# 21241 Fall 22 Final Project: PageRank and HITS

Audrey Hasson "ahasson", Doris Wang "boleiw"

December 2022

## 1 Introduction and Background

The internet today is made up of millions of sites and an overwhelming amount of information. Search engines, such as Google, sort through this massive web of content to find the portions users want to see. These engines rely on search algorithms, which take in some subset of the sites on the internet and return the subset in order, where the first site has the highest value and each following site has less value. Like all great algorithms, these work using Linear Algebra.[1]

To start, we can represent the internet as a matrix. Think of the internet as a set of sites, all of which link to some subset of those sites. The sites on the internet and the links between them can be represented using an adjacency matrix. We call this matrix A. The rows and columns of A represent the sites on the internet. $A_{ij} = 1$ if site j contains a link to site i, and 0 otherwise.

For example, consider an internet made up of Site P and Site Q. Site Q links to site P, and Site P links to nothing. Represented by an adjacency matrix, this internet looks like this:

$$\begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}$$

The columns are the source sites, and the rows are the destination sites. There is a 0 in row 1, column 1, because Site P does not link to itself. There is a 1 in row 1, column 2, because Site Q does link to Site P.

When one site links to k sites, the probability of going to any one of k sites from that source is (1/k). Thus we can adjust our matrix A to represent the probability of going to any destination from any source. In this matrix, $A_{ij} = \frac{1}{k}$ (where k is the number of destinations source j links to) if site j contains a link to site i.

To illustrate this, consider an internet made up of Sites A, B and C, with the following links.

A links to B and C. B links to A and C. C links to A.

This internet can be represented using the following matrix, where A, B, C are rows and columns 1, 2, 3 respectively:

$$\begin{pmatrix} 0 & \frac{1}{2} & 1 \\ \frac{1}{2} & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 \end{pmatrix}$$

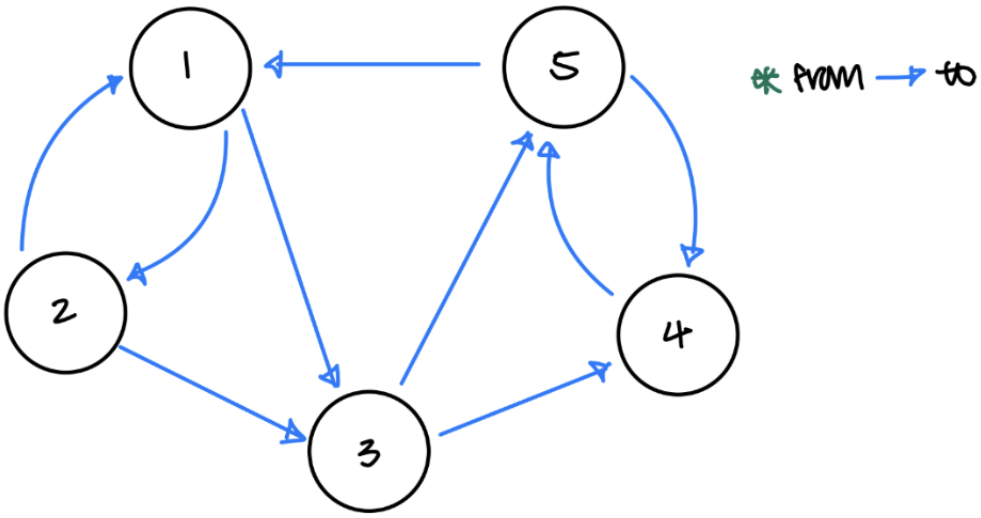
Note that site A and B both link to 2 sites. Therefore, the probability of going to any one of the sites that A links to is $\frac{1}{2}$, and the same goes for B.

We will call this matrix our transition matrix, because the component $A_{ij}$ gives the probability that the user will transition from site j to site i.

These key matrices, the adjacency matrix and the transition matrix, heavily inform two major search algorithms: the Pagerank algorithm and the HITS algorithm. Below we explore these algorithms further.

## 2 Definitions in this Article

A node graph is a visual representation of the relationships between nodes (websites), and the links between them. Our example node graph consists of five nodes, with nine different relationships between them, as visualized below.




We chose to represent the node graph as a list for the input of our function. Each element of the list is a 2-element tuple, with the first element being the 'from' node, and the second element being a list of the 'to' nodes for that 'form' node. This input reflects the structure of an adjacency matrix for the node graph, necessary for the first step in using both the PageRank and the HITS algorithms. The first element of each of the tuples represents the columns, while the list for the second element of the tuple represents the rows of those columns that contain a '1'.

Our input for our sample internet looks this:

```
inputExample = [(1, [2, 3]), (2, [3,1]), (3, [5,4]), (4, [5]), (5, [4, 1])]
```

Which corresponds to this adjacency matrix:

$$\begin{pmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{pmatrix}$$

## 3 PageRank

*What is PageRank?*

PageRank is an algorithm that uses matrix calculation to rank the importance of a site through how often the website is referenced by other websites. This algorithm is one of the first algorithms used by Google to rank web pages for their search results. [6]

*How does it work?*
The PageRank algorithm takes in a mathematical representation of the previously mentioned node graph for N nodes to construct and output a N x 1 vector, such that the value in the ith row of the vector represents the importance of the ith website. From this output, we can conclude websites corresponding with a greater value in the vector have a higher importance than those corresponding to lower values.

*Components of PageRank in Matrix Notation*
To calculate the ranks of web pages using matrix calculation at each move, we follow the formula:

$$\mathbf{R}(t+1) = d\mathcal{M}\mathbf{R}(t) + (1-d)\frac{1}{N} \cdot \mathbf{1}$$

- N represents the number of web pages in our internet space.
- M represents the transition probability matrix. which displays the probability of an internet user going from one webpage to another, for all the N webpages in our internet space. This matrix is calculated via the adjacency matrix (see introduction).
- $R(t)$ is the rankings of the web pages from that last move.
- d is the damping factor, 0.85, which is the probability that an internet user arrives at any of the N web pages by clicking on the links between websites. On the other hand, the complement of d, (1-d), is the probability that an internet user arrives at any of the web pages via other methods and not clicking on the links between sites (bookmarked links, search bar, etc.).
- $\mathbf{1} \cdot \frac{1}{N}$ is the initial probability distribution vector which has dimensions N x 1, where each element of the vector is $\frac{1}{N}$.

We can interpret the formula as a summation of two pairwise disjoint probability cases, where the first case is the internet user using links between web pages to arrive at a web page, and the second is the internet user using other methods to arrive at a web page.

For the first case, we multiply by the damping factor to take into account the probability that the user is using links. Then we multiply the rankings of the web pages from the last move ($R(t)$), by the transition probability matrix to get the probability of making a move from the user's current webpage to another webpage. In this case, the rankings on the very first move (such that t = 0), will be the initial probability distribution vector.

For the second case, we multiply by the complement of the damping factor to take into account the probability that the user is not using links. Then we multiply by the initial probability distribution vector because each webpage is equally likely to be arrived upon via this method.

In order to calculate the final rankings of the web pages, we compute the ranks for more and more moves until the values for the ranking vector converge. Finally, we rank the web pages in correspondence of the rows of the ranking vector such that the higher the value of that row, the more important that page is.

To summarize, we:

1. Initialize a probability distribution vector with equal values (1/N) for every row, and derive a transition probability matrix using the adjacency matrix
2. For many iterations, plug in the ranking vector from the previous move into the PageRank formula
3. Return the ranking vector of the final move

*Julia Program*
Our overall pagerank function uses two helper functions: one to transform the list representation of our node graph as an adjacency matrix, and another to transform the adjacency matrix into a transition matrix.

We start off with our node graph represented in list format.

```
inputExample = [(1, [2, 3]), (2, [3,1]), (3, [5,4]), (4, [5]), (5, [4, 1])]
```

Then we transform the list input into an adjacency matrix using a helper function.

Recall that the input for this function is our graph of nodes, represented as a 2D list like so: [(siteA, [list of sites that siteA points to], siteB, [list of sites that siteB points to]...]
This function initializes a square matrix of zeros with dimension equal to the number of sites in the graph. For each site $i$, it iterates through the sites that $i$ links to and places a one in the matrix at the corresponding position.

```
#takes in a list representing an iternet and outputs an adjacency matrix
function adj(input)
    size = length(input)
    A = zeros((size, size))
    for col in 1:size
        tos = input[col][2]
        for i in 1:(length(tos))
            row = tos[i]
            index = (col-1)*size + row
            setindex!(A, [1], [index])
        end
    end
    A
end
```

The adjacency matrix we receive is:

$$\begin{pmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{pmatrix}$$

Then we transform the adjacency matrix into a transitional probability matrix using another helper function.

```
# Takes input and makes markov matrix
function tranM(input)
    M = adj(input)
    size = length(input)
    for col in 1:size
        tos = input[col][2]
        divisor = length(tos)
        for row in 1:size
            curVal = getindex(M, row, col)
            if divisor == 0
                newVal = 0
            else
```

```
                newVal = curVal/divisor
            end
            index = (col-1)*size + row
            setindex!(M, [newVal], [index])
        end
    end
    M
end
```

The transition probability matrix we receive is:

$$\begin{pmatrix} 0.0 & 0.5 & 0.0 & 0.0 & 0.5 \\ 0.5 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.5 & 0.5 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.5 & 0.0 & 0.5 \\ 0.0 & 0.0 & 0.5 & 1.0 & 0.0 \end{pmatrix}$$

Finally, our overall pagerank function, which calculates in accordance with the original PageRank formula from above, goes through 43 iterations to converge.

```
function pagerank(input)
   transitionM = tranM(input)
   damp = .85
   N = length(input)
   c = (1-damp)/N * ones(N, 1)
   rank = 1/N * ones(N, 1)
   for iteration in 1:43
       rank = damp*transitionM*rank + c
   end
   rank
end
```

And returns the following matrix:

$$\begin{pmatrix} 0.20304907906226435 \\ 0.11629585865260988 \\ 0.16572159854506968 \\ 0.22405501854037257 \\ 0.29087844519968353 \end{pmatrix}$$

Meaning from the highest importance to the lowest importance, our five web pages rank:

1. 5
2. 4
3. 1
4. 3
5. 2

# 4 HITS

*Introduction to Algorithm*
The HITS (hyperlink-induced topic search) algorithm divides sites into two categories: hubs and authorities.

An authority is a site that many other sites link to; it can be considered the end-destination for information. An authority is ranked highly if it is linked to often, and if it is linked to by highly ranked hubs.

A hub is a site that links to many other sites; it can be considered like a capital city, or a crossroads, from which the user goes to many subsequent destinations. A hub is ranked highly if it links to many authorities, and if the authorities it links to are highly ranked.

The HITS algorithm takes as input a set of websites and the links between them and outputs two rankings: the rankings of each site as a hub and the rankings of each site as an authority. These rankings, in turn, inform the search results.

Note that when the HITs algorithm is used, it is assumed that an initial search query has already been performed and we are sorting the websites that resulted from that query. In this way, the algorithm differs from PageRank, which operates on the entire web. For the purposes of this paper, we utilize the same internet for each algorithm.

*Linear Algebra in Algorithm*
In order to generate the authority and hub rankings, the HITS algorithm relies on the adjacency matrix (see introduction). The authority score of each website is the sum of the hub scores of all of the websites that point to it. The hub score of each website is the sum of the authority scores of all the websites it points to.[5]

With this information, if we knew the authority scores of each website and the links between each website, we could calculate the hub scores. Similarly, if we knew the hub scores of each website and the links between each website, we could calculate the authority scores. However, to start, we know only one piece of information: the links between each website.

We represent the links between the websites using the adjacency matrix (see introduction). We start by assuming all websites have the same authority and hub score, that being a score of one. We represent the authority and hub scores as vectors a and h respectively, where $a_i$ = the authority score for site $i$ and $h_i$ = the hub score for site i. Vectors $\mathbf{a}$ and $\mathbf{h}$ are both initially the ones vector.

To calculate accurate authority and hub scores, we start with the initial ones vectors and based on those values, we compute the authority and hub scores over and over again until the values converge. In order to compute the new authority scores for each site, we add up the hub scores for all of the sites it links to. This is the same as computing $A \cdot \mathbf{h}$, where A is the adjacency matrix and h is the vector of hub scores. [3]

These computations are equivalent because $A_{ij} = 1$ if site j contains a link to site i, so that multiplying $A \cdot \mathbf{h}$ will yield a vector $\mathbf{h}'$ with the same dimensions as $\mathbf{h}$, but with the property $\mathbf{h}'_i = \sum_{j=0}^{j=N} A_{ij} h_j$, where N is the total number of sites. Likewise, the hub scores of each website calculated by adding up the authority scores of all sites they link to, so that they can be computed via $A^T \cdot \mathbf{a}$. By taking the transpose of the adjacency matrix, we set the columns to represent "to" and the rows to represent "from." This follows the same process as calculating the authority scores, but instead of adding up the scores of all of the websites that link to the website in question, we add up all the scores of the websites the website in question links to.

We want to compute these values over and over again until they converge. However, the components of the vectors will simply get larger and larger every time without converging. In order to ensure that the vectors converge, we force the sum of the vector components to be one at each step. In

other words, we divide each vector by the sum of its components before computing the next step. This will guarantee that the vectors converge at a value eventually.[5]

In summary, the steps of the algorithm are as follows:

1. Initialize two ones vectors with dimension N x 1, where N is the number of sites on the internet
2. For many iterations:
   Set a equal to $A \cdot \mathbf{h}$
   Set a equal to $\frac{1}{\text{sum of components of } \mathbf{a}} \cdot \mathbf{a}$
   Set h equal to $A^T \cdot \mathbf{a}$
   Set h equal to $\frac{1}{\text{sum of components of } \mathbf{h}} \cdot \mathbf{h}$
3. Return the vectors **a** and **h**

Note that the more times we iterate, the more accurate our algorithm is. However, these values converge relatively quickly, so that it suffices to iterate approximately 43 times [6].

*Julia Program*
To begin, we use the same adj function that we used in PageRank turn our input into an adjacency matrix.

```
#takes in a list representing an iternet and outputs an adjacency matrix
function adj(input)
    size = length(input)
    A = zeros((size, size))
    for col in 1:size
        tos = input[col][2]
        for i in 1:(length(tos))
            row = tos[i]
            index = (col-1)*size + row
            setindex!(A, [1], [index])
        end
    end
    A
end
```

From here, the steps of our code follow precisely as outlined in the previous section. We begin by initializing **a** and **h** as ones vectors. Next, we set **a** equal to $A \cdot \mathbf{h}$, then set **a** equal to $\frac{1}{\text{sum of components of } \mathbf{a}} \cdot \mathbf{a}$. Similarly, we set **h** equal to $A^T \cdot \mathbf{a}$, then set **h** equal to $\frac{1}{\text{sum of components of } \mathbf{h}} \cdot \mathbf{h}$. We repeat this process 43 times to ensure that the values approach convergence. Last, we return two vectors: our authority ranking and our hub ranking.

```
function hit(input)
    A = adj(input)
    size = length(input)
    h = ones(size, 1)
    a = ones(size, 1)
    for iteration in 1:43
        a = A * h
        a = (1/sum(a))*a
        h = A' * a
        h = (1/sum(h))*h
    end
```

```
    a, h
end
```

When we run our HITs algorithm on our internet, it returns the following:
Input: [(1, [2, 3]), (2, [3,1]), (3, [5,4]), (4, [5]), (5, [4, 1])]
Output:

$$\begin{pmatrix} 0.2846296358500217 \\ 0.08101396309145739 \\ 0.21732060650624632 \\ 0.26157080760486945 \\ 0.15546498694740515 \end{pmatrix}, \quad \begin{pmatrix} 0.15546469013308514 \\ 0.2615705548149283 \\ 0.2173209114247193 \\ 0.0810141313992346 \\ 0.2846297122280325 \end{pmatrix}$$

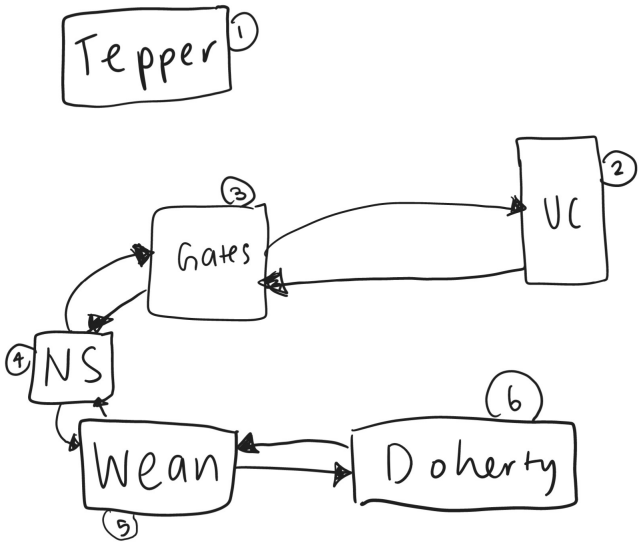
Recall that the $i$th component of $\mathbf{a}$ represents the authority score of the $i$th site, and the $i$th component of $\mathbf{h}$ represents the hub score of the $i$th site. Our sites are numbered 1, 2, 3, 4, 5. We see that the authority ranking is as follows: 1, 4, 3, 5, 2.
The hub ranking is 5, 2, 3, 1, 4.

Back in the terms of a search algorithm: these rankings tell us that site 1 is the best authority on the given topic, and that site 5 has the best links to additional information.

These rankings are both entirely different from the rankings obtained using the PageRank algorithm, but note that that is not entirely unprecedented; though both algorithms are based in analyzing connections between sites, they use different metrics. In particular, HITs typeifies websites, while PageRank returns an overall ranking for all websites.

# 5 Beyond the Web

To investigate the versatility of both algorithms, we extended them to gain more insight on one of our favorite networks: the paths and buildings of Carnegie Mellon.




Based on the graph above, we gave the following input to our algorithms: [(1, []), (2, [3]), (3, [2, 4]), (4, [3, 5]), (5, [4, 6]), (6, [5])]

The PageRank algorithm returned the following:

$$\begin{pmatrix} 0.025000000000000005 \\ 0.3083333333333334 \\ 0.025000000000000005 \\ 0.025000000000000005 \\ 0.6326393234594818 \\ 0.715100442494045 \end{pmatrix}$$

This result ranks the buildings as follows:
1) Doherty
2) Wean
3) UC
4) Tepper, Gates, Newell-Simon (all tied)
There's no distinct connection between the graph above and these rankings. However, the HITs algorithm returned these two vectors:

$$\begin{pmatrix} 0.0 \\ 0.125 \\ 0.25 \\ 0.25 \\ 0.25 \\ 0.125 \end{pmatrix}, \begin{pmatrix} 0.0 \\ 0.14285714285714285 \\ 0.21428571428571427 \\ 0.2857142857142857 \\ 0.21428571428571427 \\ 0.14285714285714285 \end{pmatrix}$$

From this, we see that the authority rankings are as follows:

1. Gates, Newell-Simon, and Wean (tied)
2. UC and Doherty (tied)
3. Tepper

And the hub rankings are as follows:

1. Newell-Simon
2. Gates and Wean (tied)
3. UC and Doherty (tied)
4. Tepper

When considering the map, these rankings are much more intuitive. Tepper is last in both rankings because it neither links to anything nor is linked to by anything. The most authoritative buildings are all central, and the hub are ranked directly in terms of centrality, with Newell-Simon being in the middle of the graph. If this graph was an exhaustive map of Carnegie Mellon, the hub rankings tell us that you could get anywhere from Newell-Simon, and that would be accurate.

# 6 Conclusion

The varied methods of traversing and ranking the vast internet adapt to the needs of users and clients. Though HITs may more accurately describe the small map of Carnegie Mellon, PageRank is the predecessor of today's search engines because it is more applicable and accurate on a very large scale.[4] As we push into and beyond the Age of Information, our algorithms and algebra must be refined to follow us.

# 7 Works Cited

[1] R. R. Raluca Tanase, "Lecture 3," PageRank Algorithm - The Mathematics of Google Search. [Online]. Available: http://pi.math.cornell.edu/ mec/Winter2009/RalucaRemus/Lecture3/lecture3.html. [Accessed: 09-Dec-2022].

[2] R. R. Raluca Tanase, "Lecture 2," Directed Graphs - Transition Matrices. [Online]. Available: http://pi.math.cornell.edu/ mec/Winter2009/RalucaRemus/Lecture2/lecture2.html. [Accessed: 09-Dec-2022].

[3] R. R. Raluca Tanase, "Lecture 4," HITS Algorithm - Hubs and Authorities on the Internet. [Online]. Available: http://pi.math.cornell.edu/ mec/Winter2009/RalucaRemus/Lecture4/lecture4.html. [Accessed: 09-Dec-2022].

[4] N. Grover and R. Wason, "Comparative analysis of pagerank and hits algorithms," International Journal of Engineering Research Technology, 29-Oct-2012. [Online]. Available: https://www.ijert.org/comparative-analysis-of-pagerank-and-hits-algorithms. [Accessed: 09-Dec-2022].

[5] "Hits algorithm and hubs and authorities explained," YouTube, 16-Oct-2018. [Online]. Available: https://www.youtube.com/watch?v=-kiKUYM9Qq8. [Accessed: 09-Dec-2022].

[6] "PageRank," Wikipedia, 22-Nov-2022. [Online]. Available: https://en.wikipedia.org/wiki/PageRank. [Accessed: 09-Dec-2022].