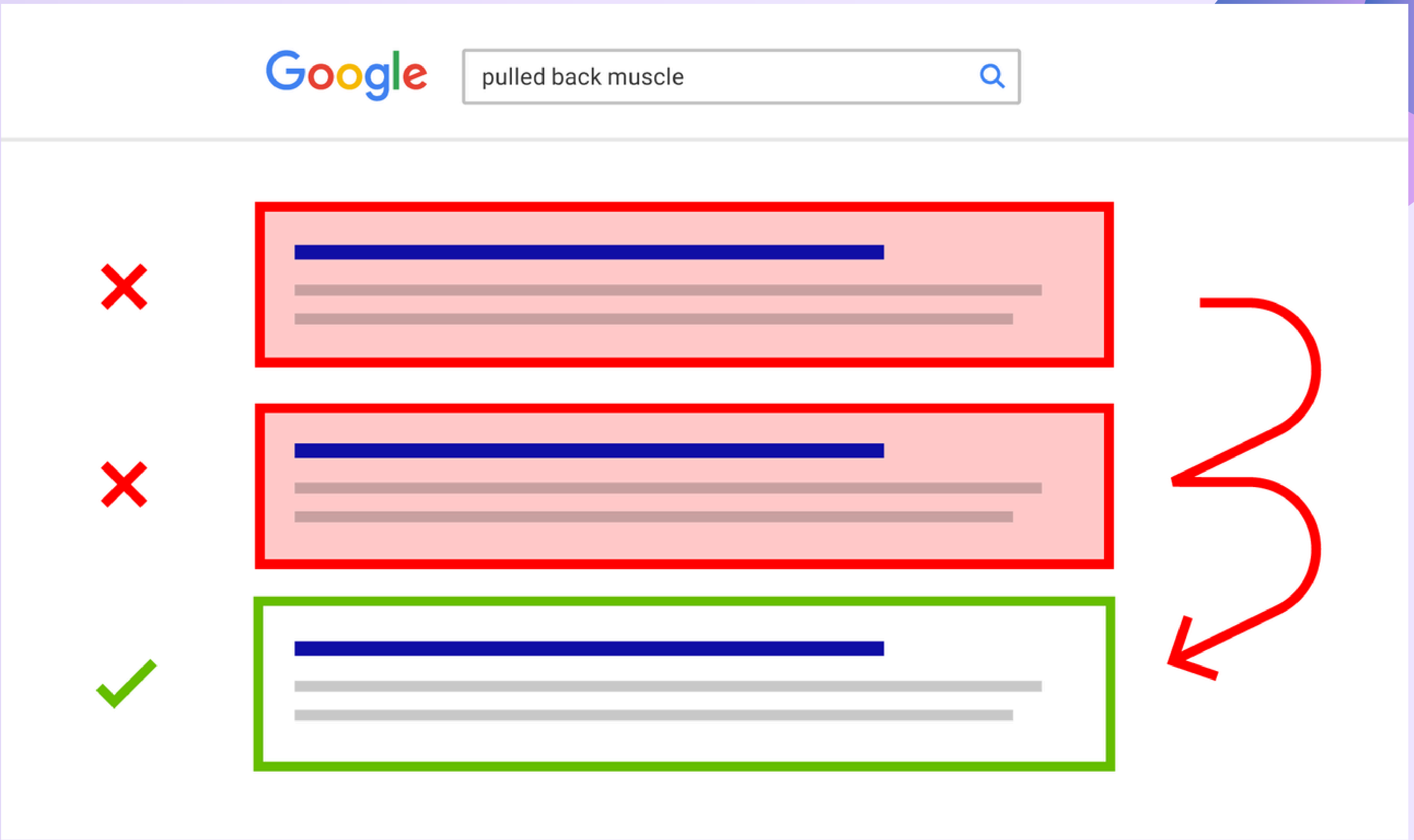
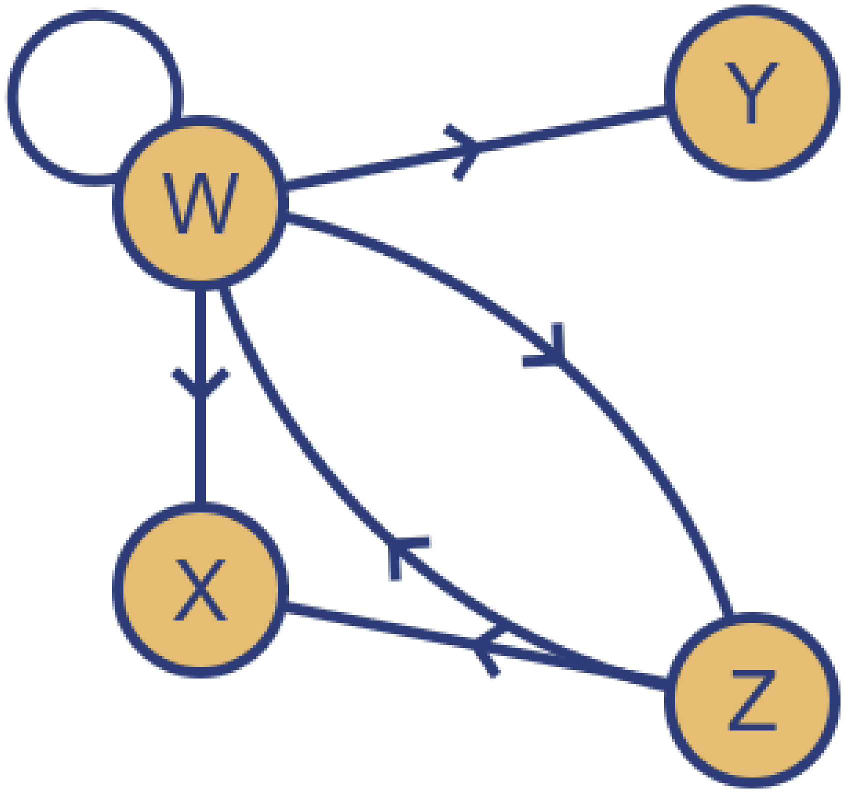
Power Method
PageRank

How does Google Rank Web Pages?





Directed graph with loop

	W	X	Y	Z
W	1	1	1	1
X	0	0	0	0
Y	0	0	0	0
Z	1	1	0	0

PageRank

- **Adjacency Matrix**

  The letters represent web page. The 1 represents if there is a link between that webpage and another. Also this matrix represents a directed graph.

- **PageRank Algorithm**

  R = R x T
  R = PageRank Vector
  T = Transition Matrix (Probability Distribution)

- **PageRank Goal**

  Convert a Matrix into a single Vector that shows a steady state. Using that vector you can see the rank for web pages.

$$\underbrace{A}_{\text{n × n Matrix}}\;\underbrace{\vec{x}}_{\text{Eigenvector}} = \underbrace{\lambda}_{\text{Eigenvalue}}\;\underbrace{\vec{x}}_{\text{Eigenvector}}$$

$$Av - \lambda v = 0$$

$$v(A - \lambda I) = 0$$

$$det(A - \lambda I)$$

Eigenvalue & Eigenvector

- Given our Equation: **TR = R**
- Which is the same as: **TR = 1*R**
- **Eigenvector Definition**: If A is an square nxn matrix, then some scalar multiplied on the eigenvector is equal to the original multiplication of that vector and matrix A.

  Eigenvalue Definition: Scalar used to transform the Eigenvector
- **T = Matrix A**

  R = Eigenvector

  1 = Eigenvalue

Why we should use another Method

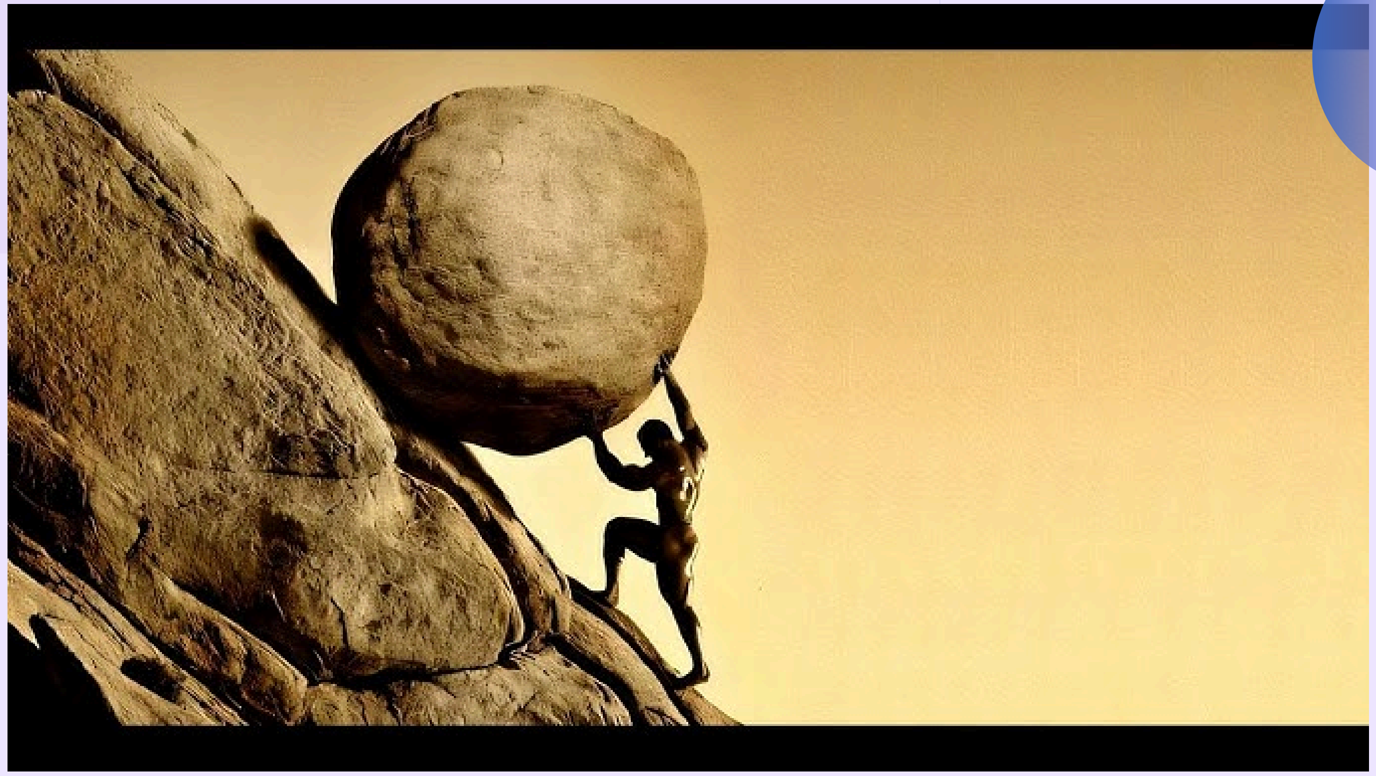
We can compute the determinant, but when the systems get larger the becomes more computationally challenging and expensive.

$$\det\left(\begin{bmatrix}4 & -1\\ 2 & 1\end{bmatrix} - \lambda I\right) = 0$$

$$\det\left(\begin{bmatrix}4 & -1\\ 2 & 1\end{bmatrix} - \lambda\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}\right) = 0$$

$$\det\left(\begin{bmatrix}4 & -1\\ 2 & 1\end{bmatrix} - \begin{bmatrix}\lambda & 0\\ 0 & \lambda\end{bmatrix}\right) = 0$$

$$\det\left(\begin{bmatrix}4-\lambda & -1\\ 2 & 1-\lambda\end{bmatrix}\right) = 0$$



How this relates to the Power Method

- **The Goal of the Power Method**

  To iteratively converge to a single eigenvector

- **Benefits**

  Quicker for bigger systems and Easy to Implement

- **Steps**

  1. Multiply Matrix by kth Vector
  2. Normalize results
  3. Repeat until the value converges

Stop when $\left\|R^{(t+1)} - R^{(t)}\right\|_1 <$ desired tolerance.

Power Method

$$A\mathrm{v} = \lambda \mathrm{v}$$

$$b_{k+1} = \frac{Ab_k}{\|Ab_k\|}$$

$$b_{k+1} = \frac{(A - \mu I)^{-1} b_k}{\|(A - \mu I)^{-1} b_k\|}$$

Code Walkthrough

- **Link Matrix**

  Convert Adjacency Matrix to a link Matrix.

$$L_{ij} := \begin{cases} 1/\ell_j, & \text{if page j links to page i} \\ 0, & \text{otherwise} \end{cases}$$

- **Transition Matrix**

  Links Matrix and the effect of jumping between links.

$$T_{ij} := \begin{cases} e_i, & \text{if } \ell_j = 0 \text{ (i.e. page j has no outgoing links)} \\ (1-d)L_{ij} + de_i, & \text{otherwise} \end{cases}$$

- **Power Method**

  Iteratively converge to an Eigenvector.

$$R^{(t+1)} = TR^{(t)}$$

References

Google PageRank Explained via Power Iteration by Binod Pant, Ronnie Ramirez, Lee Reeves Stanford University

Power Method - Determine Largest Eigenvalue and Eigenvector in Python Geeksforgeeks

Alfio Quarteroni Fausto Saleri · Paola Gervasio Scientific Computing with MATLAB and Octave

Youtube