

Jane Doe and Max Power

Quarto CRC Book

To blah, blah, and blah.

Table of contents

Preface	v
Preface	v
Software conventions	v
Acknowledgments	v
1 CINDY LAUNDIYA MARETHA	1
2 210411100037	3
3 1. Crawling Berita	5
4 Tampilan Data hasil Crawling Berita	7
5 Dokumen 1	9
6 1. Ekstraksi Kalimat pada Berita 1	11
7 Menghilangkan Kata-kata dan Tanda baca tidak penting	13
8 2. Menghitung Nilai TF-IDF dari Dokumen Berita 1	15
9 Nilai Vektor kata antar Kalimat	17
10 Nilai Vektor antar Kata dengan Kata	19
11 Nilai Cosinus Similarity Kata Kunci	21
12 Graph Kata Kunci	23
13 Kata Kunci pada Dokumen 1	25
14 3. Cosinuss Similarity pada Dokumen 1	29
15 4. Graph Kata Penting berdasarkan nilai TF-IDF pada Dokumen 1	37
16 5. Closeness Centrality Dokumen Berita 1	39

17 6. PageRank Dokumen 1	41
18 7. EigenVector dari Dokumen Berita 1	43
19 8. Mencari Kata Kunci	45
20 Dokumen 2	51
21 1. Ekstraksi Kalimat pada Dokumen Berita 2	53
22 Menghilangkan Kata dan Tanda tidak penting pada Dokumen 2	55
23 2. TF-IDF	57
24 3. Cosinuss Similarity	63
25 4. Graph	65
26 5. Closeness Centrality	67
27 6. PageRank	69
28 7. EigenVector	71
29 8. Mencari Kata Kunci Pada Dokumen 2	73
30 Summary	81
References	83
References	83

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 beautifulsoup4)

```
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, like Gecko) Chrome/91.0.4472.164 Safari/537.36'}
```



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 Kenya', 'Waktu': '13 Mei 2020'}
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 Jakarta', 'Waktu': '13 Mei 2020'}
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 2020'}
done[3] > PBSI Maklum Singapore Open Batal, tapi...
Data berhasil ditambahkan: {'Judul': 'Kualifikasi Olimpiade Mulai 2021, Richard Mainaky: Tahun Ini Pemanasan', 'Waktu': '13 Mei 2020'}
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', 'Waktu': '13 Mei 2020'}
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', 'Waktu': '13 Mei 2020'}
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', 'Waktu': '13 Mei 2020'}
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 Hutan', 'Waktu': '13 Mei 2020'}
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"]
konten = pd.DataFrame(content_column, columns=['Content'])
konten
```

	Content
0	sekitar 8.000 peserta meriahkan ajang lari yan...
1	brand apparel olahraga under armour mengadakan...
2	pp pbsi menilai wajar keputusan badminton worl...
3	pelatih bulutangkis ganda campuran richard mai...
4	pp pbsi merespons keputusan badminton world fe...
5	\r\r\rscroll to continue with content\r"equina...
6	\r\r\rscroll to continue with content\r
7	\r\r\rscroll to continue with content\r
8	awesome ðŸ˜? #openingceremony #rio2016 #olympi...

5

Dokumen 1

```
import pandas as pd

# Ambil satu dokumen dari baris pertama
dokumen_pertama = df.at[0, 'Content']

# Buat DataFrame dengan satu kolom dan satu baris
df_dokumen = pd.DataFrame({'Dokumen1': [dokumen_pertama]})

# Tampilkan DataFrame
df_dokumen
```

Dokumen1
0 sekitar 8.000 peserta meriahkan ajang lari yan...



6

1. Ekstraksi Kalimat pada Berita 1

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_dokumen['Dokumen1'].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...
5	scroll to continue with content\rintan melanju...
6	karena mencakup semua masyarakat, kegiatan ini...
7	harapannya yang bisa sehat bukan hanya individ...
8	"kemudian kita ingin berkontribusi, terutama d...
9	salah satunya adalah melalui kegiatan lazada run.
10	"kita ingin menambahkan sehat dalam hidup mere...
11	kata intan.selain itu, deputi bidang produk wi...
12	menurutnya kegiatan lazada bisa ikut mendorong...
13	sebab, baru pertama kali digelar kegiatan ini ...
14	"ini baru pertama kali sudah 8.000. artinya apa?
15	kegiatan kedua, ketiga, nanti pasti akan naik,...
16	"kami juga menantang lazada untuk menggelar ev...
17	tentunya kemenparekraf, khususnya bagian event...
18	kalau bisa suatu saat, maraton ini digelar di ...

Tokenisasi

19 sebelum berlari sesuai kategori, para peserta ...
20 bersama dengan 3 temannya, nunu telah datang s...
21 "saya suka lari, hobi.
22 terus lazada run disponsori produk bagus.
23 kegiatan ini seru, larinya disemangatin," untk...
24 bersama kedua temannya, dia mengikuti maraton ...
25 jadi sekalian latihan nanti ada half marathon ...
26 mereka juga bahkan memiliki pelatih dan menjag...
27 selain itu, tata dan temannya berharap kegiata...
28 setelah sebelumnya pertama kali digelar di vie...

7

Menghilangkan Kata-kata dan Tanda baca tidak penting

```
import pandas as pd
from nltk.tokenize import sent_tokenize
from nltk.corpus import stopwords
import string

# Misalnya, jika df adalah DataFrame yang memiliki kolom 'Content'
teks_berita = df_dokumen['Dokumen1'].values.tolist()

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

# Membuang kata-kata atau tanda baca yang tidak penting
stopwords_list = set(stopwords.words('english'))
cleaned_sentences = []

for sentence in kalimat:
    # Menghapus tanda baca
    sentence = sentence.translate(str.maketrans("", "", string.punctuation))

    # Menghapus angka
    sentence = ''.join([char for char in sentence if not char.isdigit()])

    # Mengubah teks menjadi huruf kecil
    sentence = sentence.lower()

    # Membuang kata-kata yang merupakan stopwords
    words = sentence.split()
    words = [word for word in words if word not in stopwords_list]

    # Menggabungkan kata-kata kembali menjadi kalimat
    cleaned_sentence = " ".join(words)
```

```

cleaned_sentences.append(cleaned_sentence)

# Membuat DataFrame baru
df_cleaned1 = pd.DataFrame(cleaned_sentences, columns=['Tokenisasi Dokumen1'])
df_cleaned1

```

	Tokenisasi Dokumen1
0	sekitar peserta meriahkan ajang lari yang dige...
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 runners...
5	scroll continue content intan melanjutkan kegi...
6	karena mencakup semua masyarakat kegiatan ini ...
7	harapannya yang bisa sehat bukan hanya individ...
8	kemudian kita ingin berkontribusi terutama dar...
9	salah satunya adalah melalui kegiatan lazada run
10	kita ingin menambahkan sehat dalam hidup mereka
11	kata intanselain itu deputi bidang produk wisa...
12	menurutnya kegiatan lazada bisa ikut mendorong...
13	sebab baru pertama kali digelar kegiatan ini s...
14	ini baru pertama kali sudah artinya apa
15	kegiatan kedua ketiga nanti pasti akan naik uc...
16	kami juga menantang lazada untuk menggelar eve...
17	tentunya kemenparekraf khususnya bagian event ...
18	kalau bisa suatu saat maraton ini digelar di d...
19	sebelum berlari sesuai kategori para peserta j...
20	bersama dengan temannya nunu telah datang seja...
21	saya suka lari hobi
22	terus lazada run disponsori produk bagus
23	kegiatan ini seru larinya disemangatin ungkap ...
24	bersama kedua temannya dia mengikuti maraton d...
25	jadi sekalian latihan nanti ada half marathon ...
26	mereka juga bahkan memiliki pelatih dan menjag...
27	selain itu tata dan temannya berharap kegiatan...
28	setelah sebelumnya pertama kali digelar di vie...

8

2. Menghitung Nilai TF-IDF dari Dokumen Berita 1

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

$$w_{ij} = tf_{ij} \times idf_j$$
$$w_{ij} = tf_{ij} \times \log(D/df_j)$$

Dimana w_{ij} merupakan bobot dari term(j) terhadapn dokumen(i). Sedangkan tf_{ij} merupakan jumlah kemunculan term(j) dalam dokumen(i). Untuk D sendiri merupakan jumlah semua dokumen yang ada pada data dan df_j merupakan jumlah dokumen yang mengandung term(j)



9

Nilai Vektor kata antar Kalimat

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

# Data kalimat (contoh)
kalimat = df_cleaned1['Tokenisasi Dokumen1']

# Membuat objek TfidfVectorizer
tfidf_vectorizer = TfidfVectorizer()

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

# Mengonversi matriks TF-IDF ke DataFrame Pandas
tfidf_kata1 = pd.DataFrame(tfidf_matrix.toarray(), columns=tfidf_vectorizer.get_feature_names_out())

# Menampilkan tabel TF-IDF
tfidf_kata1
```

	acaranya	ada	adalah	aja	ajang	akan	anakanak	antusias	antusiasme	apa
0	0.000000	0.00000	0.000000	0.00000	0.29658	0.000000	0.000000	0.000000	0.00000	0.000000
1	0.000000	0.00000	0.000000	0.00000	0.00000	0.000000	0.000000	0.000000	0.00000	0.000000
2	0.000000	0.00000	0.000000	0.00000	0.00000	0.000000	0.000000	0.000000	0.00000	0.000000
3	0.000000	0.00000	0.000000	0.00000	0.00000	0.000000	0.000000	0.000000	0.00000	0.000000
4	0.000000	0.00000	0.000000	0.00000	0.00000	0.000000	0.229011	0.000000	0.00000	0.000000
5	0.000000	0.00000	0.000000	0.00000	0.00000	0.000000	0.000000	0.000000	0.00000	0.000000
6	0.000000	0.00000	0.000000	0.00000	0.00000	0.000000	0.000000	0.000000	0.00000	0.000000
7	0.000000	0.00000	0.000000	0.00000	0.00000	0.000000	0.000000	0.000000	0.00000	0.000000
8	0.000000	0.00000	0.000000	0.00000	0.00000	0.000000	0.000000	0.000000	0.00000	0.000000
9	0.000000	0.00000	0.429424	0.00000	0.00000	0.000000	0.000000	0.000000	0.00000	0.000000
10	0.000000	0.00000	0.000000	0.00000	0.00000	0.000000	0.000000	0.000000	0.00000	0.000000
11	0.000000	0.00000	0.000000	0.00000	0.00000	0.000000	0.000000	0.000000	0.00000	0.000000
12	0.000000	0.00000	0.000000	0.00000	0.00000	0.000000	0.000000	0.000000	0.00000	0.000000
13	0.000000	0.00000	0.000000	0.00000	0.00000	0.000000	0.000000	0.000000	0.00000	0.000000
14	0.000000	0.00000	0.000000	0.00000	0.00000	0.000000	0.000000	0.000000	0.00000	0.437

	acaranya	ada	adalah	aja	ajang	akan	anakanak	antusias	antusiasme	apa
15	0.000000	0.00000	0.000000	0.00000	0.00000	0.176607	0.000000	0.176607	0.00000	0.00000
16	0.000000	0.00000	0.000000	0.00000	0.00000	0.000000	0.000000	0.000000	0.00000	0.00000
17	0.000000	0.00000	0.000000	0.00000	0.00000	0.335657	0.000000	0.000000	0.00000	0.00000
18	0.000000	0.00000	0.000000	0.00000	0.00000	0.000000	0.000000	0.000000	0.18112	0.00000
19	0.000000	0.00000	0.000000	0.00000	0.00000	0.000000	0.000000	0.203517	0.00000	0.00000
20	0.000000	0.00000	0.000000	0.00000	0.00000	0.000000	0.000000	0.000000	0.00000	0.00000
21	0.000000	0.00000	0.000000	0.00000	0.00000	0.000000	0.000000	0.000000	0.00000	0.00000
22	0.000000	0.00000	0.000000	0.00000	0.00000	0.000000	0.000000	0.000000	0.00000	0.00000
23	0.000000	0.00000	0.221011	0.00000	0.00000	0.000000	0.000000	0.000000	0.00000	0.00000
24	0.295853	0.00000	0.000000	0.00000	0.00000	0.000000	0.000000	0.000000	0.00000	0.00000
25	0.000000	0.19434	0.000000	0.19434	0.00000	0.000000	0.000000	0.000000	0.00000	0.00000
26	0.000000	0.00000	0.000000	0.00000	0.00000	0.000000	0.000000	0.000000	0.00000	0.00000
27	0.000000	0.00000	0.165160	0.00000	0.00000	0.000000	0.000000	0.000000	0.00000	0.00000
28	0.000000	0.00000	0.000000	0.00000	0.00000	0.000000	0.000000	0.000000	0.00000	0.00000

10

Nilai Vektor antar Kata dengan Kata

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

# Data kalimat (contoh)
kalimat = df_cleaned1['Tokenisasi Dokumen1']

# Membuat objek TfidfVectorizer
tfidf_vectorizer = TfidfVectorizer()

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

# Mengonversi matriks TF-IDF ke DataFrame Pandas
tfidf_kata1 = pd.DataFrame(tfidf_matrix.toarray(), columns=tfidf_vectorizer.get_feature_names_out())

# Mengganti NaN dengan 0
tfidf_kata1 = tfidf_kata1.fillna(0)

# Menampilkan nilai TF-IDF dengan kata-kata
tfidf_values_with_words = pd.concat([pd.Series(tfidf_vectorizer.get_feature_names_out(), name='Kata', index=0),
                                     tfidf_kata1], axis=1)

# Menampilkan tabel TF-IDF dengan kata-kata
tfidf_values_with_words
```

	Kata	acaranya	ada	adalah	aja	ajang	akan	anakanak	antusias	antusiasn
0	acaranya	0.0	0.0	0.0	0.0	0.29658	0.0	0.000000	0.0	0.0
1	ada	0.0	0.0	0.0	0.0	0.00000	0.0	0.000000	0.0	0.0
2	adalah	0.0	0.0	0.0	0.0	0.00000	0.0	0.000000	0.0	0.0
3	aja	0.0	0.0	0.0	0.0	0.00000	0.0	0.000000	0.0	0.0
4	ajang	0.0	0.0	0.0	0.0	0.00000	0.0	0.229011	0.0	0.0
...
244	vietnam	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

11

Nilai Cosinus Similarity Kata Kunci

```
from sklearn.metrics.pairwise import cosine_similarity

# Drop the 'Kata' column before calculating cosine similarity
tfidf_vectors_only = tfidf_values_with_words.drop(columns=['Kata'])

# Handling NaN values by filling them with 0
tfidf_vectors_only = tfidf_vectors_only.fillna(0)

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

# Menampilkan matriks kesamaan kosinus
cosine_sim_dfl = pd.DataFrame(cosine_sim_matrix, columns=tfidf_values_with_words.index, index=t
cosine_sim_dfl
```

	0	1	2	3	4	5	6	7	8	9
0	1.000000	0.105866	0.000000	0.000000	0.124813	0.037802	0.044076	0.123862	0.079240	0.13
1	0.105866	1.000000	0.060728	0.029568	0.010703	0.071772	0.103548	0.145574	0.153927	0.15
2	0.000000	0.060728	1.000000	0.284872	0.000000	0.153018	0.123767	0.000000	0.000000	0.00
3	0.000000	0.029568	0.284872	1.000000	0.000000	0.050439	0.159831	0.000000	0.113162	0.00
4	0.124813	0.010703	0.000000	0.000000	1.000000	0.059062	0.000000	0.081330	0.148080	0.07
...
244	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00
245	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00
246	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00
247	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00
248	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00



12

Graph Kata Kunci

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

# Assuming tfidf_kata1 is your DataFrame with TF-IDF values
# and top_keywords is the top keywords for each document

# Handling NaN values by filling them with 0
tfidf_kata1 = tfidf_kata1.fillna(0)

# Calculate the sum of TF-IDF values for each term
term_sums = tfidf_kata1.sum()

# Sort terms based on their sum of TF-IDF values in descending order
sorted_terms = term_sums.sort_values(ascending=False)

# Extract the top N keywords (adjust N as needed)
N = 5
top_keywords = sorted_terms.head(N)

# Menghitung kesamaan kosinus antara kalimat-kalimat
cosine_sim_matrix = cosine_similarity(tfidf_kata1, tfidf_kata1)

# Menampilkan matriks kesamaan kosinus
cosine_sim_df1 = pd.DataFrame(cosine_sim_matrix, columns=tfidf_kata1.index, index=tfidf_kata1.index)

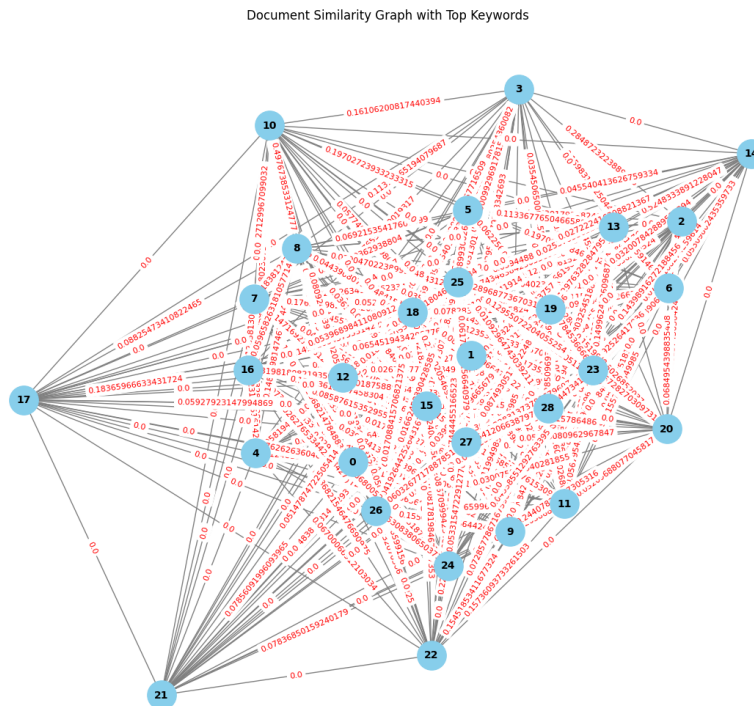
# Creating a graph
G = nx.Graph()

# Adding nodes to the graph with top keywords as labels
for document in tfidf_kata1.index:
    keywords_indices = tfidf_kata1.loc[document].values.argsort()[-N:][::-1]
    keywords_list = tfidf_kata1.columns[keywords_indices].tolist()
    G.add_node(document, label=", ".join(map(str, keywords_list)))
```

```
# Adding edges to the graph
for i in range(len(cosine_sim_df1.index)):
    for j in range(i + 1, len(cosine_sim_df1.index)):
        G.add_edge(cosine_sim_df1.index[i], cosine_sim_df1.index[j], weight=cosine_sim_df1.iloc[

# Visualizing the graph
pos = nx.spring_layout(G) # You can use different layouts depending on your preference
edge_labels = nx.get_edge_attributes(G, 'weight')

plt.figure(figsize=(12, 10))
nx.draw(G, pos, with_labels=True, font_size=10, font_color="black", node_size=800, node_color="
nx.draw_networkx_edge_labels(G, pos, edge_labels=edge_labels, font_color="red", font_size=8)
plt.title('Document Similarity Graph with Top Keywords')
plt.show()
```



13

Kata Kunci pada Dokumen 1

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

# Assuming tfidf_kata1 is your DataFrame with TF-IDF values
# and top_keywords is the top keywords for each document

# Handling NaN values by filling them with 0
tfidf_kata1 = tfidf_kata1.fillna(0)

# Calculate the sum of TF-IDF values for each term
term_sums = tfidf_kata1.sum()

# Sort terms based on their sum of TF-IDF values in descending order
sorted_terms = term_sums.sort_values(ascending=False)

# Extract the top N keywords (adjust N as needed)
N = 5
top_keywords = sorted_terms.head(N)

# Menghitung kesamaan kosinus antara kalimat-kalimat
cosine_sim_matrix = cosine_similarity(tfidf_kata1, tfidf_kata1)

# Menampilkan matriks kesamaan kosinus
cosine_sim_df1 = pd.DataFrame(cosine_sim_matrix, columns=tfidf_kata1.index, index=tfidf_kata1.index)

# Creating a graph
G = nx.Graph()

# Adding nodes to the graph with top keywords as labels
for document in tfidf_kata1.index:
    keywords_indices = tfidf_kata1.loc[document].values.argsort()[-N:][::-1]
    keywords_list = tfidf_kata1.columns[keywords_indices].tolist()
    G.add_node(document, label=", ".join(map(str, keywords_list)))
```

```

# Adding edges to the graph
for i in range(len(cosine_sim_df1.index)):
    for j in range(i + 1, len(cosine_sim_df1.index)):
        G.add_edge(cosine_sim_df1.index[i], cosine_sim_df1.index[j], weight=cosine_sim_df1.iloc[i][j])

# Visualizing the graph
pos = nx.spring_layout(G) # You can use different layouts depending on your preference
edge_labels = nx.get_edge_attributes(G, 'weight')
node_labels = nx.get_node_attributes(G, 'label')

plt.figure(figsize=(12, 10))
nx.draw(G, pos, with_labels=True, font_size=10, font_color="black", node_size=800, node_color="black")
nx.draw_networkx_edge_labels(G, pos, edge_labels=edge_labels, font_color="red", font_size=8)
nx.draw_networkx_labels(G, pos, labels=node_labels, font_size=8, font_color="green")

# Display top keywords for each node
for node, keywords in node_labels.items():
    print(f"Node {node} Keywords: {keywords}")

plt.title('Document Similarity Graph with Top Keywords')
plt.show()

```

```

Node 0 Keywords: tangerang, sekitar, kabupaten, ajang, bertajuk
Node 1 Keywords: kota, luar, dari, ini, kenya chief
Node 2 Keywords: umum, semuanya, terbuka, masyarakat, dan
Node 3 Keywords: pesertanya, terbuka, semua, kita, untuk
Node 4 Keywords: yang, community, mau, memang, mendapatkan
Node 5 Keywords: scroll, content, melanjutkan, continue, pola
Node 6 Keywords: karena, mencakup, ke, beberapa, semua
Node 7 Keywords: harapannya, keluarga, individual, bukan, sehat
Node 8 Keywords: ingin, berkontribusi, memberikan, dalam, kita
Node 9 Keywords: melalui, adalah, satunya, salah, run
Node 10 Keywords: menambahkan, mereka, sehat, dalam, hidup
Node 11 Keywords: dan, berterima, pihaknya, mengapresiasi, deputi
Node 12 Keywords: ikut, pariwisata, menurutnya, mendorong, bisa
Node 13 Keywords: diikuti, sebab, hampir, baru, sudah
Node 14 Keywords: apa, artinya, sudah, baru, kali
Node 15 Keywords: kegiatan, yang, biasa, dilakukan, rutin
Node 16 Keywords: event, menantang, kami, global, menggelar
Node 17 Keywords: mendukung, siap, tentunya, khususnya, akan
Node 18 Keywords: bisa, di, bajo, antusiasme, kawasan
Node 19 Keywords: peserta, juga, serpong, berlari, diajak
Node 20 Keywords: menyelesaikan, telah, berhasil, datang, sejak
Node 21 Keywords: hobi, suka, saya, lari, yang

```


Node 22 Keywords: bagus, disponsori, produk, terus, run

Node 23 Keywords: ini, disemangatin, jakarta, nunuselain, larinya

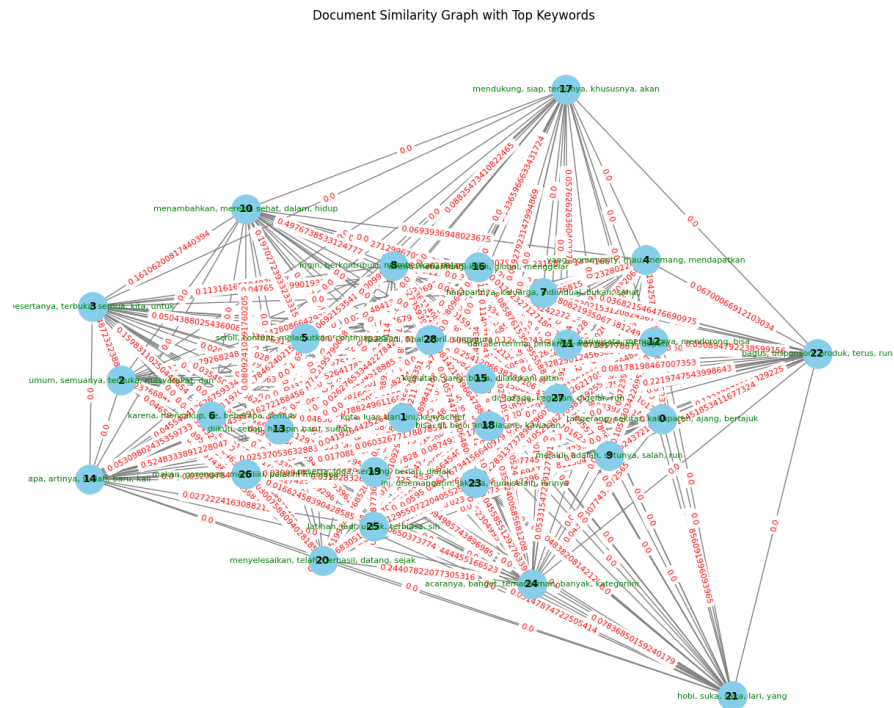
Node 24 Keywords: acaranya, banget, temanteman, banyak, kategoriini

Node 25 Keywords: latihan, jadi, untuk, terbiasa, sih

Node 26 Keywords: makan, gorengan, memiliki, pelatih, menjaga

Node 27 Keywords: di, lazada, kegiatan, digelar, run

Node 28 Keywords: pada, di, final, april, singapura





14

3. Cosinuss Similarity pada Dokumen 1

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_dfl = pd.DataFrame(cosine_sim_matrix, columns=df_cleaned1.index, index=df_cleaned1.index)
cosine_sim_dfl
```

	0	1	2	3	4	5	6	7	8	9
0	1.000000	0.105866	0.000000	0.000000	0.124813	0.037802	0.044076	0.123862	0.079240	0.132162
1	0.105866	1.000000	0.060728	0.029568	0.010703	0.071772	0.103548	0.145574	0.153927	0.158091
2	0.000000	0.060728	1.000000	0.284872	0.000000	0.153018	0.123767	0.000000	0.000000	0.000000
3	0.000000	0.029568	0.284872	1.000000	0.000000	0.050439	0.159831	0.000000	0.113162	0.000000
4	0.124813	0.010703	0.000000	0.000000	1.000000	0.059062	0.000000	0.081330	0.148080	0.072391
5	0.037802	0.071772	0.153018	0.050439	0.059062	1.000000	0.197093	0.076796	0.069215	0.038778
6	0.044076	0.103548	0.123767	0.159831	0.000000	0.197093	1.000000	0.000000	0.000000	0.045213
7	0.123862	0.145574	0.000000	0.000000	0.081330	0.076796	0.000000	1.000000	0.225818	0.000000
8	0.079240	0.153927	0.000000	0.113162	0.148080	0.069215	0.000000	0.225818	1.000000	0.020855
9	0.132162	0.158091	0.000000	0.000000	0.072391	0.038778	0.045213	0.000000	0.020855	1.000000
10	0.000000	0.057750	0.000000	0.161062	0.069394	0.197027	0.000000	0.108271	0.497674	0.000000
11	0.057880	0.091010	0.080717	0.000000	0.031703	0.033965	0.039602	0.000000	0.033755	0.128602
12	0.168602	0.078189	0.000000	0.000000	0.058194	0.122424	0.036346	0.214717	0.119969	0.105866
13	0.086432	0.089147	0.000000	0.000000	0.000000	0.113368	0.132182	0.000000	0.000000	0.044076
14	0.000000	0.053393	0.000000	0.000000	0.000000	0.045540	0.053098	0.000000	0.000000	0.000000
15	0.139192	0.159860	0.063183	0.030763	0.096497	0.062492	0.050924	0.045389	0.036114	0.205860
16	0.073862	0.096634	0.044281	0.047655	0.062316	0.029148	0.000000	0.070313	0.059658	0.039602
17	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.088255	0.000000

	0	1	2	3	4	5	6	7	8	9
18	0.182502	0.092105	0.057712	0.028099	0.088783	0.059134	0.048908	0.176509	0.044391	0.000000
19	0.053138	0.243644	0.032940	0.035451	0.000000	0.062256	0.096405	0.052305	0.065363	0.027000
20	0.000000	0.121827	0.101574	0.049455	0.000000	0.030249	0.068495	0.000000	0.000000	0.000000
21	0.078561	0.050484	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
22	0.122321	0.022846	0.000000	0.000000	0.067001	0.000000	0.000000	0.000000	0.019302	0.154000
23	0.057429	0.184108	0.000000	0.000000	0.039879	0.096529	0.112548	0.031111	0.035695	0.222000
24	0.043453	0.045984	0.000000	0.000000	0.000000	0.000000	0.000000	0.033861	0.030519	0.000000
25	0.028543	0.083919	0.056031	0.060301	0.000000	0.071390	0.040233	0.022243	0.020047	0.023000
26	0.000000	0.130624	0.046598	0.000000	0.000000	0.068875	0.000000	0.030566	0.055977	0.155000
27	0.261593	0.128471	0.035443	0.000000	0.146197	0.096525	0.112543	0.164565	0.090160	0.345000
28	0.143458	0.022357	0.000000	0.000000	0.164911	0.050026	0.058328	0.062872	0.038166	0.108000

Kata Penting pada setiap Kalimat berdasarkan nilai TF-IDF pada Dokumen Berita 1

```
# Menampilkan kata-kata dengan nilai TF-IDF tertinggi untuk setiap dokumen
for i, row in tfidf_kata1.iterrows():
    print(f"Kata-kata penting dalam Ringkasan Berita pada Kalimat {i + 1}:")
    top_keywords = row.sort_values(ascending=False).head(5) # Ganti 5 dengan jumlah kata-kata
    print(top_keywords)
    print("\n")
```

Kata-kata penting dalam Ringkasan Berita pada Kalimat 1:

```
ajang      0.29658
tangerang  0.29658
bertajuk   0.29658
kabupaten  0.29658
meriahkan  0.29658
Name: 0, dtype: float64
```

Kata-kata penting dalam Ringkasan Berita pada Kalimat 2:

```
kota      0.381169
luar      0.339489
dari      0.268237
ini       0.215727
officer   0.190584
Name: 1, dtype: float64
```

Kata-kata penting dalam Ringkasan Berita pada Kalimat 3:

```
semuanya   0.493895
```

umum 0.493895
 terbuka 0.439889
 masyarakat 0.401571
 dan 0.293559
 Name: 2, dtype: float64

Kata-kata penting dalam Ringkasan Berita pada Kalimat 4:

pesertanya 0.531533
 terbuka 0.473411
 semua 0.473411
 kita 0.432173
 untuk 0.287165
 Name: 3, dtype: float64

Kata-kata penting dalam Ringkasan Berita pada Kalimat 5:

yang 0.259223
 enthusiast 0.229011
 community 0.229011
 memang 0.229011
 running 0.229011
 Name: 4, dtype: float64

Kata-kata penting dalam Ringkasan Berita pada Kalimat 6:

content 0.325112
 continue 0.325112
 melanjutkan 0.325112
 scroll 0.325112
 pola 0.289562
 Name: 5, dtype: float64

Kata-kata penting dalam Ringkasan Berita pada Kalimat 7:

ke 0.379066
 beberapa 0.379066
 mencakup 0.379066
 karena 0.379066
 semua 0.337616
 Name: 6, dtype: float64

Kata-kata penting dalam Ringkasan Berita pada Kalimat 8:

harapannya 0.357314

individual 0.357314
keluarga 0.357314
bukan 0.318243
tetapi 0.290521
Name: 7, dtype: float64

Kata-kata penting dalam Ringkasan Berita pada Kalimat 9:

ingin 0.392765
memberikan 0.322043
berkontribusi 0.322043
dalam 0.261843
kita 0.261843
Name: 8, dtype: float64

Kata-kata penting dalam Ringkasan Berita pada Kalimat 10:

melalui 0.528151
adalah 0.429424
satunya 0.429424
salah 0.349716
run 0.313920
Name: 9, dtype: float64

Kata-kata penting dalam Ringkasan Berita pada Kalimat 11:

menambahkan 0.408240
ingin 0.372679
hidup 0.372679
mereka 0.372679
dalam 0.372679
Name: 10, dtype: float64

Kata-kata penting dalam Ringkasan Berita pada Kalimat 12:

dan 0.274960
jemadu 0.231301
intanselain 0.231301
wisata 0.231301
vinsensius 0.231301
Name: 11, dtype: float64

Kata-kata penting dalam Ringkasan Berita pada Kalimat 13:

menurutnya 0.424572

pariwisata 0.424572
ikut 0.424572
mendorong 0.345207
bisa 0.319657
Name: 12, dtype: float64

Kata-kata penting dalam Ringkasan Berita pada Kalimat 14:
hampir 0.371674
sebab 0.371674
diikuti 0.371674
sudah 0.331033
baru 0.331033
Name: 13, dtype: float64

Kata-kata penting dalam Ringkasan Berita pada Kalimat 15:
apa 0.437311
artinya 0.437311
baru 0.389492
sudah 0.389492
kali 0.355564
Name: 14, dtype: float64

Kata-kata penting dalam Ringkasan Berita pada Kalimat 16:
kegiatan 0.282693
yang 0.224448
ucap 0.198289
menjadikan 0.198289
ketiga 0.198289
Name: 15, dtype: float64

Kata-kata penting dalam Ringkasan Berita pada Kalimat 17:
event 0.547165
global 0.307171
menantang 0.307171
menggelar 0.307171
kami 0.307171
Name: 16, dtype: float64

Kata-kata penting dalam Ringkasan Berita pada Kalimat 18:
mendukung 0.376866

khususnya 0.376866
siap 0.376866
tentunya 0.376866
akan 0.335657
Name: 17, dtype: float64

Kata-kata penting dalam Ringkasan Berita pada Kalimat 19:
bisa 0.409091
di 0.215306
dilihat 0.181120
saat 0.181120
suatu 0.181120
Name: 18, dtype: float64

Kata-kata penting dalam Ringkasan Berita pada Kalimat 20:
peserta 0.286150
juga 0.258648
dirinya 0.228503
melakukan 0.228503
berlari 0.228503
Name: 19, dtype: float64

Kata-kata penting dalam Ringkasan Berita pada Kalimat 21:
berhasil 0.318774
telah 0.318774
menyelesaikan 0.318774
pagi 0.283917
datang 0.283917
Name: 20, dtype: float64

Kata-kata penting dalam Ringkasan Berita pada Kalimat 22:
suka 0.534887
saya 0.534887
hobi 0.534887
lari 0.376412
pasti 0.000000
Name: 21, dtype: float64

Kata-kata penting dalam Ringkasan Berita pada Kalimat 23:
bagus 0.488822

disponsori 0.488822
 produk 0.435370
 terus 0.435370
 run 0.290543
 Name: 22, dtype: float64

Kata-kata penting dalam Ringkasan Berita pada Kalimat 24:
 ini 0.307682
 larinya 0.271823
 jakarta 0.271823
 disemangatin 0.271823
 seru 0.271823
 Name: 23, dtype: float64

Kata-kata penting dalam Ringkasan Berita pada Kalimat 25:
 acaranya 0.295853
 banget 0.295853
 kategoriini 0.295853
 temanteman 0.295853
 banyak 0.295853
 Name: 24, dtype: float64

Kata-kata penting dalam Ringkasan Berita pada Kalimat 26:
 jadi 0.388681
 latihan 0.388681
 untuk 0.209988
 half 0.194340
 time 0.194340
 Name: 25, dtype: float64

Kata-kata penting dalam Ringkasan Berita pada Kalimat 27:
 makan 0.534127
 tidak 0.267064
 memiliki 0.267064
 gorengan 0.267064
 tersendiri 0.267064
 Name: 26, dtype: float64

Kata-kata penting dalam Ringkasan Berita pada Kalimat 28:
 di 0.362208

36

14 3. Cosinuss Similarity pada Dokumen 1

```
lazada      0.301775
kegiatan    0.289596
digelar      0.254377
run         0.241472
Name: 27, dtype: float64
```

Kata-kata penting dalam Ringkasan Berita pada Kalimat 29:

```
pada      0.443601
di         0.296036
bakal     0.249031
singapura 0.249031
setelah   0.249031
Name: 28, dtype: float64
```

15

4. Graph Kata Penting berdasarkan nilai TF-IDF pada Dokumen 1

Graph disini dibuat untuk menggambarkan nilai jarak antara kata satu dengan kata yang lain berdasarkan nilai Cosinuss Similarity dari Dokumen berita 1

```
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_cleaned1.index[i]) # Menggunakan label kalimat

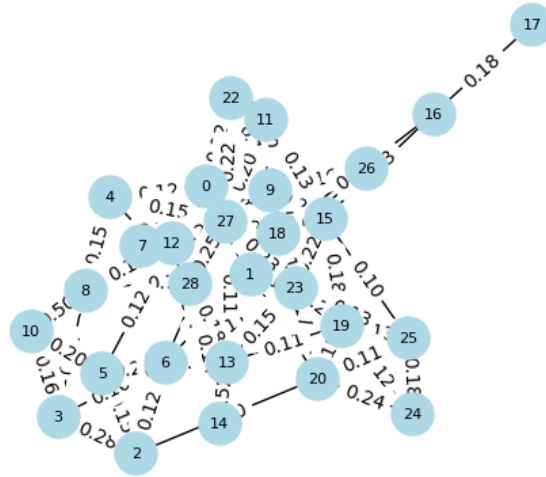
# Menambahkan tepian (hubungan) berdasarkan kesamaan kosinus
for i in range(len(cosine_sim_matrix)):
    for j in range(i+1, len(cosine_sim_matrix)):
        similarity = cosine_sim_matrix[i][j]
        if similarity > 0.1: # Atur threshold sesuai kebutuhan
            G.add_edge(i, j, weight=similarity)

# 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_color='black')
nx.draw_networkx_edge_labels(G, pos, edge_labels={(i, j): f"{similarity:.2f}" for i, j, similarity in G.edges(data=True)})

plt.show()
```

21



16

5. Closeness Centrality Dokumen Berita 1

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 27: Closeness Centrality = 0.6199
Simpul 1: Closeness Centrality = 0.5786
Simpul 23: Closeness Centrality = 0.5540
Simpul 0: Closeness Centrality = 0.5424
Simpul 9: Closeness Centrality = 0.5313
Simpul 15: Closeness Centrality = 0.5313
Simpul 18: Closeness Centrality = 0.5313
Simpul 12: Closeness Centrality = 0.5207
Simpul 6: Closeness Centrality = 0.5105
Simpul 7: Closeness Centrality = 0.5007
Simpul 13: Closeness Centrality = 0.5007
Simpul 19: Closeness Centrality = 0.4912
Simpul 20: Closeness Centrality = 0.4734
Simpul 28: Closeness Centrality = 0.4649
Simpul 8: Closeness Centrality = 0.4568
Simpul 5: Closeness Centrality = 0.4413
Simpul 11: Closeness Centrality = 0.4413
Simpul 4: Closeness Centrality = 0.4268
Simpul 25: Closeness Centrality = 0.4199
```

Simpul 2: Closeness Centrality = 0.4133
Simpul 22: Closeness Centrality = 0.4068
Simpul 26: Closeness Centrality = 0.4005
Simpul 16: Closeness Centrality = 0.3945
Simpul 3: Closeness Centrality = 0.3886
Simpul 10: Closeness Centrality = 0.3886
Simpul 14: Closeness Centrality = 0.3667
Simpul 24: Closeness Centrality = 0.3567
Simpul 17: Closeness Centrality = 0.2830
Simpul 21: Closeness Centrality = 0.0000

17

6. PageRank Dokumen 1

Pagerank merupakan suatu proses mengukur atau mencari nilai penting dalam suatu dokumen

$$PR(S_i) = \frac{1 - \alpha}{\text{NodeCount}} + \alpha \sum_{S_j \in \text{Neighbors } S_i} \frac{PR(S_j)}{\text{CountEdge}(S_j)}$$

Keterangan:

- $PR(S_i)$: Nilai PageRank untuk kalimat S_i
- $PR(S_j)$: Nilai PageRank dari vertex yang bertetangga dengan S_i
- $\text{CountEdge}(S_j)$: Jumlah edge dari kalimat S_j
- α_i : Damping faktor yang nilainya antara 0 dan 1

```
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}
```



18

7. *EigenVector* dari Dokumen Berita 1

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

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

```
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')]
```

Tampilan Kalimat penting berdasarkan nilai EigenVector pada Dokumen Berita 1

```
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_cleaned1['Tokenisasi Dokumen1']

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

```
Kalimat berital: 'sekitar peserta meriahkan ajang lari yang digelar lazada indonesia bertajuk lazada ru
Kalimat berital: 'para peserta lomba lari ini tak hanya dari dalam kota tetapi juga datang dari luar kot
```

Kalimat berital: 'semuanya terbuka untuk umum dan masyarakat' memiliki Eigenvector Centrality sebesar

Kalimat berital: 'pesertanya kita terbuka untuk semua' memiliki Eigenvector Centrality sebesar 0.37

19

8. Mencari Kata Kunci

Nilai TF-IDF Kata Kunci Dokumen 1

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

# Data kalimat (contoh)
kalimat = df_cleaned1['Tokenisasi Dokumen1']

# Membuat objek TfidfVectorizer
tfidf_vectorizer = TfidfVectorizer()

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

# Mengonversi matriks TF-IDF ke DataFrame Pandas
tfidf_kata1 = pd.DataFrame(tfidf_matrix.toarray(), columns=tfidf_vectorizer.get_feature_names_out())

# Mengganti NaN dengan 0
tfidf_kata1 = tfidf_kata1.fillna(0)

# Menampilkan nilai TF-IDF dengan kata-kata
tfidf_values_with_words = pd.concat([pd.Series(tfidf_vectorizer.get_feature_names_out(), name='Kata', index=0),
                                     tfidf_kata1], axis=1)

# Menampilkan tabel TF-IDF dengan kata-kata
tfidf_values_with_words
```

	Kata	acaranya	ada	adalah	aja	ajang	akan	anakanak	antusias	antusiasn
0	acaranya	0.0	0.0	0.0	0.0	0.29658	0.0	0.000000	0.0	0.0
1	ada	0.0	0.0	0.0	0.0	0.00000	0.0	0.000000	0.0	0.0
2	adalah	0.0	0.0	0.0	0.0	0.00000	0.0	0.000000	0.0	0.0
3	aja	0.0	0.0	0.0	0.0	0.00000	0.0	0.000000	0.0	0.0
4	ajang	0.0	0.0	0.0	0.0	0.00000	0.0	0.229011	0.0	0.0
...
244	vietnam	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

	Kata	acaranya	ada	adalah	aja	ajang	akan	anakanak	antusias	antusiasn
245	vinsensius	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
246	vinsensiusmelihat	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
247	wisata	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
248	yang	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

Nilai Cosinus Similarity Kata Kunci Dokumen 1

```

from sklearn.metrics.pairwise import cosine_similarity

# Drop the 'Kata' column before calculating cosine similarity
tfidf_vectors_only = tfidf_values_with_words.drop(columns=['Kata'])

# Handling NaN values by filling them with 0
tfidf_vectors_only = tfidf_vectors_only.fillna(0)

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

# Menampilkan matriks kesamaan kosinus
cosine_sim_df1 = pd.DataFrame(cosine_sim_matrix, columns=tfidf_values_with_words.index, index=t
cosine_sim_df1

```

	0	1	2	3	4	5	6	7	8	9
0	1.000000	0.105866	0.000000	0.000000	0.124813	0.037802	0.044076	0.123862	0.079240	0.13
1	0.105866	1.000000	0.060728	0.029568	0.010703	0.071772	0.103548	0.145574	0.153927	0.15
2	0.000000	0.060728	1.000000	0.284872	0.000000	0.153018	0.123767	0.000000	0.000000	0.00
3	0.000000	0.029568	0.284872	1.000000	0.000000	0.050439	0.159831	0.000000	0.113162	0.00
4	0.124813	0.010703	0.000000	0.000000	1.000000	0.059062	0.000000	0.081330	0.148080	0.07
...
244	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00
245	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00
246	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00
247	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00
248	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00

Graph Dan Kata Kunci pada Dokumen 1

```

import networkx as nx
import matplotlib.pyplot as plt

```

```

# Assuming tfidf_kata1 is your DataFrame with TF-IDF values
# and top_keywords is the top keywords for each document

# Handling NaN values by filling them with 0
tfidf_kata1 = tfidf_kata1.fillna(0)

# Calculate the sum of TF-IDF values for each term
term_sums = tfidf_kata1.sum()

# Sort terms based on their sum of TF-IDF values in descending order
sorted_terms = term_sums.sort_values(ascending=False)

# Extract the top N keywords (adjust N as needed)
N = 1
top_keywords = sorted_terms.head(N)

# Menghitung kesamaan kosinus antara kalimat-kalimat
cosine_sim_matrix = cosine_similarity(tfidf_kata1, tfidf_kata1)

# Menampilkan matriks kesamaan kosinus
cosine_sim_df1 = pd.DataFrame(cosine_sim_matrix, columns=tfidf_kata1.index, index=tfidf_kata1.index)

# Create a list of nodes sorted by the highest TF-IDF value
sorted_nodes = sorted(tfidf_kata1.index, key=lambda x: tfidf_kata1.loc[x].max(), reverse=True)

# Creating a graph
G = nx.Graph()

# Adding nodes to the graph with top keywords as labels
for document in sorted_nodes:
    keywords_indices = tfidf_kata1.loc[document].values.argsort()[-N:][::-1]
    keywords_list = tfidf_kata1.columns[keywords_indices].tolist()
    keywords_values = tfidf_kata1.loc[document, keywords_list].tolist()

    # Sorting keywords and values in descending order
    sorted_keywords_values = sorted(zip(keywords_list, keywords_values), key=lambda x: x[1], reverse=True)

    node_label = ", ".join([f"{kw} ({val:.4f})" for kw, val in sorted_keywords_values])
    G.add_node(document, label=node_label)

# Adding edges to the graph
for i in range(len(cosine_sim_df1.index)):
    for j in range(i + 1, len(cosine_sim_df1.index)):
        G.add_edge(cosine_sim_df1.index[i], cosine_sim_df1.index[j], weight=cosine_sim_df1.iloc[i][j])

```

```

# Visualizing the graph
pos = nx.spring_layout(G) # You can use different layouts depending on your preference
edge_labels = nx.get_edge_attributes(G, 'weight')
node_labels = nx.get_node_attributes(G, 'label')

plt.figure(figsize=(12, 10))
nx.draw(G, pos, with_labels=True, font_size=10, font_color="black", node_size=800, node_color="black",
        edge_labels=edge_labels, font_color="red", font_size=8)
nx.draw_networkx_labels(G, pos, labels=node_labels, font_size=8, font_color="green")

# Display top keywords and values for each node
for node, label in node_labels.items():
    print(f"Node {node} Keywords and Values: {label}")

plt.title('Document Similarity Graph with Top Keywords and Values (Sorted)')
plt.show()

```

```

Node 16 Keywords and Values: event (0.5472)
Node 21 Keywords and Values: hobi (0.5349)
Node 26 Keywords and Values: makan (0.5341)
Node 3 Keywords and Values: pesertanya (0.5315)
Node 9 Keywords and Values: melalui (0.5282)
Node 2 Keywords and Values: umum (0.4939)
Node 22 Keywords and Values: bagus (0.4888)
Node 28 Keywords and Values: pada (0.4436)
Node 14 Keywords and Values: apa (0.4373)
Node 12 Keywords and Values: ikut (0.4246)
Node 18 Keywords and Values: bisa (0.4091)
Node 10 Keywords and Values: menambahkan (0.4082)
Node 8 Keywords and Values: ingin (0.3928)
Node 25 Keywords and Values: latihan (0.3887)
Node 1 Keywords and Values: kota (0.3812)
Node 6 Keywords and Values: karena (0.3791)
Node 17 Keywords and Values: mendukung (0.3769)
Node 13 Keywords and Values: diikuti (0.3717)
Node 27 Keywords and Values: di (0.3622)
Node 7 Keywords and Values: harapannya (0.3573)
Node 5 Keywords and Values: scroll (0.3251)
Node 20 Keywords and Values: menyelesaikan (0.3188)
Node 23 Keywords and Values: ini (0.3077)
Node 0 Keywords and Values: tangerang (0.2966)
Node 24 Keywords and Values: acaranya (0.2959)
Node 19 Keywords and Values: peserta (0.2861)

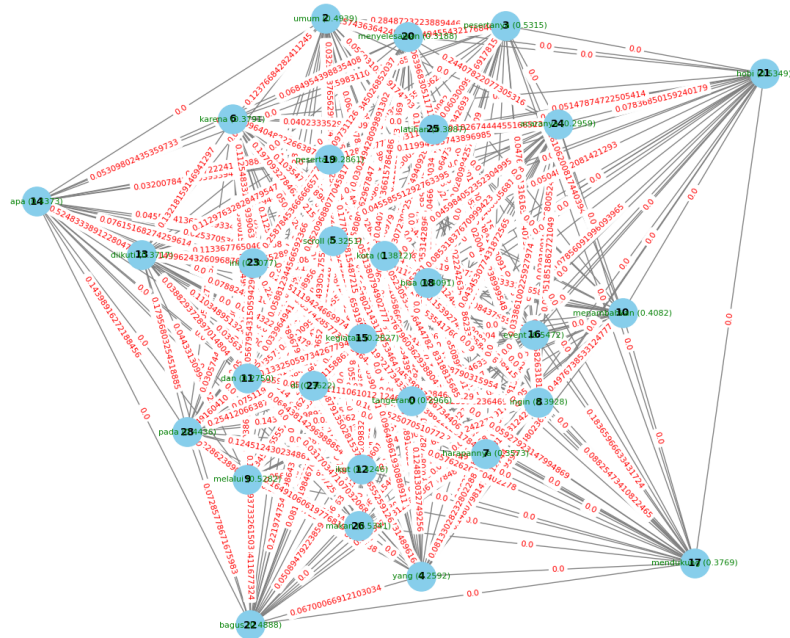
```

Node 15 Keywords and Values: kegiatan (0.2827)

Node 11 Keywords and Values: dan (0.2750)

Node 4 Keywords and Values: yang (0.2592)

Document Similarity Graph with Top Keywords and Values (Sorted)





20

Dokumen 2

```
import pandas as pd

# Ambil satu dokumen dari baris pertama
dokumen_pertama = df.at[1, 'Content']

# Buat DataFrame dengan satu kolom dan satu baris
df_dokumen2 = pd.DataFrame({'Dokumen2': [dokumen_pertama]})

# Tampilkan DataFrame
df_dokumen2
```

Dokumen2	
0	brand apparel olahraga under armour mengadakan...



21

1. Ekstraksi Kalimat pada Dokumen Berita 2

```
from nltk.tokenize import sent_tokenize

# Misalnya, jika df adalah DataFrame yang memiliki kolom 'Content'
teks_berita = df_dokumen2['Dokumen2'].values.tolist()

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

df_kalimat2 = pd.DataFrame(kalimat, columns=['Tokenisasi'])
df_kalimat2
```

	Tokenisasi
0	brand apparel olahraga under armour mengadakan...
1	kegiatan yang diadakan di senayan, jakarta, in...
2	under armour juga mengundang 11 komunitas lari.
3	"kita undang komunitas lari.
4	sebenarnya kita cuma memberikan slot (peserta)...
5	tapi ini acaranya super sukses karena ramai ba...
6	intinya kita undang semua masyarakat untuk iku...
7	sekaligus sebagai pemanasan menghadapi challen...
8	dikatakannya, selama fun run 4k berlangsung pe...
9	scroll to continue with content\r"kita menduku...
10	untuk membantu mempersiapkan diri juga sebelum...
11	adapun proses pendaftarannya sudah dibuka seja...
12	agenda ini diikuti 400 peserta dari eropa, amer...
13	selama periode itu, peserta individu maupun se...
14	hasil data berlari peserta secara otomatis aka...
15	mengingat jaraknya yang hanya 1 mil atau 1,6 k...
16	"kita juga mengajak masyarakat kembali ke real...
17	jadi semacam reuni lagi.
18	apalagi sudah berapa tahun kita (dilanda) pand...
19	jadi rasanya ini waktu yang tepat untuk mendap...



22

Menghilangkan Kata dan Tanda tidak penting pada Dokumen 2

```
import pandas as pd
from nltk.tokenize import sent_tokenize
from nltk.corpus import stopwords
import string

# Misalnya, jika df adalah DataFrame yang memiliki kolom 'Content'
teks_berita = df_dokumen2['Dokumen2'].values.tolist()

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

# Membuang kata-kata atau tanda baca yang tidak penting
stopwords_list = set(stopwords.words('english'))
cleaned_sentences = []

for sentence in kalimat:
    # Menghapus tanda baca
    sentence = sentence.translate(str.maketrans("", "", string.punctuation))

    # Menghapus angka
    sentence = ''.join([char for char in sentence if not char.isdigit()])

    # Mengubah teks menjadi huruf kecil
    sentence = sentence.lower()

    # Membuang kata-kata yang merupakan stopwords
    words = sentence.split()
    words = [word for word in words if word not in stopwords_list]

    # Menggabungkan kata-kata kembali menjadi kalimat
    cleaned_sentence = " ".join(words)
```

```

cleaned_sentences.append(cleaned_sentence)

# Membuat DataFrame baru
df_cleaned2 = pd.DataFrame(cleaned_sentences, columns=['Tokenisasi Dokumen2'])
df_cleaned2

```

	Tokenisasi Dokumen2
0	brand apparel olahraga armour mengadakan kompe...
1	kegiatan yang diadakan di senayan jakarta ini ...
2	armour juga mengundang komunitas lari
3	kita undang komunitas lari
4	sebenarnya kita cuma memberikan slot peserta t...
5	tapi ini acaranya super sukses karena ramai ba...
6	intinya kita undang semua masyarakat untuk iku...
7	sekaligus sebagai pemanasan menghadapi challen...
8	dikatakannya selama fun run k berlangsung pese...
9	scroll continue content kita mendukung semua o...
10	untuk membantu mempersiapkan diri juga sebelum...
11	adapun proses pendaftarannya sudah dibuka seja...
12	agenda ini diikuti peserta dari eropa amerika s...
13	selama periode itu peserta individu maupun sec...
14	hasil data berlari peserta secara otomatis aka...
15	mengingat jaraknya yang hanya mil atau km maka...
16	kita juga mengajak masyarakat kembali ke reali...
17	jadi semacam reuni lagi
18	apalagi sudah berapa tahun kita dilanda pandemi
19	jadi rasanya ini waktu yang tepat untuk mendap...

23

2. TF-IDF

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

# Data kalimat (contoh)
kalimat = df_cleaned2['Tokenisasi Dokumen2']

# Membuat objek TfidfVectorizer
tfidf_vectorizer = TfidfVectorizer()

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

# Mengonversi matriks TF-IDF ke DataFrame Pandas
tfidf_kata2 = pd.DataFrame(tfidf_matrix.toarray(), columns=tfidf_vectorizer.get_feature_names_out())

# Menampilkan tabel TF-IDF
tfidf_kata2
```

	acaranya	adapun	ade	agenda	ajang	akan	amerika	anakanak	apalagi	apli
0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00
1	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.289316	0.000000	0.00
2	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00
3	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00
4	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00
5	0.369011	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00
6	0.000000	0.000000	0.356671	0.000000	0.178336	0.000000	0.000000	0.000000	0.000000	0.00
7	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00
8	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00
9	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00
10	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00
11	0.000000	0.233685	0.000000	0.000000	0.000000	0.205412	0.000000	0.000000	0.000000	0.00
12	0.000000	0.000000	0.000000	0.320139	0.000000	0.000000	0.320139	0.000000	0.000000	0.00
13	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00
14	0.000000	0.000000	0.000000	0.000000	0.000000	0.263623	0.000000	0.000000	0.000000	0.14

	acaranya	adapun	ade	agenda	ajang	akan	amerika	anakanak	apalagi	apli
15	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00
16	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00
17	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00
18	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.429416	0.00
19	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00

```
# Menampilkan kata-kata dengan nilai TF-IDF tertinggi untuk setiap dokumen
for i, row in tfidf_kata2.iterrows():
    print(f"Kata-kata penting dalam Dokumen Berita 2 pada Kalimat {i + 1}:")
    top_keywords = row.sort_values(ascending=False).head(5) # Ganti 5 dengan jumlah kata-kata
    print(top_keywords)
    print("\n")
```

Kata-kata penting dalam Dokumen Berita 2 pada Kalimat 1:

```
lari      0.404653
olahraga  0.300996
rangkap   0.300996
brand     0.300996
sedunia   0.300996
Name: 0, dtype: float64
```

Kata-kata penting dalam Dokumen Berita 2 pada Kalimat 2:

```
senayan    0.289316
jakarta    0.289316
dewasa     0.289316
anakanak   0.289316
sebanyak   0.289316
Name: 1, dtype: float64
```

Kata-kata penting dalam Dokumen Berita 2 pada Kalimat 3:

```
mengundang 0.543802
komunitas   0.478010
juga        0.431330
armour      0.395122
lari        0.365538
Name: 2, dtype: float64
```

Kata-kata penting dalam Dokumen Berita 2 pada Kalimat 4:

```
komunitas  0.568672
```


undang 0.568672
 lari 0.434868
 kita 0.405111
 acaranya 0.000000
 Name: 3, dtype: float64

Kata-kata penting dalam Dokumen Berita 2 pada Kalimat 5:
 cuma 0.415793
 memberikan 0.415793
 slot 0.415793
 sebenarnya 0.415793
 terbatas 0.415793
 Name: 4, dtype: float64

Kata-kata penting dalam Dokumen Berita 2 pada Kalimat 6:
 acaranya 0.369011
 banget 0.369011
 sukses 0.369011
 super 0.369011
 tapi 0.369011
 Name: 5, dtype: float64

Kata-kata penting dalam Dokumen Berita 2 pada Kalimat 7:
 ade 0.356671
 untuk 0.223345
 ini 0.209134
 marketing 0.178336
 ditemui 0.178336
 Name: 6, dtype: float64

Kata-kata penting dalam Dokumen Berita 2 pada Kalimat 8:
 sesungguhnya 0.404343
 menghadapi 0.404343
 sebagai 0.404343
 pemanasan 0.404343
 sekaligus 0.404343
 Name: 7, dtype: float64

Kata-kata penting dalam Dokumen Berita 2 pada Kalimat 9:
 memilih 0.240828

```
grup          0.240828
sesuai        0.240828
beberapa      0.240828
dengan        0.240828
Name: 8, dtype: float64
```

Kata-kata penting dalam Dokumen Berita 2 pada Kalimat 10:

```
content      0.347498
continue     0.347498
scroll        0.347498
bersenangsenang 0.347498
mendukung    0.347498
Name: 9, dtype: float64
```

Kata-kata penting dalam Dokumen Berita 2 pada Kalimat 11:

```
negara        0.209419
belum         0.209419
membantu      0.209419
bersamaan     0.209419
mempersiapkan 0.209419
Name: 10, dtype: float64
```

Kata-kata penting dalam Dokumen Berita 2 pada Kalimat 12:

```
mei           0.233685
mendatang     0.233685
competition   0.233685
sejak         0.233685
dibuka        0.233685
Name: 11, dtype: float64
```

Kata-kata penting dalam Dokumen Berita 2 pada Kalimat 13:

```
serikat       0.320139
amerika       0.320139
asia          0.320139
dikuti        0.320139
south         0.320139
Name: 12, dtype: float64
```

Kata-kata penting dalam Dokumen Berita 2 pada Kalimat 14:

```
sejauh        0.29284
```

itu 0.29284
 maupun 0.29284
 ditantang 0.29284
 periode 0.29284
 Name: 13, dtype: float64

Kata-kata penting dalam Dokumen Berita 2 pada Kalimat 15:
 akan 0.263623
 berlari 0.263623
 dalam 0.237879
 di 0.217911
 peserta 0.187800
 Name: 14, dtype: float64

Kata-kata penting dalam Dokumen Berita 2 pada Kalimat 16:
 mengingat 0.295803
 pemula 0.295803
 cocok 0.295803
 jaraknya 0.295803
 hanya 0.295803
 Name: 15, dtype: float64

Kata-kata penting dalam Dokumen Berita 2 pada Kalimat 17:
 realita 0.385156
 back 0.385156
 kembali 0.338558
 run 0.338558
 ke 0.338558
 Name: 16, dtype: float64

Kata-kata penting dalam Dokumen Berita 2 pada Kalimat 18:
 reuni 0.531094
 semacam 0.531094
 jadi 0.466840
 lagi 0.466840
 para 0.000000
 Name: 17, dtype: float64

Kata-kata penting dalam Dokumen Berita 2 pada Kalimat 19:
 dilanda 0.429416

```
apalagi    0.429416
berapa     0.429416
tahun      0.377464
sudah      0.340602
Name: 18, dtype: float64
```

Kata-kata penting dalam Dokumen Berita 2 pada Kalimat 20:

```
running    0.351484
jogging    0.199931
selengkapnyahalaman 0.199931
selanjutnya 0.199931
detikolahraga 0.199931
Name: 19, dtype: float64
```

24

3. Cosinuss Similarity

```
from sklearn.metrics.pairwise import cosine_similarity

# Menghitung kesamaan kosinus antara kalimat-kalimat
cosine_sim_matrix2 = cosine_similarity(tfidf_matrix, tfidf_matrix)

# Menampilkan matriks kesamaan kosinus
cosine_sim_df2 = pd.DataFrame(cosine_sim_matrix2, columns=df_cleaned2.index, index=df_cleaned2.index)
cosine_sim_df2
```

	0	1	2	3	4	5	6	7	8	9
0	1.000000	0.000000	0.234330	0.175971	0.000000	0.000000	0.028339	0.000000	0.045604	0.000000
1	0.000000	1.000000	0.000000	0.000000	0.047170	0.036705	0.095177	0.040219	0.051276	0.112246
2	0.234330	0.000000	1.000000	0.430792	0.000000	0.000000	0.051199	0.000000	0.000000	0.000000
3	0.175971	0.000000	0.430792	1.000000	0.105478	0.000000	0.134385	0.000000	0.000000	0.088153
4	0.000000	0.047170	0.000000	0.105478	1.000000	0.000000	0.029076	0.000000	0.039265	0.056656
5	0.000000	0.036705	0.000000	0.000000	0.000000	1.000000	0.045250	0.000000	0.000000	0.000000
6	0.028339	0.095177	0.051199	0.134385	0.029076	0.045250	1.000000	0.000000	0.022674	0.072183
7	0.000000	0.040219	0.000000	0.000000	0.000000	0.000000	0.000000	1.000000	0.033479	0.048308
8	0.045604	0.051276	0.000000	0.000000	0.039265	0.000000	0.022674	0.033479	1.000000	0.028772
9	0.000000	0.112246	0.000000	0.088153	0.056656	0.000000	0.072183	0.048308	0.028772	1.000000
10	0.138945	0.076575	0.123103	0.061216	0.000000	0.000000	0.178788	0.094540	0.075695	0.025199
11	0.054348	0.101994	0.000000	0.000000	0.000000	0.029647	0.028656	0.000000	0.035406	0.000000
12	0.000000	0.104481	0.000000	0.000000	0.052196	0.040616	0.039257	0.000000	0.078736	0.000000
13	0.000000	0.033222	0.000000	0.000000	0.047745	0.000000	0.068527	0.000000	0.119378	0.000000
14	0.080621	0.096843	0.000000	0.000000	0.048897	0.000000	0.082857	0.000000	0.124384	0.000000
15	0.000000	0.178812	0.000000	0.000000	0.000000	0.037528	0.110831	0.041121	0.024492	0.035406
16	0.000000	0.000000	0.131770	0.097706	0.062796	0.000000	0.070146	0.000000	0.143340	0.052199
17	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
18	0.000000	0.000000	0.000000	0.108934	0.070012	0.000000	0.030029	0.000000	0.000000	0.058443
19	0.117923	0.062455	0.106524	0.058443	0.000000	0.025365	0.126400	0.027794	0.016554	0.023103



4. Graph

```
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_matrix2)):
    G.add_node(i, label=df_cleaned2.index[i]) # Menggunakan label kalimat

# Menambahkan tepian (hubungan) berdasarkan kesamaan kosinus
for i in range(len(cosine_sim_matrix2)):
    for j in range(i+1, len(cosine_sim_matrix2)):
        similarity = cosine_sim_matrix2[i][j]
        if similarity > 0.1: # 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_color='black')
nx.draw_networkx_edge_labels(G, pos, edge_labels={(i, j): f"{similarity:.2f}" for i, j, similarity in G.edges(data='weight')})

plt.show()
```


26

5. 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 6: Closeness Centrality = 0.4346
Simpul 15: Closeness Centrality = 0.4111
Simpul 10: Closeness Centrality = 0.3900
Simpul 3: Closeness Centrality = 0.3803
Simpul 2: Closeness Centrality = 0.3710
Simpul 13: Closeness Centrality = 0.3710
Simpul 19: Closeness Centrality = 0.3622
Simpul 0: Closeness Centrality = 0.3537
Simpul 14: Closeness Centrality = 0.3380
Simpul 1: Closeness Centrality = 0.3236
Simpul 16: Closeness Centrality = 0.3236
Simpul 8: Closeness Centrality = 0.3169
Simpul 4: Closeness Centrality = 0.2716
Simpul 18: Closeness Centrality = 0.2716
Simpul 17: Closeness Centrality = 0.2623
Simpul 9: Closeness Centrality = 0.2414
Simpul 11: Closeness Centrality = 0.2414
Simpul 12: Closeness Centrality = 0.2414
Simpul 5: Closeness Centrality = 0.0000
```

Simpul 7: Closeness Centrality = 0.0000

27

6. PageRank

Pagerank merupakan suatu proses mengukur atau mencari nilai penting dalam suatu dokumen

$$PR(S_i) = \frac{1 - \alpha}{\text{NodeCount}} + \alpha \sum_{S_j \in \text{Neighbors } S_i} \frac{PR(S_j)}{\text{CountEdge}(S_j)}$$

Keterangan:

- $PR(S_i)$: Nilai PageRank untuk kalimat S_i
- $PR(S_j)$: Nilai PageRank dari vertex yang bertetangga dengan S_i
- $\text{CountEdge}(S_j)$: Jumlah edge dari kalimat S_j
- α_i : Damping faktor yang nilainya antara 0 dan 1

```
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}
```



7. *EigenVector*

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

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

```
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')]
```

Kalimat penting pada Dokumen 2 berdasarkan 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_cleaned2['Tokenisasi Dokumen2']

# Menampilkan kalimat dari eigenvector centrality dan mengaitkannya dengan dokumen berita
for node, centrality_score in centrality.items():
    if 0 <= node < len(berita):
        kalimat = f"Kalimat berita2: '{berita[node]}' memiliki Eigenvector Centrality sebesar {centrality_score}"
        print(kalimat)
```

```
Kalimat berita2: 'brand apparel olahraga armour mengadakan kompetisi lari mile dalam rangkap hari lari' memiliki Eigenvector Centrality sebesar 0.37
Kalimat berita2: 'kegiatan yang diadakan di senayan jakarta ini diikuti sebanyak peserta mulai dari ora' memiliki Eigenvector Centrality sebesar 0.60
Kalimat berita2: 'armour juga mengundang komunitas lari' memiliki Eigenvector Centrality sebesar 0.60
```

Kalimat berita2: 'kita undang komunitas lari' memiliki Eigenvector Centrality sebesar 0.37

8. Mencari Kata Kunci Pada Dokumen 2

Nilai TF-IDF Kata Kunci Pada Dokumen 2

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

# Data kalimat (contoh)
kalimat = df_cleaned2['Tokenisasi Dokumen2']

# Membuat objek TfidfVectorizer
tfidf_vectorizer = TfidfVectorizer()

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

# Mengonversi matriks TF-IDF ke DataFrame Pandas
tfidf_kata2 = pd.DataFrame(tfidf_matrix.toarray(), columns=tfidf_vectorizer.get_feature_names_out())

# Mengganti NaN dengan 0
tfidf_kata2 = tfidf_kata2.fillna(0)

# Menampilkan nilai TF-IDF dengan kata-kata
tfidf_values_with_words2 = pd.concat([pd.Series(tfidf_vectorizer.get_feature_names_out(), name=
                                                tfidf_kata2], axis=1)

# Menampilkan tabel TF-IDF dengan kata-kata
tfidf_values_with_words2
```

	Kata	acaranya	adapun	ade	agenda	ajang	akan	amerika	anakanak	apalagi	...	ujian
0	acaranya	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	...	0.0
1	adapun	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.289316	0.0	...	0.0
2	ade	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	...	0.0
3	agenda	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	...	0.0
4	ajang	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	...	0.0
...
202	video	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN

	Kata	acaranya	adapun	ade	agenda	ajang	akan	amerika	anakanak	apalagi	...	ujan
203	waktu	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN
204	window	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN
205	yaitu	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN
206	yang	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN

Nilai Cosinus Similarity Pada Dokumen 2

```

from sklearn.metrics.pairwise import cosine_similarity

# Drop the 'Kata' column before calculating cosine similarity
tfidf_vectors_only = tfidf_values_with_words2.drop(columns=['Kata'])

# Handling NaN values by filling them with 0
tfidf_vectors_only = tfidf_vectors_only.fillna(0)

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

# Menampilkan matriks kesamaan kosinus
cosine_sim_df2 = pd.DataFrame(cosine_sim_matrix, columns=tfidf_values_with_words2.index, index=
cosine_sim_df2

```

	0	1	2	3	4	5	6	7	8	9
0	1.000000	0.000000	0.234330	0.175971	0.000000	0.000000	0.028339	0.000000	0.045604	0.000000
1	0.000000	1.000000	0.000000	0.000000	0.047170	0.036705	0.095177	0.040219	0.051276	0.112000
2	0.234330	0.000000	1.000000	0.430792	0.000000	0.000000	0.051199	0.000000	0.000000	0.000000
3	0.175971	0.000000	0.430792	1.000000	0.105478	0.000000	0.134385	0.000000	0.000000	0.088000
4	0.000000	0.04717	0.000000	0.105478	1.000000	0.000000	0.029076	0.000000	0.039265	0.056000
...
202	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
203	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
204	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
205	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
206	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000

Graph Kata Kunci pada Dokumen 2

```

import networkx as nx
import matplotlib.pyplot as plt

```



```

# Assuming tfidf_kata1 is your DataFrame with TF-IDF values
# and top_keywords is the top keywords for each document

# Handling NaN values by filling them with 0
tfidf_kata2 = tfidf_kata2.fillna(0)

# Calculate the sum of TF-IDF values for each term
term_sums = tfidf_kata2.sum()

# Sort terms based on their sum of TF-IDF values in descending order
sorted_terms = term_sums.sort_values(ascending=False)

# Extract the top N keywords (adjust N as needed)
N = 1
top_keywords = sorted_terms.head(N)

# Menghitung kesamaan kosinus antara kalimat-kalimat
cosine_sim_matrix = cosine_similarity(tfidf_kata2, tfidf_kata2)

# Menampilkan matriks kesamaan kosinus
cosine_sim_df2 = pd.DataFrame(cosine_sim_matrix, columns=tfidf_kata2.index, index=tfidf_kata2.index)

# Creating a graph
G = nx.Graph()

# Adding nodes to the graph with top keywords as labels
for document in tfidf_kata2.index:
    keywords_indices = tfidf_kata2.loc[document].values.argsort()[-N:][::-1]
    keywords_list = tfidf_kata2.columns[keywords_indices].tolist()
    G.add_node(document, label=", ".join(map(str, keywords_list)))

# Adding edges to the graph
for i in range(len(cosine_sim_df2.index)):
    for j in range(i + 1, len(cosine_sim_df2.index)):
        G.add_edge(cosine_sim_df2.index[i], cosine_sim_df2.index[j], weight=cosine_sim_df2.iloc[i, j])

# Visualizing the graph
pos = nx.spring_layout(G) # You can use different layouts depending on your preference
edge_labels = nx.get_edge_attributes(G, 'weight')

plt.figure(figsize=(12, 10))
nx.draw(G, pos, with_labels=True, font_size=10, font_color="black", node_size=800, node_color="black",
        nx.draw_networkx_edge_labels(G, pos, edge_labels=edge_labels, font_color="red", font_size=8)

```

```
plt.title('Document Similarity Graph with Top Keywords')
plt.show()
```



Kata Kunci Pada Dokumen 2

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

# Assuming tfidf_kata1 is your DataFrame with TF-IDF values
# and top_keywords is the top keywords for each document

# Handling NaN values by filling them with 0
tfidf_kata2 = tfidf_kata2.fillna(0)

# Calculate the sum of TF-IDF values for each term
term_sums = tfidf_kata2.sum()

# Sort terms based on their sum of TF-IDF values in descending order
```

```

sorted_terms = term_sums.sort_values(ascending=False)

# Extract the top N keywords (adjust N as needed)
N = 1
top_keywords = sorted_terms.head(N)

# Menghitung kesamaan kosinus antara kalimat-kalimat
cosine_sim_matrix = cosine_similarity(tfidf_kata2, tfidf_kata2)

# Menampilkan matriks kesamaan kosinus
cosine_sim_df2 = pd.DataFrame(cosine_sim_matrix, columns=tfidf_kata2.index, index=tfidf_kata2.index)

# Create a list of nodes sorted by the highest TF-IDF value
sorted_nodes = sorted(tfidf_kata2.index, key=lambda x: tfidf_kata2.loc[x].max(), reverse=True)

# Creating a graph
G = nx.Graph()

# Adding nodes to the graph with top keywords as labels
for document in sorted_nodes:
    keywords_indices = tfidf_kata2.loc[document].values.argsort()[-N:][::-1]
    keywords_list = tfidf_kata2.columns[keywords_indices].tolist()
    keywords_values = tfidf_kata2.loc[document, keywords_list].tolist()

    # Sorting keywords and values in descending order
    sorted_keywords_values = sorted(zip(keywords_list, keywords_values), key=lambda x: x[1], reverse=True)

    node_label = ", ".join([f"{kw} ({val:.4f})" for kw, val in sorted_keywords_values])
    G.add_node(document, label=node_label)

# Adding edges to the graph
for i in range(len(cosine_sim_df2.index)):
    for j in range(i + 1, len(cosine_sim_df2.index)):
        G.add_edge(cosine_sim_df2.index[i], cosine_sim_df2.index[j], weight=cosine_sim_df2.iloc[i, j])

# Visualizing the graph
pos = nx.spring_layout(G) # You can use different layouts depending on your preference
edge_labels = nx.get_edge_attributes(G, 'weight')
node_labels = nx.get_node_attributes(G, 'label')

plt.figure(figsize=(12, 10))
nx.draw(G, pos, with_labels=True, font_size=10, font_color="black", node_size=800, node_color="white",
        edge_color="black", edge_labels=edge_labels, font_color="red", font_size=8)
nx.draw_networkx_labels(G, pos, labels=node_labels, font_size=8, font_color="green")

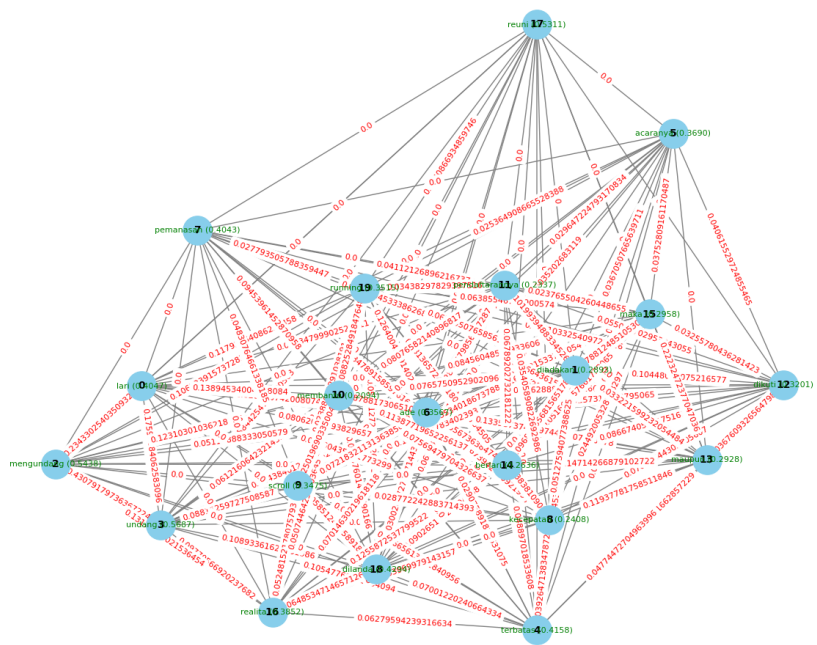
```

```
# Display top keywords and values for each node
for node, label in node_labels.items():
    print(f"Node {node} Keywords and Values: {label}")

plt.title('Document Similarity Graph with Top Keywords and Values (Sorted)')
plt.show()
```

```
Node 3 Keywords and Values: undang (0.5687)
Node 2 Keywords and Values: mengundang (0.5438)
Node 17 Keywords and Values: reuni (0.5311)
Node 18 Keywords and Values: dilanda (0.4294)
Node 4 Keywords and Values: terbatas (0.4158)
Node 0 Keywords and Values: lari (0.4047)
Node 7 Keywords and Values: pemanasan (0.4043)
Node 16 Keywords and Values: realita (0.3852)
Node 5 Keywords and Values: acaranya (0.3690)
Node 6 Keywords and Values: ade (0.3567)
Node 19 Keywords and Values: running (0.3515)
Node 9 Keywords and Values: scroll (0.3475)
Node 12 Keywords and Values: dikuti (0.3201)
Node 15 Keywords and Values: maka (0.2958)
Node 13 Keywords and Values: maupun (0.2928)
Node 1 Keywords and Values: diadakan (0.2893)
Node 14 Keywords and Values: berlari (0.2636)
Node 8 Keywords and Values: kecepatan (0.2408)
Node 11 Keywords and Values: pendaftarannya (0.2337)
Node 10 Keywords and Values: membantu (0.2094)
```

Document Similarity Graph with Top Keywords and Values (Sorted)





30

Summary

In summary, this book has no content whatsoever.



References

