TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN TOÁN ỨNG DỤNG VÀ TIN HỌC

BÁO CÁO
THỰC TẬP KỸ THUẬT

THUẬT TOÁN PAGERANK VÀ ỨNG DỤNG

Chuyên ngành: Toán Tin

Chuyên sâu: Tin học

Sinh viên thực hiện: **Nguyễn Đức Thắng**

Mã số sinh viên: **20166769**

Lớp: **KSTN Toán Tin K61**

Giảng viên phụ trách: **TS. Nguyễn Cảnh Nam**

HÀ NỘI, 07/2020

Mục lục

Lời mở đầu

Trong những năm gần đây, sự lên ngôi của các mạng xã hội và gia tăng về số lượng các trang web trên internet đòi hỏi những thách thức lớn cho việc xếp hạng các trang web, phân tích mạng xã hội. Việc gia tăng đáng kể khối lượng dữ liệu này đòi hỏi con người cần có những cách tiếp cận thông tin tốt hơn. Nếu như không có các công cụ tìm kiếm như Google thì việc tìm kiếm dữ liệu và tiếp cận dữ liệu là vô cùng khó khăn.

Thuật toán Pagerank là một trong những thuật toán nền móng xây dựng lên đế chế của Google. Tuy đơn giản nhưng đã đem lại những hiệu quả trong việc đánh giá điểm cho các trang web để có thể cải thiện tìm kiếm cho người dùng. Pagerank không dừng lại ở mức đơn thuần là xếp hạng các trang web, mà còn có thể ứng dụng trong nhiều lĩnh vực khác như xếp hạng điểm tín dụng, đánh giá các cuộc gọi spam, gợi ý kết bạn trên mạng xã hội, ...
Nội dung báo cáo được chia làm 3 chương:

1. **Đặt vấn đề:** Tổng quan về thuật toán pagerank và ứng dụng.
2. **Cơ sở lý thuyết:** Các kiến thức cơ sở toán học, ý tưởng của thuật toán pagerank.
3. **Xây dựng chương trình và ứng dụng:** Xây dựng chương trình cho thuật toán pagerank trên bộ dữ liệu mẫu, nhận xét và đánh giá kết quả.

Em xin chân thành cảm ơn công ty Grooo International, thầy TS. Lê Chí Ngọc và các anh chị trong công ty đã tận tình giúp đỡ em trong quá trình thực tập. Báo cáo vẫn còn nhiều thiếu sót, em mong nhận được sự góp ý của thầy cô và phía công ty để báo cáo hoàn thiện hơn. Em xin chân thành cảm ơn!

Chương 1

Đặt vấn đề

Vào năm 1989, mạng toàn cầu (World Wide Web) ra đời đã bắt đầu một cuộc cách mạng trong lưu trữ, truy cập và tìm kiếm thông tin. Mạng toàn cầu, gọi tắt là Web là một phát minh của nhà khoa học người Anh Tim Berners-Lee vào năm 1989. Nó là hệ thống thông tin mà trong đó các văn bản và các tài nguyên khác được nhận dạng bằng các đường dẫn (URL - Uniform Resource Locator) và có thể liên kết tới nhau bằng các hyperlink (siêu liên kết) và có thể truy cập qua Internet. Berners-Lee nảy ra ý tưởng về Web sau khi cảm thấy rắc rối mỗi lần ông muốn tìm kiếm thông tin trên nhiều máy tính. Trong ý tưởng ban đầu của mình, ông mô tả một hệ thống quản lý thông tin dựa trên các đường liên kết được gắn vào trong các văn bản, để khi người đọc gặp các tài liệu tham khảo trong một văn bản thì có thể chuyển tức thời đến tài liệu được đề cập chỉ với một cú kích chuột vào các đường dẫn.

Mặc dù mạng toàn cầu là một đột phá trong lưu trữ và truy cập thông tin, người dùng lại gặp khó khăn trong việc tìm kiếm. Khối lượng thông tin là khổng lồ và càng ngày càng mở rộng trong khi lượng thông tin người dùng muốn tìm kiếm và truy cập là rất nhỏ, công việc tìm kiếm được ví như mò kim đáy bể. Hầu hết thông tin trên Web lúc bấy giờ là gần như không thể tiếp cận

Mọi chuyện đều thay đổi khi phân tích liên kết ra đời vào năm 1998. Phân tích liên kết (Link Analysis) là một kỹ thuật phân tích dữ liệu dùng để đánh giá các qua hệ hoặc các liên kết giữa các nút trong một mạng lưới. Những công cụ tìm kiếm bắt đầu sử dụng phân tích liên kết để cải thiện kết quả tìm kiếm. Hiệu quả đem lại là rõ rệt và nhiều công cụ tìm kiếm dần trở thành thú vui ưa thích của nhiều người dùng mạng. Họ sử dụng công cụ tìm kiếm để tìm kiếm tin tức từ khắp nơi trên thế giới, để kiểm tra lỗi chính tả khi sử dụng tiếng nước ngoài, để tìm kiếm các hình ảnh. Google là một trong những công cụ tìm kiếm đầu tiên sử dụng phân tích liên kết và đến năm 2004, Google chiếm thị phần lớn nhất trên thị trường tìm kiếm với 37% số tìm kiếm được thực hiện bởi Google, tiếp sau là 27% thực hiện bởi Yahoo.

Hầu hết các công cụ tìm kiếm ngày nay đều áp dụng phân tích liên kết trong các thuật toán chấm điểm và xếp hạng các trang web. Và thuật toán nổi tiếng nhất cũng đã làm nên thương hiệu Google là thuật toán PageRank.

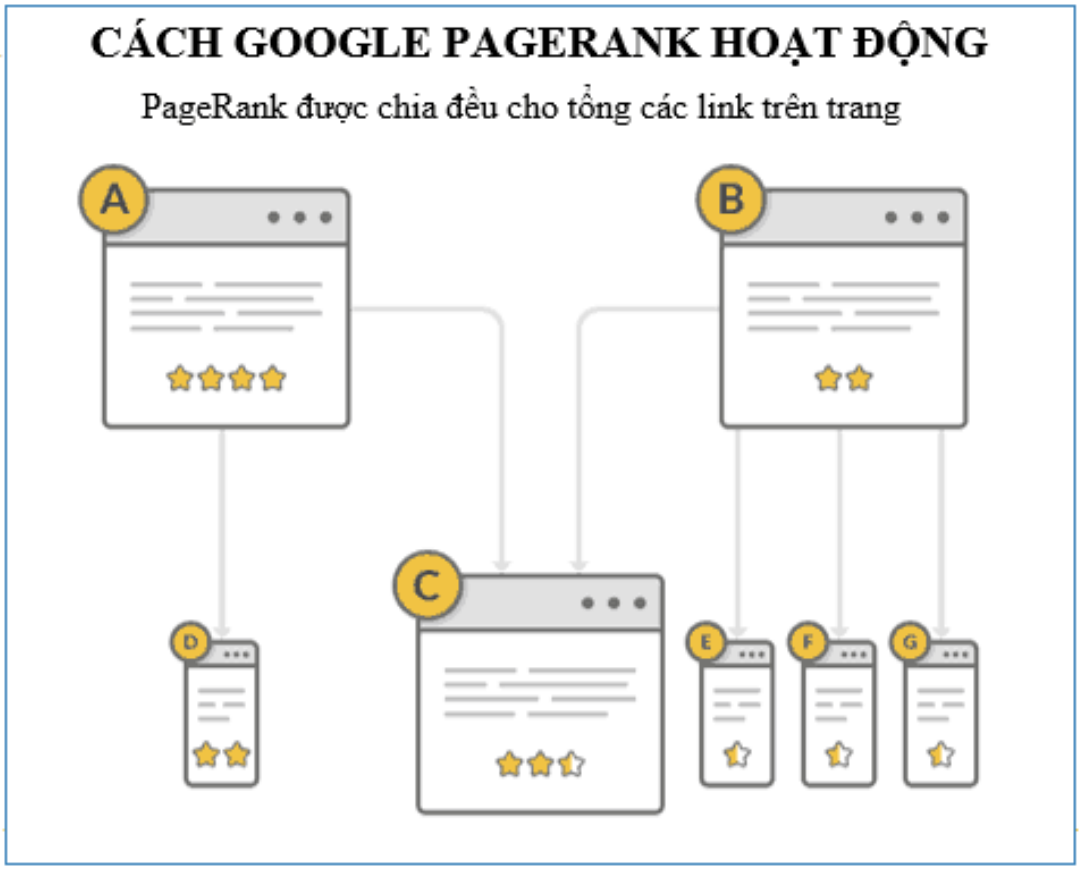
Thuật toán PageRank là một thuật toán học xếp hạng dựa trên phân tích đồ thị liên kết giữa các trang web, mỗi trang web sẽ được xem như một đỉnh, mỗi liên kết sẽ được xem như một cạnh của đồ thị. Mục đích của PageRank là đánh giá tầm quan trọng tương đối của website trong toàn bộ hệ thống world wide web.

Nhà đồng sáng lập Google, Serge Brin và Larry Page đã phát minh ra PageRank vào năm 1997 như một phần của dự án nghiên cứu tại ĐH Stanford với mong muốn cải thiện chất lượng hoạt động của các công cụ tìm kiếm. Brin và Page viết thuật toán PageRank của Google tính ra popularity score cho các trang web. Đây là thuật toán rất nổi tiếng và mặc dù công cụ tìm kiếm của Google còn có đóng góp của nhiều thuật toán khác nhưng PageRank luôn là trái tim của Google.

Bởi vì vào thời điểm đó, công cụ tìm kiếm (như Yahoo, Altavista) gặp phải các vấn đề:

- Hoạt động không hiệu quả, thường đưa ra các website không liên quan.
- Trả về kết quả tìm kiếm không phù hợp với mong đợi người dùng.

Và PageRank được lập nên nhằm giải quyết 2 vấn đề này.



Hình 1.1: Cách thuật toán Pagerank hoạt động

Chương 2

Cơ sở lý thuyết

2.1 Các kiến thức về đại số tuyến tính

2.1.1 Trị riêng, vector riêng

Cho ma trận vuông $A \in \mathbb{K}^{n\times n}$, với $\mathbb{K} = \mathbb{R}; \mathbb{C}$. Số $\lambda \in \mathbb{K}$ được gọi là giá trị riêng của A nếu tồn tại vector $x \neq 0; x \in \mathbb{K}^n$ sao cho $Ax = \lambda x$. Khi đó vector x được gọi là vector riêng của ma trận A ứng với giá trị riêng λ. Ký hiệu $\sigma(A)$ là tập tất cả các trị riêng của ma trận A

Tính chất của giá trị riêng và vector riêng

1. Các giá trị riêng của $A_{n\times n}$ là nghiệm của phương trình đa thức đặc trưng $p(\lambda) = det(A - \lambda I)$. Bậc của $p(\lambda)$ bằng n vậy nên A có n trị riêng, tuy nhiên một số có thể là số phức (ngay cả khi ma trận A là ma trận số thực), hoặc có thể là bội. Nếu ma trận A chỉ chứa số thực thì các trị riêng phức của nó (nếu có) phải theo cặp số phức liên hợp, tức là nếu $\lambda \in \sigma(A)$ thì $\bar{\lambda} \in \sigma(A)$.

2. Nếu như ma trận A có trị riêng $\lambda = 0$ thì ma trận A là không khả nghịch. Ngược lại, nếu mọi giá trị riêng của A đều khác không thì A khả nghịch.

3. Nếu λ là trị riêng của A thì λ^k là trị riêng của A^k

4. Nếu như λ là nghiệm bội k của phương trình đặc trưng thì λ là giá trị riêng bội k của ma trận A. Nếu λ là bội 1 thì λ là giá trị riêng đơn.

5. Nếu như λ là giá trị riêng bội k thì tồn tại k vector riêng độc lập tuyến tính tương ứng với λ.

Chứng minh

Định nghĩa giá trị riêng và vector riêng trội

Cho A là ma trận vuông cấp n. Giả sử A có đủ n trị riêng $\{\lambda_1, \lambda_2, ..., \lambda_n\}$ thực hoặc phức sao cho

$$|\lambda_1| > |\lambda_2| \geq |\lambda_3| \geq ... \geq |\lambda_n|$$

Các vector riêng tương ứng lần lượt là $X_1, X_2, X_3, ..., X_n$. Khi đó λ_1 được gọi là giá trị riêng trội của ma trận A và vector riêng X_1 ứng với λ_1 được gọi là vector riêng trội.

Chú ý: Theo định nghĩa trên thì trị riêng trội λ_1 phải là một số thực, vì nếu $\lambda_1 \in \mathbb{C}$ thì $\bar{\lambda}_1$ cũng là trị riêng có mô-đun bằng λ_1. Điều này trái với giả thiết $|\lambda_1| \geq |\lambda_2|$

2.1.2 Phương pháp luỹ thừa

Phương pháp luỹ thừa (power method) là một thuật toán lặp dùng để tìm ra trị riêng trội và vector riêng trội. Thuật toán bắt đầu bằng việc chọn một vector x_0 ban đầu nào đó, x_0 có thể là ngẫu nhiên hoặC là một xấp xỉ của vector riêng trội X_1. Giả sử các vector riêng của A là một hệ độc lập tuyến tính. Khi đó các vector riêng của A tạo thành một cơ sở trong $\mathbb{R}^n$. Do đó x_0 có thể biểu diễn thành:

$$x_0 = \sum_{i=1}^{n} a_i X_i$$

Sau đó, ta tính dãy:

$$x_1 = Ax_0 = A\sum_{i=1}^{n} a_i X_i = \sum_{i=1}^{n} a_i A X_i = \sum_{i=1}^{n} a_i \lambda_i X_i$$

(Vì $AX_i = \lambda X_i$)

$$x_2 = Ax_1 = A^2 x_0 = A\sum_{i=1}^{n} a_i \lambda_i X_i = \sum_{i=1}^{n} a_i \lambda_i^2 X_i$$

$$\cdots\cdots$$

$$x_k = Ax_{k-1} = A^k x_0 = \sum_{i=1}^{n} a_i \lambda_i^k X_i = \lambda_1^k \left[a_1 X_1 + \sum_{i=2}^{n} \left(\frac{\lambda_i}{\lambda_1}\right)^k a_i X_i \right]$$

Theo giả thiết trị riêng trội $|\lambda_1| > |\lambda_i|, \forall i = 2, ..., n$. Suy ra khi $k \longrightarrow \infty$ thì $\left(\frac{\lambda_i}{\lambda_1}\right)^k \longrightarrow 0, \forall i = 2, ..., n$. Do đó ta có

$$\frac{A^k x_0}{\lambda_1^k} = a_1 X_1, \text{ khi } k \longrightarrow \infty$$

hay khi k đủ lớn thì

$$A^k x_0 \approx \lambda_1^k a_1 X_1$$

đồng thời

$$A^{k+1} x_0 \approx \lambda_1^{k+1} a_1 X_1$$
$$\Rightarrow A^{k+1} x_0 \approx \lambda_1 A^k x_0$$

hay

$$A(A^k x_0) \approx \lambda_1 (A^k x_0)$$

Vậy $A^k x_0 = x^k$ là vector riêng ứng với trị riêng λ_1 và λ_1 được tính theo tỷ số $\lambda_1 \approx \frac{(Ax_k).x_k}{x_k.x_k}$

Vậy, ta thấy các vector x_k có xu hướng hội tụ về một vector riêng nào đó trong số các vector riêng ứng với trị riêng trội. Để đảm bảo x_k hội tụ tới một vector duy nhất, ta thêm bước chuẩn hoá $x_k = \frac{x_k}{||x_k||}$ với $||.||$ là một chuẩn nào đó trong không gian vector. Ngoài để cho quá trình hội tụ đến một vector duy nhất, việc chuẩn hoá còn để tránh bị tràn số.

Tốc độ hội tụ: Ta xét phương pháp luỹ thừa cho hai ví dụ sau

Ví dụ 1.

$A = \begin{bmatrix} 4 & 5 \\ 6 & 5 \end{bmatrix}$ có hai trị riêng là $\lambda_1 = 10$ và $\lambda_2 = -1$. Vì vậy nên tỷ số $\frac{|\lambda_2|}{|\lambda_1|} = 0.1$. Đối với ma trận này chỉ cần 4 vòng lặp đã thu được kết quả chính xác đến 3 chữ số sau dấu thập phân

$$x_0 = \begin{pmatrix} 1.000 \\ 1.000 \end{pmatrix}, x_1 = \begin{pmatrix} 0.818 \\ 1.000 \end{pmatrix}, x_2 = \begin{pmatrix} 0.835 \\ 1.000 \end{pmatrix}, x_3 = \begin{pmatrix} 0.833 \\ 1.000 \end{pmatrix}, x_4 = \begin{pmatrix} 0.833 \\ 1.000 \end{pmatrix}$$

Ví dụ 2.

$B = \begin{bmatrix} -4 & 10 \\ 7 & 5 \end{bmatrix}$ có hai trị riêng là $\lambda_1 = 10$ và $\lambda_2 = -9$. Đối với ma trận này tỷ số $\frac{|\lambda_2|}{|\lambda_1|} = 0.9$ và phương pháp luỹ thừa phải mất đến 68 vòng lặp mới đưa ra được kết quả với độ chính xác đến 3 chữ số sau dấu thập phân

$$x_0 = \begin{pmatrix} 1.000 \\ 1.000 \end{pmatrix}, x_1 = \begin{pmatrix} 0.500 \\ 1.000 \end{pmatrix}, x_2 = \begin{pmatrix} 0.941 \\ 1.000 \end{pmatrix}, \ldots\ldots$$

$$\ldots\ldots, x_{66} = \begin{pmatrix} 0.715 \\ 1.000 \end{pmatrix}, x_{67} = \begin{pmatrix} 0.714 \\ 1.000 \end{pmatrix}, x_{68} = \begin{pmatrix} 0.714 \\ 1.000 \end{pmatrix}$$

Dựa vào hai ví dụ trên cũng như chứng minh sự hội tụ của phương pháp luỹ thừa, ta nhận thấy tốc độ hội tụ của phương pháp phụ thuộc vào tỷ số $\frac{|\lambda_2|}{|\lambda_1|}$. Cụ thể hơn, tốc độ hội tụ của thuật toán tương đương với tốc độ tỷ số $\left[\frac{|\lambda_2|}{|\lambda_1|}\right]^k$ hội tụ về 0 khi $k \longrightarrow \infty$

2.2 Xích markov và phân bố dừng

2.2.1 Khái niệm xích Markov

Kết quả cuối cùng của thuật toán PageRank bản chất là phân phối dừng của một xích Markov. Vì vậy để hiểu rõ PageRank, ta cần nắm được một số kiến thức về xích Markov, bắt đầu bằng các khái niệm.

Định nghĩa 1 (Ma trận xác suất)**.**

Ma trận xác suất là một ma trận vuông không âm $P_{n\times n}$ trong đó tổng các phần tử trên mỗi hàng đều bằng 1.

Định nghĩa 2 (Quá trình ngẫu nhiên)**.**

Quá trình ngẫu nhiên là một tập hợp các biến ngẫu nhiên $\{X_t\}_{t=0}^{\infty}$ có cùng một tập giá trị $\{S_1, S_2, ..., S_n\}$ được gọi là không gian trạng thái của quá trình. Biến t thường được coi là biến thời gian, và X_t là trạng thái của quá trình tại thời điểm t. Ví dụ, ta xét một quá trình là sự thay đổi của thời tiết qua các ngày. Không gian trạng thái là các tình thái của thời tiết như nắng, mưa, lạnh, nhiều mây và biến X_t là trạng thái của thời tiết tại ngày thứ t.

- Theo như định nghĩa trên thì biến thời gian là biến rời rạc, đồng thời không gian trạng thái là hữu hạn.

Định nghĩa 3 (Xích Markov)**.**

Xích Markov là một quá trình ngẫu nhiên mà thoả mãn tính chất Markov

$$P(X_{t+1} = S_j | X_t = S_{i_t}, X_{t-1} = S_{i_{t-1}}, ..., X_0 = S_{i_0}) = P(X_{t+1} = S_j | X_t = S_{i_t})$$

Điều kiện này thể hiện tính chất *không nhớ* của xích Markov, có nghĩa là trạng thái của xích tại thời điểm tiếp theo chỉ phụ thuộc vào trạng thái ở thời điểm hiện tại mà không quan tâm đến các trạng thái trước. Ví dụ như, quá trình biến đổi của thời tiết là một xích Markov nếu như trạng thái thời tiết ngày mai không phụ thuộc vào trạng thái thời tiết của các ngày trước đây, mà chỉ phụ thuộc vào trạng thái của ngày hôm nay.

Định nghĩa 4 (Vector phân phối xác suất)**.**

Một vector phân phối xác suất (gọi tắt là vector xác suất) là một vector không âm $\lambda^T = (\lambda_1, \lambda_2, ..., \lambda_n)$ sao cho $\sum_k \lambda_k = 1$. Nếu ta đặt

$$\lambda_i = P(X_t = i)$$

thì $\lambda = (\lambda_i)$ là một phân phối. Ta nhận thấy tại mỗi thời điểm t, xích Markov có một vector phân phối xác suất. Ta gọi $\lambda = P(X_0 = i)$ là vector phân phối ban đầu.

2.2.2 Ma trận xác suất chuyển

Định lý 1.

Giả sử X_t là một xích Markov với vector phân phối xác suất ban đầu λ, khi đó ta có

- $P(X_n = j) = (\lambda P^n)_j$
- $P_i(X_n = j) = P(X_{n+m} = j | X_m = j) = p_{ij}^{(n)}$

theo định lý này, ta thấy $p^(n)_{ij}$ là xác suất chuyển từ trạng thái i sang trạng thái j sau n bước.

Định nghĩa 5.

Ta nói rằng trạng thái i đến được trạng thái j và ký hiệu là $i \longrightarrow j$ nếu tồn tại $n \geq 0$ sao cho $p_{ij}^{(n)} > 0$. Hai trạng thái i và j được gọi là liên thông và ký hiệu là $i \Longleftrightarrow j$ nếu $i \longrightarrow j$ và $j \longrightarrow i$

Định nghĩa 6.

Một xích Markov được gọi là tối giản nếu hai trạng thái bất kỳ đều liên thông với nhau. Ma trận chuyển P được gọi là tối giản nếu xích Markov tương ứng với nó là tối giản.

2.2.3 Phân bố dừng

Định nghĩa 7 (Vector phân phối xác suất dừng)**.**

Vector phân phối xác suất dừng của một chuỗi Markov có ma trận xác suất chuyển P là một vector xác suất π^T sao cho $\pi^T P = \pi^T$. Nếu như P là ma trận tối giản thì P có phân phối dừng.

Định nghĩa 8 (Vector phân phối xác suất bậc k)**.**

Vector phân phối xác suất bậc k của một xích Markov là

$$\lambda^T(k) = (\lambda_1(k), \lambda_2(k), ..., \lambda_n(k)), \quad \text{trong đó} \quad \lambda_j(k) = P(X_k = S_j)$$

hay nói cách khác, $\lambda_j(k)$ là xác suất xích ở trạng thái thứ j sau k lần chuyển trạng thái.

Quá trình lướt web của người dùng bằng việc chọn các đường dẫn để di chuyển qua các trang web có thể được coi là một quá trình ngẫu nhiên. Không gian trạng thái sẽ là tập hợp toàn bộ các trang web. Thuật toán PageRank coi quá trình này là một xích Markov, Với điều kiện là trang web tiếp theo người dùng chọn không phụ thuộc vào các trang đã đi qua, mà chỉ phụ thuộc vào trang hiện tại. Tức là, người dùng chọn bất kì một đường dẫn tại trang web hiện tại để đến trang tiếp theo, quá trình này gọi là *random walk* trên đồ thị hyperlink.

Cho một phân phối ban đầu $\lambda^T(0) = (\lambda_1(0), \lambda_2(0), ..., \lambda_n(0))$, câu hỏi đặt ra là, xác suất để xích ở một trạng thái là bao nhiêu sau 1 lần chuyển trạng thái. Sử dụng các quy luật xác suất, ta có.

$$\lambda_j(1) = P(X_1 = S_j) = \sum_{i=1}^{n} P\left[X_0 = S_i\right] . P\left[X_1 = S_j | X_0 = S_i\right] = \sum_{i=1}^{n} \lambda_i(0) p_{ij}$$

Tổng hợp lại cho toàn bộ các thành phần của vector phân phối xác suất, ta có $\lambda^T(1) = \lambda^T(0)P$. Phương trình nảy thể hiện sự biến đổi của phân phối xác suất ban đầu sau 1 bước chuyển trạng thái. Nhờ tính chất "không nhớ" của chuỗi Markov mà ở cuối bước chuyển thứ nhất, có thể coi xích bắt đầu lại từ đầu và lấy $\lambda^T(1)$ làm phân phối xác suất ban đầu mới. Vì vậy, ta có $\lambda^T(2) = \lambda^T(1)P$, và $\lambda^T(3) = \lambda^T(2)P$,...và cuối cùng ta có

$$\lambda^T(k) = \lambda^T(0)P^k$$

Tới đây, ta nhận thấy biểu thức trên chính là công thức của phương pháp luỹ thừa với vector ban đầu là $\lambda^T(0)$. Và nếu như $\lambda^T(k)$ hội tụ khi k tiến tới vô cùng thì nó sẽ hội tụ đến vector phân phối dừng của xích Markov X_t.

2.3 Thuật toán PageRank

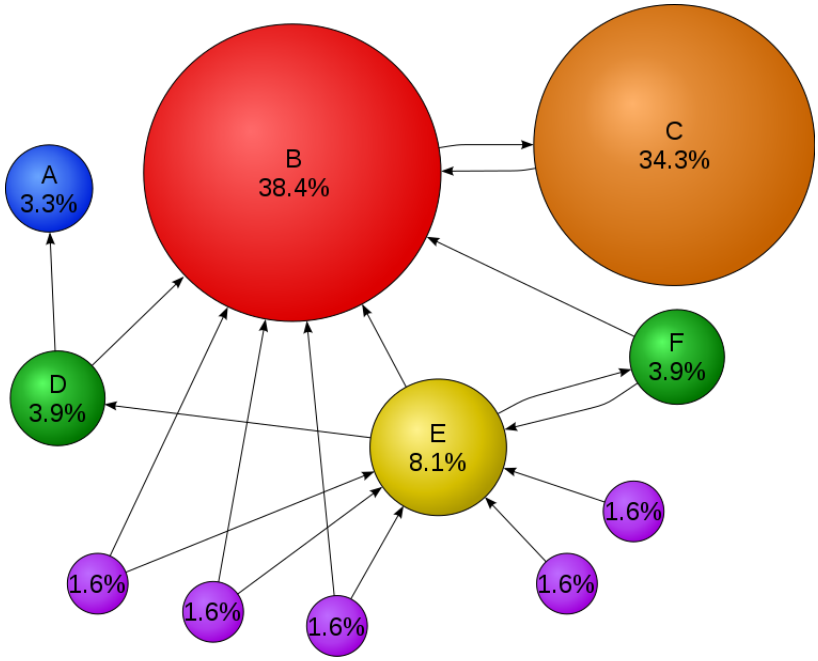
Kể từ khi xuất hiện mạng internet, đã có rất nhiều cách lưu trữ và tổ chức các trang web nhằm mục đích dễ dàng tìm kiếm. Trước đó, **Yahoo** đã lưu trữ các trang web trong các thư mục, người dùng muốn tìm kiếm trang web về thể loại gì thì có thể vào thư mục đó. Để cho kết quả tìm kiếm tốt hơn, khái niệm về **link popularity** ra đời. Theo đó, số trang liên kết tới một trang web sẽ đo độ quan trọng của trang web đó. Do đó, một trang web được coi là quan trọng hơn nếu có nhiều trang web liên kết tới nó. Trái ngược với link popularity, **PageRank** không chỉ dựa trên tổng số trang liên kết với trang web mà còn dựa trên *rank* của các trang liên kết tới nó.

PageRank là một thuật toán trước đây được **Google Search** sử dụng để xếp hạng trang web trên bộ máy tìm kiếm của họ. PageRank được sáng tạo bởi Brin và Page, là một thuật toán **học xếp hạng** dựa trên phân tích **đồ thị** liên kết giữa các trang web, mỗi trang web sẽ được xem như một đỉnh, mỗi liên kết sẽ được xem như một cạnh của đồ thị.

Hình 2.1 mô tả thuật toán pagerank với 11 trang web. Ta có 11 trang web. Trang web B có nhiều trang web liên kết với nó nhất, do đó rank của trang web B là cao nhất. Trang web C mặc dù số trang web liên kết với nó ít hơn trang E nhưng mà nó được trang B liên kết tới, do đó nó sẽ có rank cao hơn trang E. Như vậy kết quả rank cuối cùng của một trang web dựa trên cấu trúc liên kết của toàn bộ các trang web. Cách tiếp cận này nghe mặc dù rất rộng và phức tạp, nhưng Page và Brin đã có thể đưa nó vào thực tế bằng một thuật toán tương đối tầm thường.

2.3.1 Ý tưởng thuật toán

Tổng quát, giả sử chúng ta có n trang web được đánh số từ $1, \ldots, n$, PageRank của trang web i được tính dựa trên các liên kết trang web khác đến nó (trang web j liên kết trỏ đến i), nhưng không phải bất kỳ liên kết nào cũng được tính điểm như nhau. Thuật toán PageRank được xây dựng dựa trên hai ý tưởng như sau:



Hình 2.1: Mô phỏng thuật toán PageRank với 11 trang web

- Trang web A trỏ liên kết đến B, nếu A là một trang web xếp hạng cao thì phải giúp B xếp hạng cao hơn.
- Trang web A trỏ liên kết tới B, lượng trang web mà A trỏ tới nghịch biến với xếp hạng của B hay nói cách khác A trỏ đến càng nhiều trang thì giúp B tăng thứ hạng càng ít.

PageRank của một trang web P_i, ký hiệu là $r(P_i)$:

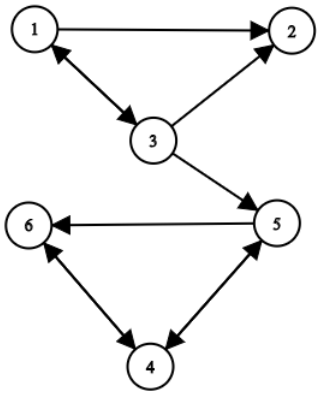
$$r(P_i) = \sum_{P_j \in B_{P_i}} \frac{r(P_j)}{|P_j|} \tag{2.1}$$

trong đó B_{P_i} là tập các trang web trỏ đến P_i, $|P_j|$ là tổng số liên kết đi ra từ P_j. Công thức trên thoả mãn ý tưởng trên. Vấn đề với phương trình (2.1) đó là giá trị $r(P_j)$, PageRank của các trang P_j là chưa biết. Brin và Page giải quyết vấn đề này bằng phương pháp lặp. Họ giả sử rằng ban đầu tất cả các trang có PageRank bằng nhau (cụ thể hơn là $1/n$, trong đó n là số trang trong danh mục Web của Google). Sau đó các giá trị đó được dùng trong phương trình (2.1) để tính $r(P_i)$ mới cho từng trang P_i trong danh mục. Sau khi tính được các $r(P_i)$ mới, chúng lại được thay vào (2.1) ở vị trí của $r(P_j)$ trong vòng lặp tiếp theo. Ký hiệu $r_{k+1}(P_i)$ là PageRank của trang P_i tại vòng lặp thứ $k+1$, khi đó,

$$r_{k+1}(P_i) = \sum_{P_j \in B_{P_i}} \frac{r_k(P_j)}{|P_j|} \tag{2.2}$$

Trong đó $r_0(P_i) = 1/n$ với mọi trang và quá trình này được lặp lại liên tục với mong muốn rằng từ một thời điểm nào đó trở đi, các giá trị PageRank sẽ hội tụ đến các giá trị cố định.

Ví dụ 3. *Xét mô hình internet gồm 6 trang web được đánh số* $1, \ldots, 6$ *như hình 2.2. Tính toán thứ hạng cho các trang web sau một vài vòng lặp.*



Hình 2.2: Ví dụ mô hình liên kết của 6 trang web

Thực hiện việc tính toán theo thuật toán, ta có bảng sau:

Vòng lặp 0	Vòng lặp 1	Vòng lặp 2	Thứ hạng ở vòng lặp 2
$r_0(P_1) = 1/6$	$r_1(P_1) = 1/18$	$r_2(P_1) = 1/36$	5
$r_0(P_2) = 1/6$	$r_1(P_2) = 5/36$	$r_2(P_2) = 1/18$	4
$r_0(P_3) = 1/6$	$r_1(P_3) = 1/12$	$r_2(P_3) = 1/36$	5
$r_0(P_4) = 1/6$	$r_1(P_4) = 1/4$	$r_2(P_4) = 17/72$	1
$r_0(P_5) = 1/6$	$r_1(P_5) = 5/36$	$r_2(P_5) = 11/72$	3
$r_0(P_6) = 1/6$	$r_1(P_6) = 1/6$	$r_2(P_6) = 14/72$	2

Để đơn giản hơn, ta sẽ mô hình hoá ví dụ trên dưới dạng ma trận. Gọi H là ma trận cấp $n \times n$ thoả mãn: $H_{ij} = \frac{1}{|P_i|}$ nếu tồn tại đường dẫn từ trang i sang trang j, và bằng 0 nếu ngược lại. Trong ví dụ trên, ma trận H sẽ là:

$$H = \begin{array}{c} \\ P_1 \\ P_2 \\ P_3 \\ P_4 \\ P_4 \\ P_6 \end{array} \begin{array}{c} \begin{array}{cccccc} P_1 & P_2 & P_3 & P_4 & P_5 & P_6 \end{array} \\ \begin{pmatrix} 0 & 1/2 & 1/2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1/3 & 1/3 & 0 & 0 & 1/3 & 0 \\ 0 & 0 & 0 & 0 & 1/2 & 1/2 \\ 0 & 0 & 0 & 1/2 & 0 & 1/2 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix} \end{array}$$

Dễ thấy các hàng của ma trận có tính chất xác suất. Các phần tử khác 0 ở dòng i tương ứng với các liên kết ra của trang thứ i, trong khi đó, các phần tử khác 0 của cột i tương ứng với liên kết vào của trang thứ i.
Tiếp theo, gọi $\pi^{(k)T}$ là vector PageRank tại vòng lặp thứ k.
Khi đó, phương trình (2.2) có thể viết thành:

$$\pi^{(k+1)T} = \pi^{(k)T} H \tag{2.3}$$

Từ phương trình (2.3), ta đặt ra câu hỏi là liệu có tồn tại trạng thái π^T nào đó mà tại đó, nếu tiếp tục thực hiện (2.3) thì vector PageRank không thay đổi:

$$\pi^T = \pi^T H \tag{2.4}$$

hay

$$\pi = H\pi \tag{2.5}$$

Nếu nhìn kỹ, chúng ta cũng sẽ thấy đây là bài toán trị riêng, vector riêng của ma trận H^T với trị riêng $\lambda = 1$. Như chúng ta đã biết, với một trị riêng λ có thể có nhiều vector riêng và hơn nữa P^T chưa chắc đã có trị riêng $\lambda = 1$. Đây là vấn đề phát sinh với ý tưởng ban đầu của các tác giả, để từ đó đưa ra thuật toán PageRank hoàn thiện hơn.

2.3.2 Thuật toán PageRank

Ma trận H trông giống ma trận xác suất chuyển của xích Markov. Ta có định lý **Perron Frobenius** chỉ ra rằng nếu như ma trận ngẫu nhiên cột (tổng thành phần từng cột bằng 1) mà ta đang xét H là ma trận dương có từng thành phần $H_{i,j} > 0$ thì ma trận H chỉ có duy nhất một phân bố dừng (duy nhất một vector riêng tương ứng trị riêng 1). Tất cả trị riêng còn lại nhỏ hơn 1.
Dựa trên định lý này, Brin và Page đã chỉnh sửa H trở thành một ma trận xác suất của một chuỗi Markov để quá trình lặp có thể hội tụ bằng cách thêm một tham số d được gọi là **Damping Factor**.

Lý thuyết PageRank cho rằng, ngay cả một người dùng giả thiết click ngẫu nhiên vào các trang web cuối cùng cũng sẽ dừng lại. Xác suất người dùng tiếp tục click trong bất cứ bước nào được gọi là yếu tố damping. Có nhiều nghiên cứu đã thử các giá trị yếu tố damping, giá trị ước lượng bằng 0.85 là người dùng sẽ tiếp tục lướt web. Công thức tính Pagerank có tính đến yếu tố damping sử dụng mô hình khi người dùng bất kỳ sẽ cảm thấy chán sau một vài lần click và chuyển đến vài trang web khác một cách ngẫu nhiên. Như vậy:

$$\pi = \left(\frac{1-d}{n}E + dH\right)\pi \tag{2.6}$$

Với E là ma trận vuông $n \times n$ mà tất cả phần tử bằng 1. Ma trận $G = \frac{1-d}{n}E + dH$ được gọi là **ma trận Google**.

Ta cũng có thể biểu diễn công thức dưới dạng sau:

$$r(P_i) = \frac{1-d}{N} + d \sum_{P_j \in B_{P_i}} \frac{r(P_j)}{|P_j|} \tag{2.7}$$

Công thức trên đã làm "mất" các số 0 ban đầu, sử dụng mô hình khi người dùng ngẫu nhiên cảm thấy chán sau khi click và được chuyển đến một số trang ngẫu nhiên. Giá trị Pagerank thể hiện những cơ hội mà người dùng ngẫu nhiên sẽ được chuyển đến trang đó bằng cách click vào các đường link. Mô hình này có thể được hiểu tương tự như Markov chain, trong đó các tỉnh là các trang web, quá trình di chuyển có xác suất ngang nhau được coi như các link giữa các trang web. Nếu như trang web không có đường link đến các trang khác, nó sẽ thành ngõ cụt và việc truy cập ngẫu nhiên sẽ dừng lại. Nhưng nếu người dùng đến trang không có các link khác, thì người dùng sẽ chọn ngẫu nhiên một trang khác để tiếp tục truy cập. **Khi tính Pagerank, những trang không có link trỏ đi các trang khác sẽ được giả định có link trỏ đến tất cả các trang trong tập văn bản**. Và như vậy giá trị Pagerank sẽ được chia đều cho các trang khác.

Quay trở lại với ví dụ 1, nếu với $d = 0.9$, ta sẽ có ma trận:

$$G = \begin{pmatrix} 1/60 & 7/15 & 7/15 & 1/60 & 1/60 & 1/60 \\ 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 \\ 19/60 & 19/60 & 1/60 & 1/60 & 19/60 & 1/60 \\ 1/60 & 1/60 & 1/60 & 1/60 & 7/15 & 7/15 \\ 1/60 & 1/60 & 1/60 & 7/15 & 1/60 & 7/15 \\ 1/60 & 1/60 & 1/60 & 11/12 & 1/60 & 1/60 \end{pmatrix}$$

Và vector PageRank là vector phân phối dừng của ma trận $\mathbf{G}$ và bằng

$$\begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array}$$
$$\pi^T = (0.03721 \quad 0.05396 \quad 0.04151 \quad 0.3751 \quad 0.206 \quad 0.2862)$$

Điều này tức là, với $\pi_1 = 0.03721$ có nghĩa là 3.721% thời gian người lướt web sẽ ở trang 1. Vì vậy, các trang trong mạng Hyperlink trên có thể xếp hạng theo thứ tự (4 6 5 2 3 1), tức là trang thứ 4 là trang quan trọng nhất và trang 1 là trang ít quan trọng nhất.

Chương 3

Xây dựng chương trình và ứng dụng

Thuật toán Pagerank có nhiều ứng dụng khác nhau, không chỉ trong xếp hạng các trang web mà còn có tác dụng trong các bài toán đánh giá như đánh giá mức độ tín dụng, đánh giá cuộc gọi spam, phân tích mạng xã hội,... Trong phần này, em trình bày chương trình thuật toán pagerank cho bài toán sắp xếp thứ hạng các trang web dựa vào các bộ dữ liệu công khai trên internet.

3.1 Chương trình

```
import networkx as nx
import pandas as pd

if __name__ == '__main__':
    data = pd.read_csv('./data/web-Stanford.txt', sep="\t", header=None,
        skiprows=4)
    G = nx.DiGraph()
    for i in range(len(data)):
        G.add_node(data.iloc[i][0])
        G.add_node(data.iloc[i][1])
        G.add_edge(data.iloc[i][0], data.iloc[i][1])
    pr = nx.pagerank(G, alpha=0.85)
    print(pr)
```

Kết quả đầu ra biến **pr** sẽ chứa rank của các node trong đồ thị, chương trình trên đang xét là đồ thị có hướng. Biến **pr** có thể được tuỳ biến sử dụng sao cho thích hợp tuỳ vào từng bài toán.

Ngoài ra, thay vì sử dụng thư viện, chúng ta cũng có thể tự thiết lập thuật toán như sau:

```
import networkx as nx
import pandas as pd
import matplotlib.pyplot as plt

if __name__ == '__main__':
    data = pd.read_csv('./data/web-Stanford.txt', sep="\t", header=None,
        skiprows=4)
    G = nx.DiGraph()
    for i in range(len(data)):
        G.add_node(data.iloc[i][0])
        G.add_node(data.iloc[i][1])
        G.add_edge(data.iloc[i][0], data.iloc[i][1])
    # pr = nx.pagerank(G, alpha=0.85)
    N = len(G)
    d = 0.9
    rank = {}
    for node in G.nodes:
        rank[node] = 1/N

    for _ in range(10000):
        for node in G.nodes:
            if len(G.out_edges(node)) == 0:
                for j in G.nodes:
                    G.add_edge(node, j)
            rank_sum = 0
            edges = G.in_edges(node)
            for j , _ in edges:
                outlinks = len(G.out_edges(j))
                if outlinks > 0:
                    rank_sum += rank[j]/len(G.out_edges(j))

            rank[node] = (1-d)/N + d*rank_sum
```

Với chương trình trên, sau khi chạy thì biến **rank** sẽ lưu trữ thứ hạng của các trang web dưới dạng dictionary.

3.2 Thực nghiệm

Trong phần này, chúng ta thực nghiệm với 4 bộ dữ liệu như sau:

Tên	Số lượng đỉnh	Số lượng cạnh
Example	6	10
Harvard500	500	2636
web-Stanford	281.903	2.312.497
web-Google	875.713	5.105.039

Bảng 3.1: Thông tin các bộ dữ liệu thực nghiệm

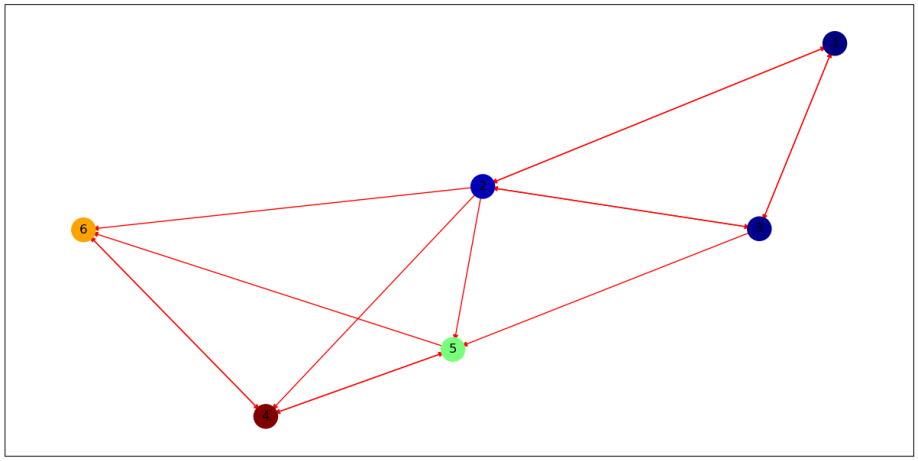
Nguồn các bộ dữ liệu:

1. Example: Bộ dữ liệu được thiết lập ở ví dụ 3 phần lý thuyết.
2. Harvard500: `https://www.cise.ufl.edu/research/sparse/matrices/MathWorks/Harvard500.html`
3. Stanford: `https://snap.stanford.edu/data/web-Stanford.html`
4. Google: `https://snap.stanford.edu/data/web-Google.html`

Thời gian chạy thuật toán tăng dần theo thứ tự các bộ dữ liệu trên. Với bộ dữ liệu *example* chúng ta thu được vector pagerank như sau:

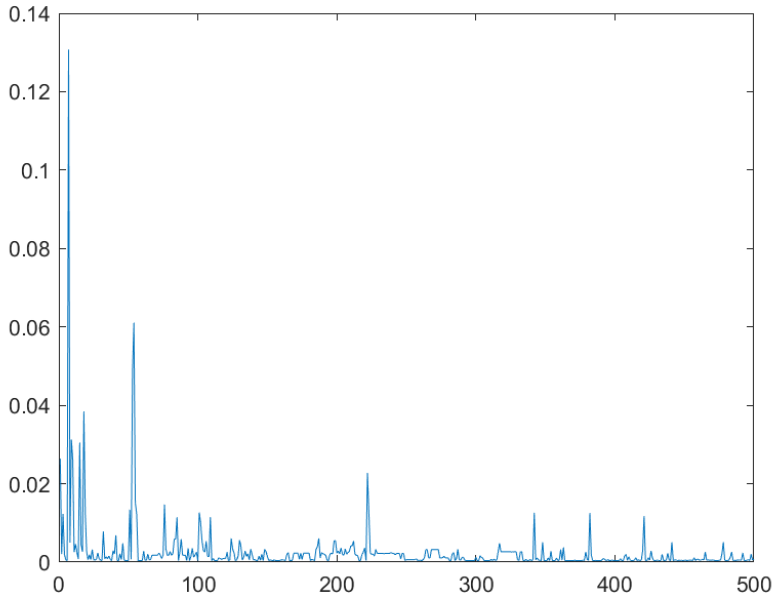
```
{1: 0.03721312715805474, 2: 0.05395936859791858, 3: 0.04150700697389663, 5:
    0.20599776987949375, 4: 0.3750784815111541, 6: 0.28624424587948205}
```

Kết quả này tương thích với kết quả chúng ta đã tìm được ở phần lý thuyết. Mô phỏng đồ thị dữ liệu *example* như hình 3.1. Sở dĩ đồ thị như vậy vì đỉnh số 2 không dẫn đến bất kỳ đỉnh nào khác, nên theo thuật toán, ta cho đỉnh đó dẫn đến tất cả các đỉnh còn lại để tránh hiện tượng rank sink.

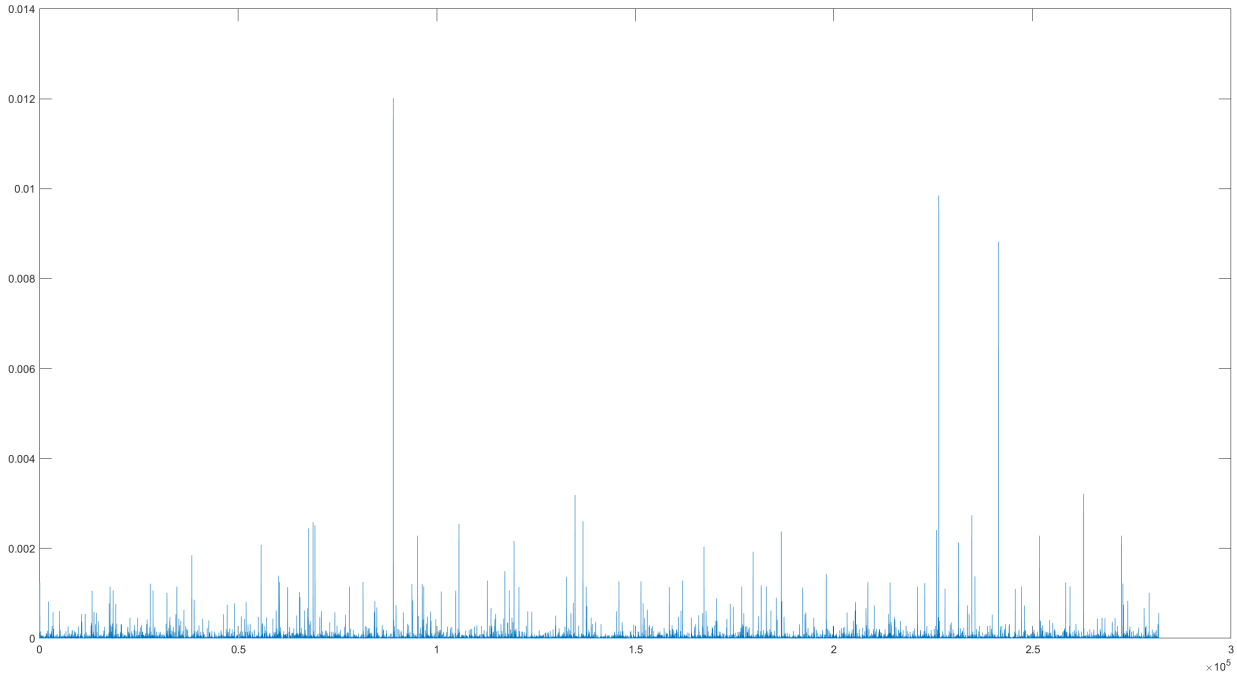


Hình 3.1: Mô phỏng đồ thị của dữ liệu example

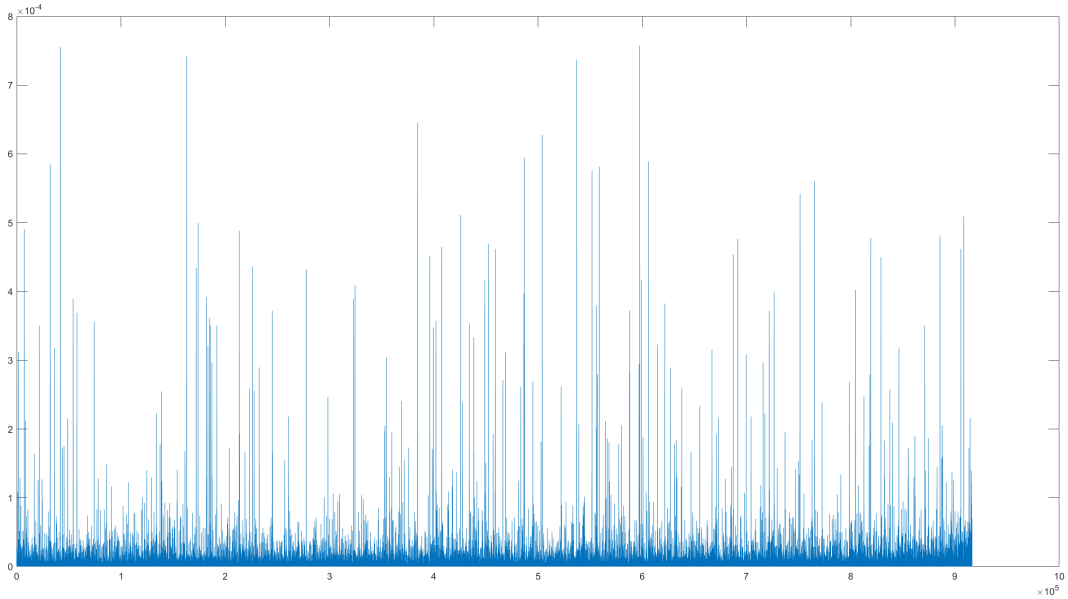
Với các bộ dữ liệu còn lại, sau khi chạy thuật toán và thống kê pagerank ta thu được phân phối pagerank của các bộ dữ liệu như hình 3.2, 3.3 và 3.4.



Hình 3.2: PageRank của bộ dữ liệu Harvard500



Hình 3.3: PageRank của bộ dữ liệu web-Stanford



Hình 3.4: PageRank của bộ dữ liệu web-Google

Nhận xét: Ở ba bộ dữ liệu đầu tiên, vector PageRank chỉ có vài phần tử lớn trội hơn hẳn các phần tử còn lại. Điều này là do đây là các mạng của chỉ 1 tên miền đó là tên miền của trường đại học Harvard và Stanford. Trong các mạng này các trang web thông thường có đường dẫn đến trang chủ và một số ít các trang web quan trọng khác. Còn ở bộ dữ liệu của Google, các trang web thuộc về nhiều tên miền khác nhau, vì vậy nên thứ hạng các trang web cũng đồng đều hơn.

Tài liệu tham khảo

[1] Amy N. Langville and Carl D. Meyer. Google's PageRank and Beyond: The Science of Search Engine Rankings, Princeton University Press, 2006.

[2] Nancy Blachman, Eric Fredricksen, and Fritz Schneider. How to Do Everything with Google. McGraw-Hill, 2003.

[3] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual Web search engine. Computer Networks and ISDN Systems, 33:107–17, 1998.

[4] Sergey Brin, Lawrence Page, R. Motwami, and Terry Winograd. The PageRank citation ranking: Bringing order to the Web. Technical Report 1999-0120, Computer Science Department, Stanford University, 1999.

[5] Carl D. Meyer. Matrix Analysis and Applied Linear Algebra. SIAM, Philadelphia, 2000.

[6] Tống Đình Quỳ, Xác suất thống kê, NXB Giáo dục, 2000.

[7] Yousef Saad. Iterative Methods for Sparse Linear Systems. SIAM, 2003.

[8] Beel, Jöran and Gipp, Bela and Wilde, Erik. Academic Search Engine Optimization (ASEO): Optimizing Scholarly Literature for Google Scholar and Co. Journal of Scholarly Publishing, 2010.

[9] Ortiz-Cordova, A. and Jansen, B. J. Classifying Web Search Queries in Order to Identify High Revenue Generating Customers. Journal of the American Society for Information Sciences and Technology, 2012.

[10] Hansell, Saul. Google Keeps Tweaking Its Search Engine. New York Times, 2007.

[11] Shari Thurow. The Most Important SEO Strategy. clickz.com. 2006.