

*Jane Doe and Max Power*

---

# *Quarto CRC Book*

To blah, blah, and blah.

---

## *Table of contents*

---

<b>Preface</b>	<b>v</b>
<b>Preface</b>	<b>v</b>
Software conventions . . . . .	v
Acknowledgments . . . . .	v
<b>1 Tugas Crawling_berita_Kompas</b>	<b>1</b>
1.1 Menginstall dan mengimport library yang dibutuhkan . . . . .	1
1.2 Web Scraping Berita Kompas dan Penyimpanan ke Google Drive . . . . .	2
1.3 Import dan Membaca Data CSV . . . . .	3
1.4 Mengambil salah satu berita untuk menjadi sampel data yang akan dianalisis . . . . .	4
1.5 Pra-pemrosesan Teks / Preprocessing . . . . .	4
1.6 Metode pengukuran TF-IDF . . . . .	5
1.7 Perhitungan Kemiripan Kosinus . . . . .	6
1.8 Visualisasi Hubungan Antar Kalimat Menggunakan Graf (Graph) berdasarkan Cosine Similarity . . . . .	7
1.9 Closeness . . . . .	9
1.9.1 Hasil penggunaan Closeness . . . . .	10
1.10 Menghitung peringkat halaman dengan Pagerank . . . . .	10
1.10.1 Hasil Pagerank . . . . .	12
1.11 menggunakan metode eigenvector centrality untuk menghitung sentralitas eigenvector pada graf . . . . .	12
1.11.1 Hasil eigen vector . . . . .	13
<b>2 Summary</b>	<b>15</b>
<b>References</b>	<b>17</b>
<b>References</b>	<b>17</b>



---

# *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

---

## *Tugas Crawling\_berita\_Kompas*

---

- Nama : Muhammad Adam Zaky Jiddyansah
  - NIM : 210411100234
  - Kelas : Penambangan dan Pencarian Web B
- 

### 1.1 Menginstall dan mengimport library yang dibutuhkan

Libraries yang Digunakan \* BeautifulSoup (bs4): Library untuk melakukan web scraping dan ekstraksi data HTML. \* Requests: Library untuk membuat permintaan HTTP ke situs web dan mendapatkan konten halaman. \* Pandas (pd): Library untuk mengelola dan menyimpan data dalam format CSV.

```
from bs4 import BeautifulSoup as soup
import requests
import pandas as pd
import nltk
from nltk.tokenize import sent_tokenize
import re
from nltk.corpus import stopwords
from sklearn.feature_extraction.text import TfidfVectorizer, CountVectorizer, TfidfTransformer
from sklearn.metrics.pairwise import cosine_similarity
import networkx as nx
import matplotlib.pyplot as plt

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

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data] Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data] Package stopwords is already up-to-date!
```

True

---

## 1.2 Web Scraping Berita Kompas dan Penyimpanan ke Google Drive

Kode ini menggunakan teknik web scraping untuk mengumpulkan judul dan isi berita dari 200 halaman indeks situs Kompas.com. Berikut adalah penjelasan komponen utama dari kode tersebut:

**Proses Web Scraping dan menyimpan data** \* Iterasi sebanyak 5 (karna yang akan dipreprocessing kali ini cuman 1 berita) halaman indeks(tergantung indeks pada website) di situs Kompas.com. \* Untuk setiap halaman, melakukan permintaan HTTP dan mendapatkan konten HTML. \* Menggunakan BeautifulSoup untuk mengekstrak daftar berita dari halaman tersebut. \* Untuk setiap berita, mengakses halaman individu untuk mengambil judul dan isi berita. \* Data judul dan isi berita ditambahkan ke dalam dictionary csv. Penyimpanan Data \* Setelah proses scraping selesai, data disimpan dalam file CSV dengan nama "Data\_BeritaKompas.csv" terlebih dahulu di environment Colab. \* File CSV tersebut disalin ke dalam Google Drive pada path "/content/drive/My Drive/Penambangan WEB/Data/".

```
from google.colab import drive
drive.mount('/content/drive')

csv = {"Judul": [], "Berita": []}

for i in range(1, 5):
    url = "https://indeks.kompas.com/?page={}".format(i)
    client = requests.get(url)
    page_html = client.content
    page_soup = soup(page_html, "html.parser")
    berita = page_soup.findAll("div", {"class": "latest--indeks mt2 clearfix"})

    for h in berita:
        r = requests.get(h.select_one('a.article__link')['href'])
        page = soup(r.content, "html.parser")
        halaman_isi = page.select_one("div", {"class": "col-bs10-10"})

        judul_berita = halaman_isi.select("h1", {"class": "read__title"})
        judul = judul_berita[0].text
```



```
isi_berita = halaman_isi.findAll('p')
isi = '\n'.join([p.get_text() for p in isi_berita])

csv["Judul"].append(judul)
csv["Berita"].append(isi)

data = pd.DataFrame(csv)
csv_path_drive = '/content/drive/My Drive/Penambangan WEB/Data/Data_BeritaKompas.csv'
data.to_csv(csv_path_drive, index=False)
```

Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive")

### 1.3 Import dan Membaca Data CSV

Menggunakan Pandas untuk membaca file CSV yang telah disimpan sebelumnya. File CSV tersebut berisi data judul dan isi berita dari Kompas.com yang telah di-web scrape sebelumnya, lalu menampilkan data yang telah dibaca dari file CSV dalam bentuk DataFrame untuk dapat dianalisis lebih lanjut.

```
from google.colab import drive
drive.mount('/content/drive')
file_path = "/content/drive/My Drive/Penambangan WEB/Data/Data_BeritaKompas.csv"
df = pd.read_csv(file_path)
df
```

Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive")

	Judul	Berita
0	Chris Martin Singgung Kemacetan Jakarta	Chris Martin Singgung Kemacetan Jakarta\nT
1	Pulihkan Nama Baik Mahasiswa yang Dituduh Laku...	Pulihkan Nama Baik Mahasiswa yang Dituduh
2	Peritel: Waspada, Kenaikan Harga Gula Bisa Bik...	Peritel: Waspada, Kenaikan Harga Gula Bisa E
3	KPK: OTT di Bondowoso Terkait Pengurusan Perka...	KPK: OTT di Bondowoso Terkait Pengurusan

---

## 1.4 Mengambil salah satu berita untuk menjadi sampel data yang akan dianalisis

```
Berita = df['Berita'].iloc[0]
Berita
```

```
'Chris Martin Singgung Kemacetan Jakarta\nTim Redaksi\nJAKARTA, KOMPAS.com - Band Coldplay ikut merasa
```

---

## 1.5 Pra-pemrosesan Teks / Preprocessing

Mengambil teks sebagai input dan melakukan beberapa langkah pra-pemrosesan pada teks dalam bahasa Indonesia. Langkah-langkah tersebut mencakup penghapusan angka, pembersihan karakter khusus, konversi teks ke huruf kecil, penghapusan kata-kata umum (stopwords), dan penggabungan kata-kata yang telah melalui proses sebelumnya. Tujuannya adalah membersihkan dan mempersiapkan teks agar dapat digunakan lebih lanjut dalam analisis atau pemrosesan lanjutan.

```
def preprocessing(text):
    text = re.sub(r'\d+', '', text)
    text = re.sub(r'^\w\s.', '', text)
    text = text.lower()

    stop_words = set(stopwords.words('indonesian'))
    words = text.split()
    filtered_words = [word for word in words if word.lower() not in stop_words]

    preprocessing_text = ' '.join(filtered_words)

    return preprocessing_text

kalimat_preprocessing = preprocessing(Berita)

kalimat = nltk.sent_tokenize(Berita)
kalimat_preprocessing = nltk.sent_tokenize(kalimat_preprocessing)
```

## 1.6 Metode pengukuran TF-IDF

Mengonversi teks yang sudah melewati tahap pra-pemrosesan menjadi representasi matriks numerik menggunakan skema TF-IDF. Matriks ini mencerminkan bobot TF-IDF dari setiap kata dalam setiap kalimat. Setelah transformasi, DataFrame pandas dibuat untuk menyajikan matriks tersebut dengan kata-kata sebagai kolom dan setiap baris mewakili satu kalimat, menunjukkan bobot TF-IDF dari kata-kata tersebut dalam kalimat tersebut.

```
tfidf_vectorizer = TfidfVectorizer()
tfidf_matrix_preprocessing = tfidf_vectorizer.fit_transform(kalimat_preprocessing)

terms = tfidf_vectorizer.get_feature_names_out()
tfidf_preprocessing = pd.DataFrame(data=tfidf_matrix_preprocessing.toarray(), columns=terms)

tfidf_preprocessing
```

	acara	adrian	adventure	anaknya	antusiasme	astrini	baca	bahagia	band	bera
0	0.00000	0.000000	0.000000	0.000000	0.000000	0.00000	0.000000	0.000000	0.243498	0.00
1	0.00000	0.000000	0.000000	0.000000	0.000000	0.00000	0.000000	0.000000	0.000000	0.00
2	0.00000	0.000000	0.000000	0.000000	0.000000	0.00000	0.000000	0.000000	0.000000	0.00
3	0.00000	0.000000	0.000000	0.000000	0.000000	0.00000	0.000000	0.000000	0.000000	0.00
4	0.00000	0.000000	0.000000	0.000000	0.000000	0.00000	0.000000	0.000000	0.000000	0.00
5	0.00000	0.000000	0.000000	0.000000	0.000000	0.00000	0.000000	0.000000	0.000000	0.00
6	0.00000	0.000000	0.000000	0.000000	0.000000	0.00000	0.000000	0.482375	0.000000	0.00
7	0.00000	0.000000	0.000000	0.000000	0.000000	0.00000	0.000000	0.000000	0.000000	0.00
8	0.00000	0.179230	0.000000	0.201959	0.201959	0.00000	0.163103	0.000000	0.000000	0.00
9	0.00000	0.000000	0.000000	0.000000	0.000000	0.00000	0.000000	0.000000	0.000000	0.00
10	0.45136	0.000000	0.000000	0.000000	0.000000	0.45136	0.000000	0.000000	0.000000	0.00
11	0.00000	0.000000	0.000000	0.000000	0.000000	0.00000	0.000000	0.000000	0.000000	0.00
12	0.00000	0.000000	0.000000	0.000000	0.000000	0.00000	0.000000	0.000000	0.000000	0.00
13	0.00000	0.000000	0.000000	0.000000	0.000000	0.00000	0.000000	0.000000	0.000000	0.00
14	0.00000	0.000000	0.000000	0.000000	0.000000	0.00000	0.000000	0.000000	0.000000	0.00
15	0.00000	0.000000	0.000000	0.000000	0.000000	0.00000	0.000000	0.000000	0.000000	0.00
16	0.00000	0.203862	0.000000	0.000000	0.000000	0.00000	0.185519	0.000000	0.000000	0.22
17	0.00000	0.000000	0.000000	0.000000	0.000000	0.00000	0.000000	0.000000	0.000000	0.00
18	0.00000	0.000000	0.000000	0.000000	0.000000	0.00000	0.000000	0.000000	0.000000	0.00
19	0.00000	0.000000	0.000000	0.000000	0.000000	0.00000	0.000000	0.000000	0.000000	0.00
20	0.00000	0.000000	0.000000	0.000000	0.000000	0.00000	0.000000	0.000000	0.000000	0.00
21	0.00000	0.000000	0.000000	0.000000	0.000000	0.00000	0.000000	0.000000	0.000000	0.00
22	0.00000	0.000000	0.000000	0.000000	0.000000	0.00000	0.000000	0.000000	0.000000	0.00
23	0.00000	0.000000	0.000000	0.000000	0.000000	0.00000	0.000000	0.000000	0.000000	0.00

	acara	adrian	adventure	anaknya	antusiasme	astrini	baca	bahagia	band	bera
24	0.00000	0.000000	0.189761	0.000000	0.000000	0.00000	0.000000	0.000000	0.000000	0.00
25	0.00000	0.000000	0.000000	0.000000	0.000000	0.00000	0.236662	0.000000	0.000000	0.00

## 1.7 Perhitungan Kemiripan Kosinus

Menghitung kemiripan kosinus antara semua pasangan kalimat menggunakan matriks TF-IDF yang telah dihasilkan sebelumnya. Kemudian, hasil kemiripan kosinus tersebut disimpan dalam DataFrame untuk kemudahan analisis.

**Rumus Perhitungan Kemiripan Kosinus:** \* Untuk dua vektor A dan B, kemiripan kosinus dihitung dengan rumus

$$\text{Similarity}(A, B) = \frac{A \cdot B}{\|A\| \cdot \|B\|}$$

- Di sini,  $\cdot$  adalah operasi perkalian dot (dot product), dan  $\| \cdot \|$  adalah norma Euclidean. Dalam konteks ini, vektor A dan B adalah vektor representasi TF-IDF dari dua kalimat. Hasilnya adalah skalar yang menunjukkan seberapa mirip dua kalimat tersebut, dengan nilai 1 menunjukkan kemiripan sempurna.

```
cosine_sim_preprocessing = cosine_similarity(tfidf_matrix_preprocessing, tfidf_matrix_preprocessing)
cosine_sim_df = pd.DataFrame(cosine_sim_preprocessing, columns=range(len(kalimat_preprocessing)))
cosine_sim_df
```

	0	1	2	3	4	5	6	7	8	9	...	1
0	1.000000	0.111710	0.179270	0.0	0.0	0.122519	0.075810	0.235482	0.120405	0.026906	...	0
1	0.111710	1.000000	0.063221	0.0	0.0	0.129621	0.080204	0.083044	0.429423	0.000000	...	0
2	0.179270	0.063221	1.000000	0.0	0.0	0.000000	0.000000	0.209022	0.049563	0.000000	...	0
3	0.000000	0.000000	0.000000	1.0	0.0	0.000000	0.000000	0.000000	0.000000	0.000000	...	0
4	0.000000	0.000000	0.000000	0.0	1.0	0.000000	0.000000	0.000000	0.000000	0.000000	...	0
5	0.122519	0.129621	0.000000	0.0	0.0	1.000000	0.242712	0.000000	0.000000	0.000000	...	0
6	0.075810	0.080204	0.000000	0.0	0.0	0.242712	1.000000	0.000000	0.000000	0.000000	...	0
7	0.235482	0.083044	0.209022	0.0	0.0	0.000000	0.000000	1.000000	0.065103	0.000000	...	0
8	0.120405	0.429423	0.049563	0.0	0.0	0.000000	0.000000	0.065103	1.000000	0.044632	...	0
9	0.026906	0.000000	0.000000	0.0	0.0	0.000000	0.000000	0.000000	0.044632	1.000000	...	0
10	0.000000	0.000000	0.000000	0.0	0.0	0.000000	0.000000	0.000000	0.000000	0.119726	...	0
11	0.035919	0.038001	0.000000	0.0	0.0	0.114998	0.071156	0.000000	0.000000	0.000000	...	0
12	0.177402	0.062125	0.000000	0.0	0.0	0.188002	0.233883	0.000000	0.000000	0.099111	...	0
13	0.000000	0.000000	0.000000	0.0	0.0	0.000000	0.000000	0.000000	0.000000	0.203673	...	0
14	0.196650	0.000000	0.000000	0.0	0.0	0.000000	0.194784	0.000000	0.000000	0.000000	...	0

### 1.8 Visualisasi Hubungan Antar Kalimat Menggunakan Graf (Graph) berdasarkan Cosine Similarity7

	0	1	2	3	4	5	6	7	8	9	...	1
15	0.072315	0.076507	0.000000	0.0	0.0	0.231523	0.143257	0.000000	0.000000	0.057155	...	0
16	0.236566	0.587631	0.112749	0.0	0.0	0.115584	0.071519	0.148102	0.644671	0.050766	...	1
17	0.028012	0.000000	0.000000	0.0	0.0	0.000000	0.000000	0.000000	0.046467	0.694276	...	0
18	0.000000	0.000000	0.000000	0.0	0.0	0.000000	0.000000	0.000000	0.000000	0.000000	...	0
19	0.000000	0.000000	0.000000	0.0	0.0	0.000000	0.000000	0.000000	0.000000	0.155534	...	0
20	0.092397	0.130328	0.082015	0.0	0.0	0.000000	0.000000	0.107731	0.058152	0.000000	...	0
21	0.000000	0.000000	0.000000	0.0	0.0	0.000000	0.000000	0.000000	0.000000	0.000000	...	0
22	0.000000	0.000000	0.000000	0.0	0.0	0.000000	0.000000	0.000000	0.000000	0.000000	...	0
23	0.108834	0.115143	0.000000	0.0	0.0	0.348442	0.215603	0.000000	0.000000	0.000000	...	0
24	0.017488	0.074414	0.000000	0.0	0.0	0.000000	0.000000	0.000000	0.105862	0.346171	...	0
25	0.179077	0.036470	0.000000	0.0	0.0	0.000000	0.000000	0.000000	0.173488	0.032380	...	0

### 1.8 Visualisasi Hubungan Antar Kalimat Menggunakan Graf (Graph) berdasarkan Cosine Similarity

Membuat graf yang merepresentasikan hubungan antara kalimat-kalimat berdasarkan nilai cosine similarity. Setiap kalimat direpresentasikan sebagai node, dan garis yang menghubungkan dua node menunjukkan tingkat kemiripan antara kalimat tersebut. Semakin panjang dan tebal garis, semakin tinggi tingkat kemiripan antara dua kalimat. Graf ini membantu memvisualisasikan struktur hubungan antar kalimat dalam teks.

```
G_preprocessing = nx.Graph()

for i in range(len(cosine_sim_preprocessing)):
    G_preprocessing.add_node(i)

for i in range(len(cosine_sim_preprocessing)):
    for j in range(len(cosine_sim_preprocessing)):
        similarity = cosine_sim_preprocessing[i][j]
        if similarity > 0.1 and i != j:
            G_preprocessing.add_edge(i, j, weight=similarity)

pos = nx.circular_layout(G_preprocessing)

labels = {i: f'K {i}' for i in G_preprocessing.nodes()}

edge_labels = {(i, j): round(data['weight'], 2) for i, j, data in G_preprocessing.edges(data=True)}
```

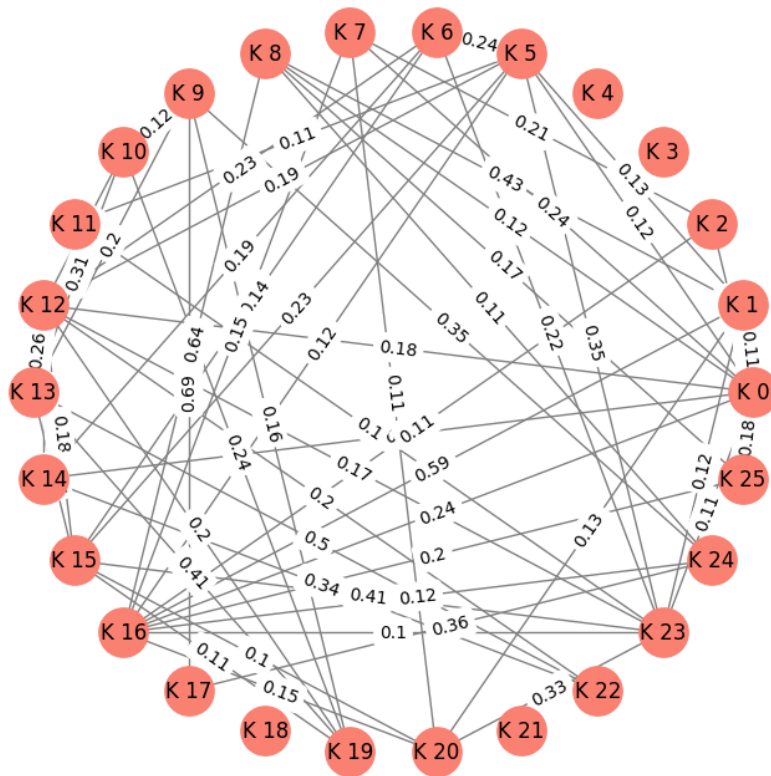
```

node_size = 800

node_color = 'salmon'
edge_color = 'gray'

plt.figure(figsize=(7, 7))
nx.draw(G_preprocessing, pos, with_labels=True, labels=labels, node_size=node_size, node_color=
nx.draw_networkx_edge_labels(G_preprocessing, pos, edge_labels=edge_labels)
plt.show()

```



## 1.9 Closeness

Menghitung dan menganalisis Closeness Centrality pada graf G\_preprocessing. Rumus Perhitungan Closeness Centrality:

```
# Menghitung closeness centrality
closeness_centrality_preprocessing = nx.closeness_centrality(G_preprocessing)

# Membuat DataFrame dari nilai Closeness Centrality
centrality_df_preprocessing = pd.DataFrame(closeness_centrality_preprocessing.items(), columns=

# Mengurutkan DataFrame berdasarkan Closeness Centrality dari yang terbesar
centrality_df_sorted_preprocessing = centrality_df_preprocessing.sort_values(by='Closeness Cent

# Menampilkan tabel Closeness Centrality yang telah diurutkan
centrality_df_sorted_preprocessing
```

	Node	Closeness Centrality Preprocessing
0	0	0.518824
12	12	0.518824
23	23	0.504000
16	16	0.490000
5	5	0.490000
15	15	0.441000
20	20	0.420000
1	1	0.420000
13	13	0.410233
19	19	0.400909
24	24	0.400909
8	8	0.400909
14	14	0.400909
6	6	0.392000
7	7	0.383478
10	10	0.383478
25	25	0.367500
2	2	0.367500
22	22	0.360000
9	9	0.360000
11	11	0.326667
17	17	0.309474
3	3	0.000000
21	21	0.000000

	Node	Closeness Centrality Preprocessing
18	18	0.000000
4	4	0.000000

### 1.9.1 Hasil penggunaan Closeness

```
# Mengambil tiga baris teratas dari DataFrame yang sudah diurutkan
top_three_closeness_preprocessing = centrality_df_sorted_preprocessing.head(3)

# Mendapatkan indeks node dari tiga baris teratas
top_node_indices_preprocessing = top_three_closeness_preprocessing['Node']

# Menampilkan kalimat yang sesuai dengan node-node teratas
for node_index in top_node_indices_preprocessing:
    print(f"Node Node Preprocessing {node_index}: {kalimat[node_index]}")
```

Node Node Preprocessing 0: Chris Martin Singgung Kemacetan Jakarta

Tim Redaksi

JAKARTA, KOMPAS.com – Band Coldplay ikut merasakan kemacetan Kota Jakarta saat pertama kali ke Indonesia.

Node Node Preprocessing 12: Chris Martin juga menyampaikan ucapan terima kasih kepada penonton dan meng-

Node Node Preprocessing 23: Saya sangat senang di sini.

## 1.10 Menghitung peringkat halaman dengan Pagerank

Rumus Perhitungan Manual PageRank:

- PageRank mengukur pentingnya suatu node dalam sebuah graf berdasarkan seberapa banyak node lain yang mengarah ke node tersebut. Rumusnya adalah sebagai berikut:

$$PR(x) = (1 - d) + d \left( \frac{L(y)}{PR(y)} \right)$$

- PR(x) adalah PageRank dari node x.
- d adalah faktor damping, biasanya diatur sebagai 0,85.
- PR(y) adalah PageRank dari node yang memiliki tautan ke node y.
- L(y) adalah jumlah tautan keluar dari node y.
- Iterasi dilakukan hingga konvergensi atau sejumlah iterasi tertentu. Semakin tinggi nilai PageRank, semakin “penting” node tersebut dalam graf.

```
# Menghitung closeness centrality
pagerank_preprocessing = nx.pagerank(G_preprocessing)
```



```
# Membuat DataFrame dari nilai Closeness Centrality
pagerank_df_preprocessing = pd.DataFrame(pagerank_preprocessing.items(), columns=['Node', 'Page Rank Preprocessing'])

# Mengurutkan DataFrame berdasarkan Closeness Centrality dari yang terbesar
pagerank_df_sorted_preprocessing = pagerank_df_preprocessing.sort_values(by='Page Rank Preprocessing', ascending=False)

# Menampilkan tabel Closeness Centrality yang telah diurutkan
pagerank_df_sorted_preprocessing
```

	Node	Page Rank Preprocessing
16	16	0.080798
12	12	0.075970
23	23	0.065561
13	13	0.060866
0	0	0.058958
9	9	0.053334
5	5	0.052513
1	1	0.050648
8	8	0.050344
15	15	0.045809
14	14	0.044665
19	19	0.039017
17	17	0.038757
6	6	0.036289
22	22	0.035917
24	24	0.034727
20	20	0.031017
10	10	0.030352
7	7	0.029462
2	2	0.022710
25	25	0.022676
11	11	0.013060
4	4	0.006637
3	3	0.006637
18	18	0.006637
21	21	0.006637

### 1.10.1 Hasil Pagerank

```
# Mengambil tiga baris teratas dari DataFrame yang sudah diurutkan
top_three_pagerank_preprocessing = pagerank_df_sorted_preprocessing.head(3)

# Mendapatkan indeks node dari tiga baris teratas
top_node_indices = top_three_pagerank_preprocessing['Node']

# Menampilkan kalimat yang sesuai dengan node-node teratas
for node_index in top_node_indices:
    print(f"Node {node_index}: {kalimat[node_index]}")
```

Node 16: Baca juga: Momen-momen Seru di Konser Coldplay di Jakarta

KOMPAS.com/RODERICK ADRIAN Vokalis Coldplay Chris Martin beraksi dalam konser Music of the Spheres Wor

Node 12: Chris Martin juga menyampaikan ucapan terima kasih kepada penonton dan mengaku senang bisa tam

Node 23: Saya sangat senang di sini.

---

## 1.11 menggunakan metode eigenvector centrality untuk menghitung sentralitas eigenvector pada graf

Rumus Perhitungan Manual Eigenvector Centrality: \* Eigenvector Centrality mengukur pentingnya suatu node dalam suatu graf berdasarkan pada seberapa banyak node lain yang terhubung ke node tersebut. Rumusnya adalah sebagai berikut:

$$x_i = \frac{1}{\lambda} \sum_{j=1}^n A_{ij} x_j$$

- $x_i$  adalah eigenvector centrality dari node  $i$
- $A_{ij}$  adalah elemen matriks ketetanggaan yang menunjukkan apakah ada tautan antara node  $i$  dan  $j$
- $\lambda$  adalah nilai eigen (eigenvalue) yang sesuai dengan eigenvector yang dicari.
- Iterasi dilakukan hingga konvergensi atau sejumlah iterasi tertentu. Semakin tinggi nilai eigenvector centrality, semakin “penting” node tersebut dalam graf.

```
# Menghitung closeness centrality
eigenvector_preprocessing = nx.eigenvector_centrality(G_preprocessing)
# Membuat DataFrame dari nilai Closeness Centrality
eigenvector_df_preprocessing = pd.DataFrame(eigenvector_preprocessing.items(), columns=['Node',
# Mengurutkan DataFrame berdasarkan Closeness Centrality dari yang terbesar
```

### 1.11 menggunakan metode eigenvector centrality untuk menghitung sentralitas eigenvector pada graf13

```
eigenvector_df_sorted_preprocessing = eigenvector_df_preprocessing.sort_values(by='Eigen Vector')
# Menampilkan tabel Closeness Centrality yang telah diurutkan
eigenvector_df_sorted_preprocessing
```

	Node	Eigen Vector Preprocessing
23	23	3.597098e-01
0	0	3.528884e-01
5	5	3.327743e-01
16	16	3.299490e-01
12	12	3.241979e-01
1	1	2.645868e-01
15	15	2.596958e-01
6	6	2.145504e-01
20	20	2.058069e-01
8	8	1.759210e-01
7	7	1.527652e-01
13	13	1.498928e-01
14	14	1.483843e-01
19	19	1.376246e-01
25	25	1.293166e-01
2	2	1.258297e-01
11	11	1.042786e-01
10	10	1.036360e-01
22	22	9.373707e-02
24	24	9.150515e-02
9	9	7.649236e-02
17	17	2.529830e-02
4	4	2.627584e-24
18	18	2.627584e-24
3	3	2.627584e-24
21	21	2.627584e-24

#### 1.11.1 Hasil eigen vector

```
# Mengambil tiga baris teratas dari DataFrame yang sudah diurutkan
top_three_eigenvector_preprocessing = eigenvector_df_sorted_preprocessing.head(3)
# Mendapatkan indeks node dari tiga baris teratas
top_node_indices = top_three_eigenvector_preprocessing['Node']
# Menampilkan kalimat yang sesuai dengan node-node teratas
for node_index in top_node_indices:
    print(f"Node {node_index}: {kalimat[node_index]}")
```

Node 23: Saya sangat senang di sini.

Node 0: Chris Martin Singgung Kemacetan Jakarta

Tim Redaksi

JAKARTA, KOMPAS.com - Band Coldplay ikut merasakan kemacetan Kota Jakarta saat pertama kali ke Indonesia.

Node 5: Kalian tidak sendirian, saya juga merasakannya," kata Chris Martin.

## 2

### *Summary*

In summary, this book has no content whatsoever.



---

## *References*

---

