Nama : Raihan Rahmanda Junianto

NIM : 222112303

Kelas : 3SD2

Responsi Praktikum 4 Information Retrieval

---

1. **Buat vector space model dengan menggunakan sekumpulan dokumen pada folder "berita".**

   Berdasarkan soal di atas, maka dibuatlah kode program sebagai berikut.

```
import os
from spacy.lang.id import Indonesian
from Sastrawi.Stemmer.StemmerFactory import StemmerFactory
from spacy.lang.id.stop_words import STOP_WORDS

nlp = Indonesian()

path = "D:/RAIHAN STIS/Perkuliahan/SEMESTER 5/Praktikum INFORMATION
RETRIEVAL/Pertemuan (2)/berita"

def read_text_file(file_path):
    with open(file_path, 'r') as f:
        content = f.read()
    return content

def preprocess_text(text):
    stemmer = StemmerFactory().create_stemmer()
    stemmed_text = stemmer.stem(text)

    doc = nlp(stemmed_text)
    tokens = [token.text for token in doc if token.text.lower() not in
STOP_WORDS]

    return tokens

inverted_index = {}
doc_dict = {}
document_index = 1

for file in os.listdir(path):
    if os.path.isfile(os.path.join(path, file)) and file.endswith(".txt"):
        file_path = os.path.join(path, file)

        text = read_text_file(file_path)
```

```
        cleaned_tokens = preprocess_text(text)
        doc_dict[document_index] = " ".join(cleaned_tokens)
        document_index += 1

        # Update the inverted index
        for token in set(cleaned_tokens):  # Use set to avoid duplicate
documents
            inverted_index.setdefault(token, []).append(file)

vocab=list(inverted_index.keys())

def termFrequencyInDoc(vocab, doc_dict):
    tf_docs = {}
    for doc_id in doc_dict.keys():
        tf_docs[doc_id] = {}
    for word in vocab:
        for doc_id,doc in doc_dict.items():
            tf_docs[doc_id][word] = doc.count(word)
    return tf_docs

def wordDocFre(vocab, doc_dict):
    df = {}
    for word in vocab:
        frq = 0
        for doc in doc_dict.values():
            if word in doc:
                frq = frq + 1
        df[word] = frq
    return df

import numpy as np
def inverseDocFre(vocab,doc_fre,length):
    idf= {}
    for word in vocab:
        idf[word] = idf[word] = 1 + np.log((length + 1) /
(doc_fre[word]+1))
    return idf

def tfidf(vocab,tf,idf_scr,doc_dict):
    tf_idf_scr = {}
    for doc_id in doc_dict.keys():
        tf_idf_scr[doc_id] = {}
    for word in vocab:
        for doc_id,doc in doc_dict.items():
            tf_idf_scr[doc_id][word] = tf[doc_id][word] * idf_scr[word]
    return tf_idf_scr
```

```
tf_idf = tfidf(vocab, termFrequencyInDoc(vocab, doc_dict),
inverseDocFre(vocab, wordDocFre(vocab, doc_dict), len(doc_dict)),
doc_dict)
# Term - Document Matrix
TD = np.zeros((len(vocab), len(doc_dict)))
for word in vocab:
    for doc_id,doc in tf_idf.items():
        ind1 = vocab.index(word)
        ind2 = list(tf_idf.keys()).index(doc_id)
        TD[ind1][ind2] = tf_idf[doc_id][word]
print(TD)
```

Proses pembuatan vektor space model diawali dengan menyusun indeks terbalik yang telah dilakukan pada praktikum pertemuan sebelumnya. Pada praktikum kali ini, library yang digunakan masih sama seperti sebelumnya, hanya saja ditambahkan library “NumPy” untuk mengakomodasi operasi numerik seperti array multidimensi (matriks).

Setelah indeks terbalik tersedia, indeks tersebut akan disimpan ke dalam variabel vocab yang berisi kumpulan term pada corpus. Selanjutnya, terdapat juga inisialisasi variabel doc_dict yang nantinya akan memuat isi dokumen (kalimat atau paragraf) yang telah dilakukan preprocessing, termasuk tokenisasi dan stemming dengan doc_id sebagai key-nya. Pada potongan kode program ini juga menjelaskan bahwa terdapat inisialisasi document_index yang berperan sebagai indeks bagi variabel doc_dict untuk kemudahan pengaksesan kamus tersebut. Sebagai gambaran, potongan kode program dapat dilihat pada gambar berikut.

```
inverted_index = {}
doc_dict = {}
document_index = 1

for file in os.listdir(path):
    if os.path.isfile(os.path.join(path, file)) and file.endswith(".txt"):
        file_path = os.path.join(path, file)

        text = read_text_file(file_path)

        cleaned_tokens = preprocess_text(text)
        doc_dict[document_index] = " ".join(cleaned_tokens)
        document_index += 1

        # Update the inverted index
        for token in set(cleaned_tokens):  # Use set to avoid duplicate documents
            inverted_index.setdefault(token, []).append(file)

vocab=list(inverted_index.keys())
```

Proses selanjutnya adalah menghitung document frequency serta inverse document frequency. Fungsi pertama yang dijalankan adalah fungsi “termFrequencyInDoc(vocab, doc_dict)”. Fungsi tersebut akan menghitung Term Frequency (TF) yaitu jumlah kemunculan setiap kata dalam setiap dokumen yang terdapat di dalam variabel doc_dict. Dalam kata lain, TF mengukur seberapa sering sebuah kata muncul dalam dokumen tertentu. Fungsi ini mengembalikan kamus (dictionary) yang berisi TF untuk setiap kata dalam setiap dokumen.

```
def termFrequencyInDoc(vocab, doc_dict):
    tf_docs = {}
    for doc_id in doc_dict.keys():
        tf_docs[doc_id] = {}
    for word in vocab:
        for doc_id,doc in doc_dict.items():
            tf_docs[doc_id][word] = doc.count(word)
    return tf_docs
```

Fungsi selanjutnya adalah fungsi “wordDocFre(vocab, doc_dict)” yang menghitung Document Frequency (DF) untuk setiap kata dalam vocab. DF mengukur berapa banyak dokumen yang mengandung sebuah kata tertentu. Fungsi ini mengembalikan variabel kamus yang berisi DF untuk setiap kata dalam vocab. Pada fungsi ini tidak perlu dilakukan tokenisasi lagi pada doc karena variabel doc_dict sudah melalui proses preprocessing, termasuk tokensasi.

```
def wordDocFre(vocab, doc_dict):
    df = {}
    for word in vocab:
        frq = 0
        for doc in doc_dict.values():
            if word in doc:
                frq = frq + 1
        df[word] = frq
    return df
```

Selanjutnya, terdapat fungsi “inverseDocFre(vocab,doc_fre,length)” yang menggunakan library “NumPy” untuk menghitung Inverse Document Frequency (IDF) untuk setiap kata dalam vocab. IDF digunakan untuk memberikan bobot kepada kata-kata yang jarang muncul dalam dokumen tetapi mungkin memiliki nilai informatif yang tinggi. Fungsi ini mengembalikan kamus yang berisi IDF untuk setiap kata dalam vocab. Fungsi

ini menggunakan dengan rumus logaritma dari total dokumen yang ada dibagi oleh Document Frequency dari kata tersebut.

```
import numpy as np
def inverseDocFre(vocab,doc_fre,length):
    idf= {}
    for word in vocab:
        idf[word] = idf[word] = 1 + np.log((length + 1) / (doc_fre[word]+1))
    return idf
```

Setelah proses di atas selesai, maka dilanjutkan dengan menghitung nilai TF-IDF yang nantinya akan dimasukkan ke dalam matriks Term-Document (TD). Pada fungsi "tfidf(vocab,tf,idf_scr,doc_dict)", nilai TF-IDF dihitung dengan mengalikan variabel "tf" dan "idf" berdasarkan vocab serta doc_id.

```
def tfidf(vocab,tf,idf_scr,doc_dict):
    tf_idf_scr = {}
    for doc_id in doc_dict.keys():
        tf_idf_scr[doc_id] = {}
    for word in vocab:
        for doc_id,doc in doc_dict.items():
            tf_idf_scr[doc_id][word] = tf[doc_id][word] * idf_scr[word]
    return tf_idf_scr

tf_idf = tfidf(vocab, termFrequencyInDoc(vocab, doc_dict), inverseDocFre(vocab, wordDocFre(vocab, doc_dict), len(doc_dict)), doc_dict)
```

Setelah nilai TF-IDF masing-masing kata dalam setiap dokumen diperoleh, maka akan dimasukkan ke dalam matriks TD.

```
TD = np.zeros((len(vocab), len(doc_dict)))
for word in vocab:
    for doc_id,doc in tf_idf.items():
        ind1 = vocab.index(word)
        ind2 = list(tf_idf.keys()).index(doc_id)
        TD[ind1][ind2] = tf_idf[doc_id][word]
print(TD)
```

Setelah semua proses dilakukan, maka output yang tertampil adalah sebagai berikut.

```
(base) D:\RAIHAN STIS\Perkuliahan\SEMESTER 5\Praktikum INFORMATION RETRIEVAL\Pertemuan (4)>python penugasan4_1.py
[[ 3.38629436  0.          1.69314718  0.          0.        ]
 [ 2.          2.          2.          2.          2.        ]
 [ 4.19722458  0.          0.          0.          0.        ]
 [ 2.09861229  0.          0.          0.          0.        ]
 [ 2.09861229  0.          0.          0.          0.        ]
 [ 2.09861229  0.          0.          0.          0.        ]
 [ 4.19722458  0.          0.          0.          0.        ]
 [ 5.91160778  0.          1.18232156  2.36464311  1.18232156]
 [ 2.09861229  0.          0.          0.          0.        ]
 [ 2.09861229  0.          0.          0.          0.        ]
 [ 4.19722458  0.          0.          0.          0.        ]
 [ 6.          3.         11.          9.          5.        ]
 [ 2.09861229  0.          0.          0.          0.        ]
 [ 2.09861229  0.          0.          0.          0.        ]
 [ 1.69314718  1.69314718  0.          0.          0.        ]
 [ 4.19722458  0.          0.          0.          0.        ]
 [ 4.19722458  0.          0.          0.          0.        ]
 [ 3.38629436  0.          0.          0.          1.69314718]
 [ 2.09861229  0.          0.          0.          0.        ]
 [ 2.09861229  0.          0.          0.          0.        ]
 [ 1.69314718  0.          0.          1.69314718  0.        ]
 [ 3.          7.          4.          2.          3.        ]
 [ 6.29583687  0.          0.          0.          0.        ]
 [ 2.09861229  0.          0.          0.          0.        ]
 [16.         18.         19.         12.         16.        ]
 [ 1.          1.          1.          1.          1.        ]

 [ 7.          4.         11.          2.          2.        ]
 [ 1.40546511  1.40546511  1.40546511  0.          0.        ]
 [ 2.          2.          2.          2.          2.        ]
 [ 1.          1.          1.          1.          1.        ]
 [ 1.69314718  0.          0.          0.          1.69314718]
 [11.         16.         22.         17.         21.        ]
 [ 1.          1.          1.          4.          1.        ]
 [ 2.09861229  0.          0.          0.          0.        ]
 [ 2.09861229  0.          0.          0.          0.        ]
 [ 1.          1.          1.          1.          1.        ]
 [ 2.09861229  0.          0.          0.          0.        ]
 [ 2.09861229  0.          0.          0.          0.        ]
 [ 3.38629436  0.          0.          0.          1.69314718]
 [ 1.69314718  0.          0.          1.69314718  0.        ]
 [ 6.29583687  0.          0.          0.          0.        ]
 [ 2.09861229  0.          0.          0.          0.        ]
 [ 2.09861229  0.          0.          0.          0.        ]
 [ 4.19722458  0.          0.          0.          0.        ]
 [ 2.09861229  0.          0.          0.          0.        ]
 [ 3.          7.          4.          2.          3.        ]
 [ 2.09861229  0.          0.          0.          0.        ]
 [ 1.69314718  0.          0.          3.38629436  0.        ]
 [ 1.          1.          1.          1.          1.        ]
 [ 1.69314718  1.69314718  0.          0.          0.        ]
 [ 2.09861229  0.          0.          0.          0.        ]
 [ 2.09861229  0.          0.          0.          0.        ]
 [ 2.09861229  0.          0.          0.          0.        ]
 [ 2.09861229  0.          0.          0.          0.        ]

 [ 3.          7.          4.          2.          3.        ]
 [ 2.09861229  0.          0.          0.          0.        ]
 [ 1.69314718  0.          0.          3.38629436  0.        ]
 [ 1.          1.          1.          1.          1.        ]
 [ 1.69314718  1.69314718  0.          0.          0.        ]
 [ 2.09861229  0.          0.          0.          0.        ]
 [ 2.09861229  0.          0.          0.          0.        ]
 [ 2.09861229  0.          0.          0.          0.        ]
 [ 2.09861229  0.          0.          0.          0.        ]
 [ 2.09861229  0.          0.          0.          0.        ]
 [ 4.19722458  0.          0.          0.          0.        ]
 [ 1.69314718  1.69314718  0.          0.          0.        ]
 [ 4.19722458  0.          0.          0.          0.        ]
 [ 2.09861229  0.          0.          0.          0.        ]
 [ 2.09861229  0.          0.          0.          0.        ]
 [ 2.09861229  0.          0.          0.          0.        ]
 [ 2.09861229  0.          0.          0.          0.        ]
 [ 0.          1.69314718  1.69314718  0.          0.        ]
 [ 0.          1.69314718  5.07944154  0.          0.        ]
 [ 0.          1.69314718  1.69314718  0.          0.        ]
 [ 0.         11.85203026  8.4657359   0.          0.        ]
 [ 0.          1.69314718  1.69314718  0.          0.        ]
 [ 0.          1.69314718  1.69314718  0.          0.        ]
 [ 0.          1.69314718  1.69314718  0.          0.        ]
 [ 0.          2.09861229  0.          0.          0.        ]
 [ 0.          1.40546511  1.40546511  0.          1.40546511]
 [ 0.          2.81093022  2.81093022  1.40546511  0.        ]
 [ 0.          4.19722458  0.          0.          0.        ]
```

```
[ 0.          6.29583687  0.          0.          0.        ]
[ 0.          1.69314718  1.69314718  0.          0.        ]
[ 0.          2.09861229  0.          0.          0.        ]
[ 0.          4.19722458  0.          0.          0.        ]
[ 0.          2.09861229  0.          0.          0.        ]
[ 0.          1.69314718  1.69314718  0.          0.        ]
[ 0.          3.38629436  5.07944154  0.          0.        ]
[ 0.          2.09861229  0.          0.          0.        ]
[ 0.          1.69314718  1.69314718  0.          0.        ]
[ 0.          2.09861229  0.          0.          0.        ]
[ 0.          1.69314718  5.07944154  0.          0.        ]
[ 0.          0.          2.09861229  0.          0.        ]
[ 0.          0.          1.69314718  0.          3.38629436]
[ 0.          0.          3.38629436  0.          1.69314718]
[ 0.          0.          2.09861229  0.          0.        ]
[ 0.          0.          2.09861229  0.          0.        ]
[ 0.          0.          6.29583687  0.          0.        ]
[ 6.          8.          7.          6.          8.        ]
[ 0.          0.          7.02732554  9.83825576  2.81093022]
[ 0.          0.          2.09861229  0.          0.        ]
[ 0.          0.          2.09861229  0.          0.        ]
[ 0.          0.          2.09861229  0.          0.        ]
[ 0.          0.          4.19722458  0.          0.        ]
[ 0.          0.          4.19722458  0.          0.        ]
[ 0.          0.          2.09861229  0.          0.        ]
[ 0.          0.          6.29583687  0.          0.        ]
[ 0.          0.          2.09861229  0.          0.        ]
[ 0.          0.          4.19722458  0.          0.        ]

[ 0.          0.          4.19722458  0.          0.        ]
[ 0.          0.          5.62186043  7.02732554  5.62186043]
[ 0.          0.          2.09861229  0.          0.        ]
[ 0.          0.          4.19722458  0.          0.        ]
[ 0.          0.          2.09861229  0.          0.        ]
[ 0.          0.          1.69314718  1.69314718  0.        ]
[ 0.          0.          2.09861229  0.          0.        ]
[ 0.          0.          2.09861229  0.          0.        ]
[ 0.          0.          0.         10.49306144  0.        ]
[ 0.          0.          0.          2.09861229  0.        ]
[ 0.          0.          0.          2.09861229  0.        ]
[ 0.          0.          0.          2.09861229  0.        ]
[ 0.          1.40546511  1.40546511  1.40546511  0.        ]
[ 0.          0.          0.          2.09861229  0.        ]
[ 0.          0.          0.          2.09861229  0.        ]
[ 0.          0.          0.          1.69314718  1.69314718]
[ 0.          0.          0.          2.09861229  0.        ]
[ 0.          0.          0.          2.09861229  0.        ]
[ 0.          0.          0.          2.09861229  0.        ]
[ 0.          0.          0.          2.09861229  0.        ]
[ 1.69314718  0.          0.          1.69314718  0.        ]
[ 0.          0.          0.          1.69314718  1.69314718]
[ 0.          0.          0.          4.19722458  0.        ]
[ 0.          0.          0.          2.09861229  0.        ]
[ 0.          0.          0.          4.19722458  0.        ]
[ 0.          0.          0.          2.09861229  0.        ]
[ 0.          0.          0.          2.09861229  0.        ]
[ 0.          0.          0.          0.          2.09861229]

[ 0.          0.          0.          0.          2.09861229]
[ 0.          0.          0.          0.          2.09861229]
[ 0.          0.          0.          0.          2.09861229]
[ 0.          0.          0.          0.          2.09861229]
[ 0.          0.          0.          0.          2.09861229]
[ 0.          0.          0.          0.          2.09861229]
[ 0.          0.          0.          0.          2.09861229]
[ 0.          0.          0.          0.          2.09861229]
[ 0.          0.          0.          0.          2.09861229]
[ 0.          0.          0.          0.          2.09861229]
[ 0.          0.          0.          0.          2.09861229]
[ 0.          0.          0.          0.          6.29583687]
[ 0.          0.          0.          0.          2.09861229]
[ 0.          0.          0.          0.          2.09861229]
[ 0.          0.          0.          0.          4.19722458]
[ 5.91160778  7.09392934  5.91160778  0.         10.64089401]
[ 0.          0.          0.          0.          2.09861229]
[ 0.          0.          0.          0.          2.09861229]
[ 0.          0.          0.          0.          2.09861229]
[ 0.          0.          0.          0.          2.09861229]
[ 0.          0.          0.          0.          2.09861229]
[ 0.          0.          0.          0.          2.09861229]
[ 0.          0.          0.          0.          4.19722458]
[ 0.          0.          0.          0.          2.09861229]
[ 0.          0.          0.          0.          6.29583687]
[ 0.          0.          0.          0.          2.09861229]
[ 0.          0.          0.          0.          2.09861229]
[ 0.          0.          0.          0.          2.09861229]
```

2. **Dari 5 file pada folder "berita", hitung skor kemiripan antara berita yang satu dan lainnya masing-masing dengan edit distance, jaccard similarity, euclidian distance, dan cosine similarity.**

Melanjutkan program pada nomor 1, dilakukan penghitungan skor kemiripan antara berita yang satu dan lainnya masing-masing dengan edit distance, jaccard similarity, euclidian distance, dan cosine similarity.

Proses edit distance dilakukan oleh fungsi "edit_distance(string1, string2)", yaitu untuk menghitung jarak edit (edit distance) antara dua string string1 dan string2. Jarak edit adalah jumlah minimum dari operasi (penghapusan, penggantian, atau penyisipan) yang diperlukan untuk mengubah satu string menjadi string lainnya. Fungsi tersebut akan mengembalikan nilai berupa angka jarak kedua string tersebut.

```
def edit_distance(string1, string2):
    if len(string1) > len(string2):
        difference = len(string1) - len(string2)
        string1[:difference]
        n = len(string2)
    elif len(string2) > len(string1):
        difference = len(string2) - len(string1)
        string2[:difference]
        n = len(string1)
    for i in range(n):
        if string1[i] != string2[i]:
            difference += 1

    return difference
```

Selanjutnya, terdapat fungsi "jaccard_sim(list1, list2)" untuk menghitung indeks kesamaan Jaccard antara dua list list1 dan list2. Indeks ini mengukur kesamaan antara dua himpunan (set) dengan menghitung rasio jumlah elemen yang sama dengan jumlah elemen yang berbeda antara kedua set. Fungsi ini mengembalikan nilai kesamaan dalam bentuk pecahan (float).

```
def jaccard_sim(list1, list2):
    intersection = len(list(set(list1).intersection(list2)))
    union = (len(list1) + len(list2)) - intersection
    return float(intersection) / union
```

Selain itu, terdapat juga fungsi "euclidian_dist(vec1, vec2)" yang berfungsi untuk menghitung jarak Euclidean antara dua vektor vec1 dan vec2. Jarak Euclidean adalah

panjang garis lurus antara dua titik dalam ruang berdimensi n. Fungsi ini mengembalikan nilai jarak antara dua vektor.

```
def euclidian_dist(vec1, vec2):
    temp = vec1 - vec2
    sum_sq = np.dot(temp.T, temp)
    return np.sqrt(sum_sq)
```

Dan yang terakhir terdapat fungsi “cosine_sim(vec1, vec2)” menggunakkan library “Math” untuk melakukan operasi matematika. Fungsi ini digunakan untuk menghitung kesamaan kosinus antara dua vektor vec1 dan vec2. Kesamaan kosinus mengukur sejauh mana dua vektor memiliki arah yang serupa dalam ruang berdimensi n. Fungsi ini mengembalikan nilai kesamaan kosinus dalam bentuk pecahan

```
import math
def cosine_sim(vec1, vec2):
    vec1 = list(vec1)
    vec2 = list(vec2)
    dot_prod = 0
    for i, v in enumerate(vec1):
        dot_prod += v * vec2[i]
    mag_1 = math.sqrt(sum([x**2 for x in vec1]))
    mag_2 = math.sqrt(sum([x**2 for x in vec2]))

    return dot_prod / (mag_1 * mag_2)
```

Berikut disajikan tampilan keseluruhan kode program beserta output yang dihasilkan.

```
import os
from spacy.lang.id import Indonesian
from Sastrawi.Stemmer.StemmerFactory import StemmerFactory
from spacy.lang.id.stop_words import STOP_WORDS

nlp = Indonesian()

path = "D:/RAIHAN STIS/Perkuliahan/SEMESTER 5/Praktikum INFORMATION
RETRIEVAL/Pertemuan (2)/berita"

def read_text_file(file_path):
    with open(file_path, 'r') as f:
        content = f.read()
    return content

def preprocess_text(text):
    stemmer = StemmerFactory().create_stemmer()
    stemmed_text = stemmer.stem(text)
```

```
    doc = nlp(stemmed_text)
    tokens = [token.text for token in doc if token.text.lower() not in
STOP_WORDS]

    return tokens

inverted_index = {}
doc_dict = {}
document_index = 1

for file in os.listdir(path):
    if os.path.isfile(os.path.join(path, file)) and file.endswith(".txt"):
        file_path = os.path.join(path, file)

        text = read_text_file(file_path)

        cleaned_tokens = preprocess_text(text)
        doc_dict[document_index] = " ".join(cleaned_tokens)
        document_index += 1

        # Update the inverted index
        for token in set(cleaned_tokens):  # Use set to avoid duplicate
documents
            inverted_index.setdefault(token, []).append(file)

vocab=list(inverted_index.keys())

def termFrequencyInDoc(vocab, doc_dict):
    tf_docs = {}
    for doc_id in doc_dict.keys():
        tf_docs[doc_id] = {}
    for word in vocab:
        for doc_id,doc in doc_dict.items():
            tf_docs[doc_id][word] = doc.count(word)
    return tf_docs

def wordDocFre(vocab, doc_dict):
    df = {}
    for word in vocab:
        frq = 0
        for doc in doc_dict.values():
            if word in doc:
                frq = frq + 1
        df[word] = frq
    return df

import numpy as np
def inverseDocFre(vocab,doc_fre,length):
```

```
    idf= {}
    for word in vocab:
        idf[word] = idf[word] = 1 + np.log((length + 1) /
(doc_fre[word]+1))
    return idf

def tfidf(vocab,tf,idf_scr,doc_dict):
    tf_idf_scr = {}
    for doc_id in doc_dict.keys():
        tf_idf_scr[doc_id] = {}
    for word in vocab:
        for doc_id,doc in doc_dict.items():
            tf_idf_scr[doc_id][word] = tf[doc_id][word] * idf_scr[word]
    return tf_idf_scr

tf_idf = tfidf(vocab, termFrequencyInDoc(vocab, doc_dict),
inverseDocFre(vocab, wordDocFre(vocab, doc_dict), len(doc_dict)), doc_dict)
# Term - Document Matrix
TD = np.zeros((len(vocab), len(doc_dict)))
for word in vocab:
    for doc_id,doc in tf_idf.items():
        ind1 = vocab.index(word)
        ind2 = list(tf_idf.keys()).index(doc_id)
        TD[ind1][ind2] = tf_idf[doc_id][word]

def edit_distance(string1, string2):
    if len(string1) > len(string2):
        difference = len(string1) - len(string2)
        string1[:difference]
        n = len(string2)
    elif len(string2) > len(string1):
        difference = len(string2) - len(string1)
        string2[:difference]
        n = len(string1)
    for i in range(n):
        if string1[i] != string2[i]:
            difference += 1

    return difference

print("\nSkor Kemiripan dengan Edit Distance")
print("Berita 1 dengan berita 2 : ", edit_distance(doc_dict[1],
doc_dict[2]))
print("Berita 1 dengan berita 3 : ", edit_distance(doc_dict[1],
doc_dict[3]))
print("Berita 1 dengan berita 4 : ", edit_distance(doc_dict[1],
doc_dict[4]))
```

```
print("Berita 1 dengan berita 5 : ", edit_distance(doc_dict[1],
doc_dict[5]))
print("Berita 2 dengan berita 3 : ", edit_distance(doc_dict[2],
doc_dict[3]))
print("Berita 2 dengan berita 4 : ", edit_distance(doc_dict[2],
doc_dict[4]))
print("Berita 2 dengan berita 5 : ", edit_distance(doc_dict[2],
doc_dict[5]))
print("Berita 3 dengan berita 4 : ", edit_distance(doc_dict[3],
doc_dict[4]))
print("Berita 3 dengan berita 5 : ", edit_distance(doc_dict[3],
doc_dict[5]))
print("Berita 4 dengan berita 5 : ", edit_distance(doc_dict[4],
doc_dict[5]))

def jaccard_sim(list1, list2):
    intersection = len(list(set(list1).intersection(list2)))
    union = (len(list1) + len(list2)) - intersection
    return float(intersection) / union

print("\nSkor Kemiripan dengan Jaccard Similarity")
print("Berita 1 dengan berita 2 : ", jaccard_sim(doc_dict[1].split(" "),
doc_dict[2].split(" ")))
print("Berita 1 dengan berita 3 : ", jaccard_sim(doc_dict[1].split(" "),
doc_dict[3].split(" ")))
print("Berita 1 dengan berita 4 : ", jaccard_sim(doc_dict[1].split(" "),
doc_dict[4].split(" ")))
print("Berita 1 dengan berita 5 : ", jaccard_sim(doc_dict[1].split(" "),
doc_dict[5].split(" ")))
print("Berita 2 dengan berita 3 : ", jaccard_sim(doc_dict[2].split(" "),
doc_dict[3].split(" ")))
print("Berita 2 dengan berita 4 : ", jaccard_sim(doc_dict[2].split(" "),
doc_dict[4].split(" ")))
print("Berita 2 dengan berita 5 : ", jaccard_sim(doc_dict[2].split(" "),
doc_dict[5].split(" ")))
print("Berita 3 dengan berita 4 : ", jaccard_sim(doc_dict[3].split(" "),
doc_dict[4].split(" ")))
print("Berita 3 dengan berita 5 : ", jaccard_sim(doc_dict[3].split(" "),
doc_dict[5].split(" ")))
print("Berita 4 dengan berita 5 : ", jaccard_sim(doc_dict[3].split(" "),
doc_dict[5].split(" ")))

def euclidian_dist(vec1, vec2):
    temp = vec1 - vec2
    sum_sq = np.dot(temp.T, temp)
    return np.sqrt(sum_sq)

print("\nSkor Kemiripan dengan Euclidian Distance")
```

```
print("Berita 1 dengan berita 2 : ", euclidian_dist(TD[:, 0], TD[:, 1]))
print("Berita 1 dengan berita 3 : ", euclidian_dist(TD[:, 0], TD[:, 2]))
print("Berita 1 dengan berita 4 : ", euclidian_dist(TD[:, 0], TD[:, 3]))
print("Berita 1 dengan berita 5 : ", euclidian_dist(TD[:, 0], TD[:, 4]))
print("Berita 2 dengan berita 3 : ", euclidian_dist(TD[:, 1], TD[:, 2]))
print("Berita 2 dengan berita 4 : ", euclidian_dist(TD[:, 1], TD[:, 3]))
print("Berita 2 dengan berita 5 : ", euclidian_dist(TD[:, 1], TD[:, 4]))
print("Berita 3 dengan berita 4 : ", euclidian_dist(TD[:, 2], TD[:, 3]))
print("Berita 3 dengan berita 5 : ", euclidian_dist(TD[:, 2], TD[:, 4]))
print("Berita 4 dengan berita 5 : ", euclidian_dist(TD[:, 3], TD[:, 4]))

import math
def cosine_sim(vec1, vec2):
    vec1 = list(vec1)
    vec2 = list(vec2)
    dot_prod = 0
    for i, v in enumerate(vec1):
        dot_prod += v * vec2[i]
    mag_1 = math.sqrt(sum([x**2 for x in vec1]))
    mag_2 = math.sqrt(sum([x**2 for x in vec2]))

    return dot_prod / (mag_1 * mag_2)

print("\nSkor Kemiripan dengan Cosine Similarity")
print("Berita 1 dengan berita 2 : ", cosine_sim(TD[:, 0], TD[:, 1]))
print("Berita 1 dengan berita 3 : ", cosine_sim(TD[:, 0], TD[:, 2]))
print("Berita 1 dengan berita 4 : ", cosine_sim(TD[:, 0], TD[:, 3]))
print("Berita 1 dengan berita 5 : ", cosine_sim(TD[:, 0], TD[:, 4]))
print("Berita 2 dengan berita 3 : ", cosine_sim(TD[:, 1], TD[:, 2]))
print("Berita 2 dengan berita 4 : ", cosine_sim(TD[:, 1], TD[:, 3]))
print("Berita 2 dengan berita 5 : ", cosine_sim(TD[:, 1], TD[:, 4]))
print("Berita 3 dengan berita 4 : ", cosine_sim(TD[:, 2], TD[:, 3]))
print("Berita 3 dengan berita 5 : ", cosine_sim(TD[:, 2], TD[:, 4]))
print("Berita 4 dengan berita 5 : ", cosine_sim(TD[:, 3], TD[:, 4]))
```

```
Skor Kemiripan dengan Edit Distance
Berita 1 dengan berita 2 :  494
Berita 1 dengan berita 3 :  545
Berita 1 dengan berita 4 :  486
Berita 1 dengan berita 5 :  484
Berita 2 dengan berita 3 :  546
Berita 2 dengan berita 4 :  377
Berita 2 dengan berita 5 :  422
Berita 3 dengan berita 4 :  546
Berita 3 dengan berita 5 :  541
Berita 4 dengan berita 5 :  414

Skor Kemiripan dengan Jaccard Similarity
Berita 1 dengan berita 2 :  0.09615384615384616
Berita 1 dengan berita 3 :  0.07446808510638298
Berita 1 dengan berita 4 :  0.09740259740259741
Berita 1 dengan berita 5 :  0.08383233532934131
Berita 2 dengan berita 3 :  0.16993464052287582
Berita 2 dengan berita 4 :  0.08955223880597014
Berita 2 dengan berita 5 :  0.07482993197278912
Berita 3 dengan berita 4 :  0.10625
Berita 3 dengan berita 5 :  0.08620689655172414
Berita 4 dengan berita 5 :  0.08620689655172414

Skor Kemiripan dengan Euclidian Distance
Berita 1 dengan berita 2 :  28.22321658060235
Berita 1 dengan berita 3 :  32.693426518708094
Berita 1 dengan berita 4 :  29.854191685888534
Berita 1 dengan berita 5 :  29.57181348630005
Berita 2 dengan berita 3 :  24.86369597871168
Berita 2 dengan berita 4 :  29.34226261220636
Berita 2 dengan berita 5 :  26.62608375545288
Berita 3 dengan berita 4 :  29.067739735236135
Berita 3 dengan berita 5 :  29.021370077125063
Berita 4 dengan berita 5 :  27.181117499495016

Skor Kemiripan dengan Cosine Similarity
Berita 1 dengan berita 2 :  0.6268310290175129
Berita 1 dengan berita 3 :  0.6266888070702852
Berita 1 dengan berita 4 :  0.5475970699785773
Berita 1 dengan berita 5 :  0.6179189497669911
Berita 2 dengan berita 3 :  0.7987675586592119
Berita 2 dengan berita 4 :  0.5922321985735381
Berita 2 dengan berita 5 :  0.7052223626726393
Berita 3 dengan berita 4 :  0.7119077195124249
Berita 3 dengan berita 5 :  0.7246784190917823
Berita 4 dengan berita 5 :  0.6757236251866554
```