MAT 167 Final Project: Google's PageRank

Jasper Dong, Alexander Lin, Sunny Liu

8/4/2024

Our Team

Jasper Dong
jmudong@ucdavis.edu

Alexander Lin
aalin@ucdavis.edu

Sunny Liu
sunliu@ucdavis.edu

Background

- Google's PageRank algorithm was developed by its founders Larry Page and Sergey Brin at Stanford University.
- In a hyperlink network of webpages, PageRank counts the links between webpages and determines which webpages are more "important".
- Given a search query, this metric of "importance" determines the ranking of returned web pages at the time of query.

Our Goal:

Given a hyperlink network and a list of keywords for each webpage, which webpages are the most responsive to certain queries, and what are the web page rankings for each query?

Outline

Eigendecomposition

- Given some Google matrix G of a hyperlink graph, we can solve the eigenvalue problem:

$$\pi^T = \pi^T G \Rightarrow G^T \pi = \pi$$

  where $\boldsymbol{\pi}$ is the dominant eigenvector of G^T.

- The eigendecomposition works because of the initial assumption that the rank of a certain page P can be formulated as:

$$r(P) = \Sigma_{Q \in \mathcal{B}_{P_i}} \frac{r(Q)}{|Q|}$$

  where Q is some inlink page that is an element of a set of all inlink pages that link to P.

Eigendecomposition

- We quickly see that this is an iteration problem, as we do not actually know the ranks of the pages linking to P. For n pages (P_1, ..., P_n) and for j iterations, we define $\boldsymbol{\pi}_j$ to be:

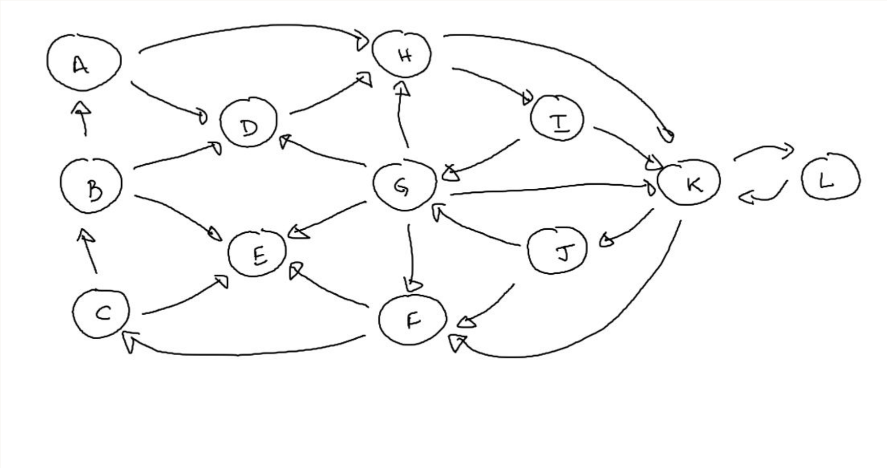
$$\pi_j := [r_j(P_1), \cdots, r_j(P_j)]$$

- We initialize with some matrix G, where each entry of the matrix is the probability that a user on a certain page will go to an outlink to another page.

- By iteratively updating our page ranks by applying the G matrix, we can quickly see that this takes on the form of an eigenvalue problem. Hence, the eigenvector corresponding to the largest eigenvalue of G^T will contain the page ranks of each web page.

Eigendecomposition

Hyperlink Network



Google Matrix

$G

	[,1]	[,2]	[,3]	[,4]	[,5]	[,6]	[,7]	[,8]	[,9]	[,10]	[,11]	[,12]
[1,]	0	0	0	1/2	0	0	0	1/2	0	0	0	0
[2,]	1/3	0	0	1/3	1/3	0	0	0	0	0	0	0
[3,]	0	1/2	0	0	1/2	0	0	0	0	0	0	0
[4,]	0	0	0	0	0	0	0	1	0	0	0	0
[5,]	0	0	0	0	0	0	0	0	0	0	0	0
[6,]	0	0	1/2	0	1/2	0	0	0	0	0	0	0
[7,]	0	0	0	1/4	1/4	1/4	0	1/4	0	0	0	0
[8,]	0	0	0	0	0	0	0	0	1/2	0	1/2	0
[9,]	0	0	0	0	0	0	1/2	0	0	0	1/2	0
[10,]	0	0	0	0	0	1/2	1/2	0	0	0	0	0
[11,]	0	0	0	0	0	1/3	0	0	0	1/3	0	1/3
[12,]	0	0	0	0	0	0	0	0	0	0	1	0

Fig 1. Example of 12 node hyperlink network with corresponding raw Google Matrix G

Eigendecomposition

- From our eigendecomposition of our matrix G, we have the resulting dominant left eigenpair:

```
$eigenvalue
[1] 0.8386825
```

```
$eigenvector
 [1] 0.05523856 0.13898284 0.23312496 0.15433545 0.49351151 0.39103564 0.22196620
 [8] 0.28311821 0.16878748 0.20353086 0.51209331 0.20353086
```

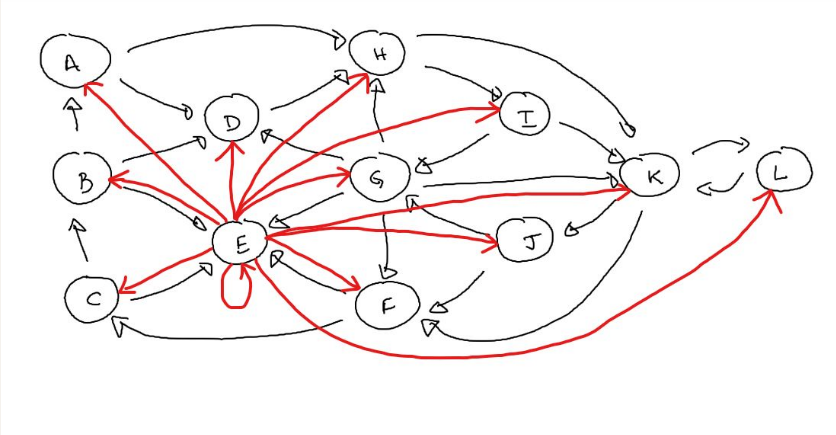
- While all the entries of the corresponding eigenvector are non-negative, the eigenvalue is not 1, which is problematic as that means that the initial assumption that our iteration problem is an eigenvalue problem is violated.

- We observe that we have a *dangling node* present in our hyperlink network, which means that our matrix G is not *row stochastic*.

- To fix this, we add artificial outlinks to our dangling node, which turns G into a row stochastic matrix.

Eigendecomposition

Hyperlink Network



Updated Google Matrix

$G

	[,1]	[,2]	[,3]	[,4]	[,5]	[,6]	[,7]	[,8]	[,9]	[,10]	[,11]	[,12]
[1,]	0	0	0	1/2	0	0	0	1/2	0	0	0	0
[2,]	1/3	0	0	1/3	1/3	0	0	0	0	0	0	0
[3,]	0	1/2	0	0	1/2	0	0	0	0	0	0	0
[4,]	0	0	0	0	0	0	0	1	0	0	0	0
[5,]	1/12	1/12	1/12	1/12	1/12	1/12	1/12	1/12	1/12	1/12	1/12	1/12
[6,]	0	0	1/2	0	1/2	0	0	0	0	0	0	0
[7,]	0	0	0	1/4	1/4	1/4	0	1/4	0	0	0	0
[8,]	0	0	0	0	0	0	0	0	1/2	0	1/2	0
[9,]	0	0	0	0	0	0	1/2	0	0	0	1/2	0
[10,]	0	0	0	0	0	1/2	1/2	0	0	0	0	0
[11,]	0	0	0	0	0	1/3	0	0	0	1/3	0	1/3
[12,]	0	0	0	0	0	0	0	0	0	0	1	0

Fig 2. Hyperlink network with artificial outlinks from dangling node E, with updated Google matrix G

Eigendecomposition

- The resulting dominant eigenpair for our updated G matrix is shown as:

```
$eigenvalue
[1] 1
```

```
$eigenvector
 [1] -0.08618217 -0.14789463 -0.22202134 -0.18955054 -0.44260751 -0.37027477
 [7] -0.24110915 -0.32980287 -0.20178540 -0.20666499 -0.50934308 -0.20666499
```

- Our dominant eigenvalue is now 1, but the entries in our eigenvector are negative, and is not a probabilistic vector. To solve this, we normalize by the 1-norm:

```
$eigenvector
 [1] 0.02732557 0.04689260 0.07039578 0.06010034 0.14033651 0.11740214 0.07644790
 [8] 0.10456981 0.06397961 0.06552677 0.16149619 0.06552677
```

Eigendecomposition

- A sorted list of the pageranks as well as their corresponding web pages is shown as follows. Included is also a visual representation of the pageranks of each web page:

webpage	pagerank
k	0.16149619
e	0.14033651
f	0.11740214
h	0.10456981
g	0.07644790
c	0.07039578
j	0.06552677
l	0.06552677
i	0.06397961
d	0.06010034
b	0.04689260
a	0.02732557

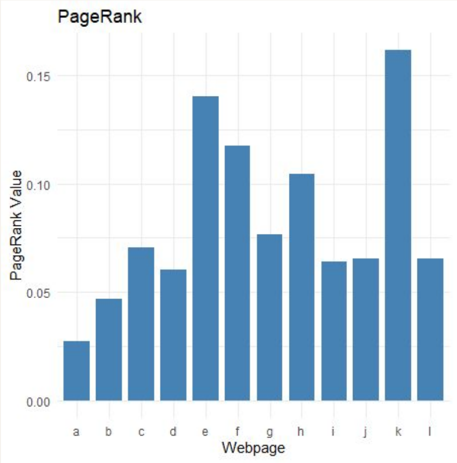


Fig 3. Sorted PageRanks of each web page, along with bar plot of PageRank values

- As we can see, from our eigendecomposition, K is our most important webpage, while A is our least important webpage.

Outline

Power Method

- Since it may be hard to solve the eigenvalue problem for such a large matrix, we can directly compute the power method. This is a way to find the dominant eigenvalue and eigenvector, although with limited precision.
- We can say that

$$\boldsymbol{\pi}_j^{\mathsf{T}} = \boldsymbol{\pi}_{j-1}^{\mathsf{T}} G$$

where $\boldsymbol{\pi}_j$ is the j^{th} iteration of $\boldsymbol{\pi}$, and we repeat the process until the relative error is below a certain threshold

- After ~28 iterations, the relative error stabilizes at around 0.000, which means that directly computing the power method converges to our previous eigendecomposition

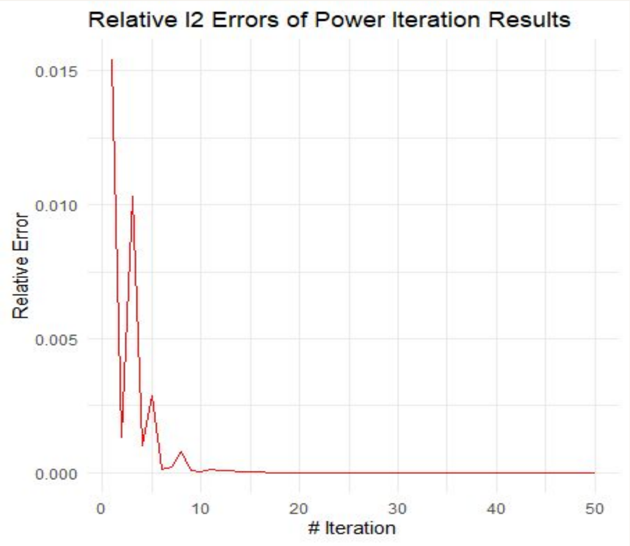


Fig 4. Relative l2 Errors of Power Iteration results

Outline

I. Eigendecomposition

II. Power Method

III. Teleportation

IV. Example Queries

V. Further Implications

Teleportation

- A possible problem that networks can have is not being strongly connected. This would lead to parts of the graph not being accessed.

- We can solve this problem by adjusting G:

$$\tilde{G} = \alpha G + (1 - \alpha)E$$

- Usually, α = 0.85, which means that 15% of the time, the network will reset and be dropped onto a random node, thus ensuring that all parts of the graph can be accessed.

Teleportation

- After applying this teleportation factor, the PageRanks of the webpages have changed.

- Some of the values have gone up while others have gone down.

- The rankings between the two are very similar, with some exceptions. Most notably, for the webpages {I, J, L} our original matrix G ranks them as {J, L, I} while the adjusted matrix G ranks them as {I, J, L}

webpage	pagerank_G	pagerank_G_tilde
k	0.16149619	0.15965440
e	0.14033651	0.13683322
f	0.11740214	0.11527365
h	0.10456981	0.10651347
g	0.07644790	0.07700645
c	0.07039578	0.06958456
j	0.06552677	0.06533434
l	0.06552677	0.06533434
i	0.06397961	0.06537146
d	0.06010034	0.06207266
b	0.04689260	0.04761099
a	0.02732557	0.02941046

Fig 5. Comparison of PageRank between G and G_tilde, sorted by webpage and PageRank

webpage	pagerank_G	pagerank_G_tilde
k	0.16149619	0.15965440
e	0.14033651	0.13683322
f	0.11740214	0.11527365
h	0.10456981	0.10651347
g	0.07644790	0.07700645
c	0.07039578	0.06958456
i	0.06397961	0.06537146
j	0.06552677	0.06533434
l	0.06552677	0.06533434
d	0.06010034	0.06207266
b	0.04689260	0.04761099
a	0.02732557	0.02941046

Outline

Example Queries

- We were provided the following term-document matrix, which is the data representation of certain terms on specific webpages.

- If there were queries for certain terms, what would be the order of webpages shown?

Fig 6. Term-document matrix, with webpages on the top and terms on the side

	A	B	C	D	E	f	G	H	I	J	K	L
Ash	1	0	1	0	0	1	1	1	0	0	0	0
Butternut	1	1	0	1	0	0	1	0	0	0	0	1
Cherry	1	0	0	1	1	0	0	1	0	0	1	0
Elm	1	0	1	0	0	0	0	0	1	0	1	1
Katsura	1	0	1	0	0	0	0	0	1	0	0	1
Magnolia	1	1	0	0	0	1	0	0	1	1	0	0
Teak	1	0	0	1	0	0	0	0	0	0	1	1
Ginkgo	1	0	1	0	0	1	1	0	0	0	1	1
Fir	0	1	0	1	0	1	0	0	1	0	1	0
Hickory	0	1	1	0	1	0	0	1	0	0	1	0
Pine	0	1	0	0	1	0	0	0	1	0	0	1
Willow	0	1	0	0	1	0	0	1	0	1	0	0
Redwood	0	1	1	0	1	1	1	1	0	1	1	0
Sassafras	0	1	0	1	0	1	0	0	1	1	1	1
Oak	0	0	1	0	1	0	1	0	0	1	0	1
Spruce	0	0	0	1	0	1	1	0	1	0	0	1
Aspen	0	0	0	1	0	1	0	1	0	1	0	1

Example Queries

- For the query of “ash,” five webpages contain that term.

- They all have the score of one because there is only one term in the query, and they have it.

$ash

	webpage	score
1	A	1
3	C	1
6	f	1
7	G	1
8	H	1
2	B	0
4	D	0
5	E	0
9	I	0
10	J	0
11	K	0
12	L	0

Fig 7. PageRank of the example query of “ash”

Example Queries

- For the query of “fir” or “hickory,” webpages B and K rank higher because they have both terms, while the next webpages only have one of the terms.

- Webpages B and K were ranked higher because they fit the query better than the other webpages.

$fir_or_hickory

	webpage	score
2	B	2
11	K	2
3	C	1
4	D	1
5	E	1
6	f	1
8	H	1
9	I	1
1	A	0
7	G	0
10	J	0
12	L	0

Fig 8. PageRank of the example query of “ash” or “hickory”

Example Queries

- For the query of “katsura” and “oak,” only webpages C and L contain both terms.

- The other webpages don’t show up in the PageRank because they do not fit the query, as both terms are specifically needed to fit the query.

```
$katsura_and_oak
   webpage score
3        C     2
12       L     2
```

Fig 9. PageRank of the example query of “katsura” and “oak”

Example Queries

- For the query of “aspen” not “sassafras,” webpage H contains the term “aspen” and doesn’t contain the term “sassafras,” therefore fitting the query best.
- The other webpages don’t contain “aspen” as well as “sassafras,” which doesn’t fit the query well but doesn’t violate it completely.
- The webpages that are not listed include “sassafras,” as it directly violates the query.

```
$aspen_not_sassafras
  webpage score
8       H     1
1       A     0
3       C     0
5       E     0
7       G     0
```

Fig 10. PageRank of the example query of “aspen” not “sassafras”

Outline

Further Implications

- We further expand upon our discussion and analysis of the PageRank algorithm by applying a statistical interpretation to the algorithm, which is distantly motivated by the analysis done by David Gleich (2009) on the PageRank algorithm.
- Specifically, we are interested in the analysis of the teleportation method. Recall the formula for the adjusted G:

$$\tilde{G} = \alpha G + (1 - \alpha)E$$

  where $\alpha \in [0, 1]$ and $E := \frac{1}{n} 1_n 1_n^T$

- We are particularly interested in the teleportation parameter α, which emulates the random behavior of a surfer in a hyperlink network.
- The purpose of our extension on this discussion is use the idea motivated by Gleich (2009) in order to briefly analyze the choice of a teleportation parameter.

Further Implications

- The main source of motivation from Gleich (2009) is the idea of a choice for a “best” teleportation parameter α.
- Gleich develops a model that treats α as a parameter for a random variable in order to find the “best” α from a probabilistic point of view.
- However, we decided to take another approach and iterate through all possible choices for α, and calculate the errors against the results of our initial eigendecomposition.
- Intuitively speaking, the α that produces the error closest to 0 should be the one that matches our eigendecomposition.
- However, this is not necessarily our “best” choice of α; remember that our previous choice for α of 0.85 produced a slightly different ranking than from our initial eigendecomposition.

Further Implications

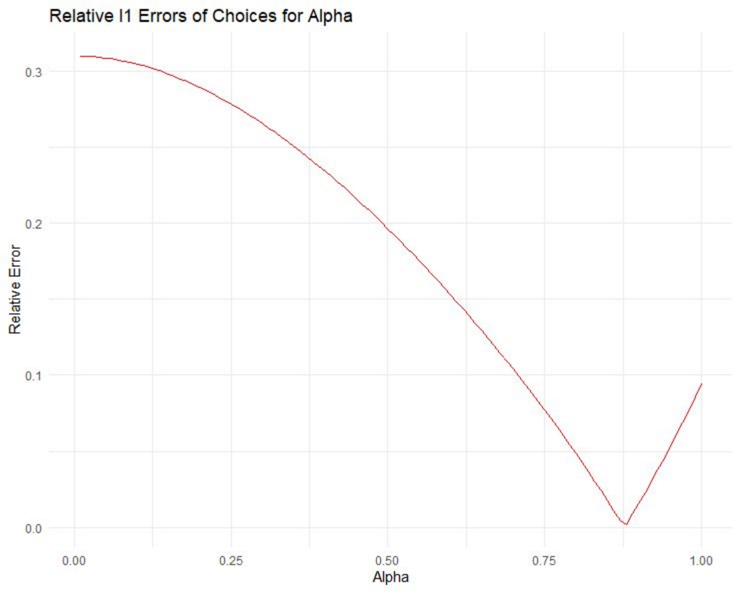


Fig 11. Relative l1 Errors of different choices for the teleportation parameter α

- From *Fig 11.*, we can see that the teleportation parameter that minimizes the error lies around 0.85.
- More specifically, from our direct computations, the α that minimizes the error is ~0.87.
- This further supports the common choice of α for 0.85, as we have shown that a choice of α around that range will be the closest to the original eigendecomposition.

Future Discussion

- Future methods that we can consider would be direct implementation of the Random Alpha Pagerank (RAPr) model developed by Gleich (2009).
- The model directly computes the expectation and standard deviation of a random teleportation parameter α, which can provide a better metric for what the “best” α would be.
- The choice of distribution for the random variable is ultimately up to statistical intuition - although it seems the Beta distribution is the one that the paper decides upon.
- One can even attempt to apply estimation methods to determine the “best” α. Although techniques such as Method of Moments or Maximum Likelihood Estimation may seem simple, applying them to this context could yield interesting and meaningful results.

References and Resources

Downey, Allen. “Data Structures and Information Retrieval in Python” 2021, https://allendowney.github.io/DSIRP/index.html#.

Gleich, David. “Models and Algorithms for PageRank Sensitivity” 2009, https://web.stanford.edu/group/SOL/dissertations/pagerank-sensitivity-thesis-online.pdf.

Sullivan, Danny. “What Is Google Pagerank? A Guide for Searchers & Webmasters.” *Search Engine Land*, 3 Mar. 2022, searchengineland.com/what-is-google-pagerank-a-guide-for-searchers-webmasters-11068.

Group Member Contributions

- Writing the code: Jasper Dong, Alexander Lin
- Writing the presentation: Jasper Dong, Sunny Liu
- Preparation of graphics: Jasper Dong, Alexander Lin
- Assembly of the presentation: Jasper Dong, Sunny Liu