Lab on PageRank

Mauro Sozio
`name.lastname@telecom-paristech.fr`

In this lab session, we are going to learn and practice with Python and then move to implement the PageRank algorithm. Only the part about PageRank will be evaluated. You should use IPython/Jupyter to edit and run your code and use Python 3. We recommend those who are not familiar with Python to check the tutorial on Python on the Web page of the course and proceed to Section 1. After that, move to Section 2. Those who are familiar with Python can go directly to Section 2.

1 Practicing with Python

This section contains a few suggestions in order to practice with the basics of Python. This section will not be evaluated.

- construct a list of integers L. Then build another list M with the same size of L that contains the square of the elements in L.
- define a function f that receives in input a list of integers and returns a new list containing only even integers.
- write a few lines in Python that read a list of integers from a file and store them into a list L. Then run f on L.

2 Implementing PageRank in Python

The following questions will be evaluated, with the number of points being specified next to the text of the exercise. You should not use any library that implements PageRank or matrix multiplication and implement the exercise from scratch.

1. (8/20 pts.) You should implement the PageRank algorithm in Python so as to deal with very large graphs. To this end, you should consider the version of the PageRank algorithm which deals with sparse matrices (slide 17). The algorithm receives in input a directed graph G which is represented as a list of lines of the kind "i j" denoting that there is an edge between node i and j. The output is the PageRank vector for G. For this question, we can assume that there are no dead ends in G. The matrix M_G should be represented using a sparse matrix representation, i.e. only non-zero entries should be represented. To this end, you should store the matrix as a list of lines, one for each non-zero entry, where the first element is the row, the second is the column while the last one is the value of the

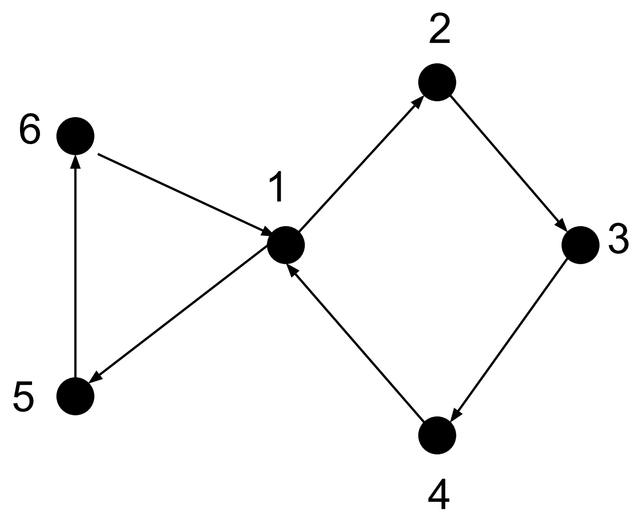

Figure 1: A simple web graph.

matrix in that row and column. For example, the line "1 2 0.4" specifies that $A_{12} = 0.4$. The product between the matrix M and the vector r should also be done while using the sparse representation of M. Run your algorithm on the graph shown in Figure 1 for 100 iterations, while using first $\beta = 1$ and $\epsilon = 0.1$ and then $\beta = 0.8$ and $\epsilon = 0.1$. Report the results you obtain.

2. (7/20 pts.) Construct the Web graph from the webpages extracted from Wikipedia provided to you ("WebPages.zip"). To do so, for each page, extract all links using the *findall()* method of the regular expression module *re* (`https://docs.python.org/2/library/re.html`) and add an edge from page i to page j if there is a link on page i to page j. Beware of duplicate link and selfloop. Tip: all links are preceded by 'a href="'. After having constructed the graph, run the PageRank algorithm on that and report the results.

3. (5/20 pts.) Implement in Python an algorithm that given an directed graph G in input, it removes dead-ends iteratively until no dead-ends are left. We recall that dead-ends are nodes with no out-link.

What to send: You should send us your Jupyter notebook including the answers to the questions and the code in Python. The answer for each question should not exceed 20 lines. The answers for all lab sessions and the project should be sent together at the end of the course to the teaching assistants and professor. All those informations are provided on the course website.