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

```
Mounted at /content/drive
```

```
!ls /content/drive/MyDrive/Cosinus-Similarity
```

```
16595-37036-1-PB.pdf  feature_extraction.py  source4.txt                     term-frequency.csv
17167-37507-1-PB.pdf  README.md              source5.txt                     tf-idf.csv
17342-37530-1-PB.pdf  source1.txt            source_raw_for_extraction.txt  untitled
17526-37540-1-PB.pdf  source2.txt            source.txt                      Untitled.ipynb
17838-37545-1-PB.pdf  source3.txt            stopwords.txt
```

```
!cp -r /content/drive/MyDrive/Cosinus-Similarity/* /content/
```

```
pip install numpy==1.26.4 -U scikit-learn Sastrawi
```

```
Requirement already satisfied: numpy==1.26.4 in /usr/local/lib/python3.11/dist-packages (1.26.4)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.11/dist-packages (1.6.1)
Collecting Sastrawi
  Downloading Sastrawi-1.0.1-py2.py3-none-any.whl.metadata (909 bytes)
Requirement already satisfied: scipy>=1.6.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn) (1.13.1)
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn) (1.4.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn) (3.5.0)
Downloading Sastrawi-1.0.1-py2.py3-none-any.whl (209 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 209.7/209.7 kB 2.5 MB/s eta 0:00:00
Installing collected packages: Sastrawi
Successfully installed Sastrawi-1.0.1
```

```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
from Sastrawi.Stemmer.StemmerFactory import StemmerFactory
import re
```

```
REGEX = re.compile(r"\s")
def tokenize(text):
    return [tok.strip().lower() for tok in REGEX.split(text)]

def stopwords(text):
    reg = re.compile(r"\n")
    return reg.split(text)
```

```
file = open("source1.txt","r");
raw1 = file.read()

file = open("source2.txt","r");
raw2 = file.read()

file = open("source3.txt","r");
raw3 = file.read()

file = open("source4.txt","r");
raw4 = file.read()

file = open("source5.txt","r");
raw5 = file.read()
```

```
# menghilangkan tanda baca
tandabaca = [".",",","-","%"]
for td in tandabaca:
    raw1=raw1.replace(td,"")
    raw2=raw2.replace(td,"")
    raw3=raw3.replace(td,"")
    raw4=raw4.replace(td,"")
    raw5=raw5.replace(td,"")
```

```
# menghilangkan stop words
file = open("stopwords.txt","r");
st = file.read()
stopwords = stopwords(st)

for word in stopwords:
    raw1=raw1.replace(" "+word+" "," ")
    raw2=raw2.replace(" "+word+" "," ")
    raw3=raw3.replace(" "+word+" "," ")
    raw4=raw4.replace(" "+word+" "," ")
    raw5=raw5.replace(" "+word+" "," ")


# stemming
factory = StemmerFactory()
stemmer = factory.create_stemmer()

hasilstem1 = stemmer.stem(raw1)
hasilstem2 = stemmer.stem(raw2)
hasilstem3 = stemmer.stem(raw3)
hasilstem4 = stemmer.stem(raw4)
hasilstem5 = stemmer.stem(raw5)


#tokenization
train_set = [hasilstem1,hasilstem2,hasilstem3,hasilstem4,hasilstem5]


count_vectorizer = CountVectorizer(tokenizer=tokenize)
data = count_vectorizer.fit_transform(train_set).toarray()
vocab = count_vectorizer.get_feature_names_out()
```

```
/usr/local/lib/python3.11/dist-packages/sklearn/feature_extraction/text.py:517: UserWarning: The parameter 'token_pattern' will not be u
  warnings.warn(
```

```
print("Jumlah Term FREQUENCY=============================")
print(data)
```

```
Jumlah Term FREQUENCY=============================
[[0 0 0 ... 0 0 1]
 [1 1 1 ... 3 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 5 0]
 [0 0 0 ... 0 0 0]]
```

```
print("VECTOR FITUR=============================")
print(vocab)
```

```
VECTOR FITUR=============================
['120' '30' '80' '90' '9286' 'accumulative' 'aceh' 'adi' 'adu' 'ahp' 'aju'
 'akses' 'aksi' 'akurasi' 'alternatif' 'ambil' 'analisa' 'analytical'
 'anggap' 'apache' 'aparatur' 'atas' 'badan' 'baik' 'bakat' 'balap'
 'banding' 'bangun' 'bantu' 'bas' 'basis' 'bayes' 'beda' 'belimbing'
 'bidang' 'bkd' 'blue' 'bobot' 'borda' 'buah' 'buka' 'bukti' 'burundi'
 'butuh' 'calon' 'cctv' 'change' 'cipta' 'circuit' 'ciri' 'citra' 'closed'
 'cocok' 'coratcoret' 'coretcoret' 'cropping' 'daerah' 'dalam' 'dan'
 'dari' 'dasar' 'data' 'database' 'dekat' 'dengan' 'detection' 'deteksi'
 'di' 'diamana' 'differences' 'dinding' 'dindingdinding' 'dukung' 'dunia'
 'e' 'efektif' 'efisien' 'egovernment' 'ekspektasi' 'ekstraksi'
 'elektronik' 'end' 'fitur' 'front' 'fungsi' 'gabung' 'gam' 'gatewayhasil'
 'gdss' 'gera' 'gerak' 'gihosha' 'government' 'green' 'guna' 'hadap'
 'harap' 'harus' 'hasil' 'hierarchy' 'ideal' 'identifikasi' 'ijin'
 'illumination' 'images' 'industri' 'internet' 'invariant' 'isi' 'jabat'
 'jadi' 'judi' 'kabupaten' 'kamera' 'kandidat' 'kantor' 'kelahi' 'kelola'
 'kelompok' 'keluh' 'kembang' 'kemudian' 'kerja' 'ketidaksesuaian'
 'klasifikasi' 'kompetensi' 'kompetisi' 'komputer' 'komunikasi' 'kondisi'
 'kota' 'kriteria' 'kualitas' 'kulit' 'kurang' 'lalu' 'lapang' 'latih'
 'layan' 'lebih' 'liar' 'lingkung' 'lowong' 'lulus' 'mahasiswa' 'maju'
 'maksimal' 'maksimum' 'malam' 'mampu' 'mana' 'manajemen' 'manis'
 'manusia' 'masalah' 'masingmasing' 'masyarakat' 'mata' 'matching'
 'mengganggap' 'metode' 'milik' 'model' 'modern' 'mysql' 'nai' 'nakal'
 'negara' 'negaranegara' 'nepotisme' 'nilai' 'objektif' 'oleh' 'pada'
 'pagi' 'panen' 'pasca' 'pegawai' 'pemrosesan' 'pengaruh' 'perankingan'
 'perintah' 'php' 'pilih' 'pkl' 'pns' 'politeknik' 'praktek' 'process'
 'prodi' 'produksi' 'profil' 'profile' 'properti' 'proses' 'publik'
 'putus' 'rangking' 'red' 'referensi' 'rekomendasi' 'remaja' 'rendah'
 'republik' 'rgb' 'saat' 'saing' 'salah' 'sedia' 'segi' 'selatan'
 'sepenuh' 'server' 'sesuai' 'siang' 'simpul' 'simulasi' 'sipil' 'sistem'
 'skala' 'sms' 'solusi' 'sore' 'sortir' 'struktural' 'studi' 'subkriteria'
 'sulit' 'swasta' 'tambah' 'tara' 'teknik' 'teknologi' 'television'
 'teliti' 'tempat' 'tengahtengah' 'tentu' 'terap' 'tinggi' 'tingkat'
 'tuju' 'tunjang' 'ubah' 'uji' 'undangundang' 'untuk' 'upaya' 'usah'
 'usaha' 'usul' 've' 'video' 'visual' 'warga' 'warna' 'web' 'webcam']
```

```
print("JUMLAH VECTOR FITUR=============================")
print(len(vocab))
```

```
JUMLAH VECTOR FITUR=============================
258
```

```
tfidf = TfidfVectorizer().fit_transform(train_set)
pairwise_similarity = tfidf * tfidf.T
```

```
print("Jumlah Term FREQUENCY-Inverse Document Frequency=============================")
print(tfidf)
```

```
Jumlah Term FREQUENCY-Inverse Document Frequency=============================
   (0, 172)     0.06697710374982785
   (0, 162)     0.06697710374982785
   (0, 165)     0.1339542074996557
   (0, 200)     0.1339542074996557
   (0, 235)     0.06697710374982785
   (0, 155)     0.1339542074996557
   (0, 56)      0.054036705583222766
   (0, 129)     0.054036705583222766
   (0, 115)     0.06697710374982785
   (0, 25)      0.06697710374982785
   (0, 139)     0.06697710374982785
   (0, 110)     0.06697710374982785
   (0, 54)      0.06697710374982785
   (0, 70)      0.2679084149993114
   (0, 101)     0.06697710374982785
   (0, 12)      0.20093131124948355
   (0, 53)      0.20093131124948355
   (0, 71)      0.06697710374982785
   (0, 27)      0.06697710374982785
   (0, 114)     0.06697710374982785
   (0, 192)     0.06697710374982785
   (0, 194)     0.054036705583222766
   (0, 227)     0.06697710374982785
   (0, 97)      0.1132011958606759
   (0, 242)     0.21614682233289106
   :     :
   (4, 209)     0.15215061287635417
   (4, 122)     0.05071687095878472
   (4, 124)     0.05071687095878472
   (4, 208)     0.05071687095878472
   (4, 248)     0.2028674838351389
   (4, 245)     0.05071687095878472
   (4, 21)      0.05071687095878472
   (4, 126)     0.10143374191756944
   (4, 117)     0.2028674838351389
   (4, 87)      0.05071687095878472
   (4, 9)       0.10143374191756944
   (4, 17)      0.05071687095878472
   (4, 98)      0.05071687095878472
   (4, 187)     0.05071687095878472
   (4, 38)      0.10143374191756944
   (4, 236)     0.15215061287635417
   (4, 37)      0.05071687095878472
   (4, 14)      0.2028674838351389
   (4, 196)     0.15215061287635417
   (4, 247)     0.05071687095878472
   (4, 154)     0.05071687095878472
   (4, 179)     0.05071687095878472
   (4, 58)      0.05071687095878472
   (4, 109)     0.05071687095878472
   (4, 188)     0.05071687095878472
```

```
print("Jumlah COSINE-SIMILARITY=============================")
print(pairwise_similarity)
```

```
Jumlah COSINE-SIMILARITY=============================
   (0, 4)       0.041108800282407286
   (0, 1)       0.15788391163208024
   (0, 3)       0.0632432820289876
   (0, 2)       0.06346285110717129
   (0, 0)       1.0000000000000004
   (1, 3)       0.02802293346893107
   (1, 4)       0.02471163501230499
   (1, 2)       0.05430007173060675
   (1, 1)       0.9999999999999998
   (1, 0)       0.15788391163208024
   (2, 3)       0.04733465662690203
   (2, 4)       0.12971176285436797
```

```
  (2, 1)        0.05430007173060675
  (2, 2)        0.9999999999999989
  (2, 0)        0.06346285110717129
  (3, 4)        0.03499910912100677
  (3, 2)        0.04733465662690203
  (3, 1)        0.02802293346893107
  (3, 3)        0.9999999999999986
  (3, 0)        0.0632432820289876
  (4, 3)        0.03499910912100677
  (4, 4)        0.9999999999999996
  (4, 2)        0.12971176285436797
  (4, 1)        0.02471163501230499
  (4, 0)        0.041108800282407286
```