222112058_Feza Raffa Arnanda_Penugasan Praktikum 4

September 18, 2023

1 Praktikum 4

Referensi : https://github.com/peermohtaram/Vector-Space-Model/blob/master/Vector_Space_Model.ipynb

1.0.1 Raw Term Frequency

[ ]:
```
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
```

[ ]:
```
doc1_term = ["pengembangan", "sistem", "informasi", "penjadwalan"]
doc2_term = ["pengembangan", "model", "analisis", "sentimen", "berita"]
doc3_term = ["analisis", "sistem", "input", "output"]
corpus_term = [doc1_term, doc2_term, doc3_term]

from Sastrawi.Stemmer.StemmerFactory import StemmerFactory

stemmer_factory = StemmerFactory()
stemmer = stemmer_factory.create_stemmer()

inverted_index = {}
for i in range(len(corpus_term)):
    for item in corpus_term[i]:
        item = stemmer.stem(item)
        if item not in inverted_index:
            inverted_index[item] = []
        if (item in inverted_index) and ((i+1) not in inverted_index[item]):
            inverted_index[item].append(i+1)

print("--- Inverted Index ---")
print(inverted_index)
print("\n")
```

```
vocab = list(inverted_index.keys())
doc_dict = {}
#clean after stemming
doc_dict['doc1'] = "kembang sistem informasi jadwal"
doc_dict['doc2'] = "kembang model analisis sentimen berita"
doc_dict['doc3'] = "analisis sistem input output"

print("--- Term Frequency -- ")
print(termFrequencyInDoc(vocab, doc_dict))
```

```
--- Inverted Index ---
{'kembang': [1, 2], 'sistem': [1, 3], 'informasi': [1], 'jadwal': [1], 'model':
[2], 'analisis': [2, 3], 'sentimen': [2], 'berita': [2], 'input': [3], 'output':
[3]}


--- Term Frequency --
{'doc1': {'kembang': 1, 'sistem': 1, 'informasi': 1, 'jadwal': 1, 'model': 0,
'analisis': 0, 'sentimen': 0, 'berita': 0, 'input': 0, 'output': 0}, 'doc2':
{'kembang': 1, 'sistem': 0, 'informasi': 0, 'jadwal': 0, 'model': 1, 'analisis':
1, 'sentimen': 1, 'berita': 1, 'input': 0, 'output': 0}, 'doc3': {'kembang': 0,
'sistem': 1, 'informasi': 0, 'jadwal': 0, 'model': 0, 'analisis': 1, 'sentimen':
0, 'berita': 0, 'input': 1, 'output': 1}}
```

Penjelasan kode di atas :

Inverted index untuk list apa saja kata yang mau dicari di suatu document

Nah, term frequencynya itu biar kita cari setiap kata yang ada di inverted index di suatu document tuh muncul berapa kalii. Gitu bro.

1.0.2 Document Frequency

[ ]:
```
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize

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

1.0.3 Inverse Document Frequency

Formula : IDF = log (N/Df)

[ ]:
```python
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
```

[ ]:
```python
print(inverseDocFre(vocab, wordDocFre(vocab, doc_dict),len(doc_dict)))
```

```
{'kembang': 1.2876820724517808, 'sistem': 1.2876820724517808, 'informasi':
1.6931471805599454, 'jadwal': 1.6931471805599454, 'model': 1.6931471805599454,
'analisis': 1.2876820724517808, 'sentimen': 1.6931471805599454, 'berita':
1.6931471805599454, 'input': 1.6931471805599454, 'output': 1.6931471805599454}
```

1.0.4 Vector Space Model

Function dibawah ini menghasilkan w = TF*IDF

[ ]:
```python
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
```

Term-Document Matrix doc1 doc2 doc3 t1 |w11 w12 w13| t2 |w21 w22 w23| t3 |w31 w32 w33|

[ ]:
```python
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)
```

```
[[1.28768207 1.28768207 0.        ]
 [1.28768207 0.         1.28768207]
 [1.69314718 0.         0.        ]
 [1.69314718 0.         0.        ]
 [0.         1.69314718 0.        ]
 [0.         1.28768207 1.28768207]
 [0.         1.69314718 0.        ]
 [0.         1.69314718 0.        ]
 [0.         0.         1.69314718]]
```

```
[0.         0.         1.69314718]]
```

1.0.5 Text Similarity

Edit Distace Ref : https://www.w3resource.com/python-exercises/challenges/1/python-challenges-1-exercise-52.php

```
[ ]: 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(edit_distance(doc_dict['doc1'], doc_dict['doc2']))
     print(edit_distance(doc_dict['doc1'], doc_dict['doc3']))
```

```
30
31
```

Jaccard Similarity Ref : https://www.w3resource.com/python-exercises/extended-data-types/python-extended-data-types-index-counter-exercise-9.php

```
[ ]: def jaccard_sim(list1, list2):
         intersection = len(list(set(list1).intersection(list2)))
         union = (len(list1) + len(list2)) - intersection

         return float(intersection) / union
```

```
[ ]: print(jaccard_sim(doc_dict['doc1'].split(" "), doc_dict['doc2'].split(" ")))
     print(jaccard_sim(doc_dict['doc1'].split(" "), doc_dict['doc3'].split(" ")))
```

```
0.125
0.14285714285714285
```

1.0.6 Euclidian Distance

```
[ ]: def euclidian_dist(vec1, vec2):
         # subtracting vector
         temp = vec1 - vec2
         # doing dot product
```

```python
    # for finding
    # sum of the squares
    sum_sq = np.dot(temp.T, temp)
    # Doing squareroot and
    # printing Euclidean distance

    return np.sqrt(sum_sq)
```

[ ]:
```python
print(euclidian_dist(TD[:, 0], TD[:, 1])) #doc1 & doc2
print(euclidian_dist(TD[:, 0], TD[:, 2])) #doc1 & doc3
```

```
4.201188773980275
3.844897884155026
```

1.0.7 Cosine Similarity

Ref : https://algoritmaonline.com/kemiripan-teks/

[ ]:
```python
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)
```

[ ]:
```python
print(cosine_sim(TD[:, 0], TD[:, 1])) #doc1 & doc2
print(cosine_sim(TD[:, 0], TD[:, 2])) #doc1 & doc3
```

```
0.15967058203849993
0.1832234081332565
```

1.0.8 Penugasan

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

[ ]:
```python
import os
from spacy.lang.id import Indonesian
from spacy.lang.id.stop_words import STOP_WORDS
from Sastrawi.Stemmer.StemmerFactory import StemmerFactory

nlp = Indonesian()

path = "C:/Users/FEZA/My Drive/00. Drive PC/1.STIS/5. Semester 5/Information␣
↪Retrieval [IR] P/Pertemuan 3/berita"
```

```python
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 = {}

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[file] = cleaned_tokens

        # Ini utk inverted index nya
        for token in set(cleaned_tokens):  # Menghindari duplikat
            inverted_index.setdefault(token, []).append(file)

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

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

[ ]:
```python
print(TD)
```

```
[[ 2.09861229  0.          0.          0.          0.        ]
 [ 4.19722458  0.          0.          0.          0.        ]
 [ 2.09861229  0.          0.          0.          0.        ]
 [ 4.19722458  0.          0.          0.          0.        ]
 [ 1.40546511  1.40546511  1.40546511  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.        ]
 [ 1.69314718  1.69314718  0.          0.          0.        ]
 [ 2.09861229  0.          0.          0.          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.        ]
 [ 4.19722458  0.          0.          0.          0.        ]
 [ 2.09861229  0.          0.          0.          0.        ]
 [ 6.29583687  0.          0.          0.          0.        ]
 [ 2.09861229  0.          0.          0.          0.        ]
 [ 1.69314718  0.          0.          1.69314718  0.        ]
 [ 1.          1.          1.          1.          1.        ]
 [ 2.09861229  0.          0.          0.          0.        ]
 [ 2.81093022  0.          2.81093022  1.40546511  0.        ]
 [ 3.          7.          4.          2.          3.        ]
 [ 2.09861229  0.          0.          0.          0.        ]
 [ 2.09861229  0.          0.          0.          0.        ]
 [ 6.29583687  0.          0.          0.          0.        ]
 [ 6.29583687  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.        ]
 [ 1.69314718  0.          0.          3.38629436  0.        ]
 [ 1.69314718  0.          3.38629436  0.          0.        ]
 [ 1.          1.          1.          1.          1.        ]
 [ 2.09861229  0.          0.          0.          0.        ]
 [ 2.09861229  0.          0.          0.          0.        ]
 [ 2.09861229  0.          0.          0.          0.        ]
 [ 1.          1.          1.          1.          1.        ]
 [ 1.          1.          1.          4.          1.        ]
 [ 2.09861229  0.          0.          0.          0.        ]
 [ 2.09861229  0.          0.          0.          0.        ]
 [ 2.09861229  0.          0.          0.          0.        ]
 [ 2.09861229  0.          0.          0.          0.        ]
 [ 3.38629436  0.          0.          0.          1.69314718]
 [ 2.09861229  0.          0.          0.          0.        ]
```

```
[ 1.          1.          1.          1.          1.        ]
[ 3.38629436  0.          0.          0.          1.69314718]
[ 1.69314718  0.          0.          1.69314718  0.        ]
[ 4.19722458  0.          0.          0.          0.        ]
[ 1.          1.          1.          1.          1.        ]
[ 4.19722458  0.          0.          0.          0.        ]
[16.         18.         19.         12.         16.        ]
[ 2.09861229  0.          0.          0.          0.        ]
[ 2.09861229  0.          0.          0.          0.        ]
[ 4.19722458  0.          0.          0.          0.        ]
[ 2.09861229  0.          0.          0.          0.        ]
[ 1.69314718  0.          0.          0.          1.69314718]
[ 3.          7.          4.          2.          3.        ]
[ 4.19722458  0.          0.          0.          0.        ]
[ 1.69314718  1.69314718  0.          0.          0.        ]
[ 2.09861229  0.          0.          0.          0.        ]
[ 0.          1.69314718  5.07944154  0.          0.        ]
[ 0.          1.69314718  1.69314718  0.          0.        ]
[ 0.          6.29583687  0.          0.          0.        ]
[ 0.          2.09861229  0.          0.          0.        ]
[ 0.          1.69314718  1.69314718  0.          0.        ]
[ 0.          1.69314718  1.69314718  0.          0.        ]
[ 0.          4.19722458  0.          0.          0.        ]
[ 0.          1.69314718  1.69314718  0.          0.        ]
[ 0.          1.69314718  1.69314718  0.          0.        ]
[ 0.          1.69314718  5.07944154  0.          0.        ]
[ 0.          2.09861229  0.          0.          0.        ]
[ 0.          2.09861229  0.          0.          0.        ]
[ 0.          1.69314718  1.69314718  0.          0.        ]
[ 0.          3.38629436  5.07944154  0.          0.        ]
[ 0.          2.81093022  2.81093022  1.40546511  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.          6.77258872  8.4657359   0.          0.        ]
[ 0.          1.69314718  1.69314718  0.          0.        ]
[ 0.          1.69314718  1.69314718  0.          0.        ]
[ 0.          0.          2.09861229  0.          0.        ]
[ 0.          0.          3.38629436  0.          1.69314718]
[ 0.          0.          1.69314718  0.          3.38629436]
[ 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.          5.62186043  7.02732554  5.62186043]
[ 0.          0.          7.02732554  9.83825576  2.81093022]
[ 0.          0.          4.19722458  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.          4.19722458  0.          0.        ]
[ 0.          0.          3.38629436  1.69314718  0.        ]
[ 0.          0.          6.29583687  0.          0.        ]
[ 0.          0.          6.29583687  0.          0.        ]
[ 0.          0.          2.09861229  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.          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.          1.69314718  1.69314718]
[ 0.          0.          0.          2.09861229  0.        ]
[ 0.          0.          0.          2.09861229  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.          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.          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.          4.19722458]
[ 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.          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.          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.          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]]
```

[ ]:

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

[ ]:
```
import os
import numpy as np
from spacy.lang.id import Indonesian
from Sastrawi.Stemmer.StemmerFactory import StemmerFactory
from spacy.lang.id.stop_words import STOP_WORDS
from scipy.spatial import distance
import math
```

Pada kode dibawah kita melakukan beberapa revisi pada function edit_distance, dimana nantinya akan terdapat seluruh operasi yang ada di edit distance seperti insert, substitusi, dan delete. Penggunaan library scipy digunakan untuk mempermudah penggunaan operasi perbedaan antar dokumen pada beberap fungsi dibawah sepertin distance.

[ ]:
```
def read_text_file(file_path):
    with open(file_path, 'r', encoding='utf-8') as f:
        content = f.read()
    return content

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

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

    return tokens

def edit_distance(string1, string2):
    # Implementasi edit distance
    if len(string1) > len(string2):
        string1, string2 = string1[:len(string2)], string2
    elif len(string2) > len(string1):
        string1, string2 = string1, string2[:len(string1)]

    distance_matrix = [[0] * (len(string2) + 1) for _ in range(len(string1) +␣
 ↪1)]

    for i in range(len(string1) + 1):
        distance_matrix[i][0] = i

    for j in range(len(string2) + 1):
        distance_matrix[0][j] = j

    for i in range(1, len(string1) + 1):
        for j in range(1, len(string2) + 1):
            cost = 0 if string1[i - 1] == string2[j - 1] else 1
            distance_matrix[i][j] = min(
                distance_matrix[i - 1][j] + 1,        # Delete
                distance_matrix[i][j - 1] + 1,        # Insert
                distance_matrix[i - 1][j - 1] + cost  # Substitusi
            )

    return distance_matrix[len(string1)][len(string2)]

def jaccard_sim(list1, list2):
    intersection = len(set(list1).intersection(list2))
    union = len(set(list1).union(list2))

    return intersection / union

def euclidean_dist(vec1, vec2):
    return distance.euclidean(vec1, vec2)

def cosine_sim(vec1, vec2):
    return 1 - distance.cosine(vec1, vec2)
```

Kemudian, kita lakukan double iterasi untuk membandingkan setiap dokumen satu dengan keseluruhan dokumen yang ada di folder berita. Total ada 10 kombinasi yang akan didapatkan dari iterasi berikut dan menghitung skor kesamaan dan perbedaan setiap dokumen.

[ ]:
```python
#perhitungan untuk setiap dokumen di berita
for i, (file1, tokens1) in enumerate(doc_dict.items()):
    for file2, tokens2 in list(doc_dict.items())[i+1:]:
        edit_dist = edit_distance(' '.join(tokens1), ' '.join(tokens2))
        jaccard_similarity = jaccard_sim(tokens1, tokens2)

        # Ubah tokens menjadi vektor biner
        vec1 = [1 if token in tokens1 else 0 for token in vocab]
        vec2 = [1 if token in tokens2 else 0 for token in vocab]

        euclidean_distance = euclidean_dist(vec1, vec2)
        cosine_similarity = cosine_sim(vec1, vec2)

        print(f"Kesamaan antara {file1} and {file2}:")
        print(f"Edit Distance: {edit_dist}")
        print(f"Jaccard Similarity: {jaccard_similarity}")
        print(f"Euclidean Distance: {euclidean_distance}")
        print(f"Cosine Similarity: {cosine_similarity}")
        print()
```

```
Kesamaan antara berita1.txt and berita2.txt:
Edit Distance: 331
Jaccard Similarity: 0.17857142857142858
Euclidean Distance: 8.306623862918075
Cosine Similarity: 0.31318038399724624

Kesamaan antara berita1.txt and berita3.txt:
Edit Distance: 396
Jaccard Similarity: 0.14
Euclidean Distance: 9.273618495495704
Cosine Similarity: 0.24656448378672147

Kesamaan antara berita1.txt and berita4.txt:
Edit Distance: 313
Jaccard Similarity: 0.1744186046511628
Euclidean Distance: 8.426149773176359
Cosine Similarity: 0.3050444380432815

Kesamaan antara berita1.txt and berita5.txt:
Edit Distance: 338
Jaccard Similarity: 0.13725490196078433
Euclidean Distance: 9.38083151964686
Cosine Similarity: 0.2419553954370992

Kesamaan antara berita2.txt and berita3.txt:
Edit Distance: 264
```

```
Jaccard Similarity: 0.4126984126984127
Euclidean Distance: 6.082762530298219
Cosine Similarity: 0.5927489783638191

Kesamaan antara berita2.txt and berita4.txt:
Edit Distance: 260
Jaccard Similarity: 0.1875
Euclidean Distance: 7.211102550927978
Cosine Similarity: 0.31589887589559384

Kesamaan antara berita2.txt and berita5.txt:
Edit Distance: 285
Jaccard Similarity: 0.1375
Euclidean Distance: 8.306623862918075
Cosine Similarity: 0.24609055357847565

Kesamaan antara berita3.txt and berita4.txt:
Edit Distance: 312
Jaccard Similarity: 0.22972972972972974
Euclidean Distance: 7.54983443527075
Cosine Similarity: 0.3774982529316784

Kesamaan antara berita3.txt and berita5.txt:
Edit Distance: 343
Jaccard Similarity: 0.16483516483516483
Euclidean Distance: 8.717797887081348
Cosine Similarity: 0.28306925853614895

Kesamaan antara berita4.txt and berita5.txt:
Edit Distance: 284
Jaccard Similarity: 0.17721518987341772
Euclidean Distance: 8.06225774829855
Cosine Similarity: 0.3050695435482862
```