PageRank: как Google нашёл порядок в Интернете

Задача и история

К 1996 г. поисковики типа *AltaVista* и *Yahoo!* ранжировали страницы по тому, сколько раз искомое слово встречается в тексте. Это легко обманывалось: достаточно было набить страницу популярными словами белым цветом по белому фону, и она попадала в топ выдачи.

Лоуренс Пейдж и Сергей Брин, тогда аспиранты Стэнфорда, предложили другой принцип ранжирования: *страница важна, если на неё ссылаются другие важные страницы.*

Граф ссылок как матрица

Обозначим N — число всех страниц Интернета (на 1998 г. — около $2,4 \cdot 10^8$, сегодня — $\sim 10^{11}$). Построим матрицу $P \in \mathbb{R}^{N \times N}$:

$$P_{ij} = \begin{cases} \frac{1}{c_j}, & \text{если со страницы } \ j \ \text{ есть ссылка на } \ i \ , \\ 0, & \text{иначе,} \end{cases} \tag{1}$$

где c_j — *полное число исходящих ссылок* со страницы j. Содержательно: P_{ij} — это вероятность того, что пользователь, сидящий на странице j и случайно ткнувший на одну из её ссылок, окажется на странице i. Поэтому P называется матрицей переходов **случайного блуждания** по графу Интернета.

Стационарное распределение = собственный вектор

Пусть $r \in \mathbb{R}^N$ — вектор «важностей»: r_i — степень важности страницы i. Принцип Пейджа–Брина формализуется системой:

$$r = Pr. \tag{2}$$

То есть важность каждой страницы равна сумме важностей тех страниц, которые на неё ссылаются, делённой на число их исходящих ссылок.

Уравнение (0.7) говорит: r — это *собственный вектор* матрицы P с собственным значением 1. И в этом — вся суть PageRank.

⚠ Теорема 0.4. Перрон–Фробениус для PageRank

Если все элементы матрицы P положительны (или хотя бы граф ссылок сильно связен и непериодичен), то собственное значение 1 у P существует, оно *простое* и наибольшее по модулю, а соответствующий собственный вектор r единственен и имеет положительные компоненты. Этот r называется **стационарным распределением** случайного блуждания.

(Эта теорема — Оскар Перрон, 1907 г., и Георг Фробениус, 1912 г. — исторически появилась задолго до Интернета, в контексте моделей популяций.)

Демпфирующий множитель и формула Брина–Пейджа

В реальной матрице Интернета у некоторых страниц нет исходящих ссылок (*dangling pages*), а в других группах — циклы. Для таких страниц соответствующий столбец матрицы переходов заранее заменяют равномерным распределением по всем страницам: иначе формула $1/c_j$ при $c_j = 0$ не определена, а масса случайного блуждания «исчезала» бы в dangling pages. Чтобы гарантировать применимость теоремы Перрона–Фробениуса, Пейдж и Брин ввели **демпфирующий множитель** $\beta \in (0, 1)$ (обычно $\beta = 0,85$):

$$r = \beta P r + \frac{1-\beta}{N}\,\mathbf{1}. \tag{3}$$

Здесь $\mathbf{1} = (1, 1, \ldots, 1)^{\mathrm{T}}$. Содержательно: с вероятностью $\beta \approx 0,85$ случайный сёрфер кликает по ссылке, а с вероятностью $1 - \beta \approx 0,15$ — телепортируется на случайную страницу Интернета.

Как находить r: степенной метод

Прямо решать систему $(I - \beta P)r = \frac{1-\beta}{N}\mathbf{1}$ с $N \sim 10^{11}$ невозможно. Но и не надо: вектор r можно получить простой итерацией.

! Алгоритм

Степенной метод для PageRank.

1. Инициализация: $r^{(0)} \leftarrow \frac{1}{N}\mathbf{1}$.
2. Итерация: $r^{(t+1)} \leftarrow \beta\, P\, r^{(t)} + \frac{1-\beta}{N}\,\mathbf{1}$.
3. Остановка: $\| r^{(t+1)} - r^{(t)} \| < \varepsilon$.

Сходимость гарантируется теоремой Перрона–Фробениуса: $\| r^{(t)} - r \| \le C \cdot \beta^t$, то есть погрешность убывает геометрически со знаменателем $\beta = 0,85$. На практике хватает $50 \div 100$ итераций.

Игрушечный пример: 6 страниц

Возьмём маленький «Интернет» из шести страниц (рис. 0.9) и посчитаем для него PageRank.

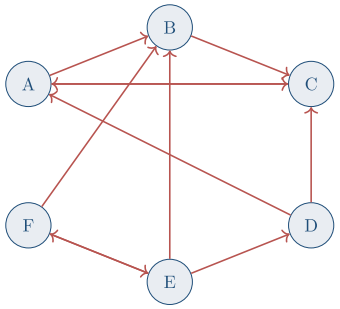

Рис. 0.9. Игрушечный «Интернет» из 6 страниц со ссылками. Видно, например, что на страницу C ссылаются три другие, а на F — только одна; будет ли C лидером по PageRank?

```python
import numpy as np

# Adjacency list -> column-stochastic transition matrix P
links = {'A': ['B','C'], 'B': ['C'], 'C': ['A'],
         'D': ['C','A'], 'E': ['D','B','F'], 'F': ['E','B']}
pages = list(links); n = len(pages); idx = {p: i for i, p in enumerate(pages)}

P = np.zeros((n, n))
for j, p in enumerate(pages):
    out = links[p]
    for q in out:
        P[idx[q], j] = 1.0 / len(out)

# PageRank via power iteration
beta, eps = 0.85, 1e-10
r = np.ones(n) / n
for t in range(200):
    r_new = beta * P @ r + (1 - beta) / n
    if np.linalg.norm(r_new - r, 1) < eps: break
    r = r_new
```

```
for p, ri in sorted(zip(pages, r), key=lambda t: -t[1]):
    print(f'{p}: {ri:.4f}')
```

Выдача программы:

C: $0,295$, A: $0,252$, B: $0,175$, E: $0,098$, F: $0,092$, D: $0,087$.

Лидер — C, хотя на неё формально ссылаются столько же страниц, сколько на A: PageRank учёл, что одна из ссылок на C приходит с уже-важной страницы B.

Историческая справка

Алгоритм PageRank был опубликован в 1998 г. в студенческой работе *Sergey Brin, Larry Page*, «*The PageRank Citation Ranking: Bringing Order to the Web*». В том же году Брин и Пейдж основали Google. К 2004 г. компания вышла на IPO, а к 2010-м стала одной из крупнейших в мире. Сам Стэнфорд получил ~ 336 миллионов долларов от лицензирования патента — который изначально просто описывал итерационный метод для собственного вектора.

Идея использовать собственный вектор графа для измерения «важности» позднее была обобщена и применена в наукометрии (*Eigenfactor* для ранжирования научных журналов), в рекомендательных системах, а также в современных нейросетях для графов (*Graph Neural Networks*).