Markov Chains

Website Traffic Prediction

Tanay Biradar

October 4, 2021

Website Traffic and PageRank

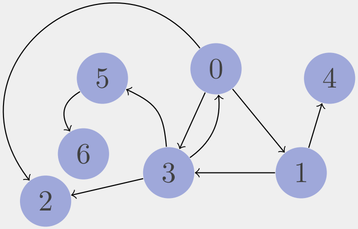
- Search engines use popularity to rank pages
- Popularity can be quantified by links to a page
  - Works in theory, can be abused in practice
- Google used a Markov Model (PageRank) to rank popularity
  - We're looking to predict traffic, but a similar process applies

Central Question

Which page is a user most likely to land on after starting on a given page?

Model

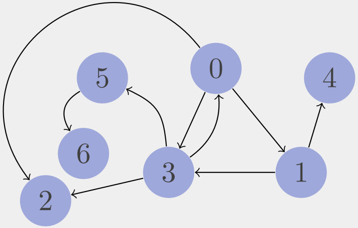
- Represent the internet as a directed graph
  - We're looking at a small slice of the web
  - Assume more links to a page means more likely to land on it
- Edges are links, vertices are web pages
  - Assume equal probability of traversing every link such that [1] $\Sigma w_{out} = 1$, where w is the edge weight
    - The probabilities coming out of every website must sum to 1



[1] https://en.wikipedia.org/wiki/PageRank

The most complex part is by far the data collection

- Perform BFS to make an adjacency list of the internet
    - Keep track of visited nodes to avoid duplicate processing
- Stop after storing 2000 links
    - I don't have the computing power of Google



Data (cont'd)

Adjacency list A stores links between pages
If $A_{ij} = 1$, there is a link from page i to page j

$$A = \begin{bmatrix} a_{00} & \cdots & a_{0n} \\ \vdots & \ddots & \vdots \\ a_{n0} & \cdots & a_{nn} \end{bmatrix}$$

Normalize the adjacency list to satisfy $\forall i \; \Sigma w_{out} = \Sigma_j w_i = 1$
We now have a transition matrix T with probabilities!

Data (cont'd 2)

We can only work with rows/column numbers for the transition matrix
Solution: Map website URLs to unique IDs

```
1 page_rank_markov.tex  2 link_ids.json +
    https://github.com/features/code-review: 366,
    https://github.com/features/project-management: 367,
    https://github.com/team: 368,
    https://help.github.com/en/articles/github-subprocessors-and-cookies: 369,
    https://www.cisa.gov/executive-order-improving-nations-cybersecurity: 370,
    https://partner.github.com: 371,
    https://desktop.github.com: 372,
    https://www.youtube.com/about/: 373,
    https://www.youtube.com/about/press/: 374,
    https://www.youtube.com/about/copyright/: 375,
    https://www.youtube.com/creators/: 376,
    https://www.youtube.com/ads/: 377,
    https://developers.google.com/youtube: 378,
    https://policies.google.com/privacy?hl=en: 379,
    https://www.youtube.com/about/policies/: 380,
```

Benchmarking

- Create adjacency list starting at my personal website
- Create a state vector
  - Create a 0 vector with same number of dimensions as rows in T
  - Start at a given page, use lookup table to make the corresponding entry 1 (web page visits are discrete)
- Transition matrix is not diagonalizable
  - We must simulate and let the state vector converge

Benchmarking (cont'd)

Steps for simulation

1. Multiply transition matrix by state vector
2. Take web page the user has the greatest probability of landing on and set the state vector probability of that page to 1, all others to 0
    2.1 Web page visits are discrete
3. Repeat 1-2 either to satisfaction or to convergence

Results

- After 1000 steps:
- https://github.com → https://services.github.com
  - Landing on as subdomain seems reasonable
- https://irs.gov/ → https://irs.gov/
  - Since our model of the internet is limited, we likely didn't explore many links leading away from this website

- Limitations: We only have access to a limited snapshot of the internet
  - Most links are internal (same domain)
  - Many pages have few outgoing links due
- Potential abuse: Websites can artificially inflate their likelihood of being landed on by creating pages with many links to another page
- If the model is larger, it might work in predicting website traffic, but it is not reliable in practice
- Potential improvement: Consider "quality" of incoming links like PageRank does [2]

[2] https://web.stanford.edu/class/cs54n/handouts/24-GooglePageRankAlgorithm.pdf

Thank you!