PageRank

Оглавление

Вступление

PageRank («пэйдж-ранк», «ранг страницы») — один из алгоритмов ссылочного ранжирования. Алгоритм применяется к коллекции документов, связанных ссылками (таких, как веб-страницы), и назначает каждому из них некоторое численное значение, измеряющее его «важность» или «авторитетность» среди остальных документов. Вообще говоря, алгоритм может применяться не только к веб-страницам, но и к любому набору объектов, связанных между собой взаимными ссылками, то есть к любому **графу**.

В 1996 году Сергей Брин и Ларри Пейдж, тогда ещё аспиранты Стэнфордского университета, начали работу над исследовательским проектом BackRub — поисковой системой по Интернету, использующей новую тогда идею о том, что веб-страница должна считаться тем «важнее», чем больше на неё ссылается других страниц, и чем более «важными», в свою очередь,

являются эти страницы. Через некоторое время BackRub была переименована в Google.

В 1998 году Google был одной из первых поисковых систем, внедривших ссылочное ранжирование, благодаря чему добился значительного улучшения качества поиска по сравнению с конкурентами. В дальнейшем многие крупные поисковые системы разработали и внедрили свои аналоги PageRank и другие методы статического (то есть запросо-независимого) ранжирования документов.

Описание проблемы

Представьте себе библиотеку, содержащую 25 миллиардов документов, но без централизованной организации и без библиотекарей. Кроме того, в каждый в любой момент может добавить документ, никому не сообщая об этом. Представьте, что в одном из документов, содержащихся в библиотеке, есть нужная вам информация и вы хотели бы найти ее за считанные секунды. Как бы вы это сделали?

В такой постановке задача кажется неразрешимой. Тем не менее, описание данной библиотеки не слишком отличается от Всемирной паутины, огромной, крайне неорганизованной коллекции документов в самых разных форматах. Поскольку все знакомы с поисковыми системами, очевидно, что данная задача решена. Ниже будет описан алгоритм ***Google PageRank*** и то, как он возвращает страницы из Интернета, состоящего из 25 миллиардов документов, которые настолько хорошо соответствуют критериям поиска, что слово «google» стало широко используемым глаголом.

Каждый раз, когда пользователь запрашивает веб-поиск с помощью поисковой фразы, такой как «поисковая система», поисковая система определяет все страницы в Интернете, которые содержат слова в поисковой фразе. Примерно 95% текста на веб-страницах состоит всего из 10 000 слов. Это означает, что для большинства поисковых запросов будет огромное количество страниц, содержащих слова в поисковой фразе. Следовательно, нам необходимо, средство ранжирования важности страниц, которые

соответствуют критериям поиска, чтобы страницы можно было отсортировать так, что наиболее важные страницы находились в верхней части списка. Один из способов определить важность страниц — использовать ранжирование, созданное людьми. Например, вы могли видеть страницы, которые в основном состоят из большого количества ссылок на другие ресурсы в определенной области интересов. Предполагая, что лицо, поддерживающее эту страницу, является надежным, упомянутые страницы, вероятно, будут полезными. Конечно, список может быстро устареть, и человек, поддерживающий список, может пропустить некоторые важные страницы либо непреднамеренно, либо в результате неустановленной предвзятости. Алгоритм ***Google PageRank*** оценивает важность веб-страниц без человеческой оценки содержания. Как мы увидим, хитрость заключается в том, чтобы попросить саму сеть ранжировать важность страниц.

Как понять, кто важен

Если вы когда-либо создавали веб-страницу, вы, вероятно, добавляли ссылки на другие страницы, содержащие ценную и достоверную информацию. Тем самым вы подтверждаете важность страниц, на которые ссылаетесь. Алгоритм Google PageRank проводит ежемесячный конкурс популярности среди всех страниц в Интернете, чтобы определить, какие страницы являются наиболее важными. Фундаментальная идея, выдвинутая создателями PageRank Сергеем Брином и Лоуренсом Пейджем, заключается в следующем: о важности страницы судят по количеству страниц, ссылающихся на нее, а также по их важности.

Мы назначим каждой веб-странице $\boldsymbol{P}$ меру ее важности $\boldsymbol{I(P)}$, называемую **PageRank** страницы.

Предположим, что страница $\boldsymbol{P_j}$ имеет $\boldsymbol{l_j}$ ссылок. Если одна из этих ссылок ведет на страницу $\boldsymbol{P_i}$, то $\boldsymbol{P_j}$ передаст $\boldsymbol{P_i}$ - $\frac{\mathbf{1}}{\boldsymbol{l_j}}$ своей важности. Тогда рейтинг важности $\boldsymbol{P_i}$ представляет собой сумму всех вкладов, сделанных

страницами, ссылающимися на нее. То есть, если обозначить множество страниц, ссылающихся на $\boldsymbol{P_i}$, через $\boldsymbol{B_i}$, то:

$$\boldsymbol{I(P_i)} = \sum_{\boldsymbol{P_j \in B_i}} \frac{\boldsymbol{I(P_j)}}{\boldsymbol{l_j}}$$

Это может напомнить о вопросе, что появилось раньше курица и яйцо: чтобы определить важность страницы, нам сначала нужно узнать важность всех страниц, ссылающихся на нее. Однако мы можем переформулировать задачу, сделав ее более знакомой с математической точки зрения.

Давайте сначала создадим матрицу, называемую матрицей гиперссылок, $H = H_{ij}$, в которой элемент в i-й строке и j-м столбце равен:

$$\boldsymbol{H_{ij}} = \begin{cases} \boldsymbol{1/l_j}, & \boldsymbol{P_j \in B_i} \\ \boldsymbol{0}, & \text{иначе} \end{cases}$$

Обратите внимание, что $\boldsymbol{H}$ обладает некоторыми особыми свойствами. Во-первых, все элементы матрицы $\boldsymbol{H}$ неотрицательны. Кроме того, сумма элементов в столбце равна единице (за исключением столбцов, которые соответствуют страницам, не имеющим ссылок). Матрицы, в которых все элементы неотрицательны и сумма элементов в каждом столбце равна единице, называются ***стохастическими***.

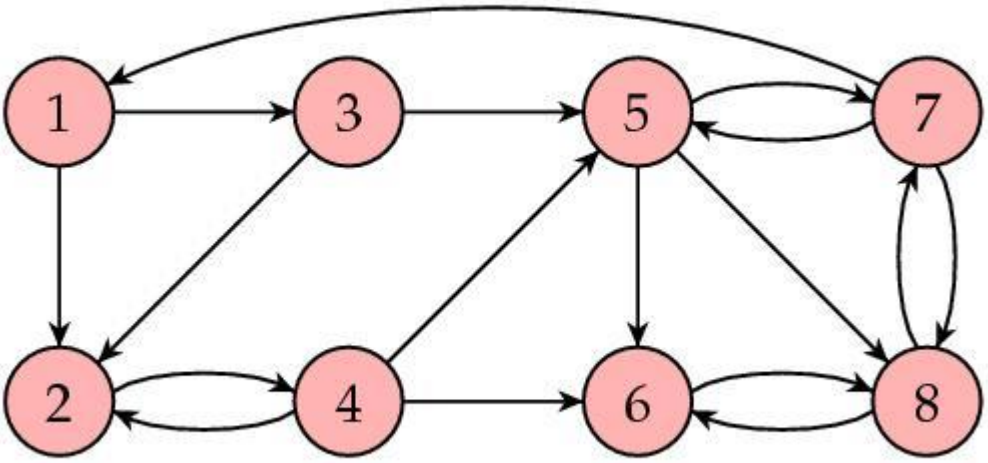
Сформируем вектор $\boldsymbol{I} = [\boldsymbol{I(P_i)}]$, компонентами которого являются PageRanks, т. е. рейтинги важности, всех страниц.

Условие выше, определяющее PageRank, может быть выражено следующим образом:

$$\boldsymbol{I} = \boldsymbol{H} \cdot \boldsymbol{I}$$

Другими словами, вектор $\boldsymbol{I}$ является собственным вектором матрицы $\boldsymbol{H}$ с собственным значением 1 или стационарным вектором матрицы $\boldsymbol{H}$.

Давайте посмотрим на пример. Ниже приведен небольшой граф из восьми веб-страниц со ссылками, представленными стрелками.



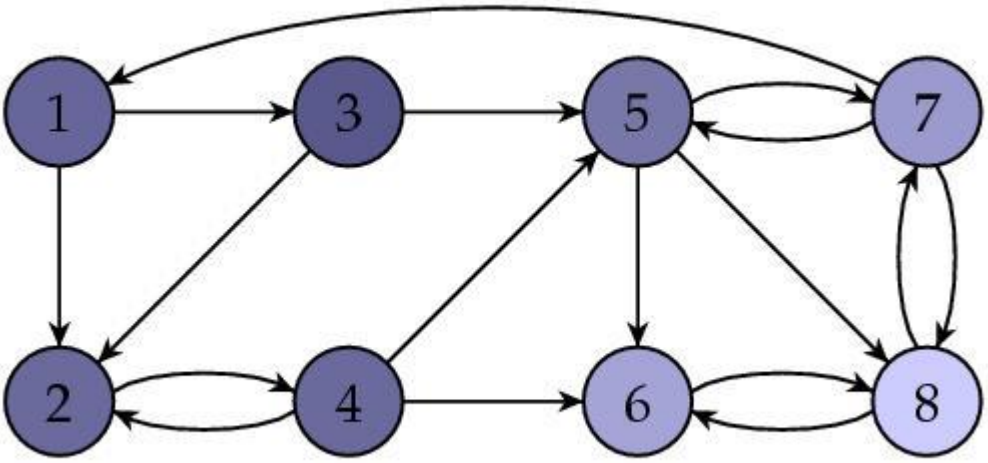
Соответствующая матрица выглядит следующим образом:

$$\mathbf{H} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1/3 & 0 \\ 1/2 & 0 & 1/2 & 1/3 & 0 & 0 & 0 & 0 \\ 1/2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1/2 & 1/3 & 0 & 0 & 1/3 & 0 \\ 0 & 0 & 0 & 1/3 & 1/3 & 0 & 0 & 1/2 \\ 0 & 0 & 0 & 0 & 1/3 & 0 & 0 & 1/2 \\ 0 & 0 & 0 & 0 & 1/3 & 1 & 1/3 & 0 \end{bmatrix}$$

Со стационарным вектором:

$$I = \begin{bmatrix} 0.0600 \\ 0.0675 \\ 0.0300 \\ 0.0675 \\ 0.0975 \\ 0.2025 \\ 0.1800 \\ 0.2950 \end{bmatrix}$$

Данный вектор показывает, что страница 8 побеждает в конкурсе популярности. Вот тот же рисунок, но веб-страницы заштрихованы таким образом, что страницы с более высоким PageRank светлее.



Вычисление стационарного вектора *I*

Существует несколько способов найти собственные векторы квадратной матрицы. Но стоит помнить, что $\boldsymbol{H}$ имеет около $n = 25$ миллиардов столбцов и строк. Однако большинство записей в $\boldsymbol{H}$ равны нулю; фактически, исследования показывают, что веб-страницы имеют в среднем около 10 ссылок, а это означает, что в среднем все, кроме 10 записей в каждом столбце, равны нулю. Мы выберем метод, известный как степенной метод для нахождения стационарного вектора $\boldsymbol{I}$ матрицы $\boldsymbol{H}$.

Суть степенного метода, следующая: выбираем вектор $\boldsymbol{I^0}$ в качестве кандидата на $\boldsymbol{I}$ и строим последовательность векторов $\boldsymbol{I^k}$, по формуле:

$$\boldsymbol{I^{k+1} = H \cdot I^k}$$

Метод основан на предположении, что последовательность $\boldsymbol{I^k}$сходится к стационарному вектору $\boldsymbol{I}$.

Проиллюстрируем на примере выше:

I^0	I^1	I^2	I^3	I^4	...	I^{60}	I^{61}
1	0	0	0	0.0278	...	0.06	0.06
0	0.5	0.25	0.1667	0.0833	...	0.0675	0.0675
0	0.5	0	0	0	...	0.03	0.03
0	0	0.5	0.25	0.1667	...	0.0675	0.0675

0	0	0.25	0.1667	0.1111	...	0.0975	0.0975
0	0	0	0.25	0.1806	...	0.2025	0.2025
0	0	0	0.0833	0.0972	...	0.18	0.18
0	0	0	0.0833	0.3333	...	0.295	0.295

Естественно задаться вопросом, что означают эти цифры. Конечно, не может быть абсолютной меры важности страницы, только относительные меры для сравнения важности двух страниц с помощью таких утверждений, как «Страница А в два раза важнее страницы Б». По этой причине мы можем умножить все рейтинги важности на некоторую фиксированную величину, не затрагивая информацию, которую они нам сообщают. Таким образом, мы всегда будем предполагать, по причинам, которые мы вскоре объясним, что сумма всех «популярностей» равна единице.

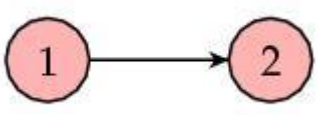
Три важных вопроса

На ум естественно приходят три вопроса:

1. Всегда ли последовательность $\boldsymbol{I^k}$ сходится?
2. Не зависит ли вектор, к которому он сходится, от начального вектора $\boldsymbol{I^0}$?
3. Содержат ли рейтинги важности нужную нам информацию?

Учитывая текущий метод, ответ на все три вопроса — «Нет!». Однако мы увидим, как изменить метод, чтобы мы могли ответить «да» на каждый из них.

Давайте сначала рассмотрим очень простой пример: сеть, состоящая из двух веб-страниц, одна из которых ссылается на другую:



с матрицей $H = \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix}$

Вот один из способов, которым может работать текущий алгоритм:

I^0	I^1	I^2	$I^3=I$
1	0	0	0

0	1	0	0

В этом случае рейтинг важности обеих страниц равен нулю, что ничего не говорит нам об относительной важности этих страниц. Проблема в том, что у $\boldsymbol{P_2}$ нет ссылок. Следовательно, на каждом итеративном шаге он берет часть важности со страницы $\boldsymbol{P_1}$, но не передает ее какой-либо другой странице. Это приводит к удалению всей важности из Интернета. Страницы без ссылок называются ***висячими узлами***, и, конечно же, в реальной сети, которую мы хотим изучить, их много.

Надо придумать, как с ними обращаться, для этого давайте рассмотрим новый способ мышления о матрице $\boldsymbol{H}$ и стационарном векторе $\boldsymbol{I}$.

Вероятностная интерпретация матрицы гиперссылок *H*

Представьте, что мы бродим по Интернету наугад; то есть, когда мы оказываемся на веб-странице, мы случайным образом переходим по одной из ее ссылок на другую страницу через некоторый фиксированный промежуток времени. Например, если мы находимся на странице $\boldsymbol{P_j}$ с $\boldsymbol{l_j}$ ссылками, одна из которых ведет на страницу $\boldsymbol{P_i}$, вероятность того, что в следующий раз мы окажемся на странице $\boldsymbol{P_i}$, равна $\boldsymbol{1/l_j}$.

Поскольку мы просматриваем страницы случайным образом, мы будем обозначать через $\boldsymbol{T_j}$ долю времени, которое мы тратим на страницу $\boldsymbol{P_j}$. Тогда доля времени, в течение которого мы оказываемся на странице $\boldsymbol{P_i}$, исходящей от $\boldsymbol{P_j}$, составляет $\boldsymbol{T_j/l_j}$. Если мы окажемся на $\boldsymbol{P_i}$, значит, мы пришли со страницы, ссылающейся на неё. Это значит, что:

$$\boldsymbol{T_i = \sum_{P_j \in B_i} \frac{T_j}{l_j}}$$

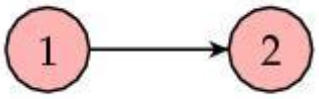
где сумма указана по всем страницам $\boldsymbol{P_j}$, ссылающимся на $\boldsymbol{P_i}$. Обратите внимание, что это то же самое уравнение, определяющее значение PageRank, поэтому $\boldsymbol{I(P_i) = T_i}$. Это позволяет интерпретировать PageRank веб-страницы как долю времени, которое случайный пользователь проводит на этой веб-

странице. Например, если вы когда-либо занимались поиском информации по теме, с которой не были знакомы: после того как вы какое-то время переходите по ссылкам, вы обнаружите, что возвращаетесь на одни страницы чаще, чем на другие. Также, как и «Все дороги ведут в Рим», вы обычно попадаете на более ценные страницы.

Обратите внимание, что при такой интерпретации естественно требовать, чтобы сумма элементов в векторе PageRank $\boldsymbol{I}$ была равна единице.

В этом описании есть сложность: если мы просматриваем сайт случайным образом, в какой-то момент мы обязательно застрянем на зависшем узле, странице без ссылок. Чтобы продолжить, мы выберем следующую страницу случайным образом; то есть мы делаем вид, что висячий узел имеет ссылку на любую другую страницу. Это приводит к изменению матрицы гиперссылок $\boldsymbol{H}$ путем замены столбца нулей, соответствующего висячему узлу, столбцом, в котором каждая запись равна $\boldsymbol{1/n}$. Назовем эту новую матрицу $\boldsymbol{S}$.

В предыдущем примере теперь имеем:



с матрицей $\mathbf{S} = \begin{bmatrix} 0 & 1/2 \\ 1 & 1/2 \end{bmatrix}$

и собственным вектором: $I = \begin{bmatrix} 1/3 \\ 2/3 \end{bmatrix}$

Другими словами, страница $\boldsymbol{P_2}$ в два раза важнее страницы $\boldsymbol{P_1}$, что может показаться логичным.

Матрица $\boldsymbol{S}$ обладает тем приятным свойством, что все ее элементы неотрицательны, а сумма элементов в каждом столбце равна единице. То есть $\boldsymbol{S}$ - стохастическая. Стохастические матрицы обладают несколькими свойствами, которые нам пригодятся. Например, стохастические матрицы всегда имеют стационарные векторы. Для дальнейших целей заметим, что $\boldsymbol{S}$ получается из $\boldsymbol{H}$ простым способом. Если $\boldsymbol{A}$ — матрица, все элементы которой

равны нулю, за исключением столбцов, соответствующих висячим узлам, в которых каждый элемент равен $\mathbf{1/n}$, то $\boldsymbol{S = H + A}$.

Как работает степенной метод

В общем, степенной метод — это метод нахождения собственного вектора квадратной матрицы, соответствующего собственному значению с наибольшей величиной. В нашем случае мы ищем собственный вектор $\boldsymbol{S}$, соответствующий собственному значению 1. В лучшем случае, который будет описан позже, другие собственные значения $\boldsymbol{S}$ будут иметь величину меньше единицы; то есть $|\lambda| < 1$, если $|\lambda|$ является собственным значением $\boldsymbol{S}$, отличным от 1.

Предположим, что собственные значения $\boldsymbol{S}$ равны $|\lambda_j|$ и что:

$$1 = \lambda_1 > |\lambda_2| \geq |\lambda_3| \geq \cdots \geq |\lambda_n|$$

Также предположим, что существует базис v_j собственных векторов для $\boldsymbol{S}$ с соответствующими собственными значениями λ_j. Это предположение не обязательно верно, но с его помощью легче проиллюстрировать, как работает степенной метод. Мы можем записать наш начальный вектор $\boldsymbol{I^0}$ следующим образом:

$$\boldsymbol{I^0} = c_1 v_1 + c_2 v_2 + \cdots + c_n v_n$$

Тогда:

$$\boldsymbol{I^1} = \boldsymbol{S} \cdot \boldsymbol{I^0} = c_1 v_1 + c_2 \lambda_2 v_2 + \cdots + c_n \lambda_n v_n$$

$$\boldsymbol{I^2} = \boldsymbol{S} \cdot \boldsymbol{I^1} = c_1 v_1 + c_2 \lambda_2^2 v_2 + \cdots + c_n \lambda_n^2 v_n$$

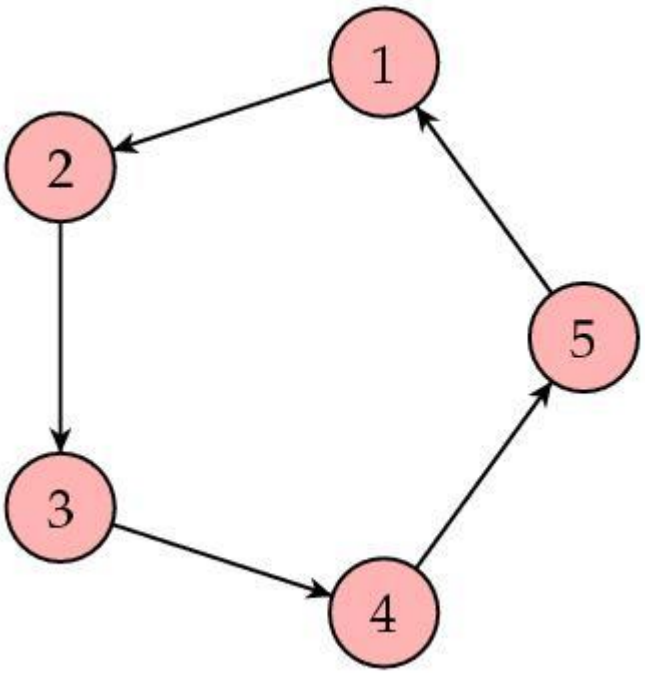
$$\cdots$$

$$\boldsymbol{I^k} = \boldsymbol{S} \cdot \boldsymbol{I^{k-1}} = c_1 v_1 + c_2 \lambda_2^k v_2 + \cdots + c_n \lambda_n^k v_n$$

Так как с $\boldsymbol{j \geq 2}$, собственные значения по модулю меньше единицы, то $\lambda_n^k \to 0$ и $\boldsymbol{I^k} \to \boldsymbol{I} = c_1 v_1$ - собственный вектор, соответствующий собственному значению 1.

Когда что-то пойдет не так

В рассуждениях выше мы предполагали, что матрица $\boldsymbol{S}$ обладает свойством, что $\lambda_1 = 1$ и $|\lambda_2| < 1$. Однако это верно не для всех матриц $\boldsymbol{S}$. Предположим, что наша сеть выглядит так:



В этом случае, матрица $\boldsymbol{S}$ выглядит следующим образом:

$$\mathbf{S} = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

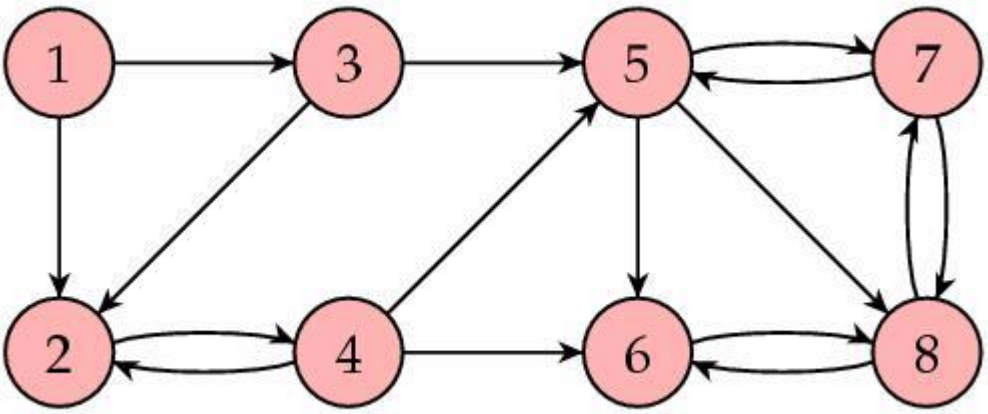
И степенной метод не работает:

I^0	I^1	I^2	I^3	I^4	I^5
1	0	0	0	0	1
0	1	0	0	0	0
0	0	1	0	0	0
0	0	0	1	0	0
0	0	0	0	1	0

В этом случае последовательность векторов $\boldsymbol{I^k}$ не сходится, поскольку второе собственное значение матрицы $\boldsymbol{S}$: $|\lambda_2| = 1$, поэтому утверждение, приведенное для обоснования степенного метода, больше не работает.

Чтобы гарантировать, что $|\lambda_2| < 1$, необходимо, чтобы матрица $\boldsymbol{S}$ была примитивной. Это означает, что для некоторого $\boldsymbol{m}$, в $\boldsymbol{S^m}$ все элементы положительны. Другими словами, если нам даны две страницы, то с первой можно перейти на вторую, пройдя по $\boldsymbol{m}$ ссылкам. Как мы можем увидеть, наш последний пример не удовлетворяет этому свойству. Далее покажем как, изменить матрицу $\boldsymbol{S}$, чтобы получить примитивную стохастическую матрицу, которая, таким образом, удовлетворяет условию $|\lambda_2| < 1$.

Вот еще один пример, показывающий, как наш метод может дать сбой. Рассмотрим сеть, показанную ниже.

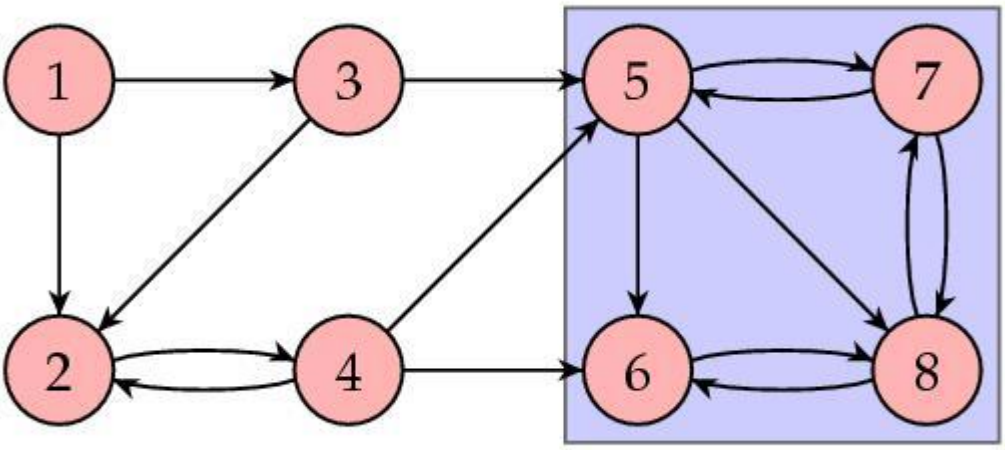


В этом случае матрица $\boldsymbol{S}$ и стационарный вектор $\boldsymbol{I}$ имеют вид:

$$\mathbf{S} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1/2 & 0 & 1/2 & 1/3 & 0 & 0 & 0 & 0 \\ 1/2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1/2 & 1/3 & 0 & 0 & 1/2 & 0 \\ 0 & 0 & 0 & 1/3 & 1/3 & 0 & 0 & 1/2 \\ 0 & 0 & 0 & 0 & 1/3 & 0 & 0 & 1/2 \\ 0 & 0 & 0 & 0 & 1/3 & 1 & 1/2 & 0 \end{bmatrix} \quad I = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0.12 \\ 0.24 \\ 0.24 \\ 0.4 \end{bmatrix}$$

Обратите внимание, что PageRank, присвоенный первым четырем веб-страницам, равен нулю. Однако это кажется неправильным: на каждую из этих

страниц есть ссылки, ведущие на них с других страниц. Кому-то нравятся эти страницы! Вообще говоря, мы хотим, чтобы рейтинг важности всех страниц был положительным. Проблема с этим примером заключается в том, что он содержит подсеть, показанную в синем поле ниже.



Ссылки приходят в эту подсеть, но не выходят. Как и в примере с висячим узлом, эти страницы образуют «приемник важности», который истощает важность других четырех страниц. Это происходит, когда матрица $\boldsymbol{S}$ приводима; то есть $\boldsymbol{S}$ можно записать в блочной форме как:

$$S = \begin{bmatrix} * & 0 \\ * & * \end{bmatrix}.$$

Действительно, если матрица $\boldsymbol{S}$ неприводима, мы можем гарантировать, что существует стационарный вектор со всеми положительными элементами. Граф называется сильно связанным, если для любых двух вершин (страниц) существует способ перейти по ребрам (ссылкам) с первой страницы на вторую. Очевидно, что последний пример не имеет сильной связи. Однако сильно связные сети дают неприводимые матрицы $\boldsymbol{S}$.

Подводя итог, матрица $\boldsymbol{S}$ является стохастической, что означает, что она имеет стационарный вектор. Однако нам нужно, чтобы $\boldsymbol{S}$ была:

- примитивной, то есть $|\lambda_2| < 1$;
- неприводимой, чтобы стационарный вектор имел все положительные элементы.

Окончательная модификация

Чтобы найти новую матрицу, одновременно примитивную и неприводимую, мы изменим способ перемещения нашего «случайного пользователя» по сети. В нынешнем виде движение нашего «случайного посетителя» определяется $\boldsymbol{S}$: либо он перейдет по одной из ссылок на своей текущей странице, либо, если находится на странице без ссылок, случайным образом выберет любую другую страницу для перехода.

Чтобы сделать модификацию алгоритма, выберем параметр $\alpha \in \overline{0,1}$. Предположим, что наш «случайный пользователь» движется немного по-другому. С вероятностью α он руководствуется $\boldsymbol{S}$. С вероятностью $1-\alpha$ он выбирает следующую страницу случайным образом. Если мы обозначим через $\mathbf{1}$ матрицу размера $n \times n$, все элементы которой равны единице, мы получим матрицу Google:

$$\boldsymbol{G} = \alpha \boldsymbol{S} + (1-\alpha)\frac{1}{n}\mathbf{1}$$

Обратите внимание, что $\boldsymbol{G}$ является стохастической, поскольку представляет собой комбинацию стохастических матриц. Кроме того, все элементы $\boldsymbol{G}$ положительны, что означает, что $\boldsymbol{G}$ одновременно примитивна и неприводима. Следовательно, $\boldsymbol{G}$ имеет единственный стационарный вектор $\boldsymbol{I}$, который можно найти с помощью степенного метода.

Роль параметра α очень важна. Обратите внимание, что если $\alpha = 1$, то $\boldsymbol{G} = \boldsymbol{S}$. Это означает, что мы работаем с исходной структурой гиперссылок сети. Однако если $\alpha = 0$, то: $\boldsymbol{G} = \frac{1}{n}\mathbf{1}$.

Другими словами, рассматриваемая нами сеть имеет связь между любыми двумя страницами, и мы потеряли первоначальную структуру гиперссылок сети. Очевидно, мы хотели бы взять α близким к 1, чтобы структура гиперссылок сети сильно учитывалась при вычислениях. Однако есть еще одно соображение. Помните, что скорость сходимости степенного метода определяется величиной второго собственного значения $|\lambda_2|$. Для матрицы Google, было доказано, что величина второго собственного значения

$|\lambda_2| = \alpha$. Это означает, что когда α близко к 1, сходимость степенного метода будет очень медленной. В качестве компромисса Серби Брин и Ларри Пейдж, создатели PageRank, выбрали $\alpha = 0.85$.

Вычисление *PageRank*

Пока что, все что описано выше, выглядит как хорошая теория, но помните, что нам нужно применить ее к матрицам размера $n \times n$, где n равно примерно 25 миллиардам! На самом деле степенной метод особенно хорошо подходит для этой ситуации. Помните, что стохастическая матрица S может быть записана как:

$$\boldsymbol{S = H + A}$$

и поэтому матрица Google имеет вид:

$$\boldsymbol{G} = \alpha \boldsymbol{H} + \alpha \boldsymbol{A} + \frac{1-\alpha}{n}\mathbf{1}$$

следовательно:

$$\boldsymbol{GI^k} = \alpha \boldsymbol{HI^k} + \alpha \boldsymbol{AI^k} + \frac{1-\alpha}{n}\mathbf{1}\boldsymbol{I^k}$$

Теперь вспомним, что большинство записей в $\boldsymbol{H}$ нулевые; в среднем только десять записей в столбце отличны от нуля. Следовательно, для вычисления $\boldsymbol{HI^k}$ требуется всего десять ненулевых членов для каждой записи результирующего вектора. Кроме того, все строки $\boldsymbol{A}$ идентичны, как и строки $\mathbf{1}$. Следовательно, оценка $\boldsymbol{AI^k}$ и $\mathbf{1}\boldsymbol{I^k}$ сводится к добавлению текущих рейтингов важности висячих узлов или всех веб-страниц. Это нужно сделать только один раз. Брин и Пейдж сообщают, что при значении α, близком к 0,85, требуется 50-100 итераций, чтобы получить достаточно хорошее приближение к $\boldsymbol{I}$. Также сообщается, что расчет занимает несколько дней.

Домашние задание

Реализовать алгоритм вычисление ***PageRank*** страниц.

Вход: csv-файл с массивом данных. (матрица NxN или список сайтов кто на кого ссылается)

Значение $\alpha = 0.85$.

Точность вычисления PageRank: 0,0001.

Слушатель, осуществивший самую быструю реализацию получает бонус на экзамене?

Тестирование на время будет происходить на закрытых данных преподавателем.