CS2302 - Data Structures

Spring 2020

Lab 6

Due Monday, April 27, 2020

Given a set of entities and relationships in the world, how do we determine which entities are the most important? For example, given a graph representation of the world-wide web, where nodes represent web pages and edges represent links, which pages are the most important in response to a web search (an efficient algorithm to answer this question made Larry Page and Sergey Brim billionaires). Other relevant examples are: which people on facebook are the most influential (nodes represent people, edges represent friendship relationships)?; which scientific publications have the most impact (nodes represent scientific papers, edges represent citations)?

The general idea to answer these questions is the following: a vertex is important if it is connected to many other vertices that are also important. It seems like this is a circular definition but there are algorithms to compute this. For this lab you will implement two of them, one based on simulation and one based on iterative probabilistic computations.

Random Walk. A random walk in a graph $G = (V, E)$ consists of selecting a random vertex v in V, then 'walk' from v to one of its neighbors u (following a randomly-chosen edge going out from v), then 'walking' to a neighbor of u and so on. The number of times a vertex is visited during a random walk of sufficient duration is a very good indicator of the vertex's importance given the graph's structure. The algorithm can be described by the following pseudocode:

```
Random Walk (V, E, steps)
    let v be a randomly chosen vertex in V
    for i in range(steps):
        visited[v] += 1
        let N be the set of neighbors of v (N = {u such that (v,u) is in E})
        if N is empty, N = V
        let u be a randomly chosen element of N
        v = u
    p = visited/steps
    return p # p[i] is the probability that the random walk will visit vertex i at any given time
```

Iterative computation. Instead of the simulation performed by a random walk, we can reach similar results by performing elementary probabilistic computations. Just as in the previous case, our goal is to find, for every vertex i, the probability that the random walk will visit vertex i at any given time. A complete explanation of the mathematical basis of the method is beyond the scope of this class, but it can be easily implemented by a repetitive multiplication of the probability vector (or 1D array) p and a transition matrix T, as explained in the following pseudocode:

```
Iterative p (V, E)
    let p be a 1D array where p[0]=p[1]=...=p[|V|-1] = 1/|V| (in numpy, p should be a 1-by-|V| array)
    let T be the transition matrix, where T[i,j] is the probability of going from vertex i to vertex j
      in a random walk of the graph, computed as follows:
        If the out-degree of i is zero, T[i,0] = T[i,1] = ... = T[i,|V|-1] = 1/|V|
        Else if there is an edge from i to j, T[i,j] = 1/out-degree(i)
             else (there is no edge from i to j), T[i,j] = 0
    Repeat until convergence # In practice we repeat for a fixed number of iterations (1000 or so)
        p = p*T - where p*T is a matrix multiplication
                  (implemented as np.dot(p,T) where p is 1-by-n and T is n-by-n)
    return p
```

After computing the vector of probabilities p, the most important vertex is i such that $p[i]$ is the maximum value in p. If we want to find the second most important vertex, we remove the influence of the first vertex and apply the algorithm again. The algorithm to find a sequence of important and independent vertices is as follows:

1. Find the most important vertex v_{imp} using the chosen algorithm (random walk or iterative).
2. Eliminate the influence of v_{imp} by removing from G all edges connected to v_{imp}

For this lab, your task is to implement both algorithms and to evaluate them on a real-world dataset that contains information about annonymized facebook friendship relationships. You will need to implement (at least) the following:

- A function to build a graph from a text file input, where each line contains the source and destination of an edge
- A function to implement the random walk algorithm using an adjacency list representation
- A function to implement the iterative algorithm using an adjacency matrix representation
- Functions to implement the algorithm for finding the sequence of important vertices using both representations

Test your program using the file *facebook_combined.txt* to find the most influential people on facebook. The file contains two integers per line, representing the source and destination vertices in an undirected graph. As usual, write a report describing your work.

Appendix: Sample output

The following shows a sample run of the program.

```
Using iterative method and adjacency matrix representation
Iteration 1 most important vertex: 107, with p = 0.00604
Iteration 2 most important vertex: 1684, with p = 0.00472
Iteration 3 most important vertex: 3437, with p = 0.00416
Iteration 4 most important vertex: 1912, with p = 0.00378
Iteration 5 most important vertex: 686, with p = 0.00187
Iteration 6 most important vertex: 0, with p = 0.00185
Iteration 7 most important vertex: 1888, with p = 0.00151
Iteration 8 most important vertex: 2543, with p = 0.00148
Iteration 9 most important vertex: 1800, with p = 0.00146
Iteration 10 most important vertex: 2347, with p = 0.00146
```