FGV EMAp
ESCOLA DE MATEMÁTICA APLICADA

RELATÓRIO DE PESQUISA SOBRE PAGERANK

Álgebra Linear

Ricael Daniel Vieria da Silva
e
Thiago Franke Melchiors

Novembro
2022

Conteúdo

1 Introdução

Este relatório propõe uma explicação elucidativa acerca do funcionamento do algoritmo de PageRank do Google, enfatizando a visão da Álbegra Linear. Nesse sentido, o objetivo é apresentar a metodologia por trás da métrica, bem como aplicações em casos reais envolvendo bases de dados.

2 Contexto Histórico

2.1 A dificuldade no gerenciamento de páginas na internet

Em 1991, o físico britânico Tim Berners-Lee criou a primeira página da web para, segundo informações do próprio site, "dar acesso universal a um grande universo de documentos". Apesar do tom bastante otimista, é pouco provável que Tim esperasse que a sua idealização se tornasse o gigante abrangente que é hoje, com mais de um bilhão de sites já criados e milhões ainda ativos. Nessa perspectiva, a organização, classificação e gerenciamento dos sites tornou-se mais difícil na mesma medida que as páginas de web se multiplicavam. Além disso, os primeiros motores de busca da internet possuíam grandes limitações, pois as combinações das palavras-chave nas pesquisas não eram bem interpretadas e a maioria dos resultados das consultas eram relacionados a publicidade de produtos.

Nesse sentido, em 1998, Larry Page e Sergey Brin, então estudantes da Universidade de Stanford, desenvolveram um algoritmo de classificação da importância dos websites com base no número de referências entre as páginas. Prevendo o potencial dessa ferramenta, mais tarde batizada de PageRank, Larry e Sergey criaram o Google Inc., empresa na qual aplicaram o referido buscador. A habilidade de recomendar sites pertinentes e reduzir a visibilidade de sites de menor qualidade foi um diferencial em relação aos mecanismos de busca concorrentes e garantiu o sucesso da companhia.

2.2 Uma breve apresentação do mecanismo

No método utilizado pelo PageRank, uma página da internet corresponde a um "nó" (vértice) e cada "ligação" (flecha) equivale a uma referência de uma página para outra (hiperligação). Outrossim, para cada nó da rede é atribuído um valor. Quanto maior o valor, mais importante é o nó e maior a chance de ele ser indicado a um usuário. Em outras palavras, um site é melhor avaliado ser existirem muitas páginas indicando para ele ou se algumas páginas muito bem avaliadas pelo algoritmo do PageRank apontarem para ele.

Profissionais da área da tecnologia avaliam que o método é bastante eficaz e democrático, visto que ele não impede a ascensão de novos sites, permitindo que todas as páginas tenham a mesma chance de ampliar a sua relevância na internet.

Não obstante, o PageRank apresentava falhas, algumas delas corrigidas com o passar do tempo. Por exemplo, no processo conhecido como Googlebombing, links descontextualizados eram atribuídos em algumas páginas, modificando a ordenação dos resultados da pesquisa e induzindo a resultados enviesados.

3 Funcionamento do algoritmo

3.1 Esclarecimentos

Primeiramente, é essencial ressaltar alguns aspectos sobre o método de ranqueamento:

- A versão do algoritmo de PageRank que será explicada nesse relatório tem a vantagem de o valor atribuído a cada página representar a probabilidade de um usuário, navegando aleatoriamente, chegar até ela. Essa interpretação é possível porque a soma de todos os ranqueamentos é igual a 1;
- Para se chegar na classificação precisa de cada site, o algoritmo precisa ser aplicado várias vezes, em um processo conhecido como iteração;
- Seja N o número total de páginas. Então, após a primeira iteração, todas as páginas terão a mesma classificação, $\frac{1}{N}$.

3.2 O algoritmo simplificado

Uma rede simplificada de quatro páginas A, B, C e D, todas com a mesma importância, é representada a seguir:

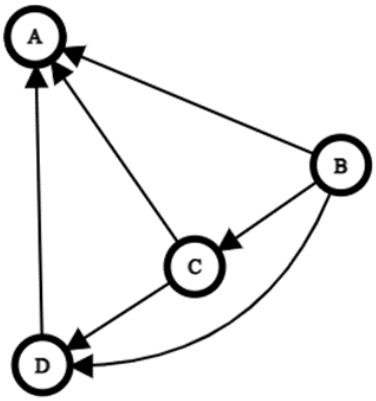

Figura 1: Exemplo de rede com 4 páginas

A partir de uma versão menos complexa do algoritmo, o primeiro passo para encontrar o PageRank do nó A é determinar que, após a primeira etapa do processo iterativo, todas as páginas têm o mesmo valor $\frac{1}{N}$ de PageRank, sendo N o número de páginas da rede. Portanto, as quatro páginas terão $0,25$ de PageRank.

Na segunda iteração, cada página transfere o seu valor de PageRank em porções iguais para as páginas que aponta. Dessa forma, o PageRank de A, $PR(A)$, da figura (1) passa a ser:

$$PR(A) = \frac{PR(B)}{3} + \frac{PR(C)}{2} + \frac{PR(D)}{1} \tag{1}$$

É possível estender a equação (1) para outros casos. Seja $M(p_i)$ o conjunto de todas as

páginas p_j que referenciam a página $\mathbf{p_i}$, e $L(p_j)$ o número de referências em p_j, o valor de PageRank para a página $\mathbf{p_i}$ é:

$$PR(p_i) = \sum_{p_j \in M(p_i)} \frac{PR(p_j)}{L(p_j)} \tag{2}$$

4 Falhas do PageRank

4.1 Páginas sem ligações

Nesse caso, uma página é referida por uma ou mais páginas, mas ela não menciona nenhuma outra, retendo, por lógica, parte ou a totalidade do PageRank da rede. Contudo, aplicando o cálculo de classificação, ambas as páginas recebem o valor zero. Portanto, essa é uma situação que não faz sentido.

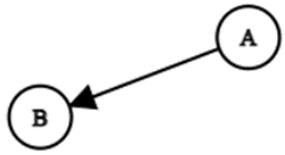

Figura 2: Exemplo de página sem ligação (B)

4.2 Ciclos (Rank Sink)

Quando uma rede possui um ciclo, após cada iteração, o valor de ranqueamento é transferido para o nó seguinte, sem que haja um momento em que a sistema entre em equilíbrio. Dessa forma, calculando o PageRank, todas as páginas receberão valor 1 de ranqueamento, o que é um absurdo, pois a soma das métricas deve resultar em 1 e é pouco provável que não exista um site mais bem avaliado do que os outros.

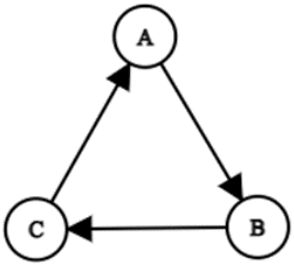

Figura 3: Exemplo de rede cíclica

4.3 Solução: Fator de Amortecimento

A fim de diminuir os desvios causados pelos casos de erro descritos nos itens anteriores, o Fator de Amortecimento, representado por **d**, foi introduzido ao algoritmo do

PageRank. Esse elemento varia de 0 a 1 e representa a probabilidade de um usuário, que está navegando casualmente pela internet, continuar seguindo as ligações entre as páginas.

Com a adição desse novo termo, o PageRank de A, $PR(A)$, passa a ser calculado pela fórmula (1), ponderada pela probabilidade **d** de o usuário seguir as hiperligações

$$d\left(\frac{PR(B)}{L(B)} + \frac{PR(C)}{L(C)} + \frac{PR(D)}{L(D)}\right)$$

e um elemento referente a possibilidade de o usuário ter selecionado a página aleatoriamente ($\frac{1}{N}$) ponderado pela probabilidade de o usuário não seguir as ligações das páginas $(1-d)$, ou seja:

$$\frac{1-d}{N}.$$

Logo, a fórmula completa do PageRank de A é:

$$PR(A) = \frac{1-d}{N} + d\left(\frac{PR(B)}{L(B)} + \frac{PR(C)}{L(C)} + \frac{PR(D)}{L(D)}\right) \tag{3}$$

A seguir, são apresentados aspectos interessantes que o Fator de Amortecimento aplica no ranqueamento:

- Todas as páginas de uma rede têm a mesma probabilidade de serem selecionada pela escolha aleatória de um usuário ($\frac{1}{N}$);
- Um nó sem ligação está ligado a todas as páginas da rede;
- Com $d = 0$, todas as páginas ficam com o PageRank de $\frac{1}{N}$ (onde N é o número de páginas da rede), resultado da primeira iteração do algoritmo. Logo, quanto mais próximo d estiver de 1, maior é a influência da estrutura da rede;
- O valor padrão para d é $0,85$.

5 Representação Matricial

A equação geral é obtida combinando-se as equações (2) e (3):

$$PR(p_i) = \frac{1-d}{N} + d\left(\sum_{p_j \in M(p_i)} \frac{PR(p_j)}{L(p_j)}\right), \tag{4}$$

na qual N é o número de páginas da web (p_1, p_2, ..., p_j,..., p_n); $M(p_i)$ é o conjunto de páginas que referenciam a página p_i; e $L(p_j)$ é o número de hiperligações que partem da página p_j.

É possível calcular o valor de PageRank de todas as páginas de uma só vez. Para isso, "empilha-se" N equações (4) referente as N páginas da rede. É, na verdade, uma representação matricial da operação. Observe os próximos passos.

O valor $PR(p_i)$ de cada uma das páginas da rede é armazenado no vetor $\mathbf{R}$:

$$R = \begin{bmatrix} PR(p_1) \\ PR(p_2) \\ \vdots \\ PR(p_j) \\ \vdots \\ PR(p_n) \end{bmatrix}$$

Além disso, a matriz $\mathbf{M}_{(\mathbf{n}\times\mathbf{n})}$ (Matriz de Markov), onde n é o número total de páginas, tem suas entradas preenchidas pelos elementos $l(p_i, p_j)$, que podem ser:

- $l(p_i, p_j) = 0$ se p_j não aponta para a página p_i; e
- $l(p_i, p_j) = \frac{1}{L(p_j)}$ se existe referência de p_j para a página p_i.

Logo, a expressão para o cálculo de PageRank de todas as páginas da rede é:

$$R = \begin{bmatrix} \frac{1-N}{d} \\ \frac{1-N}{d} \\ \vdots \\ \frac{1-N}{d} \end{bmatrix} + d \begin{bmatrix} l(p_1,p_1) & l(p_1,p_2) & \dots & l(p_1,p_n) \\ l(p_2,p_1) & \ddots & & \vdots \\ \vdots & & l(p_i,p_2j) & \\ l(p_n,p_1) & \dots & & l(p_n,p_n) \end{bmatrix} R \tag{5}$$

Ainda, seja $\mathbf{k}$ um vetor de uns de dimensões $n \times 1$ e E uma matriz $n \times n$ de uns. Uma vez que a soma de todos os elementos do vetor $\mathbf{R}$ é 1, então $E \cdot \mathbf{R} = \mathbf{k}$. Dessarte, é possível reescrever (5) para uma matriz de adjacências modificada $\hat{M}$:

$$\begin{aligned} \mathbf{R} &= d\mathbf{M}\mathbf{R} + \frac{1-d}{N}\mathbf{k} \\ \mathbf{R} &= d\mathbf{M}\mathbf{R} + \frac{1-d}{N}E\mathbf{R} \\ \mathbf{R} &= \left(d\mathbf{M} + \frac{1-d}{N}E\right)\mathbf{R} \\ \hat{M} &= d\mathbf{M} + \frac{1-d}{N}E \end{aligned} \tag{6}$$

Por fim, sejam t o número de iterações e $x(t)$ o PageRank na iteração t, então $x(0) = \frac{1}{N}$ e $x(t+1)$ é a iteração subsequente a $x(t)$. Logo, aplicando sucessivamente a equação

$$x(t+1) = \hat{M}x(t)$$

chega-se a valores de PageRank $x(t)$ mais precisos, até que, eventualmente, $x(t)$ pare de mudar.

6 Aplicando na prática

6.1 Exemplo de Resolução

Munidos dos tópicos discorridos até agora, calcularemos o PageRank das páginas A, B, C e D da rede abaixo:

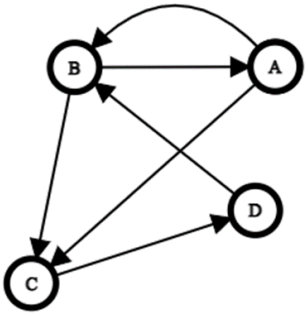

Figura 4: Exemplo de rede simplificada

O primeiro passo é preencher a matriz $\mathbf{M}$,

$$\mathbf{M} = \begin{bmatrix} l_{(A,A)} & l_{(A,B)} & l_{(A,C)} & l_{(A,D)} \\ l_{(B,A)} & l_{(B,B)} & l_{(B,C)} & l_{(B,D)} \\ l_{(C,A)} & l_{(C,B)} & l_{(C,C)} & l_{(C,D)} \\ l_{(D,A)} & l_{(C,B)} & l_{(D,C)} & l_{(D,D)} \end{bmatrix},$$

onde cada entrada $l_{(i,j)}$ significa $\frac{1}{L(j)}$ e $L(j)$ é o número de referência que partem da página j. Se j não se refere a i, a respectiva entrada da matriz recebe 0. Desse modo,

$$\mathbf{M} = \begin{bmatrix} 0 & \frac{1}{2} & 0 & 0 \\ \frac{1}{2} & 0 & 0 & 1 \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}.$$

Visto que esse exemplo possui poucos dados, a adição do fator de amortecimento ao cálculo dificultaria a operação e traria poucos benefícios, já que a aproximação dos resultados proporcionada por ele seria mínima. Por esse motivo, vamos definir $d = 1$ (o usuário vai seguir a estrutura da rede 100% das vezes) e, consequentemente, teremos a seguinte equação:

$$\begin{aligned} \mathbf{R} &= d\mathbf{M}\mathbf{R} + \frac{1-d}{N}E\mathbf{R} \\ \mathbf{R} &= 1\mathbf{M}\mathbf{R} + \frac{1-1}{N}E\mathbf{R} \\ \mathbf{M}\mathbf{R} &= \mathbf{R} \end{aligned}$$

Perceba que em $\mathbf{MR} = \mathbf{R}$ estamos buscando o vetor $\mathbf{R}$, que é nada mais nada menos que um autovetor de autovalor 1. Desse modo, prosseguiremos para calculá-lo.

$$\begin{bmatrix} 0 & \frac{1}{2} & 0 & 0 \\ \frac{1}{2} & 0 & 0 & 1 \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} PR(A) \\ PR(B) \\ PR(C) \\ PR(D) \end{bmatrix} = \begin{bmatrix} PR(A) \\ PR(B) \\ PR(C) \\ PR(D) \end{bmatrix}$$

Resolvendo essa multiplicação de matrizes pelo sistema de equações

$$\begin{cases} PR(A) = \frac{PR_B}{2} \\ PR(B) = \frac{PR_A}{2} + PR_D \\ PR(C) = \frac{PR_A}{2} + \frac{PR_B}{2} \\ PR(D) = PR_C \end{cases},$$

encontramos a solução

$$\begin{bmatrix} PR(A) \\ PR(B) \\ PR(C) \\ PR(D) \end{bmatrix} = \begin{bmatrix} 1/2 \\ 1 \\ 3/4 \\ 3/4 \end{bmatrix}$$

Contudo, apesar de que a soma dos elementos da matriz deveria resultar em 1, na matriz que obtivemos o somatório é $\frac{12}{4} = 3$. Para corrigir esse desvio, basta multiplicar tudo por $\frac{1}{3}$.

Portanto, o PageRank da rede (4) é:

$$\begin{bmatrix} PR(A) \\ PR(B) \\ PR(C) \\ PR(D) \end{bmatrix} = \begin{bmatrix} 1/6 \\ 1/3 \\ 1/4 \\ 1/4 \end{bmatrix}$$

6.2 Exemplo computacional com dados reais

Neste exemplo computacional, utilizaremos uma base de dados sobre a disputa da *Premier League* da temporada de 2020/2021. Nessa prática, analisaremos como ficaria a classificação dos times se o ranqueamento fosse feito a partir da capacidade de um time ganhar uma partida e da importância de cada vitória, em vez do atual sistema de pontos corridos.

A base de dados escolhida tem informações sobre todos os jogos da temporada e, para cada confronto, inclui o número da rodada e do jogo, a data da realização da disputa, o nome do time mandante e do visitante e, por fim, o placar da partida. Elucidações feitas, vamos para o código.

[19]:
```
import pandas as pd
import numpy as np
```

[20]:
```
data = pd.read_csv("premierleague.csv")
```

[21]:
```
data.head()
```

[21]:

	Match Number	Round Number	Date	Location	Home Team	Away Team	Result
0	4	1	12/09/2020 12:30	Craven Cottage	Fulham	Arsenal	0 - 3
1	3	1	12/09/2020 15:00	Selhurst Park	Crystal Palace	Southampton	1 - 0
2	5	1	12/09/2020 17:30	Anfield	Liverpool	Leeds	4 - 3
3	8	1	12/09/2020 20:00	London Stadium	West Ham	Newcastle	0 - 2
4	7	1	13/09/2020 14:00	The Hawthorns	West Brom	Leicester	0 - 3

Primeiramente, vamos separar a coluna *Result* em duas colunas, uma com os gols marcados pela equipe mandante (*home_tem_gols*) e outra com os gols marcados pela equipe visitante (*away_team_gols*).

[22]:
```
home_team_gols = []
away_team_gols = []

for home_gols in data["Result"]:
    home_team_gols.append(home_gols[0])

for away_gols in data["Result"]:
    away_team_gols.append(away_gols[4])
```

Em seguida, criamos dois dataframes e depois concatenamos eles, formando um novo dataframe composto pelos times que participaram da partida e pelos gols que foram marcados por cada equipe.

[23]:
```
# Dataframe com os times por partida.
df1 = data[["Home Team","Away Team"]]

# Dataframe com os gols marcados.
df2 = pd.DataFrame(list(zip(home_team_gols,away_team_gols)),columns␣
 ↪=["home_gols","away_gols"])

# Juntando os dois dataframes.
```

```
df = df1.join(df2)

# Trocando os nomes das colunas.
df.rename(columns = {"Home Team":"home_team","Away Team":"away_team",
                     "home_gols":"home_gols","away_gols":"away_gols"},
          inplace=True)
```

[24]:
```
df.head()
```

[45]:

	home_team	away_team	home_gols	away_gols
0	Fulham	Arsenal	0	3
1	Crystal Palace	Southampton	1	0
2	Liverpool	Leeds	4	3
3	West Ham	Newcastle	0	2
4	West Brom	Leicester	0	3
5	Spurs	Everton	0	1
6	Sheffield Utd	Wolves	0	2
7	Brighton	Chelsea	1	3
8	Everton	West Brom	5	2
9	Leeds	Fulham	4	3
10	Man Utd	Crystal Palace	1	3

Agora, vamos separar os times em dois grupos: vencedores (*winners*) e perdedores (*losers*). Se ocorrer empate na partida, os nomes dos participantes daquele jogo são adicionados nas duas listas.

[25]:
```
winners = []
losers = []

for index,line in df.iterrows():
    if line["home_gols"] > line["away_gols"]:
        winners.append(df.loc[index,"home_team"])
        losers.append(df.loc[index,"away_team"])

    elif line ["home_gols"] < line["away_gols"]:
        winners.append(df.loc[index,"away_team"])
        losers.append(df.loc[index,"home_team"])

# Se os times emparatam, adicionamos o nome dos times tanto na lista de␣
 ↪losers quanto na de winners.
    else:
        winners.append(df.loc[index,"home_team"])
        losers.append(df.loc[index,"away_team"])
        winners.append(df.loc[index,"away_team"])
        losers.append(df.loc[index,"home_team"])
```

[26]:
```
# Criando o dataframe com os times winners e losers.
df_matches = pd.DataFrame(list(zip(winners,losers)),columns =␣
 ↪["winner","loser"])
```

[27]:
```
df_matches.head()
```

[27]:

	winner	loser
0	Arsenal	Fulham
1	Crystal Palace	Southampton
2	Liverpool	Leeds
3	Newcastle	West Ham
4	Leicester	West Brom

A partir de agora, construiremos a matriz de trasição **M**. Primeiramente, adicionamos os nomes das colunas e das linhas.

[39]:
```
# pegando os nomes dos times exatamente uma vez e depois organizando em␣
 ↪ordem alfabética.
times = df_matches["winner"].unique()
times = sorted(times)

# Matriz M.
matrix = pd.DataFrame(index = times,columns = times)
```

Posteriormente, preenchemos a matriz **M** com dados.

[40]:
```
def preencher(valor,coluna):
    for indice in df_matches.index:
        if df_matches.loc[indice,"loser"] == coluna and df_matches.
 ↪loc[indice,"winner"] == valor.name:
            return 1

M = pd.DataFrame()

for i in matrix:
    M[i] = matrix.apply(preencher,args=(i, )).fillna(0)
```

[42]:
```
# Matriz de transição M.
M.head()[dfpreenchido.columns[:5]]
```

[42]:

	Arsenal	Aston Villa	Brighton	Burnley	Chelsea
Arsenal	0.0	0.0	1.0	1.0	1.0
Aston Villa	1.0	0.0	1.0	1.0	1.0
Brighton	0.0	1.0	0.0	1.0	1.0
Burnley	1.0	1.0	1.0	0.0	0.0
Chelsea	0.0	1.0	1.0	1.0	0.0

Vamos que criar o dataframe *dfpr*, para isso vamos pegar cada entrada de M e dividir pela soma dos valores contidos na coluna dessa entrada.

[32]:
```
dfpr = pd.DataFrame()
for i in M:
    dfpr[i]=M[i]/sum(M[i])
```

[33]:
```
dfpr.head()[dfpr.columns[:5]]
```

[33]:

	Arsenal	Aston Villa	Brighton	Burnley	Chelsea
Arsenal	0.000000	0.000000	0.058824	0.058824	0.076923
Aston Villa	0.071429	0.000000	0.058824	0.058824	0.076923
Brighton	0.000000	0.058824	0.000000	0.058824	0.076923
Burnley	0.071429	0.058824	0.058824	0.000000	0.000000
Chelsea	0.000000	0.058824	0.058824	0.058824	0.000000

[48]:
```
# Convertendo o dataframe dfpr para um array L que será utilzado nos␣
 ↪cálculos do pagerank.
L = dfpr.to_numpy()
```

Por fim, calculamos o PageRank usando a expressão:

$$PR = \frac{1-d}{n} + d \cdot L$$

onde n é a quantidade de times e d é o fator de amortecimento - utilizaremos o valor padrão 0,85. Definiremos B como um array $n \times n$ que resume a operação $\frac{1-d}{n}$.

[98]:
```
d = 0.85
n = L.shape[0]
B = ((1 - d)/n)*np.ones((n, n))
Pr = B + d*L
```

E, ainda, calculamos os autovalores u e seus autovetores v correspondentes.

[93]:
```
u, v = np.linalg.eig(Pr)

# Autovetor correspondente ao autovalor 1.
pagerank = -v[:,0].real
```

Em seguida, selecionamos os valores do PageRank de cada time e ordenamo-los em ordem decrescente. Dessa forma, obtemos a classificação final do campeonato com base na importância das vitórias de cada time.

[95]:
```
classificacao = pd.DataFrame({"Time":times,"Pagerank":pagerank})
classificacao.sort_values(by="Pagerank", ascending=False, inplace=True)
classificacao.set_index(np.arange(1,21),inplace = True)
```

[96]:
```
classificacao
```

[96]:

	Time	Pagerank		Time	Pagerank
1	Liverpool	0.273477	11	Crystal Palace	0.212731
2	Man Utd	0.272085	12	West Ham	0.212430
3	Man City	0.266215	13	Southampton	0.206356
4	Leicester	0.262372	14	Fulham	0.202996
5	Chelsea	0.261980	15	Arsenal	0.201735
6	Spurs	0.258811	16	West Brom	0.187869
7	Everton	0.243673	17	Wolves	0.182869
8	Leeds	0.238636	18	Newcastle	0.180655
9	Brighton	0.234189	19	Burnley	0.156745
10	Aston Villa	0.220853	20	Sheffield Utd	0.122887