Group 1 - 22TNT1

PAGERANK
Algorithm

Unlocking secrets: Google's millions-dollar algorithm

2024

group1@gmail.com
Ho Chi Minh city's University of Science

XUNG QUANH THUẬT TOÁN PAGERANK

NHÓM 1, LỚP 22TNT1, ĐẠI HỌC KHOA HỌC TỰ NHIÊN

NHÓM 1 LỚP 22TNT1

Lê Đại Hòa - 22120108

Bùi Trọng Trịnh - 22120390

Lê Hoàng Vũ - 22120461

Nguyễn Tường Bách Hỷ - 22120455

Mục lục

Lời Mở Đầu

Thưa quý độc giả,

Trong thời đại số hóa ngày nay, Internet đã trở thành một nguồn thông tin vô tận, nhưng việc tìm kiếm và sắp xếp thông tin này trở nên ngày càng khó khăn. Trong bối cảnh đó, thuật toán Pagerank đã nổi lên như một công cụ mạnh mẽ để xác định sự quan trọng của các trang web dựa trên cấu trúc liên kết của chúng.

Chúng tôi xin kính gửi đến bạn đọc bài báo cáo với nội dung "Xung Quanh Thuật Toán Pagerank"với hy vọng chia sẻ kiến thức, hiểu biết và sự quan tâm đối với thuật toán này. Chúng tôi sẽ đi sâu vào những khía cạnh khác nhau của Pagerank, từ cơ bản đến nâng cao, và đề cập đến ứng dụng của nó trong thế giới thực.

Tuy nhiên để hoàn thành bài báo cáo này, không thể không nhắc đến sự hỗ trợ giảng dạy cả trong lẫn ngoài giờ lên lớp của hai người thầy - hai giáo viên hướng dẫn và định hướng cho nhóm tác giả. Nhóm chúng em xin gửi lời cảm ơn chân thành và sâu sắc nhất tới Thạc Sĩ Lê Phúc Lữ và Thạc Sĩ Trần Hà Sơn ạ!

ThS. Lê Phúc Lữ

ThS. Trần Hà Sơn

Nhóm tác giả mong rằng chuyên đề này sẽ giúp bạn hiểu rõ hơn về Pagerank và ứng dụng của nó, đồng thời khơi dậy sự quan tâm và sự tò mò trong lĩnh vực này. Chúng tôi cũng rất mong nhận được phản hồi và góp ý từ phía độc giả để chúng tôi có thể cải thiện và hoàn thiện chuyên đề này trong tương lai.

Xin chân thành cảm ơn!

CHƯƠNG

I

Giới thiệu bài toán PageRank

Bảng 1. Nội dung của Chương I

Section 1

Hoàn cảnh

Trong những năm 90 của thế kỷ 20, công cụ tìm kiếm đầu tiên sử dụng hệ thống xếp hạng các trang web dựa trên văn bản, từ đó quyết định trang web nào quan trọng. Có nhiều nhược điểm với phương pháp này. Hãy lấy ví dụ, nếu một người tìm kiếm với từ khóa "Internet", có thể gặp khó khăn. Người duyệt web có thể nhận được một trang với từ khóa internet nhưng không có thông tin gì về internet trên trang hiển thị. Với phương pháp này, công cụ tìm kiếm sẽ sử dụng số lần xuất hiện của từ trong câu hỏi đã cho để tìm kiếm. Điều này không hợp lý khi trang được tìm kiếm nhiều nhất được hiển thị đầu tiên thì có thể trang web đó lại không có nội dung gì. Ngoài ra, có thể có hàng triệu trang web có tiêu đề phù hợp và khi công cụ tìm kiếm đưa ra tất cả những trang đó, điều này sẽ gây cản trở đối với người duyệt web.

Ngoài ra, người duyệt web không có kiên nhẫn để kiểm tra tất cả các trang web được đưa ra. Thông thường, người dùng mong đợi trang liên quan được hiển thị trong số 20-30 trang đầu tiên được cung cấp bởi công cụ tìm kiếm. Bởi vậy, cần có một công cụ tìm kiếm hiện đại sử dụng phương pháp cung cấp kết quả tốt nhất đầu tiên, phù hợp hơn nhiều so với phương pháp xếp hạng văn bản cũ hơn. Một trong những thuật toán mạnh mẽ nhất là thuật toán PageRank được sử dụng bởi công cụ tìm kiếm Google.

Ý tưởng chính đằng sau thuật toán xếp hạng trang là tầm quan trọng của một trang web được dự đoán bởi các trang liên kết đến nó. Nếu chúng ta tạo một trang web i có một liên kết đến trang j thì trang j được coi là quan trọng. Ngược lại, nếu trang j có một liên kết ngược từ trang k (như `www.google.com`) chúng ta có thể nói rằng k chuyển quyền kiểm soát sang j (tức là, k khẳng định rằng j quan trọng). Chúng ta có thể gán một xếp hạng cho mỗi trang dựa trên số trang chỉ đến nó theo cách lặp lại.

Section 2

Giới thiệu

PageRank là một chủ đề đã được thảo luận rộng rãi bởi các chuyên gia về Tối ưu hóa Máy tìm kiếm (Search Engine Optimization – hay viết tắt là SEO). Cốt lõi của PageRank là một công thức toán học có vẻ rất phức tạp, nhưng nếu được đơn giản hóa, nó lại dễ hiểu. Đối với các nhà phát triển web, việc hiểu về khái niệm PageRank là cần thiết để nâng cao xếp hạng của trang web.

Hiện nay, rất nhiều các trang website mọc lên như nấm trên thị trường. Nếu một người ngồi tính toán PageRank cho các trang web khác nhau, điều này có thể là một công việc tẻ nhạt. Vì tự động hóa đang làm cho cuộc sống của con người dễ dàng hơn, việc tự động hóa PageRank sẽ mang lại hiệu quả tốt hơn cho World Wide Web (WWW). PageRank là một thuật toán mà Google sử dụng để điều chỉnh thứ hạng của các trang web trong kết quả tìm kiếm của họ. Nó được sáng tạo bởi Larry Page (chủ yếu) và Sergey Brin khi họ còn là sinh viên thạc sĩ tại Đại học Stanford và sau đó trở thành thương hiệu của Google vào năm 1998.

PageRank không chỉ ảnh hưởng đến ngành công nghiệp lập trình, mà còn có tác động đến lĩnh vực kinh tế. Nó cũng được xem là mục tiêu kinh doanh của mọi công ty: xếp hạng cao hơn trong việc hiển thị trang web, điều này được coi là chiến lược SEO. Cấu trúc liên kết cùng với chiến lược SEO đóng vai trò quan trọng trong xếp hạng Page (Page Ranking).

Đằng sau mọi thuật toán đều là cơ sở toán học của nó được xây dựng tỉ mỉ và kỳ công, trong đó có thể kể đến những cơ sở về ma trận và vector. Vì số lượng trang web đang tăng lên từng ngày, các trang web phải được xếp hạng bằng cách sử dụng các thuật toán nhất định. Đặc điểm của thuật toán PageRank của Google là nó không cho phép xảy ra tình trạng spam, ý chỉ những trang web được mã hóa theo cách có thể thao túng kết quả xếp hạng và vi phạm các quy định được thiết lập bởi Google. Nó cũng tập trung vào tầm quan trọng của trang khi được trích dẫn bởi các nút quan trọng khác. Việc nắm rõ cách thức hoạt động của thuật toán PageRank sẽ giúp chúng ta nhận ra tầm quan trọng của các lý thuyết về Đại số tuyến tính và đồ thị trong việc xây dựng mô hình của những công ty lớn.

Kết luận 1 Nói tóm lại, PageRank là một công thức toán học đánh giá giá trị của trang thông qua việc xem xét số lượng và chất lượng của những trang liên kết đến nó. Mục đích chính của PageRank là đánh giá tầm quan trọng tương đối của website trong toàn bộ hệ thống WWW.

Section 3

PageRank hoạt động theo nguyên tắc gì?

Hôm nay có hàng triệu trang web và mỗi trang chứa hàng tấn thông tin. Khi một người duyệt web truy cập một trang web, điểm đó hoạt động như điểm bắt đầu cho việc liệt kê. Bằng cách sử dụng việc chọn ngẫu nhiên các liên kết, một trang web có thể được truy cập nhiều lần bởi người duyệt web. Để hiểu PageRank hoạt động như thế nào, tính toán ra sao, ta cần chú ý đến những đại lượng sau:

1. Số lượng và chất lượng liên kết vào (hay Internal Links, ký hiệu T) trên các trang web
2. Số lượng liên kết ra (Outbound Links, ký hiệu C) trên mỗi trang web

Định nghĩa 1 Từ những đại lượng như trên, sau đây là cách mà Google, hay chính xác hơn là Larry Page và Sergey Brin, đã giới thiệu về cách thức hoạt động của PageRank:

$$PR(A) = (1-d) + d \cdot \left(\frac{PR(T_1)}{C(T_1)} + \frac{PR(T_2)}{C(T_2)} + \ldots + \frac{PR(T_n)}{C(T_n)} \right) \tag{3.1}$$

Trong đó, đại lượng T và C như trên, $PR(A)$ là chỉ số PageRank của trang web A, tương tự cho $PR(T_i)$ và d là yếu tố 'damping'.

Ở công thức vừa rồi ta có đề cập đến yếu tố 'damping' (damping factor)[1] hay còn gọi là chỉ số xác suất. Đây là một tham số quan trọng giúp mô phỏng hành vi của người dùng khi họ lướt web. Yếu tố này biểu diễn xác suất mà người dùng tiếp tục lướt web sau mỗi lần nhấp chuột vào một liên kết. Vì vậy $d \in [0, 1]$. Giá trị của yếu tố damping thường được đặt là 0.85, điều này có nghĩa là người dùng sẽ tiếp tục lướt web với xác suất 85% sau mỗi lần nhấp chuột, và 15% xác suất họ sẽ rời khỏi trang hiện tại để lướt web ngẫu nhiên.

[1] *Những vấn đề khác về Damping factor như tầm ảnh hưởng đến thuật toán PageRank hay việc lựa chọn các giá trị sẽ được đề cập ở phần* (12).

Trong thuật toán PageRank gốc, PageRank của một trang T luôn được tính toán dựa trên số lượng liên kết ra $C(T)$ trên trang T. Nói cách khác, trang T càng có nhiều liên kết ra thì cũng đồng nghĩa với việc trang A được hưởng ít 'tài nguyên' hơn từ T. Từ công thức, ta cũng có thể thấy liên kết trỏ đến trang A góp phần tăng thêm chỉ số PageRank của trang A. Cuối cùng, tổng của các PageRank đã được tính toán của tất cả các trang T_i được nhân với một yếu tố 'damping' d.

Dụa trên nền tảng là công thức PageRank gốc (3.1), ta xây dựng được một phiên bản thứ hai như sau:

Định nghĩa 2 (Phiên bản thứ hai của công thức tính PageRank)

$$\begin{aligned} PR(A) &= \frac{1-d}{N} + d \cdot \sum_{i=1}^{n} \frac{PR(T_i)}{C(T_i)} \\ &= \frac{1-d}{N} + d \cdot \left(\frac{PR(T_1)}{C(T_1)} + \frac{PR(T_2)}{C(T_2)} + \ldots + \frac{PR(T_n)}{C(T_n)} \right) \end{aligned}$$

Trong đó các đại lượng được định nghĩa như ở (3.1), chỉ thêm N là số lượng trang web.

Nhận xét 1

Khi bạn đọc bài báo của Larry Page và Sergey Brin về PageRank, bạn thấy họ chỉ đề cập đến $(1-d)$ thay vì $\frac{1-d}{N}$. Điều này xuất phát từ cách hiểu cơ bản về thuật toán PageRank và cách nó hoạt động. Trong thuật toán PageRank ban đầu, $(1-d)$ đại diện cho phần trăm xác suất mà người dùng sẽ rời khỏi trang hiện tại và bắt đầu tìm kiếm trên toàn bộ web, trong khi d là phần còn lại của xác suất mà người dùng sẽ tiếp tục di chuyển qua các liên kết trên trang hiện tại.

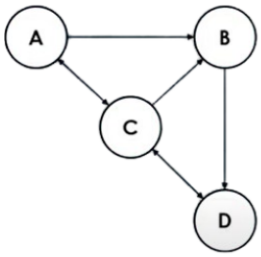
Cụ thể, $(1-d)$ không cần chia cho tổng số trang web là N vì nó chỉ phản ánh tỷ lệ phần trăm chung mà người dùng sẽ rời khỏi trang hiện tại, không phụ thuộc vào số lượng trang web tồn tại. Phần d của công thức sau đó được chia cho số lượng liên kết đến mỗi trang để tính toán PageRank tương đối của từng trang dựa trên số lượng và PageRank của các trang liên quan đến nó.

Điều này giải thích tại sao trong nguyên lý cơ bản của PageRank, $(1-d)$ được sử dụng mà không cần chia cho N, vì mục tiêu chính là xác định tỷ lệ phần trăm mà người dùng sẽ rời khỏi trang hiện tại, không phải phân phối PageRank giữa tất cả các trang web.

Ví dụ

Ta sẽ đến với một ví dụ đơn giản để có một cái nhìn tổng quan về cách tính chỉ số PageRank.

Giả sử có 4 trang web A,B,C,D. Mỗi đường link từ trang web này đến trang web khác được biểu diễn bằng một mũi tên có hướng. Hãy tính chỉ số PageRank của từng trang web. (Giả sử chỉ số PageRank ban đầu tổng cộng bằng 1 và ban đầu các trang web có chỉ số PageRank bằng nhau)



Hình 1. Mô hình ở Ví dụ

Ban đầu vì tổng các chỉ số PageRank bằng nhau và tổng bằng 1 nên $PR(A) = PR(B) = PR(C) = PR(D) = \frac{1}{4}$.

Ở lượt thứ 2 ta tính toán như sau:

- Vì trang A được trang C trỏ liên kết tới, trang C lại có tổng cộng 3 liên kết tới các trang nên ta được

$$PR(A) = \frac{PR(C)}{3} = \frac{1}{4} : 3 = \frac{1}{12}$$

- Vì trang B được trang A và C trỏ liên kết tới, trang A và C lần lượt có tổng cộng 2 và 3 liên kết ra nên ta được

$$PR(B) = \frac{PR(A)}{2} + \frac{PR(C)}{3} = \frac{1}{4} : 2 + \frac{1}{4} : 3 = \frac{5}{24}$$

- Vì trang C được trang A và D trỏ liên kết tới, trang A và D lần lượt có tổng cộng 2 và 1 liên kết ra nên ta được

$$PR(C) = \frac{PR(A)}{2} + \frac{PR(D)}{1} = \frac{1}{4} : 2 + \frac{1}{4} : 1 = \frac{3}{8}$$

- Vì trang D được trang B và C trỏ liên kết tới, trang B và C lần lượt có tổng cộng

1 và 3 liên kết ra nên ta được

$$PR(D) = \frac{PR(B)}{1} + \frac{PR(C)}{3} = \frac{1}{4} : 1 + \frac{1}{4} : 3 = \frac{1}{3}$$

Ở những lượt tính tiếp theo, ta áp dụng công thức tương tự như vậy và kết quả có thể được mô tả như ở bảng dưới, sau 4 lượt tính toán:

	Lượt tính			
Trang web	1	2	3	4
A	1/4	1/12	1/8	1/24
B	1/4	5/24	1/6	3/16
C	1/4	3/8	3/8	19/48
D	1/4	1/3	1/3	7/24

Bảng 2. Chỉ số PageRank của các trang web A, B, C, D sau 4 lần tính

Từ công thức phiên bản thứ 2 của PageRank ở (2), ta có thể mô tả giải thuật và viết được mã giả (`pseudo code`) như sau:

Mô tả Thuật toán PageRank

1. **Đầu vào:** Thuật toán nhận vào các tham số sau: N là số lượng trang web trong hệ thống, d là yếu tố 'damping', T_i là tập hợp các trang có liên kết đến trang A, $PR(T_i)$ là giá trị PageRank của mỗi trang T_i, $C(T_i)$ là số lượng liên kết đi vào từ mỗi trang T_i.

2. **Tính toán PageRank:** Sử dụng biểu thức sau để tính PageRank cho trang A:

$$PR(A) = \frac{1-d}{N} + d \cdot \left(\frac{PR(T_1)}{C(T_1)} + \frac{PR(T_2)}{C(T_2)} + \ldots + \frac{PR(T_n)}{C(T_n)} \right)$$

3. **Đầu ra:** Kết quả cuối cùng là giá trị PageRank của trang A.

Algorithm 1 PageRank Algorithm

procedure PAGERANK(G, d, ϵ) ▷ PageRank evaluation algorithm
  Input: G: adjacency matrix representing the graph
    d: damping factor
    ϵ: convergence threshold
  Output: PR: vector of PageRank scores
  $N \leftarrow$ number of nodes in the graph
  Initialize PR as a vector of length N with equal values
  repeat
    $PR_{new} \leftarrow (1-d)/N + d \times G \times PR$
    $\Delta \leftarrow ||PR_{new} - PR||_2$
    $PR \leftarrow PR_{new}$
  until $\Delta < \epsilon$
  return PR

CHƯƠNG

II

Cơ sở lý thuyết

Section 4

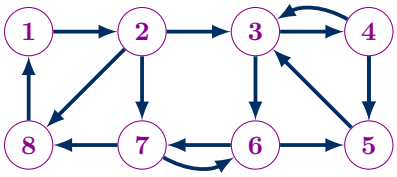
Lý thuyết đồ thị và tập hợp

Quy trình hiện tại về xếp hạng trang web trên Internet dựa trên ý tưởng tính toán phân phối xác suất của một bước đi ngẫu nhiên trên đồ thị mạng lưới Internet, nơi các nút là trang web và cạnh là liên kết giữa các trang web. Để kết quả của quá trình đó được định rõ, chúng tôi giới hạn sự chú ý của mình vào các đồ thị **liên thông mạnh**. Ta sẽ đến với những định lý, tính chất mang tính cơ sở, nền tảng sau:

Bảng 3. Nội dung của Chương II

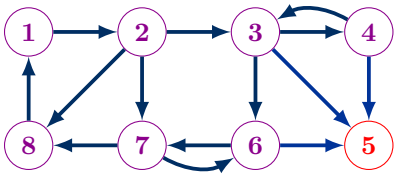
Định nghĩa 3 Một đồ thị có hướng $G = (V, E)$ được gọi là liên thông mạnh nếu với mọi cặp đỉnh phân biệt v_1, $v_2 \in V$, tồn tại đường đi từ v_1 đến v_2 trong E.

Định nghĩa này liên quan mật thiết đến khái niệm **thành phần liên thông mạnh** của một đồ thị có hướng. Strongly Connected Components (SCCs), hay thành phần liên thông mạnh là một khái niệm cơ bản trong lý thuyết đồ thị và thuật toán.



Hình 2. Đồ thị này là một đồ thị liên thông mạnh

Trong một đồ thị hướng, một thành phần liên thông mạnh là một tập con của các đỉnh sao cho mỗi đỉnh trong tập con đều có thể đi đến từ mọi đỉnh khác trong cùng tập con thông qua các cạnh hướng. Việc tìm kiếm SCCs của một đồ thị có thể cung cấp những hiểu biết quan trọng về cấu trúc và kết nối của đồ thị, với ứng dụng trong nhiều lĩnh vực như phân tích mạng xã hội, crawling web, và định tuyến mạng. Nếu mỗi thành phần liên thông mạnh được co lại thành một đỉnh, thì đồ thị sẽ trở thành một đồ thị có hướng không có chu trình.



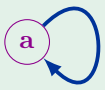
Hình 3. Đồ thị này không liên thông mạnh vì đỉnh 5 không thể đi đến đỉnh nảo khác

Độc giả có thể tìm hiểu thêm về SCCs thông qua các thuật toán **Kosaraju**, **Tarjan**, và **Gabow**. Những thuật toán này đều có thể tìm các thành phần liên thông mạnh của một đồ thị cho trước trong thời gian tuyến tính. Ta sẽ chỉ dừng lại ở đây với các khái niệm về liên thông mạnh.

Định nghĩa 4 Nếu R là một quan hệ (`Relation`) từ A đến A, thì $R \subseteq A \times A$; chúng ta nói R là một quan hệ trên A. Ta ký hiệu aRb $\Leftrightarrow (a, b) \in R$.

Tính chất 1 Một quan hệ R trên A được gọi là:

(i) Phản xạ - `Reflexive`: nếu $(a, a) \in R_a$ cho mọi $a \in A$

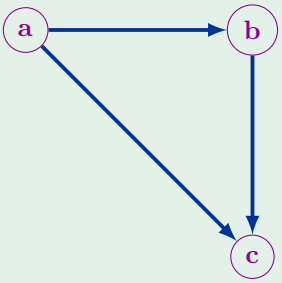


(ii) Không phản xạ - `Irreflexive`: nếu $(a, a) \notin R$ cho mọi $a \in A$

(iii) Đối xứng - `Symmetric`: nếu $(a, b) \in R \Rightarrow (b, a) \in R$ cho mọi $a, b \in A$

(iv) Phản xứng - `Anti-symmetric`: nếu $[(a,b) \in R \wedge (b,a) \in R] \Rightarrow a = b$ cho mọi $a, b \in A$

(v) Bắc cầu - `Transitive`: nếu $[(a,b) \in R \wedge (b,c) \in R] \Rightarrow (a,c) \in R$ cho mọi $a, b, c \in A$



Định nghĩa 5 **Một quan hệ trống** hay `Empty relation` R là tập con $\emptyset$. Dễ thấy nó không có tính phản xạ. Để kiểm tra tính đối xứng, chúng ta muốn biết liệu $aRb \Rightarrow bRa$ cho tất cả $a, b \in A$. Cụ thể hơn, chúng ta muốn biết liệu $(a,b) \in \emptyset \Rightarrow (b,a) \in \emptyset$. Vì $(a,b) \in \emptyset$ luôn là sai, mệnh đề kết luận luôn đúng. Do đó, mối quan hệ là đối xứng. Tương tự, nó cũng là chống đối xứng và bắc cầu.

Một quan hệ hoàn chỉnh hay `Complete relation` R là tập hợp toàn bộ $A \times A$. Dễ thấy nó phản xạ, đối xứng và bắc cầu. Đồng thời nó phản xứng khi và chỉ khi $|A| = 1$.

Một quan hệ đồng nhất hay `Identity relation` R bao gồm các cặp có dạng (a,a), với $a \in A$. Nói cách khác, aRb khi và chỉ khi $a = b$. Khi đó R phản xạ, đối xứng, chống đối xứng và bắc cầu.

Định nghĩa 6 (Sắp xếp trên một tập hợp) Với A là một tập hợp, một quan hệ $R \subseteq A \times A$ được gọi là một sắp xếp (`Ordering`) trên A nếu nó thỏa mãn các tính chất phản xạ, bắc cầu, đầy đủ và phản xứng. Đặt $L(A)$ là tập hợp các sắp xếp trên A.

Ký hiệu: Ta quy ước $\preceq$ là một sắp xếp, khi đó $a \simeq b$ nếu và chỉ nếu $a \preceq b$ và $b \preceq a$.

Định nghĩa 7 (Hệ thống xếp hạng) Đặt $\mathbb{G}_V$ là tập hợp tất cả các đồ thị liên thông mạnh với tập đỉnh V. Một hệ thống xếp hạng (`Page-ranking system`) F là một hàm số sao cho đối với mỗi tập đỉnh hữu hạn V, nó ánh xạ mọi đồ thị mạnh liên thông $G \in \mathbb{G}_V$ sang một sắp xếp $\preceq_G^F \in L(V)$.

Định nghĩa 8 Xét $G = (V, E)$ là một đồ thị có hướng, và $v \in V$ là một đỉnh trong G. Khi đó: Tập liên kết ra (`Outlinks`) của v được định nghĩa là $S_G(v) = \{u | (v,u) \in E\}$, và tập liên kết vào (`Iternal links`) của v là $P_G(v) = \{u | (u,v) \in E\}$.

Section 5

Đại số Tuyến tính & Xác suất Thống kê

Định nghĩa 9 Ma trận thưa (`Sparse Matrix`) là ma trận mà phần lớn các phần tử bằng 0. Ngược lại sẽ được coi là 'dày đặc' (`dense`).

Ma trận xác suất (`Stochastic Matrix`), hay còn được gọi là ma trận chuyển đổi, được sử dụng để mô tả sự chuyển đổi trong mô hình người lướt web ngẫu nhiên. Đây là một ma trận mà mọi phần tử đều là số thực không âm và tổng các phần tử trên mỗi hàng bằng 1.

$$A = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 0 & 5 \\ 0 & 0 & 0 \end{bmatrix}$$

Hình 4. Ma trận A như trên là một ma trận thưa kích thước 3×3

$$B = \begin{bmatrix} 0.3 & 0.5 & 0.2 \\ 0.1 & 0.4 & 0.5 \\ 0.6 & 0.3 & 0.1 \end{bmatrix}$$

Hình 5. Ma trận B như trên là một ma trận xác suất hàng kích thước 3×3 vì tổng các phần tử mỗi hàng bằng 1

Không có định nghĩa nghiêm ngặt về tỷ lệ phần tử 0 để một ma trận được coi là thưa nhưng một tiêu chí phổ biến là số lượng phần tử khác 0 gần bằng số lượng hàng hoặc cột. Ngược lại, ma trận được coi là dày đặc. Tỷ lệ của số phần tử 0 so với số phần tử của ma trận đôi khi được gọi là **độ thưa** của ma trận.

Cả hai loại ma trận này đều đóng vai trò quan trọng trong thuật toán PageRank, giúp thuật toán tận dụng cấu trúc đặc biệt của đồ thị web và quá trình di chuyển ngẫu nhiên của người dùng để đưa ra xếp hạng trang web một cách hiệu quả. Cụ thể:

- Trong thuật toán PageRank, ma trận chuyển (`transition matrix`) đại diện cho đồ thị web, với các phần tử biểu diễn khả năng di chuyển từ một trang web đến trang web khác. Do đồ thị web thường rất lớn nhưng chỉ có một số lượng hạn chế các liên kết giữa các trang web, ma trận chuyển thường là ma trận thưa.
- Trong thuật toán PageRank, ma trận xác suất được sử dụng để mô hình hóa quá trình di chuyển ngẫu nhiên của người dùng trên web. Mỗi hàng của ma trận xác suất đại diện cho khả năng di chuyển từ một trang web đến tất cả các trang web khác, với tổng các khả năng này phải bằng 1. Việc sử dụng ma trận xác suất giúp thuật toán PageRank mô phỏng chính xác hơn quá trình di chuyển ngẫu nhiên của người dùng trên web, từ đó đưa ra xếp hạng trang web một cách chính xác hơn

Ma trận xác suất còn được goi là Ma trận Markov (`Markov Matrix`). Nhiều tác giả viết chuyển vị của ma trận này là A^T và nhân vào bên phải của một vector hàng p. Trong tài liệu này ta sẽ ghi là $A \cdot p$. Điều này rõ ràng là tương đương.

Định nghĩa 10 Ta gọi một vector p với các phần tử không âm p_k sao cho tổng các p_k bằng 1 là một vector xác suất (`Stochastic vector`).

Lưu ý là chỉ cần tổng các phần tử của p bằng 1 thì p là một vector xác suất. Nghĩa là định nghĩa này có thể được áp dụng cho cả vector cột và hàng. Từ đó thì những hệ quả như dưới đây cũng đúng với cả hai loại vector này.

Hệ quả 1 Đối với một ma trận xác suất, mỗi `cột/hàng` là một vector xác suất. Nếu p là một vector xác suất và A là một ma trận xác suất, thì $A \cdot p$ là một vector xác suất.

Proof Ta sẽ chứng minh hệ quả trên với A là ma trận xác suất cột và p là một vector xác suất cột. Giả sử $v_1, \ldots, v_n$ là các vector cột của A. Khi đó $v_1, v_2, \cdots, v_n$ là các vector xác suất, đồng nghĩa với việc tổng các phần tử của mỗi v_i bằng 1.

Đặt $p = \begin{bmatrix} p_1 & p_2 & \cdots & p_n \end{bmatrix}^T$, do p là một vector xác suất nên $p_1, p_2, \ldots, p_n \geq 0$ và

$p_1 + p_2 + \cdots + p_n = 1$.

Khi đó tích $A \cdot p$ được biểu diễn như sau:

$$A \cdot p = \begin{bmatrix} | & | & \cdots & | \\ v_1 & v_2 & \cdots & v_n \\ | & | & \cdots & | \\ | & | & \cdots & | \end{bmatrix} \cdot \begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ p_n \end{bmatrix} = p_1 v_1 + p_2 v_2 + \cdots + p_n v_n$$

Đây là tích của một ma trận $n \times n$ và $n \times 1$ nên kết quả sẽ cho ra một ma trận $n \times 1$, tương đương một vector cột n hàng. Việc còn lại là chứng minh tổng các phần tử của $A \cdot p$ bằng 1. Thật vậy:

$$\begin{aligned} \sum (A \cdot p) &= \sum_{s \in (A \cdot p)} s = \sum \sum_{i=1}^{n} p_i v_i = \sum (p_1 v_1 + p_2 v_2 + \cdots + p_n v_n) \\ &= \sum_{i=1}^{n} p_i \sum_{s \in (v_i)} s = p_1 \sum v_1 + p_2 \sum v_2 + \cdots + p_n \sum v_n \end{aligned}$$

Vì $\sum v_i = 1$ cho mọi i, ta có: $\sum (A \cdot p) = p_1 + p_2 + \cdots + p_n = 1$. Vậy ta có điều phải chứng minh! □

Hệ quả 2 Một ma trận xác suất A luôn có một trị riêng (`eigenvalue`) bằng 1. Trong khi đó các trị riêng phân biệt khác sẽ có giá trị tuyệt đối nhỏ hơn 1.

Proof Ta sẽ chứng minh hệ quả trên với vector và ma trận xác suất cột. Xét ma trận chuyển vị của A là A^T, dễ thấy các vector hàng của A^T là vector xác suất. Do đó, tổng các phần tử trên mỗi vector hàng của A^T đều bằng 1. Mặt khác, ma trận chuyển vị A^T có vector riêng:

$$\begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}$$

Bởi vì A và A^T có cùng định thức, nên $A - \lambda I_n$ và $A^T - \lambda I_n$ có cùng định thức, từ đó các giá trị riêng của A và A^T là như nhau. Với việc A^T có một giá trị riêng là 1, thì A cũng có một giá trị riêng là 1.

Giả sử v là một vector riêng với giá trị riêng $|\lambda| > 1$. Khi đó, $A^n v = |\lambda|^n v$ có tốc độ tăng theo cấp số nhân khi $n \to \infty$. Điều này có nghĩa là với n đủ lớn, tồn tại một hệ số $[A^n]_{ij}$ lớn hơn 1. Tuy nhiên, A^n là một ma trận xác suất và tất cả các phần tử của nó đều ≤ 1.

Ta thấy được điểm mâu thuẫn, nên giả thiết tồn tại một trị riêng λ sao cho $|\lambda| > 1$ là sai. Ta có điều phải chứng minh! □

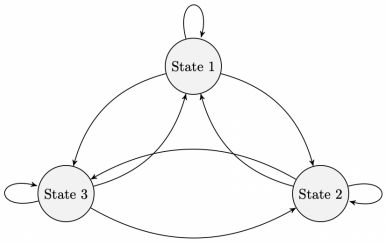
Định nghĩa 11 (Markov chain) Cho $X_0, X_1, \cdots : \Omega \to \mathbb{E}$ là một tiến trình ngẫu nhiên được định nghĩa trên không gian $(\Omega, \mathcal{F}, P)$ và các trạng thái $\mathbb{E} = \{1, 2, \cdots, n\}$. Khi đó, $X_0, X_1, \cdots$ được gọi là một xích Markov với phân phối ban đầu là $v = (v_1, v_2, \cdots, v_n)^T$ và ma

trận chuyển $A = (p_{ij})$ nếu thỏa mãn:

$$P(X_0 = i_0, X_1 = i_1, \cdots, X_n = i_n) = \nu_{i_0} p_{i_0 i_1} \cdots p_{i_{n-1} i_n} \tag{5.1}$$

với $n = 0, 1, \cdots$ tùy ý là $i_0, i_1, \cdots, i_n \in \mathbb{E}$

Dựa vào (5.1), ta suy ra được một số hệ quả sau:



Hình 6. Một xích Markov với 3 trạng thái State 1, Sate 2, State 3

Hệ quả 3 Nếu tập S là hữu hạn thì ta có một xích Markov hữu hạn. Đặt $p_{ij} = P(X_{n+1} = j | X_n = i)$, khi đó p_{ij} không phụ thuộc vào n, cụ thể:

$$p_{ij} = P(X_1 = j | X_0 = i) = P(X_2 = j | X_1 = i) = P(X_3 = j | X_2 = i) = \dots$$

Nếu $m < n$ và $P(X_{n+h} = j | X_{m+h} = i) = P(X_n = j | X_m = i)$ với mọi h, thì ta nói xích Markov này là thuần nhất.

Định nghĩa 12 (Ma trận xác suất chuyển trạng thái) Giả sử tập trạng thái $S = \{1, 2, 3, \dots, r\}$ và $p_{ij} = P(X_{n+1} = j | X_n = i)$. Khi đó, ma trận

$$P = \begin{bmatrix} p_{11} & p_{12} & \cdots & p_{1r} \\ p_{21} & p_{22} & \cdots & p_{2r} \\ \vdots & \vdots & \ddots & \vdots \\ p_{r1} & p_{r2} & \cdots & p_{rr} \end{bmatrix} \tag{5.2}$$

được gọi là ma trận xác suất chuyển trạng thái sau một bước.

Lưu ý Ta luôn có $p_{ij} \geq 0$ với mọi i và $\sum_{k=1}^{r} p_{ik} = \sum_{k=1}^{r} P(X_{n+1} = k | X_n = i) = 1$.

Tiếp sau đây là những nội dung kiến thức về **Phân phối xác suất trạng thái** - `State Probability Distribution`.

Định nghĩa 13 (Xác suất chuyển sau n bước) Xác suất chuyển sau n bước được định nghĩa như sau:

$$p_{ij}^{(n)} = P(X_{n+m} = j | X_m = i) = P(X_n = j | X_0 = i),$$

với quy ước $p_{ij}^{(0)} = \begin{cases} 1 & \text{nếu } i = j \\ 0 & \text{nếu } i \neq j \end{cases}$

Giả sử có không gian trạng thái $S = \{1, 2, 3, \dots, r\}$, phân phối của biến X_n được cho bởi

$$\pi^{(n)} = (P(X_n = 1), P(X_n = 2), \dots, P(X_n = r)).$$

Khi đó phân phối của biến X_0 $\pi^{(0)} = (P(X_0 = 1), P(X_0 = 2), \dots, P(X_0 = r))$ được gọi là phân phối ban đầu của hệ.

Định nghĩa 14 (Egordic) Xích Markov với tiến trình $X_0, X_1, \cdots$ với ma trận chuyển $P = (p_{ij})$ tương ứng với trạng thái thứ n của ma trận $P^{(n)} = (p_{ij}^{(n)})(= P^n)$ được gọi là **egordic** nếu

giới hạn

$$\pi_j = \lim_{n\to\infty} p_{ij}^{(n)}$$

1. tồn tại với mọi $j \in \mathbb{E}$
2. tất cả là số dương và không phụ thuộc vào $i \in \mathbb{E}$
3. $\pi = (\pi_1, \pi_2, \cdots, \pi_n)$ là vector biến cố, tức $\sum_{j\in\mathbb{E}} \pi_j = 1$

Định nghĩa 15 (non-negative và quasi-positivity) Ma trận non-negative và quasi-positivity được định nghĩa như sau:

1. Một ma trận vuông $n \times n$ $P = (p_{ij})$ được gọi là non-negative (không âm) nếu tất cả các phần tử $p_{ij} \geq 0$, $\forall i, j \in [1, n]$
2. Một ma trận vuông $n \times n$ $P = (p_{ij})$ được gọi là quasi-positivity nếu tồn tại n_0 nguyên dương sao cho tất cả các phần tử của P^{n_0} dương.

Định lý 1 Từ trên, ta rút ra được một ma trận biến cố hàng P được gọi là egordic nếu và chỉ nếu ma trận P là ma trận quasi-positivity.

Proof Ta chỉ cần chứng minh chiều ma trận P là ma trận quasi-positivity thì nó sẽ là egordic (vì chiều xuôi là hiển nhiên theo định nghĩa của egordic)
Với ma trận biến cố hàng P, ta có đẳng thức sau: $P^{(n+m)} = P^{(n)}P^{(m)}$ $(= P^{m+n})$
Từ đó, ta rút ra được: $p_{ij}^{(n+1)} = \sum_{k\in\mathbb{E}} p_{ik}p_{kj}^{(n)}$
Đặt $m_j^{(n)} = \min_{i\in\mathbb{E}} p_{ij}^{(n)} > 0$ và $M_j^{(n)} = \max_{i\in\mathbb{E}} p_{ij}^{(n)} > 0$ thì ta rút ra được

$$m_j^{(n+1)} = \min_{i\in\mathbb{E}} p_{ij}^{(n+1)} = \min_{i\in\mathbb{E}} p_{ij} \sum_{k\in\mathbb{E}} p_{ik}p_{kj}^{(n)} \geq \min_{i\in\mathbb{E}} p_{ij} \sum_{k\in\mathbb{E}} p_{ik} \min_{l\in\mathbb{E}} p_{lj}^{(n)} = \min_{l\in\mathbb{E}} p_{lj}^{(n)} = m_j^{(n)}$$

Do đó, ta thu được $m_j^{(n+1)} \geq m_j^{(n)}$ với $n \geq 0$. Chứng minh tương tự, ta thu được $M_j^{(n)} \leq M_j^{(n+1)}$ với $n \geq 0$

Với ma trận chuyển P, ta cần chứng minh tồn tại giới hạn $\pi_j = \lim_{n\to\infty} p_{ij}^{(n)}, \forall j \in \mathbb{E}$ hay ta cần chứng minh $\lim_{n\to\infty} (M_j^{(n)} - m_j^{(n)}) = 0$
Xét 2 trạng thái $i_0, j_0 \in \mathbb{E}$ và xét $E' = \{k \in \mathbb{E} : p_{i_0k}^{(n_0)} \geq p_{j_0k}^{(n_0)}\}$ và $E'' = E \setminus E'$
Đặt $a = \min_{i,j\in\mathbb{E}} p_{ij}^{(n_0)} > 0$ thì

$$\sum_{k\in E'} (p_{i_0k}^{(n_0)} - p_{j_0k}^{(n_0)}) = 1 - \sum_{k\in E''} p_{i_0k}^{(n_0)} - \sum_{k\in E'} p_{j_0k}^{(n_0)} \leq 1 - na$$

Và

$$\sum_{k\in E''} (p_{i_0k}^{(n_0)} - p_{j_0k}^{(n_0)}) = -\sum_{k\in E'} (p_{i_0k}^{(n_0)} - p_{j_0k}^{(n_0)})$$

Mặc khác, áp dụng công thức trên với $n \le 0$, ta được:

$$\begin{aligned}
p_{i_0 a}^{(n_0+n)} - p_{j_0 j}^{(n_0+n)} &= \sum_{k\in E} \left(p_{i_0 k}^{(n_0)} - p_{j_0 k}^{(n_0)}\right) p_{kj}^{(n)} \\
&= \sum_{k\in E'} \left(p_{i_0 k}^{(n_0)} - p_{j_0 k}^{(n_0)}\right) p_{kj}^{(n)} + \sum_{k\in E''} \left(p_{i_0 k}^{(n_0)} - p_{j_0 k}^{(n_0)}\right) p_{kj}^{(n)} \\
&\le \sum_{k\in E'} \left(p_{i_0 k}^{(n_0)} - p_{j_0 k}^{(n_0)}\right) M_j^{(n)} + \sum_{k\in E''} \left(p_{i_0 k}^{(n_0)} - p_{j_0 k}^{(n_0)}\right) m_j^{(n)} \\
&= \sum_{k\in E'} \left(p_{i_0 k}^{(n_0)} - p_{j_0 k}^{(n_0)}\right) M_j^{(n)} - \sum_{k\in E''} \left(p_{j_0 k}^{(n_0)} - p_{i_0 k}^{(n_0)}\right) m_j^{(n)} \\
&= \sum_{k\in E'} \left(p_{i_0 k}^{(n_0)} - p_{j_0 k}^{(n_0)}\right) \left(M_j^{(n)} - m_j^{(n)}\right) \\
&\le (1 - n\alpha) \left(M_j^{(n)} - m_j^{(n)}\right).
\end{aligned}$$

Từ đó, ta rút ra được $M_j^{n_0+n} - m_j^{(n_0+n)} \le \left(M_j^{(n)} - m_j^{(n)}\right)(1 - n\alpha)$
Từ đó, với $k \ge 1$, bằng quy nạp, ta thu được:

$$M_j^{n_0+kn} - m_j^{(n_0+kn)} \le \left(M_j^{(n)} - m_j^{(n)}\right)(1 - n\alpha)^k$$

Do đó

$$\lim_{k\to\infty} (M_j^{n_0+kn} - m_j^{(n_0+kn)}) = 0$$

Từ đó, ta suy ra được luôn tồn tại π_j.
Hiển nhiên $\pi_j > 0$ vì $\pi_j = \lim_{n\to\infty} p_{ij}^{(n)} \ge \lim_{n\to\infty} m_j^{(n)} \ge m_j^{(n_0)} \ge \alpha > 0$
Hơn thế nữa, ta có $\sum_{j\in\mathbb{E}} \pi_j = \sum_{j\in\mathbb{E}} \left(\lim_{n\to\infty} p_{ij}^{(n)}\right) = \lim_{n\to\infty} \left(\sum_{j\in\mathbb{E}} p_{ij}^{(n)}\right) = 1$
Định lý trên đã được chứng minh. □

Định lý 2 Cho xích Markov $\{X_n, n = 0, 1, 2, \dots\}$ với ma trận xác suất chuyển sau một bước là P, khi đó:

(i) $\pi^{(n+1)} = \pi^{(n)} P$, với $n = 0, 1, 2, \dots$

(ii) $\pi^{(n)} = \pi^{(0)} P^n$, với $n = 0, 1, 2, \dots$

Proof Để chứng minh định lý trên cho chuỗi Markov $\{X_n, n = 0, 1, 2, \dots\}$ với ma trận xác suất chuyển sau một bước là P, ta cần hiểu rằng $\pi^{(n)}$ biểu diễn phân phối xác suất của chuỗi Markov sau n bước. Định lý đề cập đến cách tính phân phối xác suất sau mỗi bước và sau một chuỗi số bước.

Chứng Minh 1: $\pi^{(n+1)} = \pi^{(n)} P$

Giả sử $\pi^{(n)}$ là phân phối xác suất sau n bước, tức là $\pi_i^{(n)}$ là xác suất gặp trạng thái i sau n bước. Để tìm $\pi^{(n+1)}$, ta cần xem xét cách mà chuỗi Markov tiến hóa từ trạng thái i sau n bước đến trạng thái j sau $(n + 1)$ bước.

- Trước tiên, từ trạng thái i sau n bước, chuỗi có thể di chuyển đến trạng thái j sau $(n + 1)$ bước thông qua một bước chuyển từ trạng thái i đến một trạng

thái tạm thời k, sau đó từ trạng thái k đến trạng thái j. Xác suất cho việc này xảy ra là $\pi_i^{(n)} P_{ik} P_{kj}$.

- Tổng hợp tất cả các cách mà chuỗi có thể di chuyển từ trạng thái i sau n bước đến trạng thái j sau $(n+1)$ bước, ta nhận được $\pi_j^{(n+1)} = \sum_{k \in S} \pi_i^{(n)} P_{ik} P_{kj}$.

Điều này chứng minh rằng $\pi^{(n+1)} = \pi^{(n)} P$.

Chứng Minh 2: $\pi^{(n)} = \pi^{(0)} P^n$

Chúng ta biết rằng $\pi^{(0)}$ là phân phối ban đầu của chuỗi Markov, tức là xác suất khởi đầu ở mỗi trạng thái. Để tìm $\pi^{(n)}$, ta cần xem xét cách mà chuỗi Markov tiến hóa từ phân phối ban đầu $\pi^{(0)}$ sau n bước.

- Theo định nghĩa, $\pi^{(n)} = \pi^{(n-1)} P$, vì vậy ta có thể viết $\pi^{(n)} = (\pi^{(0)} P^{n-1}) P = \pi^{(0)} P^n$.

Điều này chứng minh rằng $\pi^{(n)} = \pi^{(0)} P^n$.
Như vậy, cả hai phần của định lý đều được chứng minh! □

Định nghĩa 16 (Phân phối dừng) Phân phối ban đầu được gọi là dừng nếu $\pi^{(n)}$ không phụ thuộc vào n, tức là:

$$\pi = \pi^{(n)} P \quad \text{hay} \quad \pi = \pi^{(n)}$$

Hệ quả 4 Cho vector hàng $\pi = (\pi_1, \pi_2, \ldots, \pi_r)$ thỏa $\pi_i \geq 0$ và $\sum_{i \in S} \pi_i = 1$ được gọi là phân phối dừng (`Stationary Distribution`) của xích Markov với ma trận chuyển trạng thái P nếu

$$\sum_i \pi_i \cdot p_{ij} = \pi_j, \quad \forall j \in S$$

Nghĩa là π là nghiệm của hệ phương trình tuyến tính

$$\pi \cdot P = \pi$$

Ví dụ Giả sử chúng ta có một xích Markov với ma trận xác suất chuyển trạng thái sau:

$$P = \begin{bmatrix} 0.3 & 0.6 & 0.1 \\ 0.2 & 0.5 & 0.3 \\ 0.1 & 0.3 & 0.6 \end{bmatrix}$$

Phân phối dừng $\pi = (\pi_1, \pi_2, \pi_3)$ là nghiệm của hệ phương trình:

$$\begin{cases} 0.3\pi_1 + 0.2\pi_2 + 0.1\pi_3 = \pi_1 \\ 0.6\pi_1 + 0.5\pi_2 + 0.3\pi_3 = \pi_2 \\ 0.1\pi_1 + 0.3\pi_2 + 0.6\pi_3 = \pi_3 \\ \pi_1 + \pi_2 + \pi_3 = 1 \end{cases}$$

Giải hệ phương trình trên, chúng ta sẽ thu được phân phối dừng $\pi = (1/6, 1/3, 1/2)$.

Định nghĩa 17 (Bất khả quy) Một xích Markov với ma trận chuyển trạng thái P là bất khả quy nếu với mọi trạng thái $i, j \in S$, tồn tại một số nguyên dương n sao cho phần tử ở hàng i cột j của ma trận P^n là số dương.

$$P = \begin{bmatrix} 0.4 & 0.3 & 0.3 \\ 0.2 & 0.5 & 0.4 \\ 0.1 & 0.4 & 0.5 \end{bmatrix}$$

Hình 7. Ma trận P như trên là bất khả quy vì P^3 có tất cả các phần tử đều dương

Định nghĩa 18 Chu kỳ $d(i)$ của một trạng thái i được xác định bởi công thức

$$d(i) = gcd\{n : p_{ii}(n) > 0\}$$

có nghĩa là nếu gọi S là tập hợp các phần tử **số lần quay lại trạng thái i từ i** thì $d(i)$ là ước chung lớn nhất của những phần tử của S.

Chúng ta gọi trạng thái i là tuần hoàn (`periodic`) nếu $d(i) > 1$ và phi tuần hoàn (`aperiodic`) nếu $d(i) = 1$.

Hồi qui dương (`positive recurrent` hay `persistent`) nghĩa là thời gian được kì vọng trở lại trạng thái ban đầu là một giá trị dương cho mọi trạng thái.

$$Q = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

Hình 8. Ma trận Q như trên khả quy vì từ mọi trạng thái, có thể đến được mọi trạng thái khác sau một số bước.

Đối với ứng dụng dữ liệu trong cuộc sống, chúng ta sử dụng các mô hình mà ở đó **MCMC** (`Markov Chain Monte Carlo`) được xây dựng theo hướng ergodic, có nghĩa là mỗi trạng thái của chuỗi Markov mà chúng ta sử dụng ở các mô hình này đều là ergodic.

Định lý 3 (Kac–Bernstein) Với mọi xích Markov bất khả quy, nếu **hồi quy dương** thì tồn tại duy nhất một phân phối dừng và được xác định bởi $\pi_i = \dfrac{1}{\mu_i}$, trong đó μ_i là thời gian trung bình trở lại trạng thái i. Khi đó vector phân phối dừng có mọi thành phần đều dương.

Ngược lại nếu xích Markov không hồi quy dương (`null recurrent`) thì không tồn tại phân phối dừng.

Proof Ta sẽ chứng minh định lý `Kac-Bernstein` trên hai khía cạnh: Sự tồn tại của phân phối dừng và tính duy nhất (độc nhất) của nó.

1. Tính tồn tại: Mỗi xích Markov bất khả quy có hồi quy dương đều có một phân phối dừng.

Gọi một vector ν là một vector phân phối dừng nếu $\nu P = \nu$. Điều này hoàn toàn giống như một phân phối dừng, ngoại trừ việc không có điều kiện chuẩn hóa rằng tổng của nó phải bằng 1.

Giả sử rằng (X_n) là hồi quy (không quan tâm là hồi quy dương hoặc không, kể cả bất thường - `transient`).

Nhiệm vụ đầu tiên của chúng ta sẽ là tìm một vector phân phối dừng. Cố định một trạng thái ban đầu k, và gọi ν_i là số lần 'ghé thăm' trạng thái i trước khi

quay trở lại k. Cụ thể:

$$\begin{aligned}\nu_i &= \mathbb{E}(\#\text{ visits to } i \text{ before returning to } k \mid X_0 = k)\\ &= \mathbb{E}M_k \sum_{n=1}^{M_k} P(X_n = i \mid X_0 = k)\\ &= \sum_{n=1}^{\infty} \mathbb{P}(X_n = i \text{ and } n \le M_k \mid X_0 = k),\end{aligned}$$

trong đó M_k là thời gian trở lại. Theo định nghĩa này, $\nu_k = 1$, vì lần duy nhất thăm k là lần trở lại chính nó.

Vì ν có thể coi như đếm số lượt thăm các trạng thái khác nhau trong một khoảng thời gian (ngẫu nhiên) nào đó, ta nhận thấy ν phù hợp để được chuẩn hóa thành một phân phối dừng, nghĩa là ν chính nó có thể là một vector phân phối dừng.

Chúng ta muốn chứng minh rằng $\sum_i \nu_i \cdot p_{ij} = \nu_j$. Ta sẽ kiểm tra điều đó:

$$\begin{aligned}\sum_{i\in S} \nu_i p_{ij} &= \sum_{i\in S}\sum_{n=1}^{\infty} \mathbb{P}(X_n = i \text{ and } n \le M_k \mid X_0 = k) p_{ij}\\ &= \sum_{n=1}^{\infty}\sum_{i\in S} \mathbb{P}(X_n = i \text{ and } X_{n+1} = j \text{ and } n \le M_k \mid X_0 = k)\\ &= \sum_{n=1}^{\infty} \mathbb{P}(X_{n+1} = j \text{ and } n \le M_k \mid X_0 = k).\end{aligned}$$

Ở đây việc hoán đổi thứ tự của các tổng là hợp lệ, vì tính hồi quy của xích đảm bảo rằng M_k là hữu hạn với xác suất 1. Để ý thay vì đếm số lượt thăm từ 1 đến M_k, chúng ta có thể đếm số lượt thăm từ 0 đến $M_k - 1$. Ta có:

$$\begin{aligned}\sum_{i\in S} \nu_i p_{ij} &= \sum_{n=0}^{\infty} \mathbb{P}(X_{n+1} = j \text{ and } n \le M_k - 1 \mid X_0 = k)\\ &= \sum_{n+1=1}^{\infty} \mathbb{P}(X_{n+1} = j \text{ and } n+1 \le M_k \mid X_0 = k)\\ &= \sum_{n=1}^{\infty} \mathbb{P}(X_n = j \text{ and } n \le M_k \mid X_0 = k) = \nu_j.\end{aligned}$$

Vậy ν thực sự là một vector phân phối dừng. Ta đã chứng minh được tồn tại vector phân phối dừng.

Bây giờ chúng ta có thể chuẩn hóa ν thành một phân phối dừng bằng cách chia cho $\sum_i \nu_i$. Nhưng để ý điều này chỉ có thể được thực hiện nếu $\sum_i \nu_i$ là hữu hạn. Mà $\sum_i \nu_i$ chính là tổng số lượt thăm trạng thái trước khi quay lại k, tức là thời gian trung bình trở lại μ_k. Giả sử (X_n) hồi quy dương, từ đó μ_k là hữu hạn. Vậy ta có $\pi = (1/\mu_k)\nu$ là một phân phối dừng. Tính tồn tại của phân phối dừng khi xích Markov bất khả quy và hồi quy dương đã được chứng minh!

2. Tính duy nhất: Đối với một xích Markov bất khả quy, hồi quy dương, phân phối dừng là duy nhất và được cho bởi $\pi_i = 1/\mu_i$.

Giả sử π là một phân phối dừng ta đã tìm được khi xích Markov bất khả quy và hồi quy dương. Ta cần chứng minh $\pi_i = 1/\mu_i$ với mọi i.

Thật vậy, đầu tiên ta có:

$$\mu_k = 1 + \sum_j p_{kj}\eta_{jk} \tag{5.3}$$

Vì công thức này cũng liên quan đến thời gian trung bình đến trạng thái k, ta cũng có

$$\eta_{ik} = 1 + \sum_j p_{ij}\eta_{jk} \text{ cho mọi } i \neq k \tag{5.4}$$

Từ biểu thức (5.4), nhân 2 vế với π_i và lấy tổng trên các $i \neq k$, ta được

$$\sum_i \pi_i\eta_{ik} = \sum_{i\neq k} \pi_i + \sum_j \sum_{i\neq k} \pi_i p_{ij}\eta_{jk}. \tag{5.5}$$

Để ý là tổng ở vế trái có thể coi là trên toàn bộ i, vì $\eta_{kk} = 0$.) Tiếp theo, nhân 2 vế của biểu thức (5.3) với π_k để có

$$\pi_k\mu_k = \pi_k + \sum_j \pi_k p_{kj}\eta_{jk} \tag{5.6}$$

Cộng vế theo vế 2 biểu thức (5.5) và (5.6), ta suy ra

$$\sum_i \pi_i\eta_{ik} + \pi_k\mu_k = \sum_i \pi_i + \sum_j \sum_i \pi_i p_{ij}\eta_{jk}$$

Ta lại có $\sum_i \pi_i p_{ij} = \pi_j$ và $\sum_i \pi_i = 1$, nên

$$\sum_i \pi_i\eta_{ik} + \pi_k\mu_k = 1 + \sum_j \pi_j\eta_{jk}$$

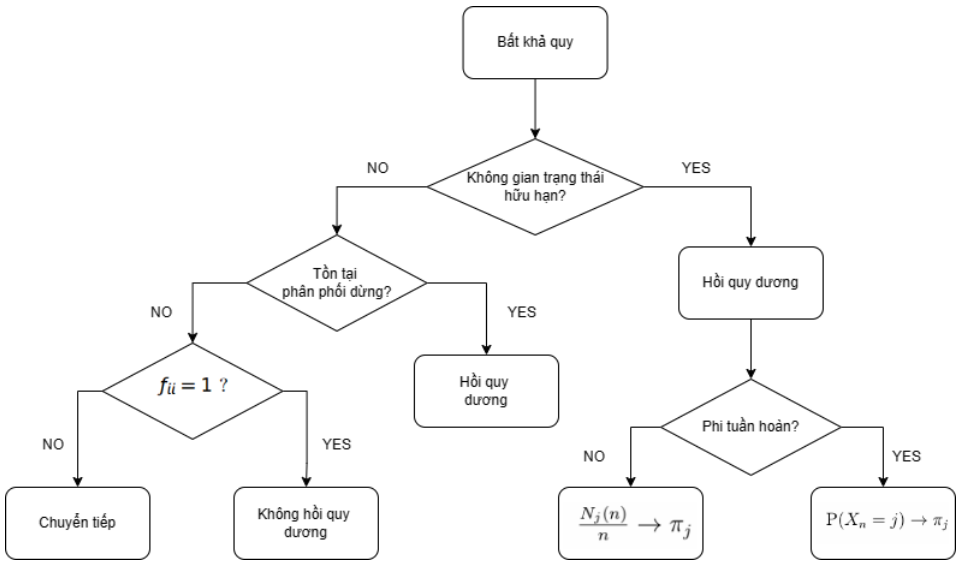
$$\Leftrightarrow \pi_k\mu_k = 1 \Leftrightarrow \pi_k = \frac{1}{\mu_k}$$

Tới đây ta có điều phải chứng minh! □

Định lý 4 (Phân phối giới hạn của chuỗi Markov - `Limiting distribution of Markov chain`) Cho ma trận xác suất chuyển trạng thái P, đặt $\pi_j = \lim_{n\to\infty} P(n)_{ij}$.

Nếu xích Markov $\{X_0, X_1, \ldots$ với ma trận xác suất chuyển P thỏa tồn tại một số nguyên dương m sao cho P^m có tất cả các phần tử đều dương, và có phân phối dừng π, thì $P(X_n = i)$ sẽ hội tụ tới π_i khi $n \to \infty$. Khi đó mọi hàng của ma trận P^n sẽ hội tụ tới π khi $n \to \infty$.

Ta có cái nhìn tổng quan về mối liên hệ giữa các trạng thái trong một xích Markov như sau:



Section 6

Định lý Perron - Frobenius

Định lý 5 (Perron - Frobenius) Cho ma trận $n \times n$ A là ma trận dương, tức là $A_{ij} > 0$, $\forall 0 \leqslant i, j \leqslant n$. Khi đó A có một trị riêng lớn nhất duy nhất. Vectơ riêng tương ứng của nó cũng có các phần tử đều dương.

Lưu ý Ta có những điều cần lưu ý như sau:

1. Ở trên có nói trị riêng lớn nhất duy nhất, điều này không có nghĩa là trị riêng này là duy nhất.

   Toán học định nghĩa trị riêng lớn nhất này là trị riêng trội (`dominant eigenvalue`). Và số lần mà một trị riêng λ xuất hiện trong đa thức đặc trưng (`characteristic polynomial`) của một ma trận là số bội đại số (`algebraic multiplicity`) của trị riêng đó, ký hiệu là $\mu(\lambda)$

   Vậy định lý muốn chỉ ra nếu ma trận có một trị riêng lớn nhất là λ thi $\mu(\lambda) = 1$.

2. Nếu A là ma trận dương thì trị riêng trội λ là duy nhất, hay $\mu(\lambda) = 1$. Còn nếu A chỉ có các phần tử đều không âm (vẫn có thể có 0) thì $\mu(\lambda) >= 1$

Định lý Perron - Frobenius đóng một vai trò lớn trong thuật toán PageRank. Có nhiều cách để chứng minh định lý trên, ở phần này nhóm tác giả sẽ giới thiệu một phương án chứng minh thiên về hình học.

Để phục vụ cho việc chứng minh, sau đây là một số những kết quả đáng chú ý:

Định nghĩa 19 (Không gian Metric) Cho X là một tập khác rỗng, xét một phiếm hàm (tức là một ánh xạ có nguồn là tập số) là

$$\begin{aligned} d : X \times X &\to [0; +\infty) \\ (x, x') &\mapsto d(x, x') \end{aligned}$$

Cặp (X, d) đó, sẽ được gọi là một không gian metric (với d gọi là khoảng cách), nếu nó thỏa mãn đồng thời các điều kiện sau đây:

1. *Xác định dương*: $d(x, x') \geq 0 \quad \forall x, x' \in X$
2. *Đối xứng*: $d(x, x') \geq 0 \quad \forall x, x' \in X$
3. *Bất đẳng thức tam giác*: $d(a, b) \leq d(a, c) + d(c, b) \quad \forall a, b, c \in X$

Ví dụ Cặp (R, d) với $d(x, y) = |x - y|$ là một không gian metric. Không gian hình học 3 chiều với d là khoảng cách hình học thông thường cũng vậy.

Định nghĩa 20 (Không gian metric đầy đủ) Không gian metric X gọi là đầy đủ nếu cho một dãy x_n các phần tử sao cho nếu n, m càng lớn thì x_n và x_m càng gần nhau (tính chất này được gọi là tính chất Cauchy) thì tồn tại một phần tử x trong X sao cho x_n càng ngày càng gần với x (tính chất này gọi là hội tụ về x).

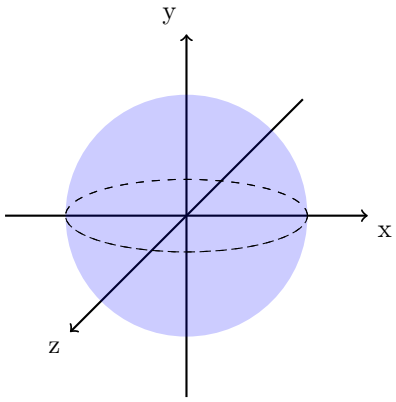
Định lý 6 (Ánh xạ co) Cho không gian metric đầy đủ X. Cho $f : X \to X$. Giả sử tồn tại $0 \leq a < 1$ sao cho với mọi x, y ta có

$$d(f(x), f(y)) \leq a \cdot d(x, y)$$

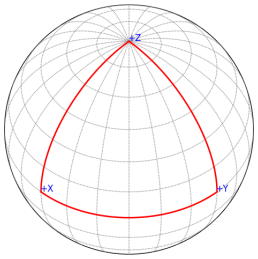
Khi đó tồn tại duy nhất x_0 thỏa mãn $f(x_0) = x_0$, và nếu ta xét dãy x_n như sau: $x_2 = f(x_1)$, $x_3 = f(x_2)$,... thì x_n hội tụ về x.

Hệ quả 5 f thỏa tính chất trong định lý này gọi là ánh xạ co vì khoảng cách giữa các điểm ảnh của f nhỏ hơn khoảng cách giữa các điểm ban đầu, nghĩa là f làm các điểm co lại.

Từ những két quả trên, ta bắt tay vào chứng minh định lý (5) như sau:



Hình 9. Mô phỏng hình cầu $x_1^2 + \ldots + x_n^2 = 1$



Hình 10. Ứng với $n = 3$

Proof Quan sát hình cầu $x_1^2 + \ldots + x_n^2 = 1$ và giao với không gian $\{x_1 \geq 0, \ldots, x_n \geq 0\}$. Trong không gian hai chiều, một hình cầu được giới hạn bởi không gian dương sẽ là một phần của mặt phẳng thứ nhất (góc phần tư thứ nhất). Trong không gian ba chiều, nó sẽ là một phần của khối lập phương đầu tiên (góc tám phần tư đầu tiên). Điều này tạo ra một tập hợp đóng và được giới hạn, ở đây ta sẽ gọi là X.

Tồn tại một ánh xạ T trên X, cho bởi công thức

$$T : X \to X$$
$$v \to T(v) = X\frac{Av}{|Av|}$$

vì các phần tử của ma trận A là không âm. Mà $T \cdot X$ dương nên nó sẽ là một tập con không chứa phần tử nào ở biên của X.

Ta thấy đây là một ánh xạ co, bởi tồn tại $0 < k < 1$ sao cho $d(Tx, Ty) \leq kd(x, y)$, với $d(x, y)$ là khoảng cách giữa 2 điểm x và y trên hình cầu.

Theo định lý (6), do T là một ánh xạ co nên tồn tại một bất động duy nhất là v. Đây cũng chính là vectơ riêng v thỏa $Av = \lambda v$ mà chúng ta đang tìm. Bây giờ ta đã thấy rằng trên X, chỉ có một vectơ riêng như vậy.

Mọi vector riêng khác $Aw = \mu w$ phải có một phần tử tọa độ là số âm. Viết $|w|$ cho vector với các tọa độ $|w_j|$. Phép tính sau

$$|\mu||w_i| = |\mu w_i| = |A_{ij} w_j| \leq |A_{ij}||w_j| = A_{ij}|w_j| = (A|w|)_i$$

chứng tỏ $|\mu|L \leq \lambda L$ vì $(A|w|)$ là một vectơ có độ dài nhỏ hơn λL, với L là độ dài của w.

Vì $|\mu|L \leq \lambda L$ với L khác không, ta có $|\mu| \leq \lambda$. Ta có điều phải chứng minh!

□

CHƯƠNG

III

Liên hệ với mô hình đồ thị

Section 7

Xây dựng

Trước tiên ta có một hướng xây dựng ma trận PageRank dưới dạng mô hình đồ thị dựa trên các lý thuyết về quan hệ sắp xếp và tập hợp, cụ thể là định nghĩa (7) và (8):

Bảng 4. Nội dung của Chương III

Định nghĩa 21 Đặt $G = (V, E)$ là một đồ thị có hướng, $V = \{v_1, v_2, ..., v_n\}$. Ma trận PageRank A_G (kích thước $n \times n$) được định nghĩa như sau:

$$[A_G]_{i,j} = \begin{cases} \frac{1}{|S_G(v_j)|} & \text{nếu} (v_j, v_i) \in E \\ 0 & \text{nếu ngược lại} \end{cases}$$

Đó chỉ là công thức được xây dựng dựa trên nền tảng về lý thuyết đồ thị và tập hợp. Ta có thể định nghĩa đơn giản hơn thông qua ý tưởng chính của thuật toán Pagerank.

Như đã định nghĩa ở phần (I), thuật toán Pagerank hỗ trợ để đánh giá một trang web có tốt hơn trang web khác hay không thông qua việc gán trọng số cho trang web khác. Nó phụ thuộc vào số lượng các trang web có trọng số cao khác liên kết đến và số lượng trang web liên kết tương đối ít với mấy trang web khác.

Định nghĩa 22 Xét đồ thị có hướng $W(V, E)$ với tập đỉnh V là tập hợp của các trang web và tập cạnh E là các hyperlink. Ta có thể biểu diễn đồ thị trên thông qua ma trận kề L với $n = |V|$. Phần tử $L_{ij} = 1$ nếu trang i có liên kết đến trang j và bằng 0 khi không có liên kết đến. Khi đó L được gọi là một ma trận liên kết (`Hyperlink Matrix`).

Đặt $m_i = \sum_{k=1}^{n} L_{ik}$ là tổng số lượng liên kết ra của trang web i. Tất cả các định nghĩa ở trên ta có thể tổng quát hóa nó thành đồ thị trọng số có hướng trong trường hợp các phần tử $L_{ij} \geq 0$. Ma trận chuyển P được định nghĩa là ma trận mà các phần tử $P_{ij} = \frac{L_{ij}}{m_i}$ trong trường hợp $m_i > 0$ và bằng 0 khi $m_i = 0$. Dễ thấy khi $m_i > 0$ thì hàng i chính là ma trận chuyển đổi hàng, tức tổng tất cả các phần tử trong hàng i bằng 1.

Ta xét mô hình du khách ngẫu nhiên. Du khách di chuyển dọc theo các cạnh của đồ thị có hướng. Nếu tại bước thứ k, du khác ở trang i. Khi đó, tại bước thứ $k + 1$ thì xác suất để du khách di chuyển đến những trang mà trang i liên kết đến là như nhau. Rõ ràng, n trang web trên có thể được coi là n trạng thái của xích Markov với ma trận chuyển P. Giả sử, tại bước thứ k ta xét $\pi^{(k)} = (\pi_i^{(k)})$ là các xác suất cho du khách đang ở trang i tại bước thứ k đến các trang khác. Khi đó, xác suất để du khách đi đến trang

j tại bước thứ $k+1$ là:

$$\pi_j^{(k+1)} = \sum_{i \to j} P_{ij}\pi_i^{(k)} = \sum_{i=1}^{n} \frac{L_{ij}}{m_i}\pi_i^{(k)} \text{ hay } \pi^{(k+1)} = \pi^{(k)}P \tag{7.1}$$

(Nếu tất cả các trang đều có liên kết ra thì $\pi_j^{(k+1)}$ chính là phân phối xác suất).Trạng thái cân bằng của biến đổi (7.1) phản ánh chính xác mục tiêu mô hình của chúng ta đang xem xét. (7.1)
Từ công thức (7.1), ta rút ra được:

- π_j ở bước $k+1$ đồng biến với π_i ở bước k: tức là nếu như trang web i là một trang web nổi tiếng và có liên kết đến với j thì sẽ tăng mức ảnh hưởng của j.
- π_j ở bước $k+1$ nghịch biến với m_i: tức là nếu như trang web i có quá nhiều liên kết tới các trang web khác thì uy tín mà i đóng góp cho j càng ít.

Từ đó, ta có định nghĩa sau:

Định nghĩa 23 Vector Pagerank (stationary vector) là vector π tại điểm cân bằng của ma trận chuyển P với các phần tử không âm (là trạng thái cân bằng của xích Markov), tức là:

$$\pi = \pi P \tag{7.2}$$

Từ định nghĩa trên, ta có các vấn đề sau:

1. Nếu ma trận P có một hàng mà tất cả các phần tử đều bằng 0, tức là trang đó không có liên kết với bất kì trang nào. Từ định nghĩa trên, ta rút ra được $m_i = 0$, trong khi đó ma trận chuyển của xích Markov là một ma trận biến cố theo hàng. Do đó, khi có một hàng mà tất cả các phần tử bằng 0 thì đến bước thứ k nào đó thì tất cả các phần tử của ma trận chuyển sẽ bằng 0, nó không khác gì hố đen, nơi mà các vật thể tiến vào thì không thể đi ra. Chính vì vậy, ta phần phải có cách giải quyết và ta sẽ giải quyết nó ở phần (??)
2. Từ công thức (7.2), ta rút ra được ma trận chuyển P có một trị riêng là 1. Nhưng trong thực tế, liệu ma trận chuyển P luôn luôn có trị riêng là 1 và ứng với trị riêng đó sẽ chỉ có duy nhất 1 vector riêng? Để có được điều đó, ma trận A phải thỏa mãn 2 điều kiện sau đây:
   (a) Đồ thị có hướng $W(V, E)$ phải là một đồ thị liên thông mạnh, tức luôn tồn tại đường đi nối từ đỉnh này đến đỉnh khác
   (b) Đồ thị có hướng $W(V, E)$ là một đồ thị không tuần hoàn. Nghĩa là ước chung lớn nhất của tất cả độ dài chu trình của đồ thị W là 1.
   Điều kiện (b) là được bảo toàn trong website thì điều kiện (a) thường vi phạm. Do đó, ta chỉ cần giải quyết vấn đề (a) và ta sẽ giải quyết ở phần

Section 8

Lũy thừa ma trận

Từ công thức (7.1), ta biến đổi như sau: $\pi^{(k+1)} = \pi^{(k)}P = \pi^{k-1}P^2 = \cdots = \pi^{(0)}P^{k+1}$

Từ đó, ta rút ra được công thức sau:

$$\pi^{(k)} = \pi^{(0)}P^k \tag{8.1}$$

Từ công thức (8.1), ta chỉ cần tính được P^k là giải quyết được bài toán. Vậy có những phương pháp nào dùng để lũy thừa ma trận? Qua tìm hiểu, ta có các cách để lũy thừa ma trận như sau:

1. Cách ngây thơ: Ta chạy một vòng lặp và nhân.
2. Cải tiến phép lũy thừa trên bằng phương pháp lũy thừa nhị phân.

Ta cũng có thể cải tiến thuật toán trên bằng việc cải tiến phép nhân 2 ma trận bằng phương pháp Strassen mà ta sẽ đề cập sau.

1. Cải tiến thuật toán nhân 2 ma trận:
1.1. Cách tiếp cận đơn giản:
Ta biết được, để có thể nhân 2 ma trận A và B với nhau và kết quả là ma trận C ($C = AB$). Ta giả sử ma trận A và B có dạng

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \text{ và } B = \begin{bmatrix} e & f \\ g & h \end{bmatrix}$$

Khi đó, nếu ta thực hiện phép nhân theo định nghĩa phép nhân ma trận thì ma trận C sẽ có dạng

$$C = \begin{bmatrix} ae + bg & af + bh \\ ce + dg & cf + dh \end{bmatrix}$$

Từ đó ta có đoạn mã giả sau: Nếu ta gọi $T(n)$ là thời gian để nhân 2 ma trận A với B thì từ phân tích trên, ta thấy ta phải thực hiện phép nhân 2 ma trận $n/2 \times n/2$ 8 lần. Do đó, ta rút ra được công thức sau:

$$T(n) = 8T(n/2) + O(n^2)$$

Từ đó, ta rút ra được $T(n) = 8^{\log(n)} + O(n^2\log(n)) = n^{\log 8} + O(n^2\log(n))$.
Do đó, theo nguyên tắc `big-O`, ta rút ra được độ phức tạp của thuật toán trên là $O(n^{\log 8}) = O(n^3)$

1.2. Phương pháp Stranssen:
Theo như cách tiếp cận trên thì thời gian cần để nhân 2 ma trận vuông là $O(n^3)$. Liệu có tồn tại thuật toán nào tốt hơn để giải quyết vấn đề trên không?
Ta xem xét lại phép nhân 2 ma trận A và B. Nếu ta không đơn thuần dùng định nghĩa

Algorithm 2 Multiply two matrices

```
1:  procedure POWER(A, B)                          ▷ Multiply two matrices algorithm
2:      Input:
3:          A: matrix is a square matrix n × n
4:          B: matrix is a square matrix n × n
5:      Output: C = AB
6:      C ← zero-matrix O_n
7:      row ← n
8:      col ← n
9:      while row > 0 do
10:         while col > 0 do
11:             while n > 0 do
12:                 C[row][col] ← C[row][col] + A[row][n] * B[n][col]
13:                 n ← n − 1
14:             col ← col − 1
15:         row ← row − 1
16:     return C
```

để nhân 2 ma trận trên lại với nhau mà sử dụng một thủ thuật như sau:

$$p_1 = a(f-h) \qquad p_2 = (a+b)h$$
$$p_3 = (c+d)e \qquad p_4 = d(g-e)$$
$$p_5 = (a+d)(e+h) \qquad p_6 = (b-d)(g+h)$$
$$p_7 = (a-c)(e+f)$$

Thì khi đó, ma trận $C = A \cdot B$ sẽ có dạng:

$$C = \begin{bmatrix} p_5 + p_4 - p_2 + p_6 & p_1 + p_2 \\ p_3 + p_4 & p_1 + p_5 - p_3 - p_7 \end{bmatrix}$$

Từ đó, ta có đoạn mã giả sau:

Algorithm 3 Strassen's algorithm

procedure StrassensAlg(A, B) ▷ Strassen Algorithm
  Input:
    A, B: matrix is a square matrix $n \times n$
  Output: $C = AB$
  if $n == 1$ **then**
    return $[A[0] * B[0]]$
  $r \leftarrow 1$
  while $r < n$ **do**
    $r \leftarrow r * 2$
  $a, b, c, d, e, f, g, h \leftarrow$ zero-matrix $r/2 \times r/2$
  for $(i = 0; i < r/2; ++i)$ **do**
    for $(j = 0; j < r/2; ++j)$ **do**
      $a[i][j] = A[i][j]$, $e[i][j] = B[i][j]$
      if $(i + r/2 < n$ and $j + r/2 < n)$ **then**
        $d[i][j] = A[i + r/2][j + r/2]$, $h[i][j] = B[i + r/2][j + r/2]$
      else if $(j + r/2 < n)$ **then**
        $b[i][j] = A[i][j + r/2]$, $f[i][j] = B[i][j + r/2]$
      else if $(i + r/2 < n)$ **then**
        $c[i][j] = A[i + r/2][j]$, $g[i][j] = B[i + r/2][j]$
      else
        $b[i][j] = c[i][j] = d[i][j] = f[i][j] = g[i][j] = h[i][j] = 0$
  $p_1 \leftarrow StrassensAlg(a, f - h)$, $p_2 \leftarrow StrassensAlg(a + b, h)$
  $p_3 \leftarrow StrassensAlg(c + d, e)$, $p_4 \leftarrow StrassensAlg(d, g - e)$
  $p_5 \leftarrow StrassensAlg(a + d, e + h)$, $p_6 \leftarrow StrassensAlg(b - d, g + h)$
  $p_7 \leftarrow StrassensAlg(a - c, e + f)$
  $c_{11} \leftarrow p_5 + p_4 - p_2 + p_6$, $c_{12} \leftarrow p_1 + p_2$
  $c_{21} \leftarrow p_3 + p_4$, $c_{22} \leftarrow p_1 + p_5 - p_3 - p_7$
  for $(i = 0; i < r/2; ++i)$ **do**
    for $(j = 0; j < r/2; ++j)$ **do**
      $C[i][j] = c_{11}[i][j]$
      if $(i + r/2 < n$ and $j + r/2 < n)$ **then**
        $C[i + r/2][j + r/2] = c_{22}[i][j]$
      else if $(j + r/2 < n)$ **then**
        $C[i][j + r/2] = c_{12}[i][j]$
      else if $(i + r/2 < n)$ **then**
        $c[i + r/2][j] = c_{21}[i][j]$
  return C

Nếu ta gọi $T(n)$ là thời gian để nhân 2 ma trận A với B thì từ phân tích trên, ta thấy ta chỉ cần thực hiện phép nhân 2 ma trận $n/2 \times n/2$ 7 lần. Do đó, ta rút ra được công thức sau:

$$T(n) = 7T(n/2) + O(n^2)$$

Từ đó, ta rút ra được $T(n) = 7^{\log(n)} + O(n^2 \log(n)) = n^{\log 7} + O(n^2 \log(n))$.
Do đó, theo nguyên tắc big-O, ta rút ra được độ phức tạp của thuật toán trên là $O(n^{\log 7}) \approx O(n^{2.8074})$

2. Cải tiến thuật toán lũy thừa ma trận:
2.1. Cách tiếp cận ngây thơ:
Đây là cách dễ hiểu và dễ thực hiện nhất. Ta chỉ cần khởi tạo ma trận $A = I_n$ là ma trận đơn vị. Sau đó, ta lần lượt nhân ma trận A với ma trận P k lần thì sẽ thu được kết quả. Ta có đoạn mã giả sau:

Algorithm 4 Power matrix (naive)

```
1:  procedure POWER(P, k)                          ▷ Power matrix algorithm
2:      Input:
3:          P: matrix is a square matrix n × n
4:          k is an exponentiation.
5:      Output: P^k
6:      A ← identity matrix I_n
7:      while k > 0 do
8:          A ← A * P
9:          k ← k − 1
10:     return A
```

Từ đoạn code trên, ta thấy độ phức tạp của thuật toán là $O(k)$ do ta phải đi qua vòng lặp. Và nếu ta dùng cách nhân ma trận của Strassen (1) thì độ phức tạp tổng thể là $O(kn^{\log 7}) \approx O(kn^{2.8074})$, nó sẽ không tối ưu thời gian chạy khi k lớn (từ 1 triệu trở lên). Vậy liệu ta có cách nào để giảm đi độ phức tạp của lũy thừa ma trận không? Câu trả lời là có và đó là phương pháp lũy thừa nhị phân.
2.2. Lũy thừa nhị phân ma trận
Ta có ý tưởng như sau:

$$P^k = \begin{cases} 1, & \text{nếu } k = 0 \\ (P^{\frac{k-1}{2}})^2 \times P, & \text{nếu } k \text{ lẻ} \\ (P^{\frac{k}{2}})^2, & \text{nếu } k \text{ chẵn} \end{cases}$$

Từ công thức trên, ta có hướng tiếp cận là sử dụng đệ quy để giải quyết bài toán, ta có mã giả sau:

Algorithm 5 Power matrix (recursion)

```
1:  procedure POWERRECUR(P, k)                     ▷ recursive exponentiation method
2:      Input:
3:          P: matrix is a square matrix n × n
4:          k is an exponentiation.
5:      Output: P^k
6:      if k == 0 then
7:          return 1
8:      if k lẻ then
9:          A ← powerRecur(P, (k-1)/2)
10:         return A × A × P
11:     A ← powerRecur(P, k/2)
12:     return A × A
```

Từ đây, ta thấy việc tính P^k sẽ thông qua việc tính $P^{[k/2]}$. Chính vì vậy, độ phức tạp của thuật toán trên là $O(\log(k))$.
Nhưng như vậy thì cứ mỗi lần đệ quy, ta lại phải tốn thời gian lẫn bộ nhớ gọi lại hàm. Vậy có cách nào để khắc phục được những hạn chế đó không, hay nói cách khác là có cách nào để ta khử đệ quy được không? Câu trả lời là có.

Từ công thức ta phân tích ở trên, ta có thể nhìn theo 1 hướng khác: Nếu ta phân tích k thành tổng các lũy thừa của 2, tức là $k = \sum 2^i$ thì việc tính P^k sẽ quy về việc tính $\prod P^{2^i}$. Mà P^{2^i} có thể được tính thông qua $P^{2^{i-1}}$ nên tổng hợp lại, ta có đoạn mã giả sau:

Algorithm 6 Power matrix (recursion)

```
1:  procedure powerBin(P, k)                    ▷ Binary exponentiation method
2:      Input:
3:          P: matrix is a square matrix n × n
4:          k is an exponentiation.
5:      Output: P^k
6:      A ← identity matrix I_n
7:      while k > 0 do
8:          if k lẻ then A ← A × P
9:          P ← P × P
10:         k ← [k/2]
11:     return A
```

Từ trên, ta thấy độ phức tạp của thuật toán trên vẫn là $O(\log k)$ nhưng ta đã xử lý được vấn đề thời gian và bộ nhớ khi gọi lại hàm đệ quy nên nó vẫn tốt.
Hơn thế nữa, nếu ta kết hợp với phương pháp nhân ma trận của Strassen (1) thì tổng độ phức tạp của thuật toán trên là $O(n^{\log 7} \log k)$. Có thể sử dụng trong thực tế với k rất lớn (tầm số có 18 chữ số vẫn có thể chạy trong $1s$ với phép lũy thừa số tự nhiên).

CHƯƠNG

IV

Xử lý trường hợp đặc biệt

Bảng 5. Nội dung của Chương IV

Section 9

Một góc nhìn khác về PageRank

Để hiểu được nguồn góc của các trường hợp đặc biệt ta đang nói đến có nguồn gốc từ đâu, xảy ra khi nào thì trước tiên ta sẽ tìm hiểu một góc nhìn khác của thuật toán PageRank.

$$\begin{bmatrix}\pi_1 & \pi_2 & \cdots & \pi_n\end{bmatrix} = \begin{bmatrix}\pi_1 & \pi_2 & \cdots & \pi_n\end{bmatrix} \cdot \begin{bmatrix} m_1 & 0 & \cdots & 0 \\ 0 & m_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & m_n \end{bmatrix}^{-1} \cdot \begin{bmatrix} L_{11} & L_{12} & \cdots & L_{1n} \\ L_{21} & L_{22} & \cdots & L_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ L_{n1} & L_{n2} & \cdots & L_{nn} \end{bmatrix}$$

Ý tưởng thô sơ của công thức trên xuất phát từ đây:

- Đặt $L_{ij} = 1$ nếu như trang web i có liên kết đến trang web j (kí hiệu $i \to j$). Ngược lại thì $L_{ij} = 0$.
- Khi đó $m_i = \sum_{k=1}^{n} L_{ik}$ là tổng số liên kết mà web i trỏ đến các trang web khác.

Ta xét mô hình người lướt web ngẫu nhiên. Những người này di chuyển dọc theo các cạnh của đồ thị có hướng. Nếu tại bước thứ k, họ ở trang i. Khi đó, tại bước thứ $k+1$ thì xác suất để họ di chuyển đến những trang mà trang j liên kết đến là như nhau.

Rõ ràng, n trang web trên có thể được coi là n trạng thái của xích Markov với ma trận chuyển đổi P. Đặt $\pi^{(k)} = \left(\pi_i^{(k)}\right)$ là vector biến cố người lướt đang ở trang i tại bước thứ k. Khi đó, biến cố người dùng tại trang j vào bước thứ $k+1$ là

$$\pi_j^{(k+1)} = \sum_{i=1}^{n} \frac{L_{i,j}}{m_i} \pi_i^{(k)} \tag{9.1}$$

Thoạt nhìn, ta dễ dàng nhận ra công thức (9.1) thỏa mãn những điều kiện căn bản của thuật toán PageRank:

- π_j ở bước $k+1$ đồng biến với π_i ở bước k: tức là nếu như trang web i là một trang web nổi tiếng và có liên kết đến với j thì sẽ tăng mức ảnh hưởng của j.
- π_j ở bước $k+1$ nghịch biến với m_i: tức là nếu như trang web i có quá nhiều liên kết tới các trang web khác thì uy tín mà i đóng góp cho j càng ít.

Từ công thức trên, ta cần tìm π sao cho $\pi = \pi \cdot P = \pi \cdot M^{-1} \cdot L$, với:

$$M = \begin{bmatrix} m_1 & 0 & \cdots & 0 \\ 0 & m_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & m_n \end{bmatrix} \text{ và } L = \begin{bmatrix} L_{11} & L_{12} & \cdots & L_{1n} \\ L_{21} & L_{22} & \cdots & L_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ L_{n1} & L_{n2} & \cdots & L_{nn} \end{bmatrix}$$

Từ $\pi = \pi \cdot P$ ta suy ra $\pi^T = P^T \cdot \pi^T$. Đặt $p = \pi^T, A = P^T$, ta được $p = A \cdot p$

Định nghĩa 24

Qua đó ta rút ra được công thức tính toán PageRank theo dạng ma trận:

$$\underbrace{\begin{bmatrix} p_1^{(k+1)} \\ p_2^{(k+1)} \\ \vdots \\ p_n^{(k+1)} \end{bmatrix}}_{p^{(k+1)}} = \underbrace{\begin{bmatrix} L_{11} & L_{12} & \cdots & L_{1n} \\ L_{21} & L_{22} & \cdots & L_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ L_{n1} & L_{n2} & \cdots & L_{nn} \end{bmatrix}}_{L} \cdot \underbrace{\begin{bmatrix} m_1 & 0 & \cdots & 0 \\ 0 & m_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & m_n \end{bmatrix}^{-1}}_{M^{-1}} \cdot \underbrace{\begin{bmatrix} p_1^{(k)} \\ p_2^{(k)} \\ \vdots \\ p_n^{(k)} \end{bmatrix}}_{p^{(k)}}$$

Trong đó:

- p : kích thước $n \times 1$, lưu trữ uy tín của từng trang web.
- L : kích thước $n \times n$, biểu diễn số lượng liên kết giữa các trang web.
- M : kích thước $n \times n$, tương đương với L nhưng được sắp xếp theo thứ tự của mj.

Đến đây có lẽ bạn đã nhận ra điều gì đó từ công thức $p = A \cdot p$ rồi đúng không? Ở đây chẳng hạn nếu ta thêm phần 'còn thiếu' này vào thì bạn sẽ nhận ra ngay:

$$\lambda \cdot p = A \cdot p$$

Kết luận 2

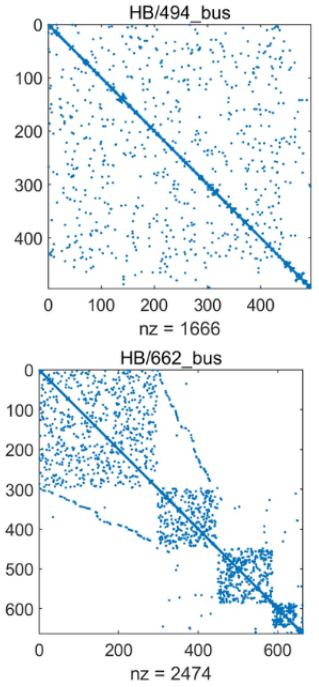
Từ đó ta thấy ngay p chính là vector riêng của A với trị riêng 1, điều này phù hợp với mục tiêu của thuật toán, là tìm một vector p phản ánh uy tín của từng trang web dựa trên số lượng và chất lượng liên kết đến từ các trang web khác.

Trong khi đó, A lại là ma trận thưa vì L là ma trận thưa, một trang web chỉ liên kết với một số ít trang web trên không gian mạng thôi. (Theo thống kê trên Yahoo! vào tháng $8, 2004$ thì chỉ có khoảng 38% trang web là không phải trang treo).

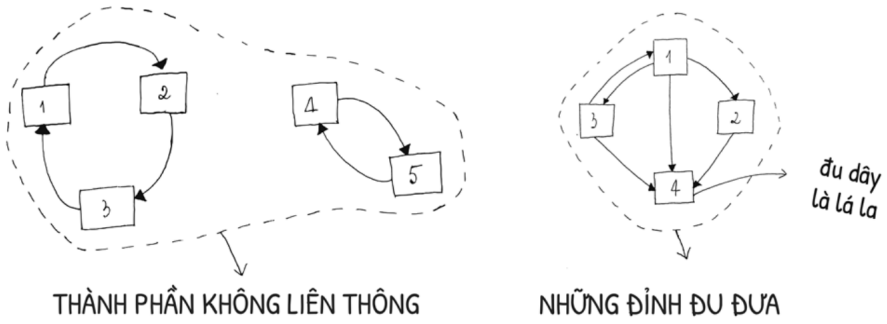
Mặt khác, chúng ta lại biết rằng có một số phương pháp tính hiệu quả dành riêng cho bài toán trị riêng - vector riêng trên ma trận thưa nữa như phương pháp lặp (`Power Method`). Như vậy liệu đã đủ để cho thuật toán hoạt động hiệu quả và tránh được các rủi ro tiềm ẩn chưa?

Câu trả lời là chưa bởi có ít nhất 2 trường hợp của các loại đồ thị mà việc lấy vector riêng của bài toán này không mang lại kết quả chúng ta mong muốn:

- Đồ thị không liên thông: nhiều hơn một vector riêng.
- Đồ thị chứa đỉnh đu đưa/ nút treo: những đỉnh không trỏ liên kết về bất kì trang web nào dù có liên kết trang web khác trỏ tới.



Hình 11. Ma trận thưa thể hiện mối quan hệ giữa số lượng xe buýt và số lượng hành khách trên hai tuyến xe buýt *HB*/494 và *HB*/662.



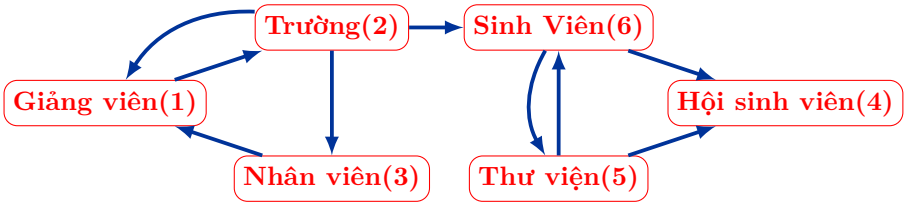
Section 10

Dangling Node

Đầu tiên hãy đến với trường hợp đồ thị chứa đỉnh đu đưa, hay còn goi là nút treo. Đây là những đỉnh không trỏ liên kết về bất kì trang web nào dù có liên kết trang web khác trỏ tới. Điều này gây ra lỗi `Dangling Nodes`.

Định nghĩa 25 Một đỉnh được gọi là đinh treo nếu đỉnh đó không có liên kết trực tiếp với bất kì đỉnh nào khác.

Từ định nghĩa trên, ta rút ra được nếu tràn i được gọi là đỉnh treo thì tổng liên kết ra của trang i sẽ bằng 0 hay $m_i = 0$. Chính vì vậy, hàng thứ i của ma trận P sẽ toàn số 0. Nhưng theo định nghĩa ma trận biến cố hàng thì mỗi hàng trong P có các phần tử không âm và tổng bằng 1. Do đó, ta cần phải xử lý trường hợp này để có thể áp dụng được xích Markov. Ta xét ví dụ một đồ thị trong một trường học như sau:

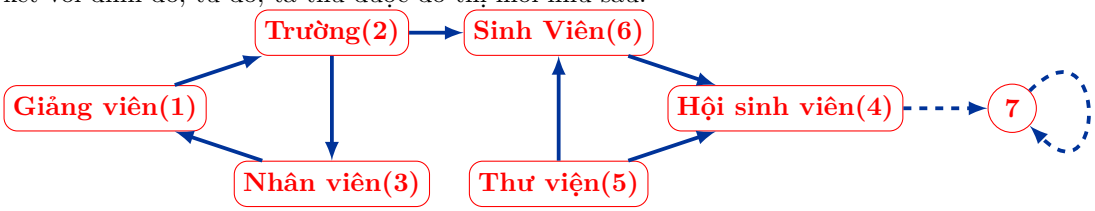


Từ trên, ta thấy nút **Hội sinh viên** không có bất kì liên kết ra nào nên nút đó chính là nút treo (dangling node)
Từ trên khi ta chuyển qua ma trận kề. Ta thu được:

$$L = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 \end{bmatrix} \quad \text{và} \quad P = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ \frac{1}{3} & 0 & \frac{1}{3} & 0 & 0 & \frac{1}{3} \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{2} & 0 & \frac{1}{2} \\ 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 \end{bmatrix}$$

Từ đó, ta thấy được ma trận chuyển P có 1 hàng toàn 0. Chính vì vậy nó không thỏa mãn là ma trận biến cố dòng nên ta không sử dụng xích Markov vào được.
Để xử lý vấn đề trên, Bianchini và Singh đã đề xuất việc thêm 1 đỉnh ảo vô, tất cả các

đỉnh đu đưa trên sẽ liên kết với đỉnh đu đưa đó và chính bản thân đỉnh đó sẽ tự liên kết với đỉnh đó, từ đó, ta thu được đồ thị mới như sau:



Từ đó ta thu được ma trận kề L' và ma trận chuyển P' như sau:

$$L' = \begin{bmatrix} 0&1&0&0&0&0&0 \\ 1&0&1&0&0&1&0 \\ 1&0&0&0&0&0&0 \\ 0&0&0&0&0&0&1 \\ 0&0&0&1&0&1&0 \\ 0&0&0&1&1&0&0 \\ 0&0&0&0&0&0&1 \end{bmatrix} \quad \text{và} \quad P' = \begin{bmatrix} 0&1&0&0&0&0&0 \\ \frac{1}{3}&0&\frac{1}{3}&0&0&\frac{1}{3}&0 \\ 1&0&0&0&0&0&0 \\ 0&0&0&0&0&0&1 \\ 0&0&0&\frac{1}{2}&0&\frac{1}{2}&0 \\ 0&0&0&\frac{1}{2}&\frac{1}{2}&0&0 \\ 0&0&0&0&0&0&1 \end{bmatrix}$$

Vì các hàng của ma trận P có các phần tử không âm và tổng tất cả các phần tử của hàng bằng 1 nên ma trận P chính là ma trận biến cố hàng. Từ đó ta có thể sử dụng xích Markov vào.
Ta có đoạn mã giả sau:

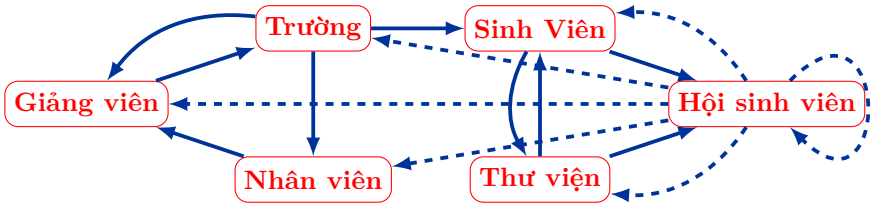
Algorithm 7 Handle Dangling Nodes (Bianchini and Singh)

```
1:  procedure BianchiniSingh(P, k)        ▷ Handle Dangling Nodes (Bianchini and Singh)
2:      Input:
3:          P: matrix is a square matrix n × n
4:      Output: P sau khi đã xử lý dangling nodes
5:      P' ← matrix(n + 1, n + 1)
6:      for (i = 0; i < n; ++i) do
7:          m_i ← 0
8:          for (j = 0; j < n; ++j) do
9:              P'[i][j] ← P[i][j]
10:             m_i ← m_i + P[i][k]
11:         if (m_i == 0) then P'[i][n] ← 1
12:     P'[n][n] ← 1
13:     return P'
```

Ta dễ dàng chứng minh được ma trận P' chính là ma trận biến cố hàng. Thật, vậy, với thuật toán trên, ta đã thêm 1 hàng $(0, 0, \cdots, 0, 1)$ vào cuối và thêm một cột thứ $n+1$ (nếu hàng i biểu diễn nút treo i thì phần tử thứ i sẽ bằng 1 và ngược lại) vào ma trận P. Nhưng cách trên có nhược điểm là ta phải tốn thêm bộ nhớ để lưu nút ảo và các cạnh nối đến nút ảo nên cách trên tỏ ra kém hiệu quả khi gặp n lớn.
Một phương pháp khác được đề ra bởi Langville và Meyer, về mặt ý tưởng thì ta thêm các cạnh ảo của nút treo đến các đỉnh khác và cả chính bản thân nút treo. Nếu diễn giải theo ngôn ngữ toán học thì ta xét vector $d = (d_i)$ với

$$d_i = \delta(m_i, 0) = \begin{cases} 0 & \text{nếu } m_i = 0 \\ 1 & \text{trường hợp còn lại} \end{cases}$$

và vector $v = e_n = (\frac{1}{n}, \frac{1}{n}, \cdots, \frac{1}{n})$. Khi đó, nếu ta đặt $P' = P + d^T.v$ thì tất cả các hàng mà có các phần tử bằng 0 sẽ được thay bằng $\frac{1}{n}$. Khi đó, ta có hướng xử lý khác cho ví dụ bên trên như sau:



Từ trên, ta thu được ma trận kề L' với ma trận chuyển P' như sau:

$$L' = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 \end{bmatrix} \quad \text{và} \quad P' = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ \frac{1}{3} & 0 & \frac{1}{3} & 0 & 0 & \frac{1}{3} \\ 1 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} \\ 0 & 0 & 0 & \frac{1}{2} & 0 & \frac{1}{2} \\ 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 \end{bmatrix}$$

Ta sẽ chứng minh ma trận P' chính là ma trận biến cố hàng. Thật vậy, với các hàng không phải là đỉnh treo thì hiển nhiên các phần tử trong hàng không âm và tổng các phần tử đó bằng 1. Với các hàng là đỉnh treo thì khi ta biến đổi P như trên thành P' thì các hàng đó sẽ thành $v = (\frac{1}{n}, \frac{1}{n}, \cdots, \frac{1}{n})$. Khi đó, do các phần tử trong v là không âm và tổng các phần tử bằng 1 nên ma trận P' chính là ma trận biến cố hàng.
Ta có đoạn mã giả sau:

Algorithm 8 Handle Dangling Nodes (Dangville and Meyer)

```
1:  procedure BianchiniSingh(P, k)        ▷ Handle Dangling Nodes (Dangville and Meyer)
2:      Input:
3:          P: matrix is a square matrix n × n
4:      Output: P sau khi đã xử lý dangling nodes
5:      for (i = 0; i < n; ++i) do
6:          m_i ← 0
7:          for (j = 0; j < n; ++j) do
8:              m_i ← m_i + P[i][j]
9:          if (m_i == 0) then
10:             for (j = 0; j < n; ++j) do
11:                 P[i][j] ← 1/n
12:     return P
```

Section 11

Web Reducible

Từ định lý 1 (ở phần trên) thì một ma trận xác suất chuyển P là egordic nếu và chỉ nếu ma trận P là quasi-positivity. Nhưng rất khó để ta có thể kiểm tra ma trận P trong trường hợp n lớn. Do đó, ta cần định nghĩa lại khái niệm egordic của một ma trận chuyển như sau:
Với 2 trạng thái $i, j \in \mathbb{E}$ bất kỳ, ta gọi trạng thái j được truy cập từ trạng thái i nếu $p_{ij}^{(n)} > 0$ với $n \geq 0$ (ở đây ta định nghĩa $P^0 = I_n$).
Phát biểu trên có thể được diễn đạt lại như sau:
Đặt $r_j = \min\{n \geq 0, X_n = j\}$ là số bước nhỏ nhất để xích Markov đạt đến trạng thái j. Ta định nghĩa $r_j = \infty$ nếu $X_n \neq j, \forall n \geq 0$.
Ta có định lý sau:

Định lý 7 Cho $i \in \mathbb{E}$ thỏa $P(X_0 = i) > 0$. Trạng thái $j \in \mathbb{E}$ có thể được truy cập (accessible) từ trạng thái i nếu và chỉ nếu $P\left(r_j < \infty | X_0 = i\right) > 0$

Proof Điều kiện trên là cần thiết do:

$$\{X_n = j\} \subset \{r_j \leq n\} \subset \{r_j < \infty\} \quad \text{và do đó} \quad 0 < p_{ij}^{(n)} \leq P\left(r_j < \infty | X_0 = i\right)$$

Với $n \geq 0$ nếu j có thể được truy cập (accessible) từ i
Ngược lại, nếu $i \neq j$ và $p_{ij}^{(n)} = 0$ với mọi $n \geq 0$ thì

$$\begin{aligned} P(r_0 < \infty | X_0 = i) &= \lim_{n\to\infty} P\left(r_j < n | X_0 = i\right) \\ &= \lim_{n\to\infty} P\left(\cup_{k=0}^{n-1}\{X_k = j\} | X_0 = i\right) \\ &\leq \lim_{n\to\infty} \sum_{k=0}^{n-1} P(X_k = j | X_0 = i) = \lim_{n\to\infty} \sum_{k=0}^{n-1} p_{ij}^{(k)} = 0 \end{aligned}$$

Vậy ta suy ra điều phải chứng minh □

Từ đó, ta rút ra được các tính chất của `accessibility` là **tính bắc cầu:** Nếu $i \to j$ và $j \to k$ thì $i \to k$ (Ta có thể dựa vào bất đẳng thức $p_{ij}^{(m+n)} = \sum_{k\in\mathbb{E}} \left(p_{ik}^{(m)}\right) p_{kj}^{(n)} \quad \forall i, j \in \mathbb{E}$ (sẽ chứng minh sau) để suy ra). Nhưng nếu $i \to j$ và $j \to i$ thì ta nói i và j có thể truy cập từ 2 chiều. Ký hiệu là $i \leftrightarrow j$

Tính 2 chiều của trạng thái có các quan hệ tương đương sau:

- **Tính phản xạ:** $i \leftrightarrow j$
- **Tính đối xứng:** Nếu $i \leftrightarrow j$ thì $j \leftrightarrow i$
- **Tính bắc cầu:** Nếu $i \leftrightarrow j$ và $j \leftrightarrow k$ thì $i \leftrightarrow k$

Từ đó, ta có các nhận xét sau:

1. Không gian trạng thái $\mathbb{E}$ có thể được phân hoạch thành các lớp tương đương khác nhau theo quan hệ $\leftrightarrow$

2. Xích Markov $\{X_n\}$ với ma trận chuyển $P = (p_{ij})$ được gọi là irreducible nếu không gian trạng thái $\mathbb{E}$ là một lớp tương đương duy nhất. Tức không gian E không thể phân hoạch thành 2 lớp tương đương khác nhau.

Từ định nghĩa (18), ta rút ra được d_i, d_j trùng nhau nếu và chỉ nếu i, j thuộc cùng một phân loại trạng thái. Vì mục đích đó, ta kí hiệu $i \to j[n]$ nếu $p_{ij}^{(n)} > 0$

Ta có định lý sau:

Định lý 8 Nếu 2 trạng thái $i, j \in \mathbb{E}$ thuộc cùng một phân loại trạng thái thì $d_i = d_j$

Proof Từ khai triển $P^{(m+n)} = P^{(n)}P^{(m)}$ ta rút ra được $p_{ij}^{(m+n)} = \sum_{k\in\mathbb{E}} \left(p_{ik}^{(m)}\right) p_{kj}^{(n)} \ \ \forall i, j \in \mathbb{E}$
Từ đó, ta rút ra $p_{ii}^{(m+n)} = \sum_{j\in\mathbb{E}} \left(p_{ij}^{(m)}\right) p_{ji}^{(n)} \geq p_{ij}^{(m)} p_{ji}^{(n)} \ \ \forall i, j \in \mathbb{E}$
Do đó, nếu $j \to j[n], i \to j[k]$ và $j \to i[m]$ với $m, n, k \geq 1$ thì theo bất đẳng thức trên, ta thu được $i \to i[m+k]$ và $i \to i[m+n+k]$. Khi đó, $d_i | m+k$ và $d_i | m+n+k$. Từ đó suy ra $d_i | n$. Từ đó d_i là ước chung của mọi n thỏa $p_{jj}^{(n)} > 0$ hay $d_i \leq d_j$.
Chứng minh tương tự, ta thu được $d_j \leq d_i$. Từ đó ta suy ra điều phải chứng minh. □

Từ trên, ta có hệ quả sau:

Hệ quả 6 Nếu một xích Markov $\{X_n\}$ được gọi là irreducible thì tất cả các trạng thái của $\{X_n\}$ thuộc cùng một lớp trạng thái.

Từ đó ta có cách diễn giải khác như sau: Chuỗi Markov là egordic tương đương với việc chuỗi Markov là irreducible và aperiodic. Để chứng minh điều này, ta cần bổ đề sau:

Bổ đề 1 Cho k là một số tự nhiên bất kỳ. Khi đó luôn tồn tại số tự nhiên $n_0 \geq 1$ sao cho

$$\{n_0, n_0 + 1, n_0 + 2, \cdots\} \subset \{n_1 k + n_2(k + 1); n_1, n_2 \geq 0\}$$

Proof Với $n_0 \geq k^2$ thì luôn tồn tại $m, k \in \mathbb{N}, d < k$ sao cho $n - k^2 = mk + d$. Khi đó $n = (k - d + m)k + d(k + 1)$ thì hiển hiên $n \in \{n_1 k + n_2(k + 1); n_1, n_2 \geq 0\}$. Do đó, ta chỉ cần chọn $n_0 = k^2$ là suy ra điều phải chứng minh □

Định lý 9 Ma trận chuyển P là quasi-positive nếu và chỉ nếu P là irreducible và aperiodic

Proof Nếu ma trận chuyển P là quasi-positive thì ta dễ dàng suy ra ma trận P là irreducible và aperiodic do dựa theo định nghĩa. Do đó ta chỉ cần chứng minh chiều ngược lại là đúng, tức nếu ma trận chuyển P là irreducible và aperiodic thì ma trận P là quasi-positive.
Với ma trận chuyển P là irreducible và aperiodic. Với mỗi $i \in \mathbb{E}$ ta xét $J(i) = \{n \geq 1 : p_{ii}^{(n)} > 0\}$, khi đó $\gcd J(i) = 1$ (do P là aperiodic).
Từ bất đẳng thức mà ta đã chứng minh bên trên:

$$p_{ii}^{(m+n)} \geq p_{ii}^{(m)} p_{ii}^{(n)}$$

Trong ví dụ sau đây,

$$P_1 = \begin{bmatrix} 1/2 & 1/2 \\ 1/2 & 1/2 \end{bmatrix},$$

$$P_2 = \begin{bmatrix} 1/2 & 1/2 \\ 1/4 & 3/4 \end{bmatrix}$$

là hai ma trận irreducible,

$P_3 = \begin{bmatrix} P_1 & 0 \\ 0 & P_2 \end{bmatrix}$ là reducible

Ta rút ra $m+n \in J(i)$ nếu $m, n \in J(i)$ $(*)$
Ta sẽ chứng minh $J(i)$ chứa 2 số nguyên liên tiếp. Thật vậy, nếu không thì khoảng cách giữa cách phần tử trong $J(i)$ nhỏ nhất sẽ là $k \geq 2$. Khi đó sẽ tồn tại vài số dạng $mk+d \in J(i)$ với $m \in \mathbb{N}, d = 1, 2, \cdots, k-1$ vì nếu không thì trong $J(i)$ chỉ toàn số có dạng $n = mk$. Điều này mâu thuẫn thì $\gcd J(i) = 1$
Xét 2 số $n_1, n_1 + n_0 k \in J(i)$. Khi đó, dựa vào tính chất $(*)$ ta rút ra được $a(n_1 + n_0 k) \in J(i)$ và $n + bn_1 \in J(i)$ với $n = mk + d \in J(i); a, b \in \mathbb{N}$.
Ta sẽ chứng minh tồn tại 2 số tự nhiên a, b sao cho khoảng cách giữa $a(n_1+k) \in J(i)$ và $n + bn_1 \in J(i)$ nhỏ hơn k.
Thật vậy, ta chọn $a = b = (m+1)$ và xét hiệu:

$$a(n_1 + k) - (n + bn_1) = (m+1)(n_1 + k) - (mk + d + (m+1)n_1) = k - d < k$$

Từ đó dẫn đến vô lý.
Chính vì vậy trong $J(i)$ tồn tại 2 số nguyên liên tiếp.
Khi đó, áp dụng bổ đề trên, ta thu được luôn tồn tại $n(j) \geq 1$ sao cho

$$\{n(j), n(j)+1, \cdots\} \subset J(i)$$

Từ khai triển $p_{ij}^{(m+n)} = \sum_{k \in \mathbb{E}} \left(p_{ik}^{(m)}\right) p_{kj}^{(n)}$ $\forall i, j \in \mathbb{E}$, ta rút ra $p_{ij}^{(m+n)} \geq p_{ik}^{m} p^{kj}$ và từ đó ta suy ra $p_{ij}^{(r+n+m)} \geq p_{ik}^{(r)} p_{kk}^{(n)} p_{kj}^{(m)}$
Từ các kết quả trên, kết hợp với ma trận P là irreducible ta rút ra được
Với mọi trạng thái $i, j \in \mathbb{E}$ thì luôn tồn tại $n(ij) \geq 1$ sao cho

$$J(ij) = \{n \geq 0 : p_{ij}^{(n)} > 0\} \supset \{n(ij), n(ij)+1, \ldots\}$$

Từ đó, ta suy ra được ma trận P là quasi-positive □

Từ đây, ta rút ra được một ma trận chuyển P có trạng thái dừng thì ma trận P phải là quasi-positive. Vậy có thuật toán nào để ta có thể xử lý điều trên? Để ma trận chuyển P là quasi-positive thì ta có một cách xử lý như sau:

$$P'' = \alpha P' + (1-\alpha)E, \quad E = v_e^T.[1, 1, \cdots, 1] \quad \text{với} \quad v_e \quad \text{là vector biến cố hàng} \quad (11.1)$$

Từ trên, ta rút ra được các hàng của ma trận E trùng với vector biến cố hàng v_e. Ta sẽ chứng minh ma trận P'' ở (11.1) là ma trận biến cố hàng.

Proof Với mọi $j \in \mathbb{E}$, ta có:

$$\begin{aligned} \sum_{i \in \mathbb{E}} v''_{ji} &= \alpha \sum_{i \in \mathbb{E}} v'_{ji} + (1-\alpha) \sum_{i \in \mathbb{E}} v_{ei} \\ &= \alpha + (1-\alpha) = 1 \end{aligned}$$

Từ đó ta suy ra điều phải chứng minh □

Với cách đặt trên, ta dễ dàng rút ra được ma trận P'' chính là quasi-positive dựa theo định nghĩa. Do đó áp dụng định lý (1) và (9), ta rút ra được ma trận P'' chính là một xích Markov do đó nó có điểm dừng.
Xét (11.1), hệ số α chính là hệ số Dampling. Nó có ý nghĩa là xác suất để người dùng ở trang i sẽ lướt tiếp đến một trang nào đó. Do đó, với trang không treo thì người lướt web sẽ theo một trong các trang mà trang i liên kết với xác suất α và nhảy đến một trang $j \in G$ với xác suất $(1-\alpha)v_{ej}$. Chính vì vậy v_e được biết đến như là vector teleportation. Trong thực tế, người ta thường dùng $v_e = (\frac{1}{n}, \frac{1}{n}, \cdots, \frac{1}{n})$ và $\alpha \in [0.85, 0.9]$.

Từ trên, ta có đoạn mã giả sau:

Algorithm 9 Handle Quasi-Positive

```
1: procedure QuasiPositive(P, k)                    ▷ Handle Quasi-Positive
2:     Input:
3:         P: matrix is a square matrix n × n
4:         α: Dampling Factor
5:     Output: P sau khi đã xử lý quasi-positive
6:     P′ ← square matrix n × n
7:     for (i = 0; i < n; ++i) do
8:         for (j = 0; j < n; ++j) do
9:             P′[i][j] ← α × P[i][j] + (1 − α) × 1/n
10:    return P′
```

CHƯƠNG

V

Ma trận Google

Section 12

Ảnh hưởng của Damping factor

Nhắc lại 1 Như đã nói ở trên, yếu tố 'damping' biểu thị tỷ lệ thời gian mà người lướt web ngẫu nhiên đi theo các liên kết trang thay vì nhảy tới các trang khác một cách ngẫu nhiên. Do đó, yếu tố này đóng một vai trò quan trọng trong việc tính toán PageRank bằng phương pháp lặp (`Power Method`) đã được nhắc lại ở dưới, và tính bởi công thức (14.1) sẽ được nhắc tới ở phần 14:

$$\pi = \pi \cdot \left(\alpha \cdot M^{-1} \cdot L + \frac{1-\alpha}{n} E \right)$$

Bảng 6. Nội dung của Chương V

Algorithm 10 Power matrix (recursion)

```
1:  procedure powerBin(P, k)                    ▷ Binary exponentiation method
2:      Input:
3:          P: matrix is a square matrix n × n
4:          k is an exponentiation.
5:      Output: P^k
6:      A ← identity matrix I_n
7:      while k > 0 do
8:          if k lẻ then A ← A × P
9:          P ← P × P
10:         k ← [k/2]
11:     return A
```

Nhiều nhà nghiên cứu đã khẳng định rằng giá trị của yếu tố 'damping' ảnh hưởng đến tốc độ hội tụ của phương pháp lặp. Để làm rõ ván đề này, xin được giới thiệu một thí nghiệm của nhóm tác giả Atul Kumar Srivastava để quan sát tác động của yếu tố 'damping' đối với tốc độ hội tụ của thuật toán PageRank, cũng như sự sắp xếp của các trang web được hiển thị trên công cụ tìm kiếm web trên các tập dữ liệu khác nhau!

Tập dữ liệu	Số trang web	Số trang treo
YouTube dataset (D1)	1157827	783042 (0.67%)
Road network dataset (D2)	1971281	6075 (0.30%)
Wikitalk dataset (D3)	2394385	2246783 (93%)

Bảng 7. Các tập dữ liệu và các thuộc tính tương ứng

Thông tin các bộ dữ liệu ta sẽ nghiên cứu nằm ở bảng 7 ở trên. Nhóm nghiên cứu đã áp dụng phương pháp lặp trên mỗi tập dữ liệu cho đến khi hội tụ và quan sát tốc độ hội tụ với các giá trị yếu tố 'damping' khác nhau. Hình 2 dưới đây cho thấy số lần lặp

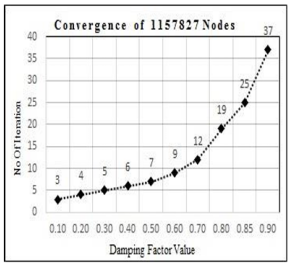
cần thiết để phương pháp PageRank hội tụ ở các giá trị hệ số damping khác nhau. Biểu đồ thể hiện rằng khi giá trị hệ số damping tăng, số lần lặp cần thiết cho phương pháp PageRank hội tụ cũng tăng với giá trị ngưỡng chấp nhận.

Từ các biểu đồ ở hình 14, chúng ta có thể thấy rằng có sự thay đổi nhỏ trong số lần lặp khi giá trị yếu tố 'damping' là $0 < \alpha < 0.65$, nhưng đồ thị lại tăng nhanh khi giá trị yếu tố 'damping' là $\alpha > 0.65$ và hướng đến 1.

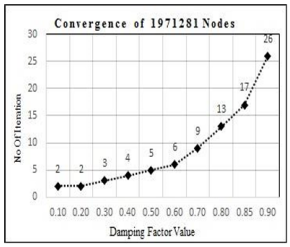
Các nghiên cứu trước đây đã quan sát rằng giá trị của yếu tố 'damping' không chỉ kiểm soát sự hội tụ của thuật toán PageRank mà còn ảnh hưởng đến thứ tự của các trang web được trả về bởi nó. Như chúng ta có thể thấy trong hình 14 , $\alpha = 0.7$ chỉ cần 12 lần lặp để hội tụ khi áp dụng cho $1,157,827$ nút của đồ thị. Tuy nhiên, với số lượng lớn dữ liệu, lựa chọn $\alpha = 0.85$ vẫn cần quá nhiều thời gian để hội tụ. Trước đây, Brin và Page đã chọn giá trị của yếu tố 'damping' là $\alpha = 0.85$ vì giá trị lớn của hệ số này tăng cường ảnh hưởng của cấu trúc liên kết thực sự của web trong khi các giá trị nhỏ hơn của hệ số này tăng ảnh hưởng của vectơ xác suất nhân tạo nhiều hơn là xác suất thực tế.

Tuy nhiên, người lướt web ngẫu nhiên hiện nay không tuân theo cấu trúc thường thấy mà có xu hướng chuyển tới các trang web khác do lượng dữ liệu không liên quan trên web họ đang ở là rất lớn. Vì vậy, giá trị của yếu tố 'damping' $\alpha \in (0.7, 0.8)$ phản ánh tốt hơn hành vi của người lướt web, và cũng cần ít lần lặp hơn để phương pháp hội tụ. Giá trị yếu tố 'damping' $\alpha - 0.7$ sẽ tăng trọng số của vector điều hướng nhân tạo, điều này hữu ích trong PageRank cá nhân hóa, công cụ tìm kiếm dựa trên miền và nhiều ứng dụng khác nơi mà vector điều hướng nhân tạo đóng một vai trò quan trọng.

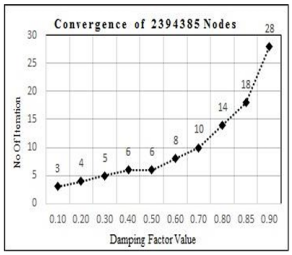
Hình 14 hiển thị số lần lặp cần thiết để thuật toán PageRank hội tụ trên các tập dữ liệu khác nhau với giá trị ngưỡng chấp nhận là $\epsilon = 10^{-7}$. Số lần lặp cần thiết để thuật toán PageRank hội tụ không hoàn toàn phụ thuộc vào kích thước tập dữ liệu, nó cũng phụ thuộc vào một số thuộc tính của tập dữ liệu như số lượng nút treo và sự kết nối của các nút (liên quan đến tính liên thông của đồ thị). Như chúng ta có thể thấy từ Hình 15 và Bảng 7 cho tập dữ liệu $D3$, thuật toán PageRank chỉ cần ít lần lặp hơn để hội tụ so với $D1$.



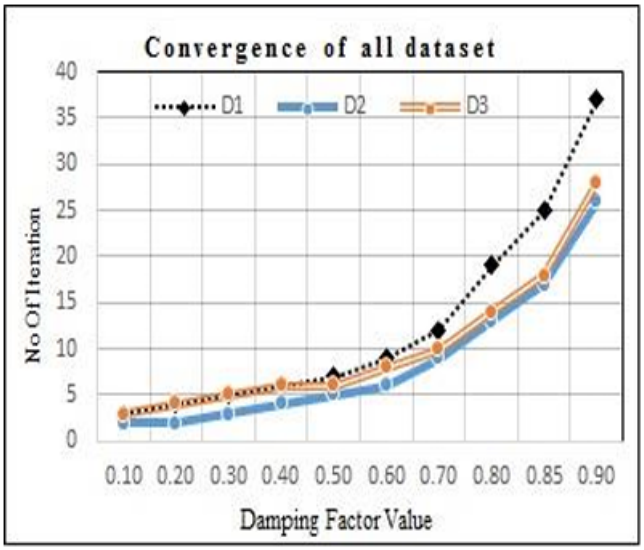
Hình 12. Figure 2*a*.



Hình 13. Figure 2*b*.



Hình 14. Figure 2*c*.

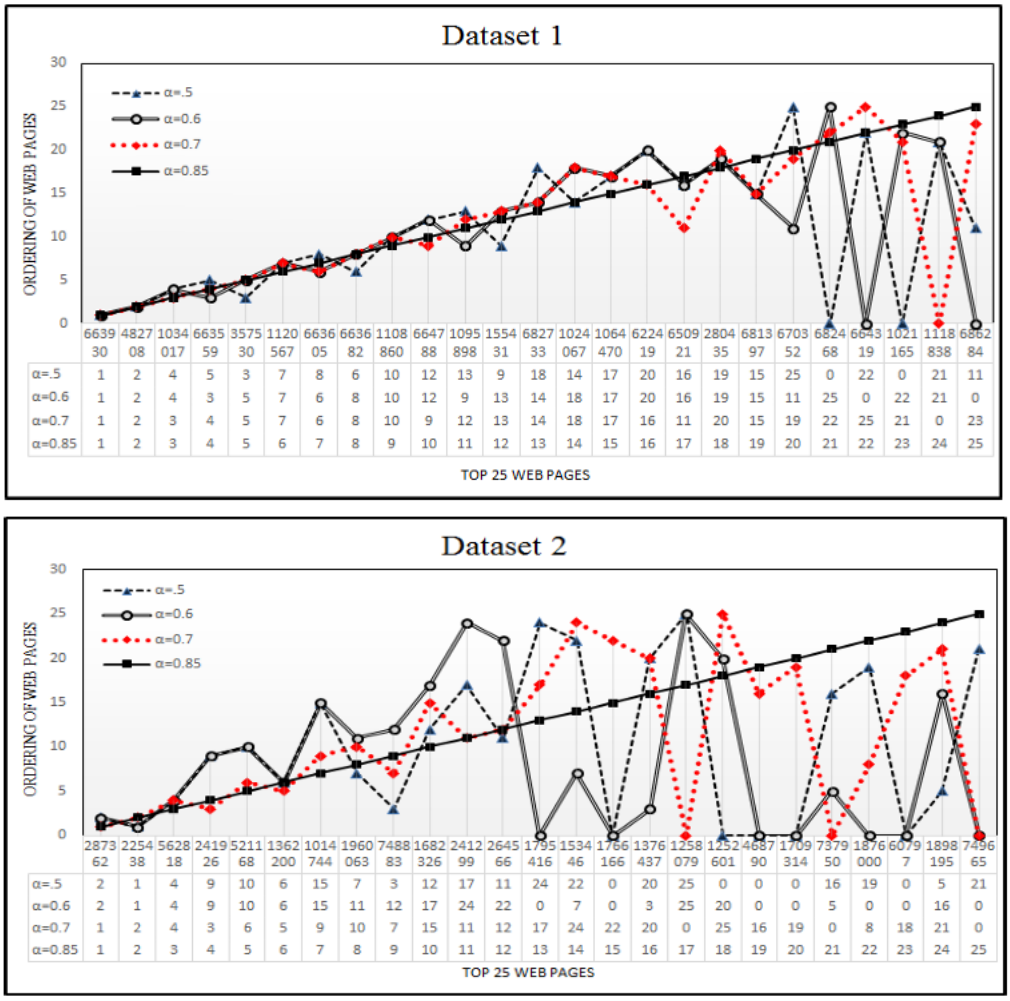


Hình 15. Figure 2d

Trong thí nghiệm này, nhóm nghiên cứu chỉ lấy ra 25 trang web hàng đầu được trả về bởi thuật toán PageRank vì **SERP** (trang kết quả của công cụ tìm kiếm) chứa gần 20 đến 30 trang web trong trang kết quả đầu tiên mà có liên quan hơn đối với người lướt web, và trung bình 90% người dùng không đi xa hơn trang kết quả đầu tiên cho bất kỳ truy vấn cụ thể nào. Nếu bất kỳ trang web nào được liệt kê vào 1 trong 25 vị trí hàng đầu cho $\alpha = 0.85$ nhưng không thể có được một vị trí cho $top25$ của bảng cho $\alpha = 0.7$ thì nhóm nghiên cứu đã lấy một tham số để chỉ ra xem trang web có được đề cập trong 25 vị trí hàng đầu hay không. Tham số được định nghĩa như sau:

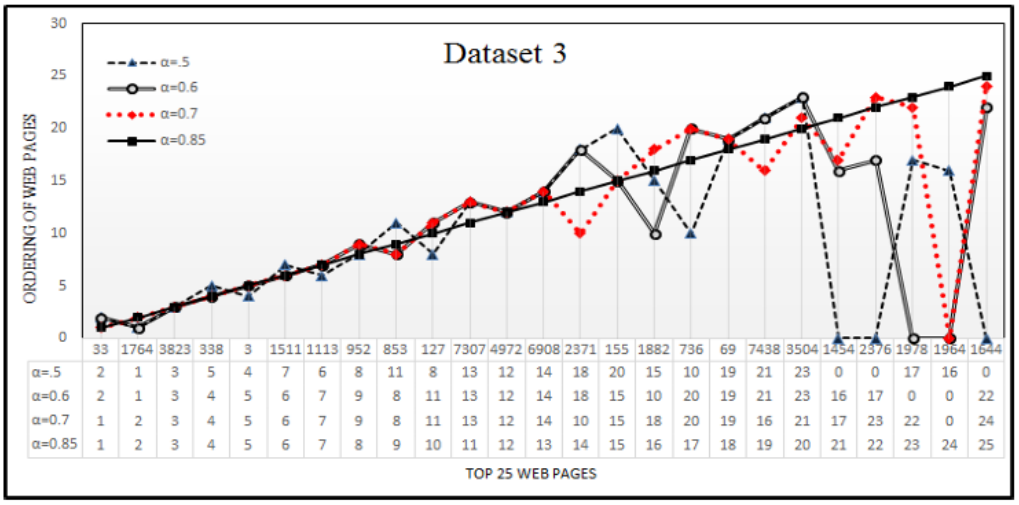
$$\lambda = \begin{cases} 0 & \text{nếu trang web không thuộc top25} \\ 1 \leqslant \lambda \leqslant 25 & \text{nếu ngược lại} \end{cases}$$

Ví dụ, trong hình $(3a)$, trang web 1454 đạt vị trí thứ 21 cho $\alpha = 0.85$ nhưng không thể đạt vị trí hàng đầu cho $\alpha = 0.7$, vì vậy gán giá trị cho nó là $\lambda = 0$. Trong Hình 3, đường thẳng chỉ ra thứ tự của các trang web cho hệ số damping . Trong Hình $3(a, b, c)$, chúng ta có thể thấy rằng các trang web cho $\alpha = 0.7$ có thứ tự gần như nhau.



Hình 16. ($P1$) Hình thể hiện sự so sánh tương đối về số thứ tự của 25 trang web hàng đầu cho các giá trị yếu tố 'damping' khác nhau trên các tập dữ liệu $D1$, $D2$, $D3$ tương ứng.

Trong hình $(3a)$, chỉ có một trang web không thể có tên trong 25 vị trí hàng đầu, đó là trang 1964. Trong hình $(3b)$, có 3 trang web không có tên trong 25 vị trí hàng



	33	1764	3823	338	3	1511	1113	952	853	127	7307	4972	6908	2371	155	1882	736	69	7438	3504	1454	2576	1978	1984	1644
α=.5	2	1	3	5	4	7	6	8	11	8	13	12	14	18	20	15	10	19	21	23	0	0	17	16	0
α=0.6	2	1	3	4	5	6	7	9	8	11	13	12	14	18	15	10	20	19	21	23	16	17	0	0	22
α=0.7	1	2	3	4	5	6	7	9	8	11	13	12	14	10	15	18	20	19	16	21	17	23	22	0	24
α=0.85	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

Hình 17. ($P2$) Hình thể hiện sự so sánh tương đối về số thứ tự của 25 trang web hàng đầu cho các giá trị yếu tố 'damping' khác nhau trên các tập dữ liệu $D1$, $D2$, $D3$ tương ứng.

đầu là trang 1259079, 737950, 749665. Và trong hình (3c), chỉ có một trang web không thể lọt vào trong 25 vị trí hàng đầu là trang 1118838 cho yếu tố 'damping' đối với các tập dữ liệu $D1$, $D2$, $D3$ tương ứng.

Kết luận 3

Nói tóm lại, có những lý do sau khiến Google đã lựa chọn giá trị cho yếu tố 'damping' là $\alpha = 0.85$:

1. Kiểm soát sự hội tụ: yếu tố 'damping' ảnh hưởng đến tốc độ hội tụ của thuật toán PageRank qua phương pháp lặp. Giá trị 0.85 được Brin và Page chọn có thể là một điểm cân bằng hợp lý giữa việc đảm bảo sự hội tụ của thuật toán và giảm thiểu thời gian tính toán.

2. Mang tính mô phỏng thực tế của hành vi người lướt web tốt hơn: Ngày nay, người dùng web ít có xu hướng tuân theo cấu trúc liên kết mà các trang web đặt ra và thường nhảy tới trang khác nhiều hơn. Do đó, yếu tố 'damping' $\alpha < 1$ (như 0.85) có thể phản ánh chính xác hơn hành vi này.

3. Chênh lệch với các giá trị α khác là thấp: Qua thí nghiệm được mô tả ở hình 16 và 17, dù có sự thay đổi về thứ tự, kết quả trả về của các trang web gần như giống nhau dù có sự thay đổi về giá trị của yếu tố 'damping'.

Những điều kể trên cho thấy việc sử dụng yếu tố 'damping' $\alpha = 0.85$ so với các giá trị khác có thể cung cấp kết quả gần đúng nhưng trong thời gian tính toán ngắn hơn, và với tính mô phỏng hành vi người lướt web thực tế tốt hơn.

Section 13

Phương án của Google

Chúng ta đã được giới thiệu các phương pháp giải quyết tình trạng `Dangling Node`, `Web Reducible` cũng như cách giải quyết đồ thị W liên thông mạnh. Mặt khác là biết được thêm thông tin: giải quyết Dangling Node là bước tiền xử lý quan trọng cho việc tính toán PageRank. Liệu đây có phải là những gì nằm trong tính toán của Google?

Trên thực tế thì đúng là vậy, những người 'cha đẻ' của PageRank từ khi xây dựng thuật toán đã tính toán được có khả năng xảy ra trường hợp đem lại kết quả không mong muốn khi tính toán, sẽ dẫn đến những rắc rối mà sau này chúng ta gọi là `Dangling Node` hay `Web Reducible`. Sau đây xin được giới thiệu phương án giải quyết của `Larry Page` và `Sergey Brin`.

Ta sẽ sử dụng yếu tố 'damping' làm yếu tố then chốt, chẳng hạn ở đây ta lấy $d = 0.85$. Ý tưởng chính là sau mỗi lần cập nhật PageRank, tất cả các nút giữ lại 85% của PageRank đã cập nhật của họ, và tổng cộng 15% PageRank còn lại được chia đều cho tất cả các nút. Điều này có nghĩa là mỗi nút nhận được một giá trị chia sẻ là $\frac{0.15}{n}$, với n là số lượng nút. Giá trị này được cộng thêm vào 85% PageRank mà mỗi nút đã giữ lại. Phương pháp này giúp ngăn chặn các nút hội tụ tới giá trị 0 hoặc 1 - điều mà ta không mong muốn!

Page và Brin đã xây dựng một 'ma trận Google' dựa trên ý tưởng vừa rồi, cụ thể là:

Định nghĩa 26

Ma trận Google được cho bởi công thức

$$T = \alpha \cdot P + (1 - \alpha) \cdot E \tag{13.1}$$

trong đó:

- T: Ma trận Google hay ma trận chuyển đổi đã điều chỉnh
- α: Yếu tố 'damping'
- P: Ma trận chuyển đổi ban đầu
- E: Ma trận các phần tử hàng đều bằng $\frac{1}{n}$, với n là tổng số trang web trong mạng lưới

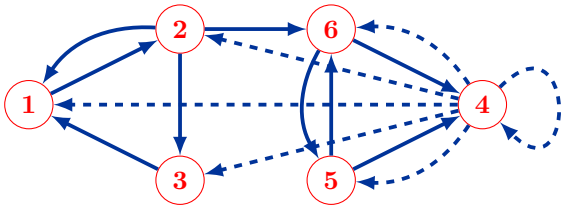
Google tin rằng ma trận mới này có xu hướng mô phỏng tốt hơn người lướt web thực tế. Trong đời thực, một người lướt web sẽ có xác suất $1-\alpha$ là nhảy đến một trang ngẫu nhiên trên Web, tức là bằng cách gõ URL vào dòng lệnh của trình duyệt, và xác suất α là nhấp vào liên kết tiếp theo trên trang hiện tại.

Và việc sử dụng ma trận E với các phần tử từ hàng bằng $\frac{1}{n}$ giúp đảm bảo rằng mọi trang web đều có cơ hội nhận PageRank từ người dùng ngẫu nhiên, giúp thuật toán PageRank phân phối PageRank một cách đồng đều hơn trong mạng lưới web.

Đến đây bạn có lẽ đã nhận ra việc thêm vào thành phần $(1 - \alpha) \cdot E$ sẽ đảm bảo cho không có hàng nào của T - ma trận chuyển đổi sau cùng, bằng 0.

Vấn đề là Google đã không chỉ dừng lại ở đây! Họ muốn phân phối PageRank một cách đồng đều hơn trong mạng lưới, đồng thời cải thiện chất lượng và ổn định của kết quả PageRank. Nếu bạn chưa quên, `Langville` đã giới thiệu phương án thêm cạnh ảo vào đồ thị. Trên thực tế thì `Langville` đã phản ánh đúng một bước quan trọng trong khâu xử lý của Google!

Trở lại với ví dụ ở phần (10), khi xử lý trường hợp `Dangling Node`, ta đã có phương án thêm cạnh ảo theo `Langville` như sau đây:



Ứng với đồ thị ta có ma trận chuyển đổi ở hình 18. Một ma trận chuyển đổi P không còn dòng nào bằng 0 như vậy chính là một ma trận đầu vào sẽ được Google sử dụng để tính toán!

$$P = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ \frac{1}{3} & 0 & \frac{1}{3} & 0 & 0 & \frac{1}{3} \\ 1 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} \\ 0 & 0 & 0 & \frac{1}{2} & 0 & \frac{1}{2} \\ 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 \end{bmatrix}$$

Hình 18. Ma trận chuyển đổi P ứng với đồ thị bên

Sau đây ta sẽ lý giải tại sao lại chọn P là một ma trận không còn phần tử 0. Hãy cùng quan sát lại định lý sau đây:

Định lý 10 Nếu ma trận xác suất P là ma trận dương, tức là các phần tử của P đều dương, thì P có duy nhất một phân phối dừng (duy nhất một vector riêng tương ứng trị riêng 1). Tất cả trị riêng còn lại đều có giá trị tuyệt đối nhỏ hơn 1.

Bạn có thấy quen không? Ta rất nhanh có thể nhận ra đây là một phiên bản khác của định lý 5, chỉ khác là định lý này áp dụng cho ma trận xác suất P và chỉ ra rằng trị riêng lớn nhất duy nhất của P bằng 1. Đồng thời tất cả trị riêng còn lại đều có giá trị tuyệt đối nhỏ hơn 1.

Ta đã có chứng minh chi tiết cho định lý 5 ở trên, và để hoàn thiện lời giải cho định lý 10, ta sẽ cùng đến với các bổ đề sau:

Bổ đề 2 Trị riêng lớn nhất của một ma trận xác suất là 1.

Proof Bổ đề này đúng với cả ma trận xác suất hàng hay cột. Ta sẽ chứng minh với ma trận xác suất hàng, trường hợp còn lại chứng minh tương tự.

Đầu tiên, nếu A là một ma trận xác suất hàng, thì

$$A \cdot \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} = \begin{bmatrix} \sum_{j=1}^{n} a_{1j} \\ \sum_{j=1}^{n} a_{2j} \\ \vdots \\ \sum_{j=1}^{n} a_{mj} \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}$$

vì tổng các phần tử mỗi hàng của A bằng 1. Điều này chứng minh rằng 1 là một trị riêng của A.

Mặt khác, giả sử tồn tại $\lambda > 1$ và vector x khác 0 sao cho $Ax = \lambda x$. Gọi

x_i là phần tử lớn nhất của x. Vì bất kỳ bội vô hướng (`scalar multiple`) của x cũng sẽ thỏa mãn phương trình này, không mất tính tổng quát, ta giả sử $x_i > 0$. Vì các hàng của A không âm và tổng bằng 1, mỗi phần tử trong λx là một tổ hợp lồi (`convex combination`) của các phần tử của x. Do đó, không có phần tử nào trong λx có thể lớn hơn x_i. Với $\lambda > 1$, ta lại có $\lambda x_i > x_i$ (mâu thuẫn)

Vậy giả thiết tồn tại $\lambda > 1$ là sai. Do đó, trị riêng lớn nhất của A là 1. Ta có điều phải chứng minh!

□

Bổ đề 3 Giả sử A là một ma trận dương với trị riêng lớn nhất $\lambda_{max} = 1$. Khi đó tất cả các trị riêng khác của A đều thỏa mãn $\lambda < 1$.

Proof Giả sử z là vectơ riêng của A với giá trị riêng λ với $|\lambda| = 1$. Khi đó $|z| = |xz| = |Az| \leq |A||z| = A|z| \Rightarrow |z| \leq A|z|$. Đặt $y = A|z| - |z|$, ta có $y \geq 0$.

Giả sử $y \neq 0$ thì $Ay > 0$ và $A|z| > 0$ nên tồn tại $\epsilon > 0$ sao cho $Ay > \epsilon \cdot Az$ và do đó $A(A|z| - z) > \epsilon Az$ hay $B(A|z|) > A|z|$, với $B = \frac{1}{1+\epsilon}A$

Suy ra $B^k \cdot A|z| > A|z|$ với mọi k. Nhưng các trị riêng của B đều có giá trị tuyệt đối nhỏ hơn 1, vì vậy $B^k \to 0$ với k đủ lớn. Do đó, tất cả phần tử của $A|z|$ đều ≤ 0. Điều này mâu thuẫn với $A|z| > 0$. Vậy z là vectơ riêng của A với trị riêng 1.

Mà ta lại có $|Az| = |z|$ hay $|Az| = A|z|$ chỉ xảy ra nếu tất cả các phần tử của z cùng dấu. Vì vậy z phải là một bội của vector riêng x bởi không có vector riêng nào khác có tất cả các phần tử cùng dấu ngoại trừ bội của x. Vì vậy $\lambda = 1$, ta có điều phải chứng minh! □

Nói tóm lại, từ việc chứng minh 2 bổ đề trên, cùng với chứng minh sẵn có ở phần trước của định lys 5, ta đã hoàn tất chứng minh định lys 10.

Ta sẽ áp dụng định lý 10 vừa được chứng minh cho ma trận chuyển đổi P là một ma trận xác suất. Khi đó ta suy ra P chỉ có lưu ý 2, ta nhận thấy P chỉ có duy nhất một phân phối dừng, đồng nghĩa với có duy nhất một vector riêng tương ứng trị riêng 1.

Việc P là ma trận dương ảnh hưởng đến số vector riêng có thể có ứng với trị riêng 1, từ đó quyết định tồn tại phân phối dừng hay không. Nhìn lại lưu ý 2, ta có thể thấy nếu P chỉ không âm thì có thể sẽ tồn tại hơn 1 vector riêng cho trị riêng 1, đây là điều ta không mong muốn khi tính toán chỉ số PageRank!

Như vậy ta có thể thấy việc xử lý các trường hợp đặc biệt như `Dangling Nodes` hay `Web Reducible` cũng đóng vai trò quan trọng trong khâu xử lý của Google khi những thao tác này giúp 'dọn dẹp' các phần tử 0 để ma trận chuyển đổi P trở thành một ma trận dương. Từ đó theo định lý 10, P mới tồn tại duy nhất một phân phối dừng để hội tụ về một trạng thái tĩnh nơi mà vector PageRank sẽ không thay đổi đáng kể trong các lần lặp từ sau đó trở đi.

Để cho sự hội tụ ta vừa đề cập được ổn định và chỉ số PageRank được phân phối đồng đều trên mạng lưới thì công thức 13.1 của Google chính là chìa khóa, thông qua việc sử dụng yếu tố 'damping' hợp lý.

Ở đây ta sẽ áp dụng công thức 13.1 để thử tính toán chỉ số PageRank cho các nút

của đồ thị 13:

$$T = \alpha \cdot \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ \frac{1}{3} & 0 & \frac{1}{3} & 0 & 0 & \frac{1}{3} \\ 1 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} \\ 0 & 0 & 0 & \frac{1}{2} & 0 & \frac{1}{2} \\ 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 \end{bmatrix} + (1-\alpha) \cdot \begin{bmatrix} \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} \\ \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} \\ \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} \\ \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} \\ \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} \\ \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} \end{bmatrix}$$

Với $\alpha = 0.85$, sau cùng ta thu được ma trận Google sau:

$$T = \begin{bmatrix} 0.025 & 0.85 & 0.025 & 0.025 & 0.025 & 0.025 \\ 0.475 & 0.025 & 0.475 & 0.025 & 0.025 & 0.475 \\ 0.85 & 0.025 & 0.025 & 0.025 & 0.025 & 0.025 \\ 0.275 & 0.275 & 0.275 & 0.275 & 0.275 & 0.275 \\ 0.025 & 0.025 & 0.025 & 0.425 & 0.025 & 0.425 \\ 0.025 & 0.025 & 0.025 & 0.425 & 0.425 & 0.025 \end{bmatrix}$$

Sau đó ma trận T này được sử dụng như một công cụ quan trọng để tính toán PageRank của từng trang web trong mạng lưới. Nó phản ánh cách mà người dùng tiềm ẩn sẽ di chuyển từ một trang web đến trang web khác, bao gồm cả việc nhấp chuột ngẫu nhiên trên toàn bộ mạng lưới web.

Qua một số lần lặp lại quy trình như vậy, ma trận chuyển đổi sẽ tiến đến một trạng thái tĩnh tức là vector PageRank sẽ không thay đổi đáng kể sau một số lần lặp nhất định. Điều này được gọi là hội tụ của thuật toán PageRank!

Khi đó ta tìm được một vector riêng của ma trận chuyển đổi sau cùng, ứng với trị riêng 1, chẳng hạn là s như bên dưới:

$$s = \begin{bmatrix} 0.043 & 0.034 & 0.047 & 0.041 & 0.044 & 0.045 \end{bmatrix}$$

Đây chính là chỉ số PageRank ta vừa tìm được cho hệ thống web trường Đại học Khoa học Tự nhiên, thông qua việc xử lý `Dangling Node`!

Section 14

Thuật toán

Để khép lại vấn đề về những trường hợp đặc biệt trong tính toán PageRank cũng như phương án của Google, chúng ta sẽ theo dõi những phát biểu sau cùng, quan sát thuật toán và đưa ra kết luận.

Nhắc lại 2 Đây là công thức xây dựng Ma trận của Google:

$$T = \alpha \cdot P + (1-\alpha) \cdot E$$

và đây là công thức dạng ma trận của bài toán PageRank :

$$\begin{bmatrix}\pi_1 & \pi_2 & \cdots & \pi_n\end{bmatrix} = \begin{bmatrix}\pi_1 & \pi_2 & \cdots & \pi_n\end{bmatrix} \cdot \begin{bmatrix} m_1 & 0 & \cdots & 0 \\ 0 & m_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & m_n \end{bmatrix}^{-1} \cdot \begin{bmatrix} L_{11} & L_{12} & \cdots & L_{1n} \\ L_{21} & L_{22} & \cdots & L_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ L_{n1} & L_{n2} & \cdots & L_{nn} \end{bmatrix}$$

Từ đây ta đúc kết được một phát biểu về thuật toán PageRank như sau:

Bài toán 1 Tìm vector p thỏa:

$$\pi = \pi \cdot \left(\alpha \cdot M^{-1} \cdot L + \frac{1-\alpha}{n} E \right) \tag{14.1}$$

trong đó E là ma trận $n \times n$ mà các phần tử đều bằng 1, n là số trang web. L mà M như đã giới thiệu ở phần 9.

Nếu bạn chú ý kỹ sẽ thấy những trường hợp như đồ thị không liên thông hay tồn tại những đỉnh treo là trường hợp đặc biệt nơi mà các ma trận chuyển đổi có chứa phần tử 0, thậm chí là thành ma trận thưa. Ma trận E được thêm vào đã giải quyết vấn đề có phân phối dừng hay không!

Sau đây sẽ là mã giả (`pseudo code`) và mô tả giải thuật để tính toán PageRank dựa trên công thức trên:

Algorithm 11 PageRank Algorithm

```
procedure PAGERANK(A, ϵ, α)                    ▷ PageRank evaluation algorithm
    Input: A: matrix with N rows and N columns
           ϵ: float - tolerance level
           α: float - damping factor
    Output: p: eigenvector with N rows and 1 column
    N ← length of A
    p ← random vector of size N × 1
    last_p ← random vector of size N × 1
    G ← (1 − α)/N× ones matrix of size N × N + α × A
    while ||p − last_p||_2 > ϵ do
        last_p ← p
        p ← G × p
    return p
```

Mô tả Thuật toán PageRank

1. **Đầu vào:** Thuật toán nhận vào 3 đối số:
- A: ma trận có N hàng và N cột, biểu diễn cấu trúc liên kết giữa các trang web.
- ϵ: ngưỡng dung sai, quyết định đến khi nào thuật toán dừng.
- α: hệ số giảm, xác định mức độ quan trọng của trang web hiện tại so với trang web khác.

2. **Khởi tạo:** Thuật toán bắt đầu bằng việc tính độ dài N của ma trận A và tạo ra hai vector ngẫu nhiên p và $last\_p$ có kích thước $N \times 1$.

3. **Tạo ma trận chuyển trạng thái:** Tạo ma trận G có kích thước $N \times N$ bằng cách thực hiện phép nhân giữa một ma trận toàn số một có kích thước $N \times N$ với hệ số $(1 - \alpha)/N$ và ma trận A với hệ số α.

4. **Lặp:** Trong khi khoảng cách Euclid giữa p và $last\_p$ lớn hơn ϵ, thực hiện các bước sau:
- Cập nhật $last\_p$ thành giá trị của p.
- Cập nhật p bằng cách nhân ma trận G với p.

5. **Đầu ra:** Kết quả cuối cùng là vector p, biểu diễn PageRank của các trang web. Thuật toán dừng lại khi hai vector p liên tiếp không thay đổi đến mức độ cho phép ϵ, và trả về vector p là kết quả cuối cùng của PageRank.

CHƯƠNG

VI

Tổng kết - Tài liệu tham khảo

Section 15

Hiện tại và tương lai của PageRank

Google loại bỏ PageRank khỏi thanh công cụ của mình vào năm 2016, sau khi không cập nhật giá trị PageRank cho công chúng trong nhiều năm. Có một số lý do chính có thể kể đến như sau:

1. Google muốn ngăn chặn các hành vi SEO xấu, như mua bán liên kết, spam liên kết, hay thao túng liên kết để tăng Page Rank nhanh chóng cho trang web của họ. Những hành vi này làm giảm chất lượng của kết quả tìm kiếm và trải nghiệm người dùng.

2. Google muốn khuyến khích các web-master tập trung vào việc tạo ra nội dung chất lượng và mang lại giá trị cho người dùng, thay vì chỉ quan tâm đến chỉ số PageRank. PageRank chỉ là một trong nhiều yếu tố trong thuật toán xếp hạng của Google và nó không phản ánh đầy đủ sức mạnh của một trang web.

3. Google muốn bảo mật thuật toán xếp hạng của mình, và không muốn để lộ giá trị PageRank cho đối thủ cạnh tranh hoặc các công cụ SEO khác. PageRank là một phần quan trọng của bí quyết thành công của Google, và Google không muốn bị sao chép hoặc lợi dụng.

Dựa vào sự loại bỏ của Google PageRank khỏi thanh công cụ vào năm 2016 và các lý do đi kèm, có thể dự đoán rằng tương lai của PageRank sẽ tiếp tục trải qua các thay đổi và điều chỉnh. Mặc dù không còn được Google sử dụng như một chỉ số chính thức để xác định xếp hạng trang web, nhưng vẫn có những vai trò và ý nghĩa trong cộng đồng SEO và tiếp thị trực tuyến.

Một trong những hướng phát triển tiềm năng cho PageRank là sự phát triển của các thuật toán và phương pháp mới để đánh giá và xác định sự uy tín và chất lượng của các trang web. Các công cụ tìm kiếm có thể phát triển các yếu tố mới hoặc cải tiến các yếu tố hiện có để thay thế PageRank và cung cấp kết quả tìm kiếm chính xác hơn và chất lượng hơn cho người dùng.

Ngoài ra, PageRank vẫn có thể được sử dụng như một công cụ đo lường tương đối để so sánh sức mạnh và ảnh hưởng của các trang web trong một lĩnh vực cụ thể. Các nhà nghiên cứu và nhà phát triển web có thể tiếp tục sử dụng PageRank trong các nghiên cứu và phát triển các công cụ và ứng dụng mới.

Tóm lại, tương lai của PageRank có thể đi theo hướng điều chỉnh và thay đổi, nhưng vẫn có vai trò và ý nghĩa trong cộng đồng SEO và tiếp thị trực tuyến, cũng như trong nghiên cứu và phát triển công nghệ web.

Section 16

Tài liệu tham khảo

I. Giới thiệu bài toán PageRank

[1] A Survey on PageRank Computing
[2] PageRank: The Trillion Dollar Algorithm

II. Cơ sở lý thuyết

[3] Markov Chains and Monte-Carlo Simulation
[4] Math 19b, Linear Algebra, Probability and Statistics, Spring, 2011
[5] Stationary distributions
[6] Properties of Relations, LibreTexts Mathematics
[7] Algebraic and geometric multiplicity of eigenvalues, StatLect
[8] Lecture12 ThePerron-Frobeniustheorem, ShlomoSternberg

III. Liên hệ với mô hình đồ thị

[9] Ranking Systems: The PageRank Axioms - Alon Altman, Moshe Tennenholtz
[10] A Survey on PageRank Computing
[11] Divide and Conquer | Set 5 (Strassen's Matrix Multiplication)
[12] Lũy thừa nhị phân

IV. Xử lý trường hợp đặc biệt

[13] The effects of dangling nodes on citation networks, Erjia Yan and Ying Ding, School of Library and Information Science, Indiana University, Bloomington, USA
[14] Markov Chains and Monte-Carlo Simulation

V. Ma trận Google

[15] Discussion on Damping Factor Value in PageRank Computation, Atul Kumar Srivastava & Co, I.J. Intelligent Systems and Applications, 2017, 9, 19-28
[16] Damping factor in Google page ranking, Hwai-Hui Fu*,y, Dennis K. J. Lin and Hsien-Tang Tsai, Appl. Stochastic Models Bus. Ind., 2006; 22:431–444

VI. Tổng kết - Tài liệu tham khảo