NEU AI fall 2018 / M. Gaidis
Pre-Course Assignment 0

Python for PageRank

What is PageRank?

From Wikipedia (https://en.wikipedia.org/wiki/PageRank):
PageRank is an algorithm once used by Google Search to rank websites in their search engine results. It is a way of measuring the importance of website pages. [PageRank was named after Larry Page, one of the founders of Google, and helped make him one of the richest men in the world!]

PageRank works by evaluating the *number* and *quality* of links to a page to determine a rough estimate of how important the website is. One assumes that more important websites are likely to receive more links from other websites. A page that is linked to by many pages (especially pages with high PageRank) receives a high rank itself. The PageRank algorithm outputs a probability that a person randomly clicking on links will arrive at a particular page. The probability is expressed as a different numeric value between 0 and 1 for each website. A PageRank of 0.5 means there is a 50% chance that a person clicking on a random link will be directed to the website with the 0.5 PageRank.

Unfortunately, web site designers have found ways to manipulate their PageRank, and so Google now relies heavily on other methods of generating search results. However, the mathematics of PageRank is still relevant, as it can apply to any graph or network in any domain. Consider some of the following applications:

1. Systems analysis of road networks linking important cities
2. In neuroscience, PageRank correlates with the relative firing rate of a neuron
3. Twitter uses a version of PageRank to recommend other accounts you may wish to follow
4. Scientific impact factor of publications can be approximated with PageRank
5. PageRank can help determine the best academic doctoral programs at different universities
6. PageRank can process language to interpret the likely meaning of a sentence
7. In biology, PageRank can tell you which species are essential to the health of the environment
8. PageRank may improve voting systems so the best candidate is elected

Your Assignment:

Use simple Python code to create an iterative version of PageRank, and demonstrate it works for a small network that you will create.

Why is this Assignment Important?

There are two main purposes to this assignment:

1. It shows you how even simple algorithms can give computing machines the power to add to human intelligence on a scale that human brains cannot manage. This algorithm by itself is not "artificial intelligence," but it is one of many tools used together to create AI. During this class, you will learn about many of these tools and how they contribute to AI. No one tool is "AI," just as a single hammer does not make someone a carpenter. AI needs many different tools, just like a carpenter needs many tools before they can even become a carpenter. Then, AI needs to figure out what to do with all of these tools to make something useful.
2. It gives you an introduction to Python programming, showing you some of the most important data structures and methods that you need to be familiar with for this class.

Simple Example of Page Rank Computation

The PageRank of a node is defined *iteratively* and depends on the number and PageRank metric of all nodes that link to it ("incoming links"). The iterative calculation proceeds as follows:

1. Start at iteration i=0 by giving all N nodes in your network an equal PageRank of 1/N
2. Compute PageRank at iteration i=1 using the values from the previous iteration step i=0
3. Continue in this way, computing PageRank at iteration i=k from the values at i=(k-1)
4. Stop iterating when the PageRank values do not change significantly (they converge).

The iteration should not take too long, as Google has revealed that a network with about 300 million links will converge in only 50 iterations. Your networks will be much smaller, and should converge very quickly.

To explain the computation (step 2 above), it is easiest to use an example. Figure 1 shows a network of 5 nodes (A, B, C, D, E) in a directed graph. It is a "directed" graph because the edges that represent links between nodes have a direction specified by an arrowhead. For example, you can imagine a webpage C that has an outlink on it to webpage B. In addition, webpages B and D have outlinks to webpage C – or we can say that C has two inlinks.

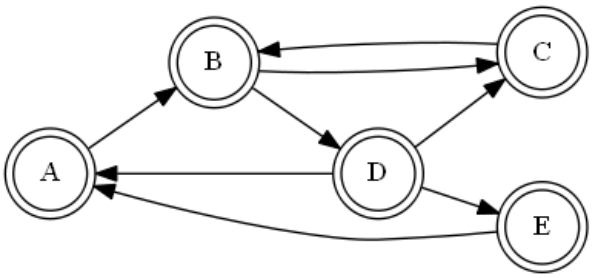

Figure 1: Example graph with 5 nodes and multiple directed links

Step 1, iteration i=0
Assign a value of 1/N PageRank (PR) to all nodes in the network.

Node	PR Value @ i=0 (PR_0)
A	1/5
B	1/5
C	1/5
D	1/5
E	1/5

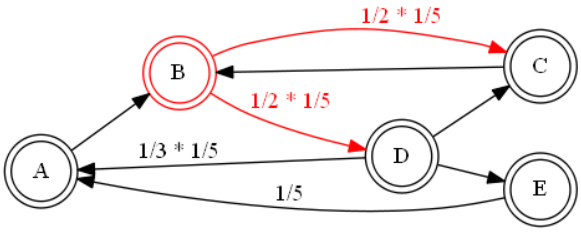

Figure 2: Node B shares its 1/5 PageRank equally with all of its outlinks (nodes it links to), shown in red. B has 2 outlinks, so we multiply the previous iteration PageRank by ½, and give that amount to B's outlinks C and D. Node A has two inlinks (shown in black), and its PR is the sum of 1/3 of D's PR (1/3 because D has 3 outlinks) plus the entire PR of node E (because E only has 1 outlink).

Step 2, iteration i=1
Compute new PageRank (PR_1) by adding together all fractional page ranks of inlinks.

n	i=0	PR_1
A	1/5	1/3*D + E = 1/3*1/5 + 1/5 = 0.2667
B	1/5	A + C = 1/5 + 1/5 = 0.4
C	1/5	1/2*B+1/3*D = 1/10+1/15 = 0.1667
D	1/5	1/2*B = 1/2*1/5 = 1/10 = 0.1
E	1/5	1/3*D = 1/3*1/5 = 1/15 = 0.0667

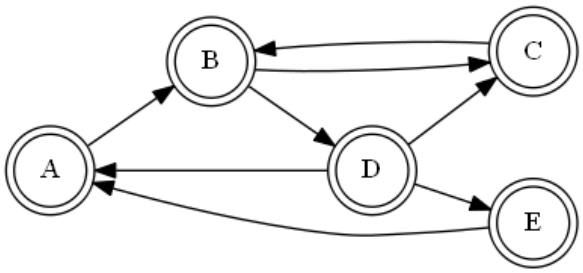

Figure 3: Continue to iterate over the nodes, with the new values of PageRank from step 2.

Step 3, iteration i=2
Compute new PageRank by adding together values computed from the previous iteration's PR.

n	i=0	i=1	PR_2
A	0.2	0.2667	1/3*D + E = 0.1
B	0.2	0.4	A + C = 0.4334
C	0.2	0.1667	1/2*B+1/3*D = 0.333
D	0.2	0.1	1/2*B = 0.2
E	0.2	0.0667	1/3*D = 0.0333

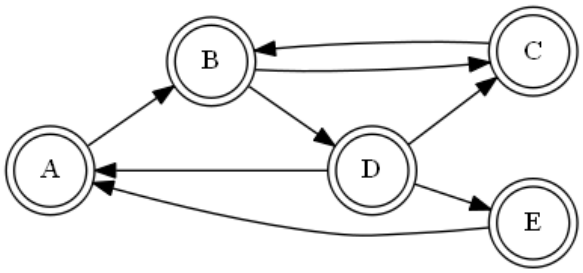

Figure 4: Continue to iterate over the nodes, with the new values of PageRank from previous iteration.

Step 4, iteration converges
Continue repeating step 3 until the values stop changing (within your margin of error).

n	i=0	i=1	i=2	i=10	i=∞
A	0.2	0.2667	0.1	0.1249	0.125
B	0.2	0.4	0.4334	0.3752	0.375
C	0.2	0.1667	0.333	0.2499	0.25
D	0.2	0.1	0.2	0.187	0.1875
E	0.2	0.0667	0.0333	0.0629	0.0625

Note how the columns always sum to 1.

This is quite straightforward, and should be relatively simple to code up in your favorite programming language. You now have an estimate of the probability that a random web surfer will be at any one of

these webpage nodes. The mathematical expression used to describe the example calculations in the previous table is given here:

$$PR(p_i) = \sum_{p_j \in IL_{p_i}} \frac{PR(p_j)}{OL(p_j)} \quad ,$$

where $PR(p_i)$ is the PageRank of a page p_i, and as shown is dependent on the PageRank values of each page p_j contained in the set IL_{p_i} (the set containing all pages that have inlinks to page p_i), and divided by the number of outlinks from page p_j.

There is one significant problem, however, that is unrelated to web designers trying to take advantage of the system. Figure 5 shows addition of two "sink" nodes: F and G. Although there are links to F and G, there are no links back to the initial grouping of 5 nodes. If a page has no links to other pages, it becomes a "sink" and it terminates the random surfing process. Since PageRank is intended to represent a probability that the user will click through to a given node, computing PageRanks for the graph in Figure 5 would suggest that the user would be at node F with 0.5 probability, or at node G with 0.5 probability.

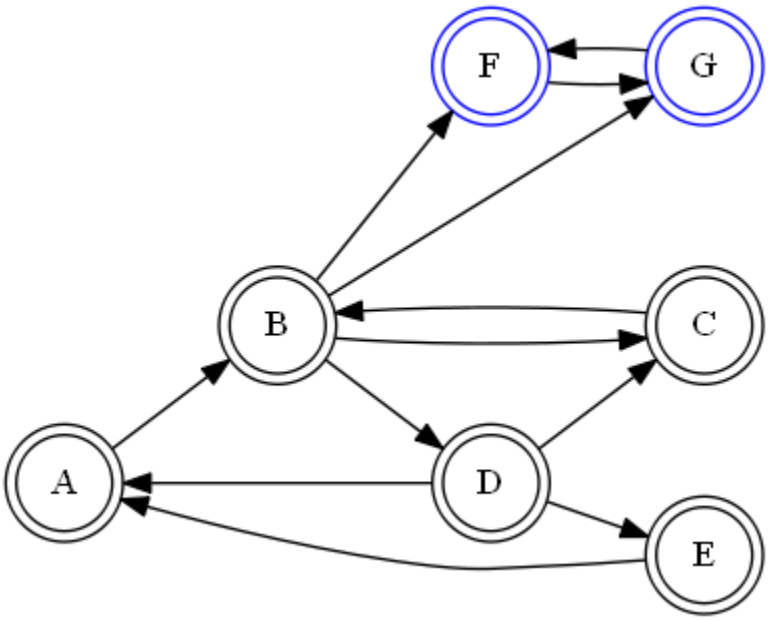

Figure 5: A network with two "sink" nodes, F and G.

Google's inventors decided to modify the PageRank algorithm to reflect that a user occasionally clicks on a random link or a link from their list of bookmarks, and that users rarely spend a large amount of time on webpages without useful outlinks. We add a "damping factor" α that represents the probability a user will click on a random link, and will *not* follow the weighted PageRanks we computed in the previous table. The formula represents a model of a web surfer who gets bored after several clicks and switches to a random page. The PageRank value of a page reflects the chance that the random surfer will land on that page by clicking on a link. If a page has no links to other pages, it

becomes a sink and therefore terminates the random surfing process. If the web surfer arrives at a sink page, it picks another page *at random* and continues surfing again. Our previous equation becomes slightly more complicated, and is given here:

$$PR(p_i) = \frac{1-\alpha}{N} + \alpha \sum_{p_j \in IL_{p_i}} \frac{PR(p_j)}{OL(p_j)} \quad ,$$

where α is the damping factor and N is the total number of pages in the network. From experience, it is known that a damping factor of 0.85 closely approximates reality, so we will use that in our computation.

Deliverables and Due Date

This assignment is in part to teach you an artificial intelligence technique, and in part to help me make sure you have the correct version of Python installed, and to make sure you know the basics of how to program in Python. Therefore, I would like you to follow the skeleton code given here: https://github.com/migai/NEU_AI_2018/tree/master/Assignment_0

Do not try to solve this assignment with advanced Python packages that may be available to do the work for you. I want you to read through the skeleton code and understand it, and make minor changes to it as you complete the assignment. If you run the skeleton code, you should find a result like the one shown in Figure 6.

You are welcome to work together on this assignment, and can help each other and share code with each other – as long as you understand what you are writing! Check with TA Mingwei Zhang if you have any serious problems.

ASSIGNMENT 0 DUE DATE:
at the *beginning* of the first day of class! We will discuss this assignment during the first class lecture.

6 ASSIGNMENT ACTIONS and DELIVERABLES:
1) Run the skeleton code on your machine.

2) Check your Python and package versions for items 0-11 in the printed list shown in Figure 6. Make sure they are all installed on your machine, and make sure the version numbers are at least as high as what my printout above is showing. (Your Python can be 3.6 or 3.7, but upgrade any other packages to match my printout above. Anaconda users: use the Anaconda Navigator executable, or the Anaconda command window function to update any necessary packages.)

3) Look closely at Figure 6 and be sure you understand the PageRank values when you look at the network graph. Can you understand why some nodes show higher probability of landing on that node, and why some are lower?

4) Change the name and the student ID in the skeleton program to be your name and your ID.

```
'''
ASSIGNMENT PART 2:  PAGERANK
'''
# 2.1 Create a network graph with nodes labeled with the letters of your name
### ENTER YOUR NAME HERE.  REMOVE MY NAME AND USE YOUR OWN NAME. ###
### IT MUST BE AT LEAST 6 CHARACTERS ###
student_name = "Michael Gaidis"
### ENTER YOUR STUDENT ID HERE ###
student_id = 20161234
```

(Simply change the two variables shown above, then save your program. You may try running your program at this point, to be sure everything still works properly.)

5) Make adjustments to the skeleton program beneath your student_id to do the following things:

5a) Change the network to be 6 nodes instead of 5 nodes in size.
This should be a simple change to an existing line.
5b) Add an extra row to the bottom of the PageRank table that prints "Sum #.###" so that the #.### is the sum of the 6 elements in the PR-Calc column, is aligned with the PR-Calc column, and is formatted to print only 3 digits after the decimal point.
For this, you might need to create a new variable and add code to sum the values shown in the PR-Calc column, and you will also need to add code to print out the formatted line.
5c) Print a blank line and then two more lines below the bottom of the PageRank table that look like the following:
Maximum PageRank is 0.31 for Node A, which has 3 inlinks.
Minimum PageRank is 0.12 for Node S, which has 1 inlink.
For this, you might need to create at least 2 new variables and add code to find the minimum and maximum of the node names, PageRank values, and number of inlinks. (Use the values shown in the PR-Calc column.) You will also need to add code to print out the blank line and the two formatted lines. Note that each format line will need 3 placeholders: a 2-decimal float, a node name string, and a single-digit integer.
5d) Print another blank line and then one more line that looks like the following:
Submitted by 'Michael Gaidis' with student ID = 20161234.
For this, you will need to add a print statement beneath the others you created.

Note: you may use existing while loops or for loops in the skeleton code to assist with your computations. You may, but do not have to, create new loops. Change the code beneath "student_id" in any way you like.

6) Run your program. Take a screenshot of your output that looks like Figure 7. Rerun your program and make sure all the PageRank values are the same. (Your graph may look different, however.) Then, print 3 copies of your 1-page screenshot, and bring them to class on the first day of class.

```
Path to my active Python executable: C:\Users\mgaidis\Anaconda3\pythonw.exe
0. Python version: 3.7.0 (default, Jun 28 2018, 08:04:48) [MSC v.1912 64 bit (AMD64)]
1. matplotlib version 2.2.3
2. networkx version 2.2
3. nltk version 3.3
4. numpy version 1.15.1
5. pandas version 0.23.4
6. pip version 18.1
7. scikit-image version 0.14.0
8. scikit-learn version 0.19.2
9. scipy version 1.1.0
10. statsmodels version 0.9.0
11. tqdm version 4.26.0

Node    PR-Calc         PR-∞
 L      0.25758         0.25758
 I      0.13196         0.13195
 M      0.18075         0.18076
 A      0.31147         0.31146
 S      0.11824         0.11825
```

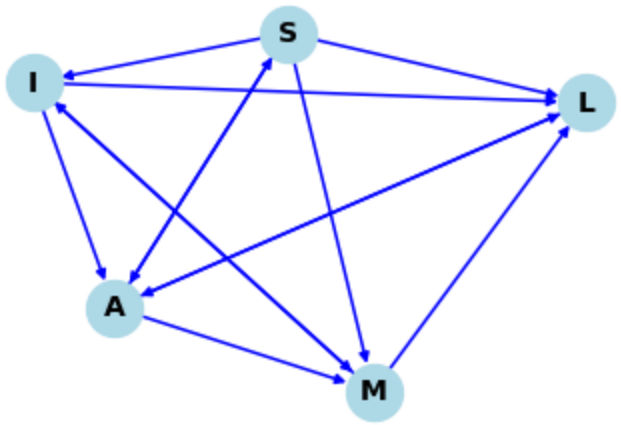

Figure 6: Output from skeleton code for pre-class Assignment 0. The upper text shows installed versions of Python and packages we will use for the course. The table underneath shows the calculated PageRank for each node in the displayed network graph, and also shows what the PageRank would be if the computation approached infinite number of iterations through the formulas described earlier.

```
Path to my active Python executable: C:\Users\mgaidis\Anaconda3\pythonw.exe
0. Python version: 3.7.0 (default, Jun 28 2018, 08:04:48) [MSC v.1912 64 bit (AMD64)]
1. matplotlib version 2.2.3
2. networkx version 2.2
3. nltk version 3.3
4. numpy version 1.15.1
5. pandas version 0.23.4
6. pip version 18.1
7. scikit-image version 0.14.0
8. scikit-learn version 0.19.2
9. scipy version 1.1.0
10. statsmodels version 0.9.0
11. tqdm version 4.26.0

Node    PR-Calc         PR-∞
 L      0.28796         0.28795
 I      0.09317         0.09317
 M      0.18549         0.18548
 A      0.17474         0.17474
 S      0.06214         0.06213
 G      0.19650         0.19652
Sum     1.000

Maximum PageRank is 0.29 for Node L, which has 4 inlinks.
Minimum PageRank is 0.06 for Node S, which has 1 inlinks.

Submitted by 'Michael Gaidis' with student ID = 20161234.
```

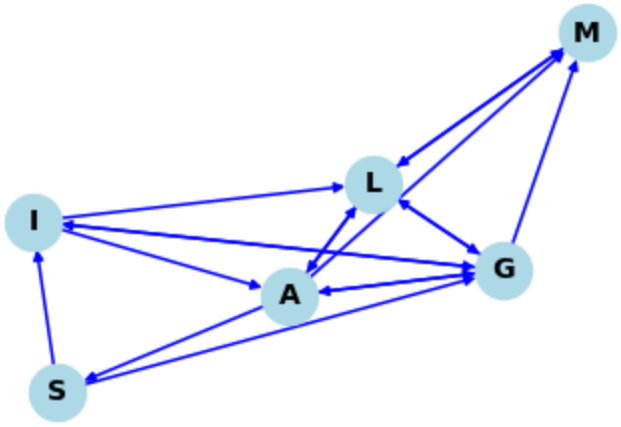

Figure 7: Example screenshot, like the 3 identical pages you will bring to class on the first day.