INFO 371, Spring 2018, Assignment #2

Calculating Eigenfactor Scores (Pseudocode)

Jevin D. West

April 10, 2018

1 Overview

The Eigenfactor (EF) Metrics measure the influence of scholarly journals [3]. They are based on the famous PageRank Algorithm [2], which measures the influence of webpages in the world wide web. The scores and details of the method can be found at Eigenfactor.org. They are also included in Clarivate's (TR) Journal Citation Reports, which are used by scholars, editors and librarians around the world for assessing the impact of scholarly journals.

For this assignment, you will replicate (in python) the method described in this document. You will submit your code with the answers commented at the end of your code. Make sure to write code that can be read (functions, commenting, proper variable names). You may be assessed on how well you (1) replicate the method, (2) your rankings, (3) the speed in which your code runs, and (4) the readability of your code. Everyone will submit their own code, but you can work together. For Quiz #1, there may be a question about this algorithm.

There are seven steps for calculating Eigenfactor Scores:

1. Data Input
2. Creating an Adjacency Matrix
3. Modifying the Adjacency Matrix
4. Identifying the Dangling Nodes
5. Calculating the Stationary Vector
6. Calculationg the EigenFactor (EF) Score

Like the well known Impact Factor metric [1], Eigenfactor measures the number of times that articles published during a *census period* provide citations to papers published during an earlier *target window*. The Impact Factor as reported by TR has a one year census period and uses the two previous years for the target window. In its current form, Eigenfactor has a one year census period and uses the five previous years for the target window.

1.1 Data Input

Four inputs — two files and two constants — are needed:

- Journal Citation File: The list will tell you how often each journal cites all other journals, where we count citations that are given during census period (e.g. 2006) to papers published during the target window (e.g. 2001–2005). This list of journals is stored as a pajek file[1]. You will not need this file since you will be just replicating the example further down in this document.

[1] See MapEquation.org for details on Pajek file formats.

- Article File: this is the file that contains the number of articles that each journal produces in the five previous years. This is the data used for assembling the teleport vector. Again, you won't need this data file because an example teleport vector is given below. These two files are mentioned so that you understand how the algorithm would be applied to real data.

  The constants below, however, will be used in replicating this algorithm.

- Alpha constant ($\alpha = 0.85$)

- Epsilon constant ($\epsilon = 0.00001$)

1.2 Creating an Adjacency Matrix

The journal citation network can be conveniently represented as an adjacency matrix $\mathbf{Z}$, where the $\mathbf{Z_{ij}}$-th entry indicates the number of times that articles published in journal j during the census period cite articles in journal i published during the target window. The dimension of this square matrix is n x n where n is the number of unique journals. For example, suppose there are journals A, B, and C.

	A	B	C
A	2	0	3
B	4	1	1
C	0	2	7

In the adjacency matrix above, journal A cites itself 2 times, it cites journal

B 4 times, and it doesn't cite journal C at all. Journal B receives 4 citations from journal A, 1 citation from itself, and 1 citation from journal C.

1.3 Modifying the Adjacency Matrix

There are some modifications that need to be done to $\mathbf{Z}$ before the eigenvectors can be calculated.

- First, we set the diagonal of $\mathbf{Z}$ to zero (i.e., we set all of the entries $Z_{ii} = 0$). This is done so that journals do not receive credit for self-citations. There is likely zero-ing of diagonals for other variants of PageRank. Search engines may not want to give credit to hyperlinks within a given domain, for example.

- Second, we normalize the columns of the matrix $\mathbf{Z}$ (i.e., divide each entry in a column by the sum of that column). To do this, compute the column sums for each column j as $Z_j = \sum_i \mathbf{Z_{ij}}$. Then divide the entries from each column by the corresponding column sum to get the entries of the $\mathbf{H}$ matrix: $\mathbf{H_{ij}} = \mathbf{Z_{ij}}/Z_j$. There may be columns that sum up to zero (i.e., journals that cite no other journals). These are dangling nodes, and we will deal with them in the next section.

In the example below, we take an adjacency matrix through these two modifications. The matrix you get after these two modifications is $\mathbf{H}$. This example matrix will be used throughout the pseudocode as an example of how to calculate the EF of a journal. The numbers in parentheses next to each journal letter represent the number of papers that each journal has published.

Example raw adjacency matrix ($\mathbf{Z}$)

	A	B	C	D	E	F
$A(3)$	1	0	2	0	4	3
$B(2)$	3	0	1	1	0	0
$C(5)$	2	0	4	0	1	0
$D(1)$	0	0	1	0	0	1
$E(2)$	8	0	3	0	5	2
$F(1)$	0	0	0	0	0	0

1. Set the diagonal to zero

↓

	A	B	C	D	E	F
$A(3)$	0	0	2	0	4	3
$B(2)$	3	0	1	1	0	0
$C(5)$	2	0	0	0	1	0
$D(1)$	0	0	1	0	0	1
$E(2)$	8	0	3	0	0	2
$F(1)$	0	0	0	0	0	0

2. Normalize the columns. This matrix is H.

↓

	A	B	C	D	E	F
$A(3)$	0	0	2/7	0	4/5	3/6
$B(2)$	3/13	0	1/7	1	0	0
$C(5)$	2/13	0	0	0	1/5	0
$D(1)$	0	0	1/7	0	0	1/6
$E(2)$	8/13	0	3/7	0	0	2/6
$F(1)$	0	0	0	0	0	0

1.4 Identifying the Dangling Nodes

As mentioned in the previous section, there will be journals that don't cite any other journals (similar to how some webpages don't cite any other webpages). These journals are called dangling nodes and can be identified by looking for columns that contain all zeros. These columns need to be identified with a row vector of 1's and 0's. Call this vector d. The 1's indicate that a journal is a dangling node; the 0's indicate a non-dangling node. For the example above, d would be the following row vector:

	A	B	C	D	E	F
d_i	0	1	0	0	0	0

1.5 Calculating the Influence Vector

The next step is to construct a transition matrix $\mathbf{P}$ and compute its leading eigenvector. This eigenvector, normalized so that its components sum to 1, will be called the influence vector π^*. This vector gives us the journal weights that we will use in assigning eigenfactor scores.

To calculate the influence vector π^*, we need six inputs: the matrix $\mathbf{H}$ that we just created, an initial start vector $\pi^{(\mathbf{0})}$, the constants α and ϵ, the dangling node vector d and the article vector a.

Article Vector. Let A_{tot} be the total number of articles published by all of the journals. The article vector a is a column vector of the number of articles published in each journal over the (five-year) target window, normalized so that its entries sum to 1. (To do this normalization, divide the number of articles that each journal publishes by A_{tot}). Using the example from above, $A_{\text{tot}} = 3 + 5 + 2 + 1 + 2 + 1 = 14$ and the article vector would be

Article Vector

	a_i
A	3/14
B	2/14
C	5/14
D	1/14
E	2/14
F	1/14

Initial start vector $\pi^{(\mathbf{0})}$. This vector is used in iterating the influence vector. Set each entry of this column vector to $1/n$. For our example, this vector would look like

	$\pi_{\mathbf{i}}^{(\mathbf{0})}$
A	1/6
B	1/6
C	1/6
D	1/6
E	1/6
F	1/6

Calculating the influence vector π^*. The influence vector π^* is the leading eigenvector (normalized so that its terms sum to one) of the matrix $\mathbf{P}$, defined as follows:[2]

$$\mathbf{P} = \alpha \mathbf{H}' + (1 - \alpha) a.e^T,$$

Here e^T is a row vector of all 1's and $a.e^T$ is thus a matrix with identical columns a. The matrix $\mathbf{H}'$ is the matrix $\mathbf{H}$, with all columns corresponding to dangling nodes replaced with the article vector a. In the example, $\mathbf{H}'$ would be the following matrix (notice the replacement of the B column):

	A	B	C	D	E	F
$A(3)$	0	3/14	2/7	0	4/5	3/6
$B(2)$	3/13	2/14	1/7	1	0	0
$C(5)$	2/13	5/14	0	0	1/5	0
$D(1)$	0	1/14	1/7	0	0	1/6
$E(2)$	8/13	2/14	3/7	0	0	2/6
$F(1)$	0	1/14	0	0	0	0

Because $\mathbf{P}$ will be an irreducible aperiodic Markov chain by construction, it will have a unique leading eigenvector by the Perron-Frobenius theorem. We could compute the normalized leading eigenvector of the matrix P directly using the power method, but this involves repeated matrix multiplication operations on the dense matrix $\mathbf{P}$ and thus is computationally

[2]This matrix describes a stochastic process in which a random walker moves through the scientific literature; it is analogous to the "google matrix" that Google uses to compute the PageRank scores of websites. The stochastic process can be interpreted as follows: a fraction α of the time the random walker follows citations and a fraction $1 - \alpha$ of the time the random walker "teleports" to a random journal chosen at a frequency proportional to the number of articles published.

intensive. Instead, we can use an alternative approach that involves only operations on the sparse matrix $\mathbf{H}$ and thus is far faster[3]. To compute the influence vector rapidly, we will iterate the following equation

$$\pi^{(k+1)} = \alpha\, \mathbf{H}\, \pi^{(k)} + [\alpha\, d.\pi^{(k)} + (1 - \alpha)]a$$

This iteration will converge uniquely to the leading eigenvector of $\mathbf{P}$, normalized so that its terms sum to 1. To find this eigenvector, iterate repeatedly. After each iteration, check to see if the residual ($\tau = \pi^{(k+1)} - \pi^{(k)}$) is less than ϵ by calculating the L1 norm. If it is, then $\pi^* \approx \pi^{(k+1)}$ is the influence vector. Typically, this does not take more than 100 iterations with $\epsilon = 0.00001$. Using the raw adjacency matrx example above and the corresponding article vector, the stationary vector converges after 16 iterations to the following vector with $\alpha = 0.85$ and $\epsilon = 0.00001$:

	$\pi^*_{\mathbf{i}}$
A	0.3040
B	0.1636
C	0.1898
D	0.0466
E	0.2753
F	0.0206

[3]Notice that the equation below uses the matrix $\mathbf{H}$, without the dangling node columns replaced, not the matrix $\mathbf{H}'$. In fact, one does not need to ever construct the matrix $\mathbf{H}'$ in the process of doing these calculations.

1.6 Calculating Eigenfactor ($\mathbf{EF_i}$)

The vector of eigenfactor values for each journal is given by the dot product of the H matrix and the influence vector π^*, normalized to sum to 1 and then multiplied by 100 to convert the values from fractions to percentages:

$$\mathrm{EF} = 100\,\frac{\mathbf{H}.\pi^*}{\sum_i [\mathbf{H}.\pi^*]_i}$$

The Eigenfactor values for our example are thus

	EF_i
A	34.0510
B	17.2037
C	12.1755
D	3.6532
E	32.9166
F	0.0000

Optimizing your code: If you calculated the same EF values as listed above, congratulations! You are almost finished. The next step involves optimizing your code so that it can run in a reasonable time on real data. On canvas, you will find the file called "links.txt". This includes 10,747 unique journals (nodes); 4,283,119 unique edges; and 14,927,795 citations. The number in the first two columns represent journals and the third column represents the number of citations from column1 to column2. In other words, (1) the first column represents the citing journal, (2) the second column represents the cited journal, and (3) the third column represents the number of citations between the citing and cited journal. You will not need an article

file, as noted above. Instead, you can assume all articles publish one paper (i.e., a uniform teleport).

To finish the assignment, you need to input this network, run your code and calculate scores for each one of these journals. If your code runs within a few minutes or less, you are finished. Good job! Your code on the small example in this pseudocode used the right data structures and your functions work well with bigger networks. If your code takes too long, that is ok. This your chance to practice optimizing your code with real world data.

Once your code is running in a reasonable time, (a) report the scores for the top 20 journals, (b) report the time it took to run your code on this real network and (c) report the number of iterations it took to get to your answer.

Clean up your code (comment and use variables that you can remember in the future). Turn in your python script (not a Jupyter notebook file) with your reported scores commented out at the bottom of your script.

References

[1] Eugene Garfield. The history and meaning of the journal impact factor. *Jama*, 295(1):90–93, 2006.

[2] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: bringing order to the web. 1999.

[3] J.D. West, T.C. Bergstrom, and C.T. Bergstrom. The eigenfactor metrics: A network approach to assessing scholarly journals. *College and Research Libraries*, 71(3):236–244, 2010.