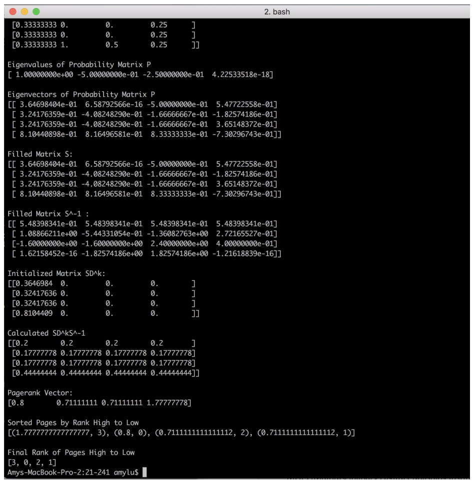
“Blink”: A Backlinks-Based Page Rank Algorithm

Kayla Leung, Amy Lu



```
 [0.33333333 0.         0.         0.25      ]
 [0.33333333 0.         0.         0.25      ]
 [0.33333333 1.         0.5        0.25      ]]

Eigenvalues of Probability Matrix P
[ 1.00000000e+00 -5.00000000e-01 -2.50000000e-01  4.22533518e-18]

Eigenvectors of Probability Matrix P
[[ 3.64698404e-01  6.58792566e-16 -5.00000000e-01  5.47722558e-01]
 [ 3.24176359e-01 -4.08248290e-01 -1.66666667e-01 -1.82574186e-01]
 [ 3.24176359e-01 -4.08248290e-01 -1.66666667e-01  3.65148372e-01]
 [ 8.10440898e-01  8.16496581e-01  8.33333333e-01 -7.30296743e-01]]

Filled Matrix S:
[[ 3.64698404e-01  6.58792566e-16 -5.00000000e-01  5.47722558e-01]
 [ 3.24176359e-01 -4.08248290e-01 -1.66666667e-01 -1.82574186e-01]
 [ 3.24176359e-01 -4.08248290e-01 -1.66666667e-01  3.65148372e-01]
 [ 8.10440898e-01  8.16496581e-01  8.33333333e-01 -7.30296743e-01]]

Filled Matrix S^-1 :
[[ 5.48398341e-01  5.48398341e-01  5.48398341e-01  5.48398341e-01]
 [ 1.08866211e+00 -5.44331054e-01 -1.36082763e+00  2.72165527e-01]
 [-1.60000000e+00 -1.60000000e+00  2.40000000e+00  4.00000000e-01]
 [ 1.62158452e-16 -1.82574186e+00  1.82574186e+00 -1.21618839e-16]]

Initialized Matrix SD^k:
[[0.3646984  0.         0.         0.        ]
 [0.32417636 0.         0.         0.        ]
 [0.32417636 0.         0.         0.        ]
 [0.8104409  0.         0.         0.        ]]

Calculated SD^kS^-1
[[0.2        0.2        0.2        0.2       ]
 [0.17777778 0.17777778 0.17777778 0.17777778]
 [0.17777778 0.17777778 0.17777778 0.17777778]
 [0.44444444 0.44444444 0.44444444 0.44444444]]

Pagerank Vector:
[0.8        0.71111111 0.71111111 1.77777778]

Sorted Pages by Rank High to Low
[(1.7777777777777777, 3), (0.8, 0), (0.7111111111111112, 2), (0.7111111111111112, 1)]

Final Rank of Pages High to Low
[3, 0, 2, 1]
Amys-MacBook-Pro-2:21-241 amylu$ 
```

Figure 1. Terminal output of the *Blink* program

Introduction

The PageRank algorithm was first invented in 1998 by Larry Page and Sergei Brin, the founders of Google, in order to rank websites by their relative importance. The basis of the algorithm is the concept that more relevant websites have a greater likelihood of being linked to by outside websites. Shortly after developing PageRank, Page and Brin created the search engine “BackRub”, so named because it used backlinks to estimate the importance of websites. “BackRub” is more famously known today as “Google” . To this day, PageRank continues to serve as the groundwork for all of Google’s web-search tools.

A key component of the PageRank algorithm is the use of Markov chains. Markov chains are defined as sequence models in which the state of the next event depends solely on the state of the previous event. To help illustrate this concept, a commonly used analogy for the Markov property is the “drunkard’s walk” – a drunk may stumble in any direction, but will move only one step from his current position. This property of Markov Chains lends itself to countless mathematical and real-world applications. More technically, the probability distributions of Markov Chains can be represented in a transition matrix that can predict future states of a system, given a vector of the current state. Some important applications of Markov Chains include calculating exchange rates of currencies, population growth, storage systems, and internet web-searches.

In this paper, we shall discuss (i) the math underlying Markov Chains and (ii) the building of a Markov chain based algorithm, “*Blink*,” to rank lists of links.

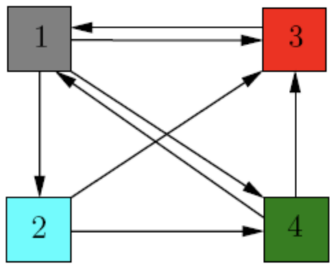


Figure 2. Representation of websites as directed graph, where the boxes are pages and the arrows are links.

Background

What happens when you enter a query in a search engine? In order to best sort the results by relevance, search engines employ methods of ranking. The most popular sorting algorithm is PageRank, which uses the number of links to a page to assess its significance. To better depict this concept, we picture a set of internet websites as a directed graph, where the websites are nodes and links are edges. In Figure 2, we can see that the four webpages are connected by a serires of links. For instance, page 1 links to all three other pages. We assign each of the pages a value, such that if a node has k outgoing edges, it will pass on $\frac{1}{k}$ of its importance to each of the nodes that it links to. Hence, we represent the directed graph in Figure 2 as a transition matrix

$$A = \begin{bmatrix} 0 & 0 & 1 & 1/2 \\ 1/3 & 0 & 0 & 0 \\ 1/3 & 1/2 & 0 & 1/2 \\ 1/3 & 1/2 & 0 & 0 \end{bmatrix}$$

A is an example of a positive, column-stochastic matrix because the entries in each of its columns sums to one. This is an important feature of A because it allows us to use the Perron-Frobenius Theorem to assert that A will have a largest eigenvalue of 1 and that A will eventually converge to the eigenvector corresponding to this eigenvalue. Computationally speaking, we can start with an all 1's vector and multiply A in order to get an approximation of this eigenvector, or the PageRank vector.

Terminology

- Markov chains: A sequence of random variables that that follow the Markov property, i.e. a linked chain of events in which what happens next depends only on the current system state.
- Perron-Frobenius Theorem: A real positive square matrix with positive entries has a unique maximal eigenvalue of multiplicity 1 and a corresponding eigenvector with positive entries.

- Column-stochastic matrix: A matrix is column-stochastic if all of its entries are nonnegative and the entries in each column sum to one.
- Transition matrix: Define a_{ij} to be the probability of the system being in state i after it was in state j (at any observation). The matrix $A = (a_{ij})$ is called the Transition matrix
- Dangling node: A node, representing a webpage in a graph, is a dangling node if it does not contain any outgoing link.

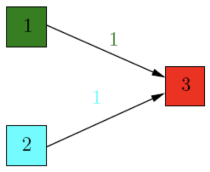


Figure 3. Visual representation of a dangling node

- Independent Eigenvector Theorem: If A is an N x N complex matrix with N distinct eigenvalues, then any set of N corresponding eigenvectors form a basis for C_N.

Materials and Methods

Our program was coded in Python Version 3.4.6. The NumPy 1.15.4 scientific computing package was used for a number of mathematical calculations, specifically for the use of finding the eigenvalues and eigenvectors of a matrix as well as finding the inverse and transpose of a matrix.

The algorithm for our program, Blink, is based on the premise that the importance of any webpage can be determined by the number of other external pages that link to it (i.e. the mos amount of backlinks).

Probability Matrix Generation

The rank function takes in a Markov chain of links, called *links*, represented as a Python list of lists whose elements are integer numbers of the links between pages. The total number of links, or the length of the outer list, is recorded in a variable n. We put this data into matrix format by creating an $n \times n$ NumPy matrix A, whose entries are defined as

$$A_{ij} = \begin{cases} \frac{1}{links[j]} & \text{if there exists a link from page j to page i} \\ \frac{1}{n} & \text{if page j is a dangling node} \\ 0 & \text{otherwise} \end{cases}$$

As defined previously, we consider a page to be a dangling node if it does not contain any outgoing links, i.e. other pages may link to it but it does not have any links to other pages. To avoid all zero columns, and to ensure that our matrix is column-stochastic, we replace entries of dangling node pages with $\frac{1}{n}$. In real life, this would represent a web surfer who jumps to a new random page on the site after reaching a dead end on a page, he or she is equally likely to choose any of the other pages to jump to.

Application of Perron-Frobenius Theorem

Our function now has an $n \times n$ probability matrix A that holds the data out all the outgoing links from each page. Matrix A is column-stochastic because all of the entries are positive and the entries in each column sum to one. Hence, we can use the Perron-Frobenius theorem, which states that any column-stochastic matrix with positive entries will have 1 as its largest eigenvalue.

Diagonalization

For the sake of simplifying computation, we assume that all the eigenvalues of A are distinct. From here, we find n eigenvectors corresponding to each of the n eigenvalues and can begin constructing our diagonalization matrices. Let S be a matrix with the normalized eigenvectors of A as its columns. Since the columns of A are linearly independent, by the Independent Eigenvector Theorem, which implies that the determinant of A is non-zero (by the TFAE), and hence A must be invertible. Let D be a diagonal nxn matrix with the eigenvalues λ_i along its diagonal. By combining SDS^{-1} and setting it equal to A, as $A = SDS^{-1}$ we have thus formed the diagonalization of A. After finding the diagonalization of A, we can much more easily compute the effect of raising A to any power.

Calculation of Pagerank Vector

To calculate the Pagerank vector, we initialize Pagerank vector x as an $n \times 1$ vector with nonnegative entries and repeated replace x with the product Ax until it converges. We construct the initial state vector x as an all ones $n \times 1$ NumPy array and apply the diagonalized form of matrix A, $SDS^{-1}x$ to get the Pagerank values. Lastly, by using the Python *enumerate* function, we convert x into a list of tuples of the page rank and page number. Our rank function then sorts x by page rank from high to low, prints the final rank of the pages, and returns the generated list.

Results

We shall walk through a simple example of the *Blink* program in action. Given an input of lists of lists of links of length 7,

input = [[1, 5], [2, 5], [1, 3, 5], [4], [1, 5], [2, 6], [0, 1]]

the rank function creates an initial state vector x, x = [1, 1, 1, 1, 1, 1, 1].

A column-stochastic probability matrix A is generated from the input data

$$A = \begin{bmatrix} 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.5 \\ 0.5 & 0.0 & 0.333 & 0.0 & 0.5 & 0.0 & 0.5 \\ 0.0 & 0.5 & 0.0 & 0.0 & 0.0 & 0.5 & 0.0 \\ 0.0 & 0.0 & 0.3 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 & 0.0 & 0.0 & 0.0 \\ 0.5 & 0.5 & 0.4 & 0.0 & 0.5 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.5 & 0.0 \end{bmatrix}$$

the eigenvalues/eigenvectors of A are solved for,

Eigenvalues of A = [1.00000000e+00+0.j, -6.51612424e-01+0.j, -1.46442432e-01+0.55098627j, -1.46442432e-01-0.55098627j, -7.52167739e-17+0.j, -2.77513562e-02+0.3124031j, -2.77513562e-02-0.3124031j]

Now we diagonalize A by expressing it in the form $A = SDS^{-1}$, and then multiply x is to SDS^{-1} to get the Pagerank vector,

x = [0.42567568-2.48637574e-17j 1.41891892-8.28791912e-17j 1.56081081-9.11671104e-17j 0.52027027-3.03890368e-17j 0.52027027-3.03890368e-17j 1.7027027 -9.94550295e-17j 0.85135135-4.97275147e-17j]

which is enumerated and sorted. At last, pulling the node index corresponding to each of the sorted values, we get the ranked list of pages,

[5, 2, 1, 6, 3, 4, 0]

Conclusion

In this paper, we present the program *Blink* as a simple, user-friendly Pagerank algorithm using Markov chains. Given any list of links, *Blink* is able to rank the pages by importance. Although, in practice, a simplified program like *Blink* would not be sufficient to handle the massive number of webpages parsed by search engines, *Blink* provides a helpful overview of how the Pagerank algorithm works at a basic level. In the future, it would be interesting to expand upon this program to include more complicated analysis and rigorous mathematical theory. But for now, *Blink* more than suffices as an integrative learning tool for linear algebra students.

Bibliography

"How Google Works: Markov Chains and Eigenvalues." *Klein Project Blog*, 7 Feb. 2017. Web. 10 Dec. 2018. blog.kleinproject.org/?p=280.

"Markov Chain." *Wikipedia*, Wikimedia Foundation, 7 Dec. 2018. Web. 10 Dec. 2018. en.wikipedia.org/wiki/Markovchain.

"PageRank." *Wikipedia*, Wikimedia Foundation, 7 Dec. 2018. Web. 10 Dec. 2018. en.wikipedia.org/wiki/PageRank.

Rogers, Ian. "Pagerank Explained Correctly with Examples." Princeton University, The Trustees of Princeton University. Web. 10 Dec. 2018.
www.cs.princeton.edu/ chazelle/courses/BIB/pagerank.htm.

Shum, Kenneth. "Notes on PageRank Algorithm." ENGG2012B Advanced Engineering Mathematics. 4 Mar. 2013, The Chinese University of Hong Kong, The Chinese University of Hong Kong. Web. 10 Dec. 2018.
http://home.ie.cuhk.edu.hk/ wkshum/papers/pagerank.pdf

Soni, Devin. "Introduction to Markov Chains – Towards Data Science." *Towards Data Science*, Towards Data Science, 5 Mar. 2018. Web. 10 Dec. 2018.
towardsdatascience.com/introduction-to-markov-chains-50da3645a50d.

Tanase, Raluca, and Remus Radu. "Lecture 3: PageRank Algorithm: The Mathematics of Google Search." Brussels Sprouts and the Euler Characteristic. Web. 10 Dec. 2018.
http://pi.math.cornell.edu/ mec/Winter2009/RalucaRemus/Lecture3/lecture3.html