PageRank

Rebecca C. Steorts
STA 325

Supplemental Material

Optional reading: ESL 14.10

Information retrieval with the web

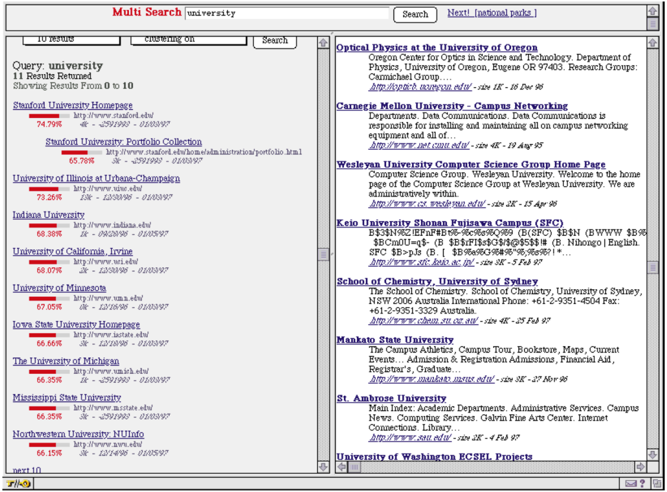
Last time: information retrieval, learned how to compute similarity scores (distances) of documents to a given query string

But what if documents are webpages, and our collection is the whole web (or a big chunk of it)? Now, two problems:

- Techniques from last lectures (normalization, IDF weighting) are computationally infeasible at this scale. There are about 30 billion webpages!
- Some webpages should be assigned more priority than others, for being more important

Fortunately, there is an underlying structure that we can exploit: links between webpages

Web search before Google



(From Page et al. (1999), "The PageRank Citation Ranking: Bringing Order to the Web")

PageRank algorithm

PageRank algorithm: famously invented by Larry Page and Sergei Brin, founders of Google. Assigns a *PageRank* (score, or a measure of importance) to each webpage

- Suppose there are n webpages.
- The PageRank of webpage i is based on its linking webpages (webpage(s) j that link to i),
- Don't just count the number of linking webpages, i.e., don't want to treat all linking webpages equally

Instead, we weight the links from different webpages

- Webpages that link to i, and have high PageRank scores themselves, should be given more weight
- Webpages that link to i, but link to a lot of other webpages in general, should be given less weight

Note that the first idea is circular! (But that's OK)

BrokenRank (almost PageRank) definition

Let $L_{ij} = 1$ if webpage j links to webpage i (written $j \to i$), and $L_{ij} = 0$ otherwise

Also let $m_j = \sum_{k=1}^n L_{kj}$, the total number of webpages that j links to

First we define something that's almost PageRank, but not quite, because it's broken. The BrokenRank p_i of webpage i is

$$p_i = \sum_{j \to i} \frac{p_j}{m_j} = \sum_{j=1}^n \frac{L_{ij}}{m_j} p_j$$

Does this match our ideas from the last slide? Yes: for $j \to i$, the weight is p_j / m_j—this increases with p_j, but decreases with m_j

BrokenRank in matrix notation

Written in matrix notation,

$$p = \begin{pmatrix} p_1 \\ p_2 \\ \vdots \\ p_n \end{pmatrix}, \quad L = \begin{pmatrix} L_{11} & L_{12} & \dots & L_{1n} \\ L_{21} & L_{22} & \dots & L_{2n} \\ \vdots & & & \\ L_{n1} & L_{n2} & \dots & L_{nn} \end{pmatrix},$$

$$M = \begin{pmatrix} m_1 & 0 & \dots & 0 \\ 0 & m_2 & \dots & 0 \\ \vdots & & & \\ 0 & 0 & \dots & m_n \end{pmatrix}$$

Dimensions: p is $n \times 1$, L and M are $n \times n$

Now re-express definition on the previous page: the BrokenRank vector p is defined as $p = LM^{-1}p$

Eigenvalues and eigenvectors

- Let $A = LM^{-1}$.
- Then A is a diagonal matrix and $p = Ap$.
- This means that p is an eigenvector of the matrix A with eigenvalue 1.

Great! Because we know how to compute the eigenvalues and eigenvectors of A, and there are even methods for doing this quickly when A is large and sparse (why is our A sparse?)

But wait ... do we know that A has an eigenvalue of 1, so that such a vector p exists? And even if it does exist, will it be unique (well-defined)?

For these questions, it helps to interpret BrokenRank in terms of a Markov chain

BrokenRank as a Markov chain

$$A = LM^{-1} \quad \text{and} \quad P = A^T$$

- Big picture: A Markov chain says that the probability of moving from state i to j only depends on state j.
- (It doesn't depend anywhere you were before!).
- Think of a Markov Chain as a random process that moves between states numbered $1, \ldots n$ (each step of the process is one move).
- For a Markov chain to have an $n \times n$ transition matrix P, this means

$$\mathrm{P}(\text{go from } i \text{ to } j) = P_{ij}$$

BrokenRank as a Markov chain

- Let $p^{(0)}$ is an n-dimensional vector giving initial probabilities.
- After one step, $p^{(1)} = P^T p^{(0)}$ gives probabilities of being in each state (why?)

As an example: Let $p^{(0)} = (1/n, \ldots, 1/n)$. Then $p^{(1)} = P_n^T p^{(0)}$ and $p_i^{(1)} = \sum_{j=1}^n P_{ji} p_j^{(0)}$.

- Now consider a Markov chain, with the states as webpages, and with transition matrix A^T.
- Note that $(A^T)_{ij} = A_{ji} = L_{ji}/m_i$, so we can describe the chain as

$$P_{ij} = \mathrm{P}(\text{go from } i \text{ to } j) = \begin{cases} 1/m_i & \text{if } i \to j \\ 0 & \text{otherwise} \end{cases}$$

- (Check: does this make sense?) This is like a random surfer, i.e., a person surfing the web by clicking on links uniformly at random

Stationary distribution

A stationary distribution of our Markov chain is a probability vector p (i.e., its entries are ≥ 0 and sum to 1) with $p = P^T p$ (Here, we have p = Ap).

If the Markov chain is strongly connected, meaning that any state can be reached from any other state, then stationary distribution p exists and is unique. Furthermore, we can think of the stationary distribution as the of proportions of visits the chain pays to each state after a very long time (the ergodic theorem):

$$p_i = \lim_{t \to \infty} \frac{\text{\# of visits to state } i \text{ in } t \text{ steps}}{t}$$

Our interpretation: the BrokenRank of p_i is the proportion of time our random surfer spends on webpage i if we let him go forever

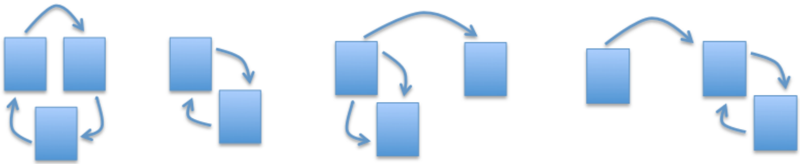
Why is BrokenRank broken?

There's a problem here. Our Markov chain—a random surfer on the web graph—is not strongly connected, in three cases (at least):

Disconnected components

Dangling links

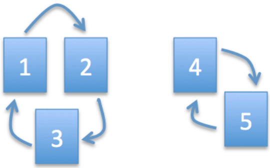
Loops



Actually, even for Markov chains that are not strongly connected, a stationary distribution always exists, but may nonunique

In other words, the BrokenRank vector p exists but is ambiguously defined

BrokenRank example



Here $A = LM^{-1} = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}$

(Check: matches both definitions?)

Here there are two eigenvectors of A with eigenvalue 1:

$$p = \begin{pmatrix} \frac{1}{3} \\ \frac{1}{3} \\ \frac{1}{3} \\ 0 \\ 0 \end{pmatrix} \quad \text{and} \quad p = \begin{pmatrix} 0 \\ 0 \\ 0 \\ \frac{1}{2} \\ \frac{1}{2} \end{pmatrix}$$

These are totally opposite rankings!

PageRank definition

PageRank is given by a small modification of BrokenRank:

$$p_i = \frac{1-d}{n} + d \sum_{j=1}^{n} \frac{L_{ij}}{m_j} p_j,$$

where $0 < d < 1$ is a constant (apparently Google uses $d = 0.85$)

In matrix notation, this is

$$p = (\frac{1-d}{n} E + dLM^{-1}) p,$$

E is the $n \times n$ matrix of 1s, subject to the constraint $\sum_{i=1}^{n} p_i = 1$

(Check: are these definitions the same? Show that the second definition gives the first. Hint: if e is the n-vector of all 1s, then $E = ee^T$, and $e^T p = 1$)

PageRank as a Markov chain

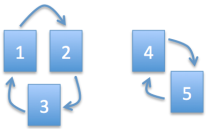
- Let $A = \frac{1-d}{n}E + dLM^{-1}$,
- Consider a Markov chain with transition matrix A^T

$$(A^T)_{ij} = A_{ji} = (1-d)/n + dL_{ji}/m_i$$

$$P_{ij} = \mathrm{P}(\text{go from } i \text{ to } j) = \begin{cases} (1-d)/n + d/m_i & \text{if } i \to j \\ (1-d)/n & \text{otherwise} \end{cases}$$

- The chain moves through a link with probability $(1-d)/n + d/m_i$,
- with probability $(1-d)/n$ it jumps to an unlinked webpage
- Like a random surfer with random jumps.
- Random jumps get rid of our problems: our Markov chain is now strongly connected.
- Stationary distribution (i.e., PageRank vector) p is unique

PageRank example



With $d = 0.85$, $A = \frac{1-d}{n}E + dLM^{-1}$

$$= \frac{0.15}{5} \cdot \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix} + 0.85 \cdot \begin{pmatrix} 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

$$= \begin{pmatrix} 0.03 & 0.03 & 0.88 & 0.03 & 0.03 \\ 0.88 & 0.03 & 0.03 & 0.03 & 0.03 \\ 0.03 & 0.88 & 0.03 & 0.03 & 0.03 \\ 0.03 & 0.03 & 0.03 & 0.03 & 0.88 \\ 0.03 & 0.03 & 0.03 & 0.88 & 0.03 \end{pmatrix}$$

Now only one eigenvector of A with eigenvalue 1: $p = \begin{pmatrix} 0.2 \\ 0.2 \\ 0.2 \\ 0.2 \\ 0.2 \end{pmatrix}$

Computing the PageRank vector

Computing the PageRank vector p via traditional methods, i.e., an eigendecomposition, takes roughly n^3 operations. When $n = 10^{10}$, $n^3 = 10^{30}$. Yikes! (But a bigger concern would be memory ...)

Fortunately, much faster way to compute the eigenvector of A with eigenvalue 1: begin with any initial distribution $p^{(0)}$, and compute

$$\begin{aligned} p^{(1)} &= Ap^{(0)} \\ p^{(2)} &= Ap^{(1)} \\ &\vdots \\ p^{(t)} &= Ap^{(t-1)}, \end{aligned}$$

Then $p^{(t)} \to p$ as $t \to \infty$. In practice, we just repeatedly multiply by A until there isn't much change between iterations

E.g., after 100 iterations, operation count: $100n^2 \ll n^3$ for large n

Computation, continued

There are still important questions remaining about computing the PageRank vector p (with the algorithm presented on last slide):

1. How can we perform each iteration quickly (multiply by A quickly)?
2. How many iterations does it take (generally) to get a reasonable answer?

Broadly, the answers are:

1. Use the sparsity of web graph (how?)
2. Not very many if A large spectral gap (difference between its first and second largest absolute eigenvalues); the largest is 1, the second largest is $\leq d$

(PageRank in R: see the function `page.rank` in package `igraph`)

A basic web search

For a basic web search, given a query, we could do the following:

1. Compute the PageRank vector p once (Google recomputes this from time to time, to stay current)
2. Find the documents containing all words in the query
3. Sort these documents by PageRank, and return the top k (e.g., $k = 50$)

This is a little too simple ... but we can use the similarity scores learned last time, changing the above to:

3. Sort these documents by PageRank, and keep only the top K (e.g., $K = 5000$)
4. Sort by similarity to the query (e.g., normalized, IDF weighted distance), and return the top k (e.g., $k = 50$)

Google uses a combination of PageRank, similarity scores, and other techniques (it's proprietary!)

Variants/extensions of PageRank

A precursor to PageRank:

- Hubs and authorities: using link structure to determine "hubs" and "authorities"; a similar algorithm was used by Ask.com (Kleinberg (1997), "Authoritative Sources in a Hyperlinked Environment")

Following its discovery, there has been a huge amount of work to improve/extend PageRank—and not only at Google! There are many, many academic papers too, here are a few:

- Intelligent surfing: pointing surfer towards textually relevant webpages (Richardson and Domingos (2002), "The Intelligent Surfer: Probabilistic Combination of Link and Content Information in PageRank")
- TrustRank: pointing surfer away from spam (Gyongyi et al. (2004), "Combating Web Spam with TrustRank")
- PigeonRank: pigeons, the real reason for Google's success (`http://www.google.com/onceuponatime/technology/pigeonrank.html`)

Recap: PageRank

PageRank is a ranking algorithm for webpages based on their importance. For a given webpage, its PageRank is based on the webpages that link to it; it helps if these linking webpages have high PageRank themselves; it hurts if these linking webpages also link to a lot of other webpages

We defined it by modifying a simpler ranking system (BrokenRank) that didn't quite work. The PageRank vector p corresponds to the eigenvector of a particular matrix A corresponding to eigenvalue 1. Can also be explained in terms of a Markov chain, interpreted as a random surfer with random jumps. These jumps were crucial, because they made the chain strongly connected, and guaranteed that the PageRank vector (stationary distribution) p is unique

We can compute p by repeatedly multiplying by A. PageRank can be combined with similarity scores for a basic web search