Jane Doe and Max Power

Quarto CRC Book

To blah, blah, and blah.

Table of contents

Preface

This is a Quarto book.

Software conventions

```
1 + 1
```

```
2
```

To learn more about Quarto books visit https://quarto.org/docs/books.

Acknowledgments

Blah, blah, blah...

1

CINDY LAUNDIYA MARETHA

2

210411100037

```
!pip install beautifulsoup4
```

```
Requirement already satisfied: beautifulsoup4 in /usr/local/lib/python3.10/dist-packages (4.11.2)
Requirement already satisfied: soupsieve>1.2 in /usr/local/lib/python3.10/dist-packages (from beautif
```

```
import requests
from bs4 import BeautifulSoup
from datetime import datetime
import csv
hades = {'user-agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, li
```

3

1. Crawling Berita

Pada proses crawling ini diambil berita melalui website detik.com dengan menggunakan kata kunci “Pemanasan Global”

```
def scrape_detik(hal, requests):
    a = 1
    # Membuka file CSV untuk menulis hasil scraping
    with open('hasil_scraping.csv', 'w', newline='', encoding='utf-8') as csvfile:
        fieldnames = ['Judul', 'Waktu', 'Link', 'Content']
        writer = csv.DictWriter(csvfile, fieldnames=fieldnames)

        # Menulis header ke dalam file CSV
        writer.writeheader()

        for page in range(1, hal):
            url = f'https://www.detik.com/search/searchall?query=pemanasan+global&siteid=2{page
            req = requests.get(url)
            sop = BeautifulSoup(req.text, 'html.parser')
            li = sop.find('div', class_='list media_rows list-berita')
            lin = li.find_all('article')

            for x in lin:
                link = x.find('a')['href']
                date = x.find('a').find('span', class_='date').text.replace('WIB','').replace('
                headline = x.find('a').find('h2').text

                ge_ = requests.get(link).text
                sop_ = BeautifulSoup(ge_, 'html.parser')
                content = sop_.find_all('div', class_='detail__body-text itp_bodycontent')

                for cont in content:
                    paragraphs = cont.find_all('p')
                    content_ = ''.join([p.text for p in paragraphs]).replace('\n', '').replace(

                    data = {
                        'Judul': headline,
```

```
                    'Waktu': date,
                    'Link': link,
                    'Content': content_
                }
                # Menulis data ke dalam file CSV
                writer.writerow(data)
                print("Data berhasil ditambahkan:", data)
                print(f'done[{a}] > {headline}')
                a += 1
```

```
scrape_detik(3, requests)
```

```
Data berhasil ditambahkan: {'Judul': 'Hampir 8.000 Pelari Meriahkan Lazada Run di ICE BSD, Ada dari Ken
done[1] > Hampir 8.000 Pelari Meriahkan Lazada Run di ICE BSD, Ada dari Kenya
Data berhasil ditambahkan: {'Judul': 'Hari Lari Sedunia, 200 Pelari Ikut Fun Run Under Armour di Jakart
done[2] > Hari Lari Sedunia, 200 Pelari Ikut Fun Run Under Armour di Jakarta
Data berhasil ditambahkan: {'Judul': 'PBSI Maklum Singapore Open Batal, tapi...', 'Waktu': ' 13 Mei 202
done[3] > PBSI Maklum Singapore Open Batal, tapi...
Data berhasil ditambahkan: {'Judul': 'Kualifikasi Olimpiade Mulai 2021, Richard Mainaky: Tahun Ini Pema
done[4] > Kualifikasi Olimpiade Mulai 2021, Richard Mainaky: Tahun Ini Pemanasan
Data berhasil ditambahkan: {'Judul': 'Piala Thomas dan Uber 2020 Jadi Oktober, PBSI: Waktunya Ideal', '
done[5] > Piala Thomas dan Uber 2020 Jadi Oktober, PBSI: Waktunya Ideal
Data berhasil ditambahkan: {'Judul': 'Liga Equestrian Digelar Akhir Pekan Ini, Ada 1.000 Kursi Penonton
done[6] > Liga Equestrian Digelar Akhir Pekan Ini, Ada 1.000 Kursi Penonton Gratis
Data berhasil ditambahkan: {'Judul': 'Begini Cara Pegolf Lokal Cari Pengalaman', 'Waktu': ' 17 Des 2017
done[7] > Begini Cara Pegolf Lokal Cari Pengalaman
Data berhasil ditambahkan: {'Judul': 'Yang Perlu Diperbaiki Agar INASGOC Lebih Siap di Asian Games 2018
done[8] > Yang Perlu Diperbaiki Agar INASGOC Lebih Siap di Asian Games 2018
Data berhasil ditambahkan: {'Judul': 'Pembukaan Olimpiade Rio: Tampilkan Favela dan Ajakan Hijaukan Hu
done[9] > Pembukaan Olimpiade Rio: Tampilkan Favela dan Ajakan Hijaukan Hutan
```

4

Tampilan Data hasil Crawling Berita

```
import pandas as pd
df = pd.read_csv('/content/hasil_scraping.csv')
df
```

	Judul	Waktu	Link
0	Hampir 8.000 Pelari Meriahkan Lazada Run di IC...	11 Jun 2023 11:58	https://sport.detik.com/spo
1	Hari Lari Sedunia, 200 Pelari Ikut Fun Run Und...	01 Jun 2022 11:17	https://sport.detik.com/spo
2	PBSI Maklum Singapore Open Batal, tapi...	13 Mei 2021 15:25	https://sport.detik.com/rak
3	Kualifikasi Olimpiade Mulai 2021, Richard Main...	04 Jun 2020 14:29	https://sport.detik.com/rak
4	Piala Thomas dan Uber 2020 Jadi Oktober, PBSI:...	29 Apr 2020 22:25	https://sport.detik.com/rak
5	Liga Equestrian Digelar Akhir Pekan Ini, Ada 1...	13 Des 2019 00:00	https://sport.detik.com/spo
6	Begini Cara Pegolf Lokal Cari Pengalaman	17 Des 2017 22:50	https://sport.detik.com/spo
7	Yang Perlu Diperbaiki Agar INASGOC Lebih Siap ...	28 Nov 2017 13:55	https://sport.detik.com/spo
8	Pembukaan Olimpiade Rio: Tampilkan Favela dan ...	06 Agu 2016 11:20	https://sport.detik.com/spo

```
%%capture
!pip install nltk
!pip install Sastrawi
```

```
import pandas as pd
import re
import nltk
import numpy as np
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.tokenize import RegexpTokenizer
# tokenizer = RegexpTokenizer(r'\w+')
from Sastrawi.Stemmer.StemmerFactory import StemmerFactory
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer,CountVectorizer
from sklearn.preprocessing import OneHotEncoder
```

```
nltk.download("punkt")
nltk.download("stopwords")
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
```

```
True
```

```
df = df.astype(str)
df["Content"] = df["Content"].apply(lambda x: x.lower())

content_column = df["Content"]
```

5

2. Ekstraksi Kalimat

Pada proses ekstraksi kalimat dilakukan tokenisasi kalimat yang ada di dalam dokumen berita

```
from nltk.tokenize import sent_tokenize

# Misalnya, jika df adalah DataFrame yang memiliki kolom 'Content'
teks_berita = df['Content'].values.tolist()

kalimat = []
for teks in teks_berita:
    kalimat.extend(sent_tokenize(teks))

df_kalimat = pd.DataFrame(kalimat, columns=['Tokenisasi'])
df_kalimat
```

	Tokenisasi
0	sekitar 8.000 peserta meriahkan ajang lari yan...
1	para peserta lomba lari ini tak hanya dari dal...
2	semuanya terbuka untuk umum dan masyarakat.
3	”pesertanya kita terbuka untuk semua.
4	mau yang sport enthusiast, professional runner...
...	...
94	pulomas jaya dan dikelola oleh pt.
95	equinara global prima.venue ini memiliki fasil...
96	\r\r\rscroll to continue with content
97	\r\r\rscroll to continue with content
98	awesome ðŸ˜? #openingceremony #rio2016 #olympi...

6

3. TF-IDF

Proses TF-IDF ini digunakan untuk mengetahui seberapa sering suatu kata muncul didalam dokumen. Berikut rumus perhitungan TF-IDF

$$w_{ij} = tf_{ij} x idf_j$$
$$w_{ij} = tf_{ij} x \log\left(D/df_j\right)$$

Dimana Wij merupakan bobot dari term(j) terhadapn dokumen(i). Sedangkan tfij merupakan jumlah kemunculan term(j) dalam dokumen(i). Untuk D sendiri merupakan jumlah semua dokumen yang ada pada data dan dfj merupakan jumlah dokumen yang mengandung term(j)

```
from sklearn.feature_extraction.text import TfidfVectorizer
import pandas as pd

# Data kalimat (contoh)
kalimat = df_kalimat['Tokenisasi']

# Membuat objek TfidfVectorizer
tfidf_vectorizer = TfidfVectorizer()

# Menghitung TF-IDF
tfidf_matrix = tfidf_vectorizer.fit_transform(kalimat)

# Mengonversi matriks TF-IDF ke DataFrame Pandas
tfidf_df = pd.DataFrame(tfidf_matrix.toarray(), columns=tfidf_vectorizer.get_feature_names_out(

# Menampilkan tabel TF-IDF
tfidf_df
```

	000	1000	10k	11	14	15	16	162	19	200	...	widianto	widjaja
0	0.245482	0.000000	0.000000	0.000000	0.0	0.0	0.0	0.000000	0.0	0.0	...	0.0	0.0
1	0.000000	0.000000	0.185612	0.000000	0.0	0.0	0.0	0.000000	0.0	0.0	...	0.0	0.0
2	0.000000	0.000000	0.000000	0.000000	0.0	0.0	0.0	0.000000	0.0	0.0	...	0.0	0.0
3	0.000000	0.000000	0.000000	0.000000	0.0	0.0	0.0	0.000000	0.0	0.0	...	0.0	0.0

	000	1000	10k	11	14	15	16	162	19	200	...	widianto	widjaja
4	0.000000	0.000000	0.000000	0.187714	0.0	0.0	0.0	0.000000	0.0	0.0	...	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
94	0.000000	0.000000	0.000000	0.000000	0.0	0.0	0.0	0.000000	0.0	0.0	...	0.0	0.0
95	0.000000	0.121002	0.000000	0.000000	0.0	0.0	0.0	0.131889	0.0	0.0	...	0.0	0.0
96	0.000000	0.000000	0.000000	0.000000	0.0	0.0	0.0	0.000000	0.0	0.0	...	0.0	0.0
97	0.000000	0.000000	0.000000	0.000000	0.0	0.0	0.0	0.000000	0.0	0.0	...	0.0	0.0
98	0.000000	0.000000	0.000000	0.000000	0.0	0.0	0.0	0.000000	0.0	0.0	...	0.0	0.0

7

4. Cosinus Similarity

Proses ini digunakan untuk mengukur jarak kedekatan antar dokumen dan diperoleh rumus sebagai berikut.

$$\cos\theta = \frac{a \cdot b}{\|a\| \cdot \|b\|}$$

```
from sklearn.metrics.pairwise import cosine_similarity

# Menghitung kesamaan kosinus antara kalimat-kalimat
cosine_sim_matrix = cosine_similarity(tfidf_matrix, tfidf_matrix)

# Menampilkan matriks kesamaan kosinus
cosine_sim_df = pd.DataFrame(cosine_sim_matrix, columns=df_kalimat.index, index=df_kalimat.inde
cosine_sim_df
```

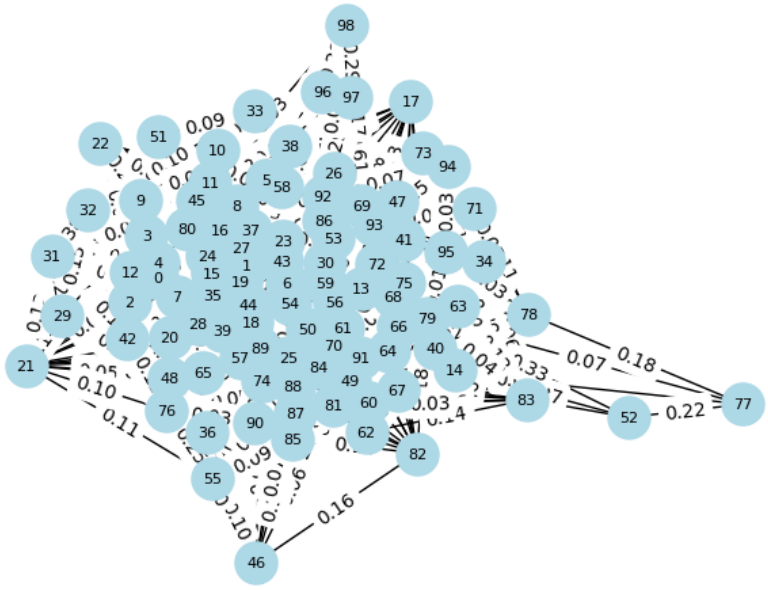
	0	1	2	3	4	5	6	7	8	9
0	1.000000	0.107981	0.000000	0.000000	0.111846	0.043594	0.055045	0.100117	0.081259	0.171
1	0.107981	1.000000	0.047697	0.025103	0.014792	0.059196	0.103742	0.140412	0.128848	0.170
2	0.000000	0.047697	1.000000	0.312559	0.000000	0.135064	0.119654	0.000000	0.000000	0.000
3	0.000000	0.025103	0.312559	1.000000	0.000000	0.045634	0.147147	0.000000	0.070556	0.000
4	0.111846	0.014792	0.000000	0.000000	1.000000	0.054399	0.000000	0.050911	0.141188	0.089
...	...	...	...	...	...	...	...	...	...	...
94	0.000000	0.023744	0.064740	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000
95	0.009203	0.022871	0.036277	0.000000	0.007038	0.008694	0.010978	0.011814	0.005436	0.000
96	0.000000	0.000000	0.000000	0.000000	0.000000	0.504372	0.000000	0.000000	0.000000	0.000
97	0.000000	0.000000	0.000000	0.000000	0.000000	0.504372	0.000000	0.000000	0.000000	0.000
98	0.000000	0.000000	0.000000	0.000000	0.000000	0.144472	0.000000	0.000000	0.000000	0.000

8

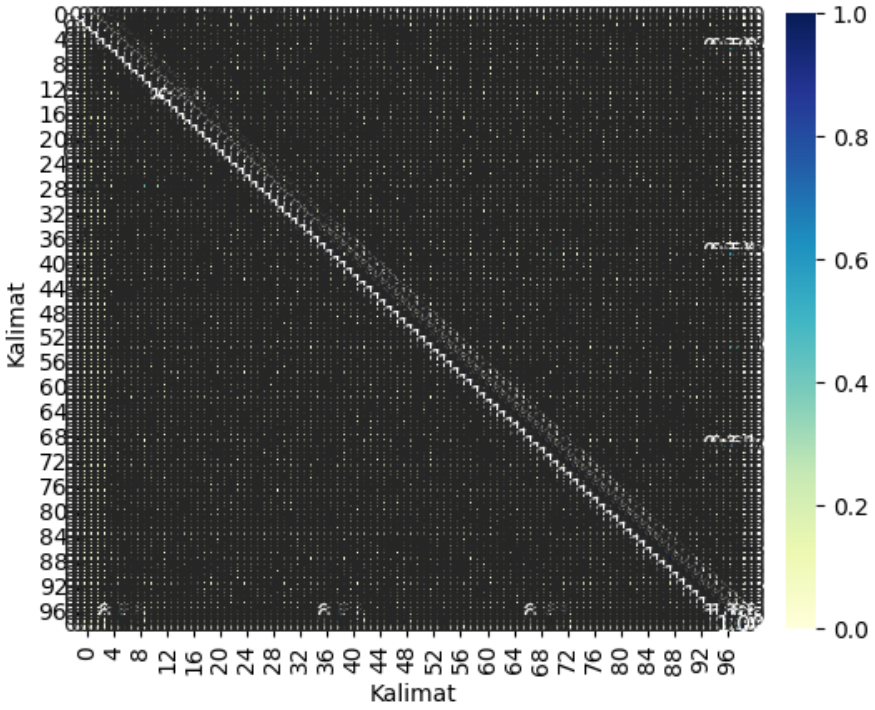
5. *Graph*

Grap disini dibuat untuk menggambarkan nilai jarak antara kalimat satu dengan kalimat yang lain berdasarkan nilai Cosinuss Similarity

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

# Membuat grafik jaringan
G = nx.Graph()

# Menambahkan simpul (kalimat)
for i in range(len(cosine_sim_matrix)):
    G.add_node(i, label=df_kalimat.index[i])  # Menggunakan label kalimat

# Menambahkan tepian (hubungan) berdasarkan kesamaan kosinus
for i in range(len(cosine_sim_matrix)):
    for j in range(i+1, len(cosine_sim_matrix)):
        similarity = cosine_sim_matrix[i][j]
        if similarity > 0:  # Atur threshold sesuai kebutuhan
            G.add_edge(i, j, weight=similarity)

# Menggambar grafik jaringan
pos = nx.spring_layout(G, seed=42)  # Menggunakan layout spring
labels = nx.get_node_attributes(G, 'label')

nx.draw(G, pos, with_labels=True, node_color='lightblue', node_size=500, font_size=8, font_colo
nx.draw_networkx_edge_labels(G, pos, edge_labels={(i, j): f"{similarity:.2f}" for i, j, similar

plt.show()
```




```
import seaborn as sns
import matplotlib.pyplot as plt

# Membuat heatmap dari matriks kesamaan kosinus
sns.heatmap(cosine_sim_matrix, cmap='YlGnBu', annot=True, fmt=".2f")

# Menambahkan label ke sumbu x dan y
plt.xlabel('Kalimat')
plt.ylabel('Kalimat')

# Menampilkan grafik
plt.show()
```


Kalimat
Kalimat
0
4
8
12
16
20
24
28
32
36
40
44
48
52
56
60
64
68
72
76
80
84
88
92
96
1.0
0.8
0.6
0.4
0.2
0.0

9

6. Closeness Centrality

Pada proses ini Closeness Centrality digunakan untuk menghitung bobot sebuah node berdasarkan jumlah jarak terpendek antara node(i) dengan node lainnya. Berikut rumus Closeness Centrality

$$C_c(i) = \frac{n-1}{\sum_{j=1}^{n} d(i,j)}$$

```
import networkx as nx

closeness_centrality = nx.closeness_centrality(G)

sorted_closeness = sorted(closeness_centrality.items(), key=lambda x: x[1], reverse=True)

for node, closeness in sorted_closeness:
    print(f"Simpul {node}: Closeness Centrality = {closeness:.4f}")
```

```
Simpul 61: Closeness Centrality = 0.8522
Simpul 1: Closeness Centrality = 0.8376
Simpul 56: Closeness Centrality = 0.8305
Simpul 18: Closeness Centrality = 0.8099
Simpul 43: Closeness Centrality = 0.8099
Simpul 27: Closeness Centrality = 0.7903
Simpul 35: Closeness Centrality = 0.7903
Simpul 39: Closeness Centrality = 0.7903
Simpul 25: Closeness Centrality = 0.7840
Simpul 48: Closeness Centrality = 0.7840
Simpul 92: Closeness Centrality = 0.7840
Simpul 8: Closeness Centrality = 0.7778
Simpul 15: Closeness Centrality = 0.7778
Simpul 79: Closeness Centrality = 0.7778
Simpul 19: Closeness Centrality = 0.7717
Simpul 37: Closeness Centrality = 0.7656
Simpul 54: Closeness Centrality = 0.7656
Simpul 57: Closeness Centrality = 0.7538
Simpul 53: Closeness Centrality = 0.7481
```

```
Simpul 59: Closeness Centrality = 0.7481
Simpul 68: Closeness Centrality = 0.7481
Simpul 91: Closeness Centrality = 0.7481
Simpul 95: Closeness Centrality = 0.7481
Simpul 44: Closeness Centrality = 0.7424
Simpul 5: Closeness Centrality = 0.7368
Simpul 30: Closeness Centrality = 0.7368
Simpul 84: Closeness Centrality = 0.7368
Simpul 93: Closeness Centrality = 0.7368
Simpul 16: Closeness Centrality = 0.7313
Simpul 64: Closeness Centrality = 0.7259
Simpul 75: Closeness Centrality = 0.7259
Simpul 89: Closeness Centrality = 0.7259
Simpul 0: Closeness Centrality = 0.7206
Simpul 23: Closeness Centrality = 0.7153
Simpul 50: Closeness Centrality = 0.7101
Simpul 70: Closeness Centrality = 0.7101
Simpul 72: Closeness Centrality = 0.7000
Simpul 74: Closeness Centrality = 0.7000
Simpul 49: Closeness Centrality = 0.6950
Simpul 7: Closeness Centrality = 0.6901
Simpul 88: Closeness Centrality = 0.6901
Simpul 4: Closeness Centrality = 0.6853
Simpul 24: Closeness Centrality = 0.6853
Simpul 40: Closeness Centrality = 0.6853
Simpul 6: Closeness Centrality = 0.6806
Simpul 86: Closeness Centrality = 0.6806
Simpul 28: Closeness Centrality = 0.6712
Simpul 41: Closeness Centrality = 0.6712
Simpul 45: Closeness Centrality = 0.6667
Simpul 11: Closeness Centrality = 0.6622
Simpul 26: Closeness Centrality = 0.6622
Simpul 81: Closeness Centrality = 0.6622
Simpul 38: Closeness Centrality = 0.6577
Simpul 87: Closeness Centrality = 0.6577
Simpul 13: Closeness Centrality = 0.6533
Simpul 65: Closeness Centrality = 0.6533
Simpul 42: Closeness Centrality = 0.6490
Simpul 20: Closeness Centrality = 0.6447
Simpul 62: Closeness Centrality = 0.6364
Simpul 66: Closeness Centrality = 0.6364
Simpul 69: Closeness Centrality = 0.6364
Simpul 12: Closeness Centrality = 0.6242
Simpul 14: Closeness Centrality = 0.6203
Simpul 2: Closeness Centrality = 0.6164
```

```
Simpul 60: Closeness Centrality = 0.6164
Simpul 76: Closeness Centrality = 0.6164
Simpul 3: Closeness Centrality = 0.6125
Simpul 34: Closeness Centrality = 0.6125
Simpul 71: Closeness Centrality = 0.6125
Simpul 85: Closeness Centrality = 0.6125
Simpul 80: Closeness Centrality = 0.6087
Simpul 90: Closeness Centrality = 0.6087
Simpul 36: Closeness Centrality = 0.6049
Simpul 58: Closeness Centrality = 0.6049
Simpul 67: Closeness Centrality = 0.6012
Simpul 78: Closeness Centrality = 0.5939
Simpul 9: Closeness Centrality = 0.5868
Simpul 47: Closeness Centrality = 0.5868
Simpul 10: Closeness Centrality = 0.5833
Simpul 63: Closeness Centrality = 0.5799
Simpul 31: Closeness Centrality = 0.5765
Simpul 33: Closeness Centrality = 0.5698
Simpul 29: Closeness Centrality = 0.5665
Simpul 94: Closeness Centrality = 0.5600
Simpul 32: Closeness Centrality = 0.5568
Simpul 55: Closeness Centrality = 0.5537
Simpul 83: Closeness Centrality = 0.5506
Simpul 21: Closeness Centrality = 0.5444
Simpul 51: Closeness Centrality = 0.5414
Simpul 17: Closeness Centrality = 0.5385
Simpul 82: Closeness Centrality = 0.5355
Simpul 22: Closeness Centrality = 0.5269
Simpul 96: Closeness Centrality = 0.5131
Simpul 97: Closeness Centrality = 0.5131
Simpul 98: Closeness Centrality = 0.5131
Simpul 46: Closeness Centrality = 0.5052
Simpul 73: Closeness Centrality = 0.4949
Simpul 77: Closeness Centrality = 0.4900
Simpul 52: Closeness Centrality = 0.4851
```

10

7. PageRank

```
  G = nx.DiGraph(nx.path_graph(4))
  pr = nx.pagerank(G, alpha=0.9)
  pr
```

```
{0: 0.1724140124772394,
 1: 0.3275859875227606,
 2: 0.3275859875227606,
 3: 0.1724140124772394}
```

11

8. EigenVector Centrality

EigenVector digunakan untuk menghitung sentralitas sebuah node dengan menambahkan sentralitas pendahulunya. Berikut nilai persamaan dari Eigen-Vector

$$\lambda x_i = \sum_{j \to i} x_j$$

```
G = nx.path_graph(4)
centrality = nx.eigenvector_centrality(G)
sorted((v, f"{c:0.2f}") for v, c in centrality.items())
```

```
[(0, '0.37'), (1, '0.60'), (2, '0.60'), (3, '0.37')]
```

12

Tampilan Dokumen dengan Nilai EigenVector

```
import networkx as nx

# Membuat grafik jaringan (contoh: grafik jalur)
G = nx.path_graph(4)

# Menghitung eigenvector centrality
centrality = nx.eigenvector_centrality(G)

# Data berita (dalam bentuk daftar)
berita =df_kalimat['Tokenisasi']

# Menampilkan kalimat dari eigenvector centrality dan mengaitkannya dengan dokumen berita
for node, centrality_score in centrality.items():
    if 0 <= node < len(berita):
        kalimat = f"Dokumen berita: '{berita[node]}' memiliki Eigenvector Centrality sebesar {c
        print(kalimat)
```

```
Dokumen berita: 'sekitar 8.000 peserta meriahkan ajang lari yang digelar lazada indonesia bertajuk laza
Dokumen berita: 'para peserta lomba lari ini tak hanya dari dalam kota, tetapi juga datang dari luar kot
Dokumen berita: 'semuanya terbuka untuk umum dan masyarakat.' memiliki Eigenvector Centrality sebesar
Dokumen berita: '"pesertanya kita terbuka untuk semua.' memiliki Eigenvector Centrality sebesar 0.37
```

13

Summary

In summary, this book has no content whatsoever.

References