csv_reader:
            g.add_edge(row[0], row[1])
```

Step 3: Personalized Weights

Create an empty dictionary (`weight_dict`) to store personalized weights. Read the weights data from the second provided CSV file (`weights_file`) and populate `weight_dict`:

```
def rwr(network_file, weights_file, outfile):
    # ... (Previous code)

    # Create a dictionary to store personalized weights
    weight_dict = {}

    # Read the weights data from the CSV file and populate weight_dict
    with open(weights_file, "r") as f:
        csv_reader = csv.reader(f, delimiter="\t")
        for name, weight in csv_reader:
            weight_dict[name] = float(weight)
```

Step 4: PageRank Calculation

Calculate PageRank with personalized weights using NetworkX's `pagerank` function. Set the damping factor `alpha` to 0.95:

```
def rwr(network_file, weights_file, outfile):
    # ... (Previous code)

    # Calculate PageRank with personalized weights
```

```
    pagerank_results = nx.pagerank(g, personalization=weight_dict, alpha=0.9)
```

Question: what is the damping factor `alpha` ? What is its accepted range of values ?

Step 5: Saving Results

Open the output file (`outfile`) for writing. Iterate through the PageRank results and save the node names and their ranks to the output file. Use formatting to ensure the ranks have 9 decimal places:

```
def rwr(network_file, weights_file, outfile):
    # ... (Previous code)

    # Open the output file for writing
    with open(outfile, "w") as f:
        for node, rank in pagerank_results.items():
            # Save node names and ranks with 9 decimal places
            print(f"{node}\t{rank:.9f}", file=f)
```

Step 6: Testing

Run the script and verify that it successfully calculates personalized PageRank and saves the results to the specified output file.

```
python3 networkx_pagerank.py
```

Question: What are the ten most interesting proteins, ranked in order?