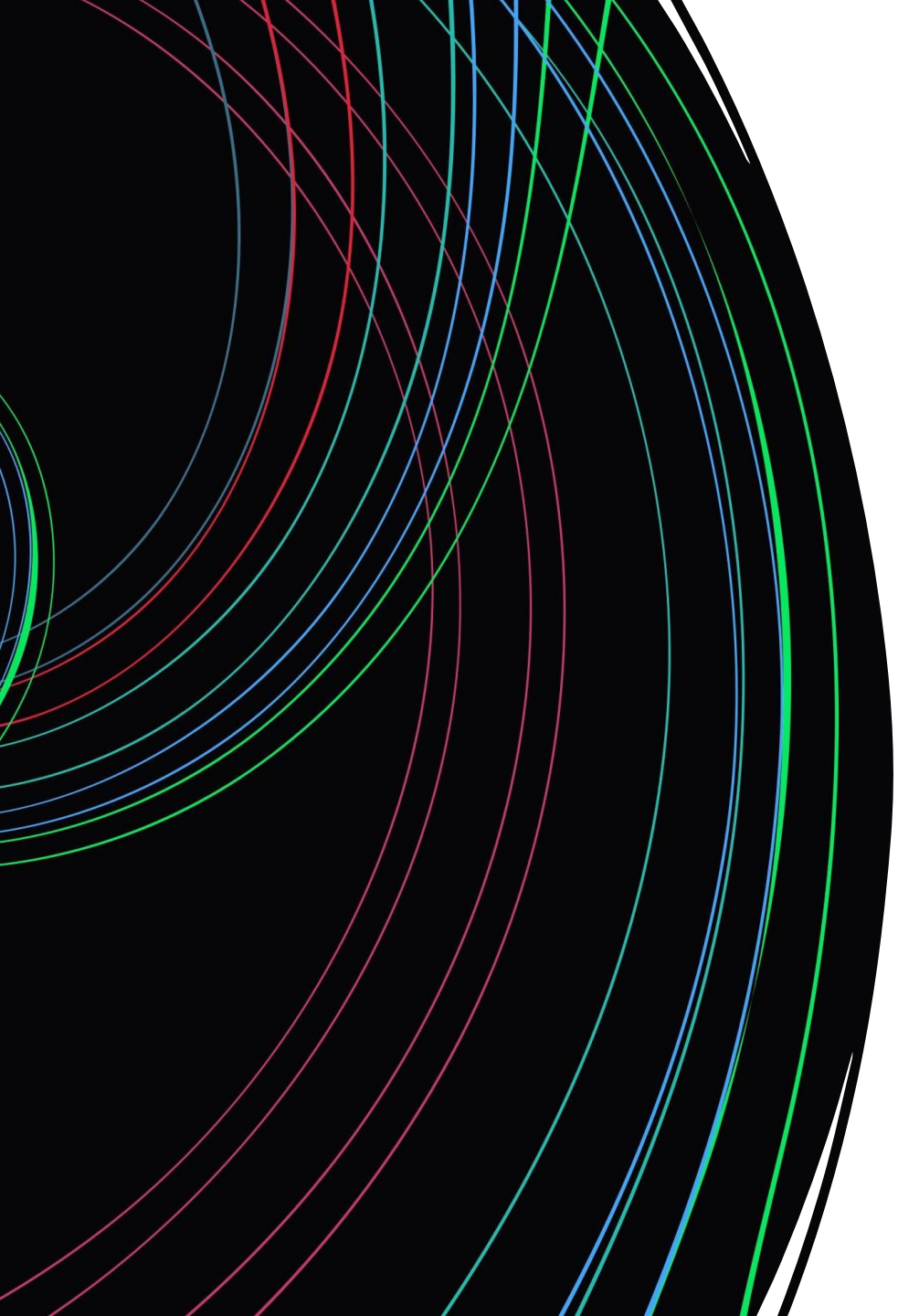


PEMBOBOTAN KATA MENGGUNAKAN TF-IDF

Chapter 4

Sains Data

TF-IDF

Term Frequency-Inverse Document Frequency

- Metode pembobotan kata dalam dokumen yang digunakan dalam Natural Language Processing (NLP) dan Information Retrieval (IR).
- Teknik ini bertujuan untuk menilai sebuah kata dalam dokumen yang relatif terhadap kumpulan dokumen (corpus).

FORMULA TF

- **Mengukur seberapa sering sebuah kata muncul dalam satu dokumen**

$$TF(t,d) = \frac{f(t,d)}{\sum_{t' \in d} f(t',d)}$$

- t = kata yang sedang dihitung frekuensinya
- d = dokumen tertentu
- $f(t,d)$ = jumlah kemunculan kata t dalam dokumen d
- $\sum_{t' \in d} f(t',d)$ = total jumlah kata dalam dokumen d

FORMULA IDF

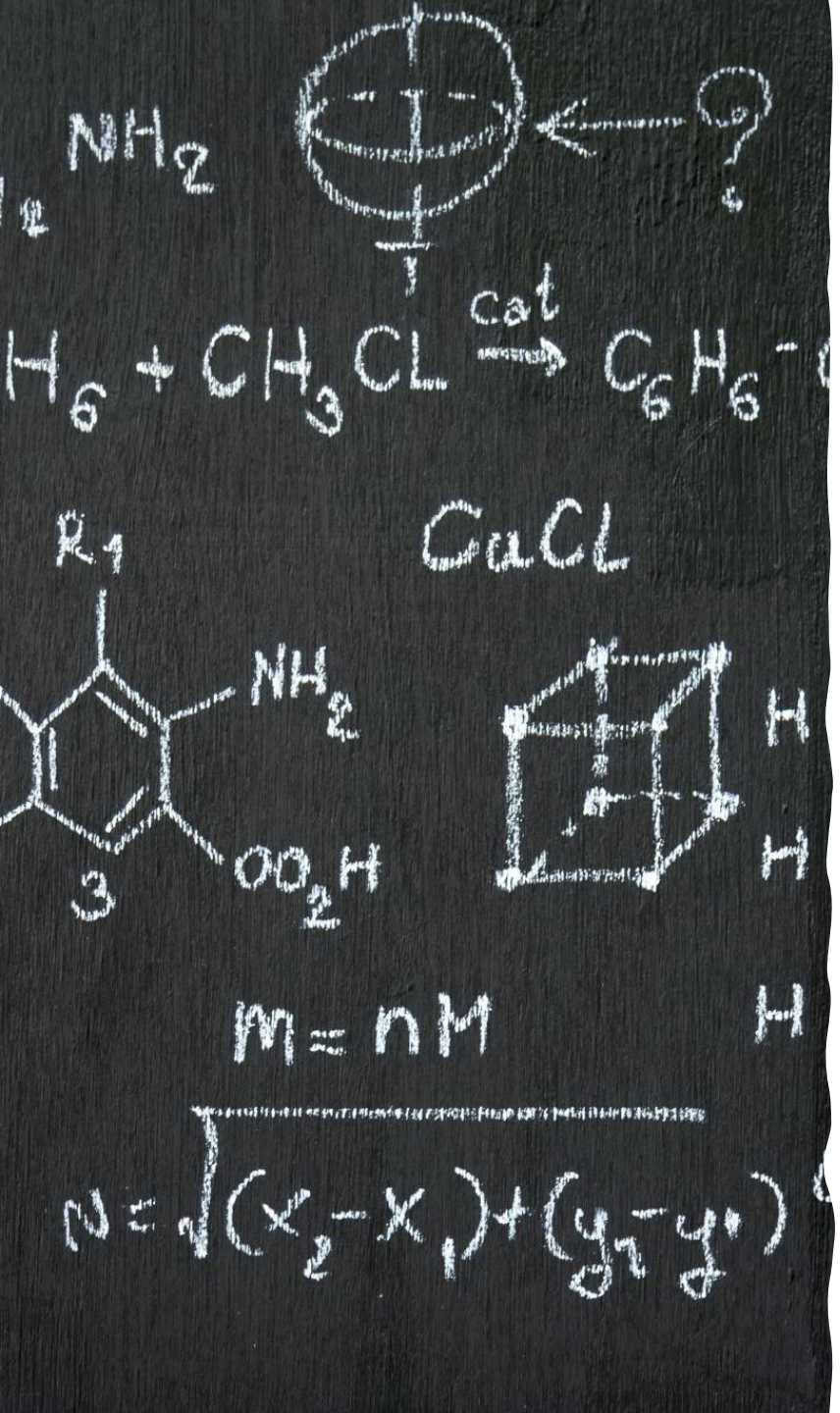
- **Mengukur seberapa jarang atau unik sebuah kata dalam seluruh koleksi dokumen. Kata yang muncul di banyak dokumen memiliki nilai IDF yang rendah, sementara kata yang jarang muncul memiliki nilai IDF yang tinggi**

$$IDF(t, D) = \log \frac{|D|}{1 + |\{d \in D : t \in d\}|}$$

- D = jumlah total dokumen dalam kumpulan
- $|\{d \in D : t \in d\}|$ = jumlah dokumen yang mengandung kata t
- Ditambahkan **1** dalam penyebut untuk menghindari pembagian dengan nol jika kata tidak ada di dokumen mana pun.

FORMULA TF-IDF

$$TF - IDF(t, d, D) = TF(t, d) \times IDF(t, D)$$



JENIS TF-IDF

- TF-IDF Unigram → ngram_range = (1,1)
- TF-IDF Unigram dan Bigram → ngram_range = (1,2)
- TF-IDF Bigram → ngram_range = (2,2)
- TF-IDF Trigram → ngram_range = (3,3)

LIB YANG DIGUNAKAN

Perhitungan TF-IDF pada Python menggunakan lib

Scikit-learn : "sklearn"

Codingnya

IMPORT DATA

```
import pandas as pd
import numpy as np

dm = pd.read_csv("tomlembong50_clean.csv", usecols=["tweet_clean", "Label"])
dm.columns = ["tweet_clean", "Label"]
dm.head(10)
```

MENJUMLAHKAN FITUR DAN STRING LIST

```
#checking the count of the dependent variable
dm['Label'].value_counts()
```

```
Positif    26
Negatif    24
Name: Label, dtype: int64
```

```
# convert list formated string to list
import ast

def convert_text_list(texts):
    texts = ast.literal_eval(texts)
    return [text for text in texts]

dm["tekslist"] = dm["tweet_clean"].apply(convert_text_list)

print(dm["tekslist"][23])

print("\ntype : ", type(dm["tekslist"][23]))
```

PERHITUNGAN TF

```python
def calc_TF(document):
    #perhitungan jumlah kata
    TF_dict = {}
    for term in document:
        if term in TF_dict:
            TF_dict[term] += 1
        else:
            TF_dict[term] = 1
    #perhitungan tf
    for term in TF_dict:
        TF_dict[term] = TF_dict[term] / len(document)
    return TF_dict

dm["TF_dict"] = dm['tekslist'].apply(calc_TF)

dm["TF_dict"].head()
```

```
0    {'acara': 0.14285714285714285, 'tahan': 0.1428...
1    {'acara': 0.04, 'tom': 0.04, 'lembong': 0.04, ...
2    {'acara': 0.03125, 'tom': 0.0625, 'lembong': 0...
3    {'acara': 0.16666666666666666, 'tom': 0.166666...
4    {'adil': 0.14285714285714285, 'hukum': 0.07142...
Name: TF_dict, dtype: object
```

```python
index = 23

print('%20s' % "term", "\t", "TF\n")
for key in dm["TF_dict"][index]:
    print('%20s' % key, "\t", dm["TF_dict"][index][key])
```

PERHITUNGAN IDF

```
def calc_DF(tfDict):
    count_DF = {}
    for document in tfDict:
        for term in document:
            if term in count_DF:
                count_DF[term] += 1
            else:
                count_DF[term] = 1
    return count_DF

DF = calc_DF(dm["TF_dict"])
```

```
#menghitung idf

n_document = len(dm)
def calc_IDF(__n_document, __DF):
    IDF_Dict = {}
    for term in __DF:
        IDF_Dict[term] = np.log(__n_document / (__DF[term] + 1))
    return IDF_Dict

#penyimpanan kamus idf
IDF = calc_IDF(n_document, DF)
```

```
#perhitungan TF-IDF
def calc_TF_IDF(TF):
    TF_IDF_Dict = {}
    for key in TF:
        TF_IDF_Dict[key] = TF[key] * IDF[key]
    return TF_IDF_Dict

#penyimpanan variabel TF-IDF
dm["TF-IDF_dict"] = dm["TF_dict"].apply(calc_TF_IDF)
```

HASIL PERHITUNGAN TF-IDF

```
# memunculkan nilai TF-IDF
index = 23

print('%20s' % "term", "\t", '%10s' % "TF", "\t", '%20s' % "TF-IDF\n")
for key in dm["TF-IDF_dict"][index]:
    print('%20s' % key, "\t", dm["TF_dict"][index][key] ,"\t" , dm["TF-IDF_dict"][index][key])
```

term	TF	TF-IDF
azab	0.043478260869565216	0.13995112282035654
lho	0.043478260869565216	0.13995112282035654
mul	0.043478260869565216	0.13995112282035654
saksi	0.043478260869565216	0.10981428888296763
kali	0.043478260869565216	0.13995112282035654
hidup	0.043478260869565216	0.13995112282035654
kalo	0.043478260869565216	0.13995112282035654
ga	0.043478260869565216	0.10981428888296763
ya	0.043478260869565216	0.12232220507652332
anak	0.043478260869565216	0.12232220507652332
turun	0.043478260869565216	0.13995112282035654
mu	0.043478260869565216	0.13995112282035654
nandur	0.043478260869565216	0.13995112282035654
ngunduh	0.043478260869565216	0.13995112282035654
mulll	0.043478260869565216	0.13995112282035654
gusti	0.043478260869565216	0.13995112282035654
mboten	0.043478260869565216	0.13995112282035654
sare	0.043478260869565216	0.13995112282035654
yg	0.043478260869565216	0.06583164054912068
sabar	0.043478260869565216	0.13995112282035654
tom	0.043478260869565216	0.0036252873451761303
lembong	0.043478260869565216	0.0026902349442646718
keluarga	0.043478260869565216	0.13995112282035654

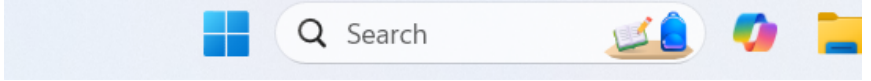
MATRIK TF-IDF

```
#matrik tf-idf
# pengurutan descending berdasarkan nilai DF
sorted_DF = sorted(DF.items(), key=lambda kv: kv[1], reverse=True)[:30]

# pembuatan list kata dari pengurutan `sorted_DF`
unique_term = [item[0] for item in sorted_DF]

def calc_TF_IDF_Vec(__TF_IDF_Dict):
    TF_IDF_vector = [0.0] * len(unique_term)

    # For each unique word, if it is in the review, store its TF-IDF value.
    for i, term in enumerate(unique_term):
        if term in __TF_IDF_Dict:
            TF_IDF_vector[i] = __TF_IDF_Dict[term]
    return TF_IDF_vector

dm["TF_IDF_Vec"] = dm["TF-IDF_dict"].apply(calc_TF_IDF_Vec)

print("tampil baris pertama matrix TF_IDF_Vec Series\n")
print(dm["TF_IDF_Vec"][0])

print("\nukuran matrix : ", len(dm["TF_IDF_Vec"][0]))
```

MENAMPILKAN KE DALAM LIST

```
#menampilkan top 30 term tf-idf

# konversi ke dalam List
TF_IDF_Vec_List = np.array(dm["TF_IDF_Vec"].to_list())

sums = TF_IDF_Vec_List.sum(axis=0)

data = []

for col, term in enumerate(unique_term):
    data.append((term, sums[col]))

ranking = pd.DataFrame(data, columns=['term', 'rank'])
ranking.sort_values('rank', ascending=False)
```

	term	rank
3		2.154247
2	anies	1.466529
26	aja	1.451356
14	bebas	1.404827
8	praperadilan	1.331056
5	gula	1.238108
7	impor	1.169731
6	sangka	1.017534
4	yg	1.015680
23	adil	0.959909
13	bukti	0.947504
10	jagung	0.927870
9	hukum	0.918525



NEXT..... PEMODELAN DENGAN ALGORITMA MACHINE LEARNING