LAPORAN TUGAS PENGENALAN POLA
DOCUMENT SIMILARITY

Nama:
Almas Fauzia Wibawa (17/409427/PA/17734)
Muhammad Hanif Muallif (17/412571/PA/17890)
Sulkha Marfuah (17/409447/PA/17754)

Impor Library

Library adalah sumber dari berbagai fungsionalitas. Dalam library kita dapat melihat rak-rak yang tertata rapi dengan modul sesuai library yang kita gunakan yang dapat kita ambil dan gunakan dalam penulisan kode. Dengan menggunakan library ,kita dapat menghasilkan kode secara efisien dan menghemat waktu tanpa harus menulis seluruh skrip. Library yang kita gunakan adalah re, nltk,string,sastrawi, dan pandas.

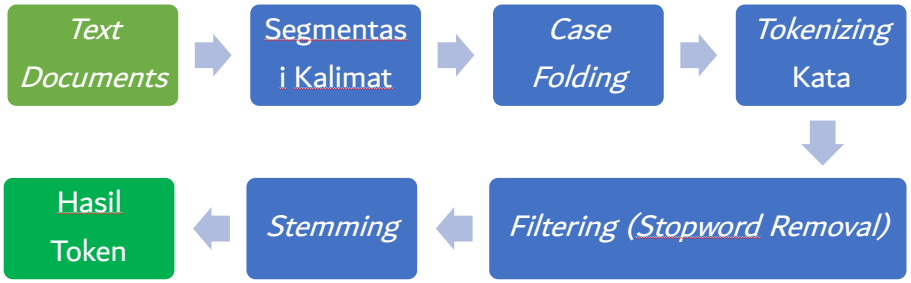
PEMBAHASAN

1. **Import library diperlukan**

```
import re
import nltk
import string
from Sastrawi.Stemmer.StemmerFactory import StemmerFactory
import pandas
pandas.set_option('display.max_rows', None)
pandas.set_option('display.max_columns', None)
```

2. **Text Preprocessing**

Langkah-langkah text processing meliputi :



PREPOCESSING

Langkah-lagkah preprocessing meliputi

1. Mendefinisikan fungsi variabel preprocessing, langkah pertama adalah membuka file abstrak dengan format .txt
2. Melakukan tokenizing yaitu memisahkan kalimat kalimat dalam abstrak
3. Mengubah abstrak menjadi lower case
4. Menghapus angka dalam abstrak
5. Menghapus tanda baca dalam abstrak
6. Menghapus karakter kosong.dalam abstrak
7. Memisahakan antar kata dalam abstrak
8. Melihat daftar stopword menggunakan sastrawi

```
#PREPROCESSING
def preprocessing(k):
    f = open("{}.txt".format(k), 'r', errors = 'ignore')
    abstrak = f.read()
    #TOKENIZING
    abstrak = nltk.tokenize.sent_tokenize(abstrak)
    #CASE FOLDING: LOWERCASE
    arrayAbstrak = []
    for i in range(len(abstrak)):
        arrayAbstrak.append(abstrak[i].lower())
    #CASE FOLDING: REMOVING NUMBER
    for i in range(len(arrayAbstrak)):
        arrayAbstrak[i] = re.sub(r"\d+", "", arrayAbstrak[i])
    #CASE FOLDING: REMOVING PUNCTUATION
    for i in range(len(arrayAbstrak)):
        arrayAbstrak[i] = arrayAbstrak[i].translate(str.maketrans("", "", string.punctuation))
    #CASE FOLDING: REMOVING WHITESPACES
    for i in range(len(arrayAbstrak)):
        arrayAbstrak[i] = arrayAbstrak[i].strip()
    #SEPARATING SENTENCES INTO WORDS
    abstrakWords = []
    for i in range(len(arrayAbstrak)):
        temp = arrayAbstrak[i].split()
        for j in range(len(temp)):
            abstrakWords.append(temp[j])
    #STEMMING USING SASTRAWI
    factory = StemmerFactory()
    stemmer = factory.create_stemmer()
    for i in range(len(abstrakWords)):
        abstrakWords[i] = stemmer.stem(abstrakWords[i])
    return abstrakWords
```

3. **Count Vectorizer**

Count Vectorizer menghitung kata/term yang terdapat dalam suatu dokumen dengan membuat Document-Term Matrix

Rows = documents

Columns = terms

```
#COUNT VECTORIZER
def countVectorizer(words, k, tabel):
    frekuensi = {}
    for kata in words[k]:
        if (kata in frekuensi):
            frekuensi[kata] = frekuensi[kata] + 1
        else:
            frekuensi[kata] = 1

    tabel.append(frekuensi)
```

4. **MENGHITUNG TF-IDF**

- **MENGHITUNG TF**

Term Frequency adalah Perhitungan banyaknya kata dalam suatu dokumen. Rumusnya adalah jumlah suatu kata dalam suatu dokumen dibagi jumlah seluruh kata dalam dokumen tersebut.

```
#MENGHITUNG TF
def computeTF(words, k, tabel):
    frekuensi = {}
    for kata in words[k]:
        if (kata in frekuensi):
            frekuensi[kata] = frekuensi[kata] + 1
        else:
            frekuensi[kata] = 1

    n = len(words[k])
    tf = {}
    for kata in frekuensi:
        tf[kata] = frekuensi[kata]/n
    tabel.append(tf)
```

- **MENGHITUNG IDF**

IDF menunjukkan hubungan ketersediaan sebuah term dalam seluruh dokumen. **Semakin sedikit jumlah dokumen yang mengandung term yang dimaksud, maka nilai IDF semakin besar.**

```
#MENGHITUNG IDF
def computeIDF(words, tabel):
    k = len(words)
    for doc in words:
        for kata in doc:
            if (kata in tabel):
                tabel[kata] = tabel[kata] + 1
            else:
                tabel[kata] = 1
    for kata in tabel:
        tabel[kata] = k/tabel[kata]
```

- **TF-IDF**

$TF - IDF$ merupakan nilai bobot suatu kata yang diperoleh dari perkalian antara bobot TF dan IDF

```
#TF-IDF
def computeTFIDF(words, tabel):
    k = len(words)
    tf = []
    for k in range(k):
        computeTF(words, k, tf)
    idf = {}
    computeIDF(words, idf)

    temp = {}
    for i in range(len(tf)):
        for kata in tf[i]:
            temp[kata] = tf[i][kata] * idf[kata]
        tabel.append(temp)
        temp = {}
```

5. **JACCARD SIMILARITY**

   Jaccard Similarity adalah salah satu metode yang dipakai untuk menghitung similarity antara dua obyek. Di sini kita akan menggunakannya untuk melihat similarity pada 10 dokumen teks abstrak.

```
#JACCARD SIMILARITY
def jaccard_similarity(list1, list2):
    s1 = set(list1)
    s2 = set(list2)
    return (len(s1.intersection(s2)) / len(s1.union(s2)))
```

6. **COSINE SIMILARITY**

   Cosine Similarity juga merupakan metode untuk menghitung **similarity** (tingkat kesamaan) antar dua buah objek. Di sini kita akan menggunakannya untuk melihat similarity pada 10 dokumen teks abstrak. Cossine similarity dan jaccard similarity merupakan 2 metode berbeda yang keduanya sama sama berfungsi untuk menghitung similarity 2 atau lebih objek.

```
#COSINE SIMILARITY
def cosine_similarity(list1, list2):

    l1 =[];l2 =[]

    s1 = set(list1)
    s2 = set(list2)

    rvector = s1.union(s2)
    for w in rvector:
        if w in s1: l1.append(1) # create a vector
        else: l1.append(0)
        if w in s2: l2.append(1)
        else: l2.append(0)
        c = 0

    # cosine formula
    for i in range(len(rvector)):
        c+= l1[i]*l2[i]
    cosine = c / float((sum(l1)*sum(l2))**0.5)
    return cosine
```

7. **Membuka List Kata Tiap Dokumen**

Di bawah ini merupakan source code untuk melihat seluruh kata dalam seluruh dokumen.

```
n = 10
words = [None] * 10
for i in range(n):
    words[i] = preprocessing(i+1)
print("LIST OF WORDS IN EVERY DOCUMENT")
pandas.DataFrame(words)
```

Didapat output sebagai berikut…

LIST OF WORDS IN EVERY DOCUMENT

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	sistem	dukung	putus	kelompok	pilih	tempat	pkl	mahasiswa	dengan	guna	metode	ahp	dan	borda
1	aplikasi	deteksi	dini	defisiensi	mineral	mikro	pada	manusia	bas	web	nina	sevani	rheinhard	unwaru
2	sistem	dukung	putus	alokasi	spare	part	rita	wiryasaputra	sri	hartati	abstrak	era	informasi	yang
3	terap	fcm	dan	tsk	untuk	tentu	cluster	rawan	pangan	di	kabupaten	cirebon	harliana	azhari
4	sistem	dukung	putus	dalam	tentu	supplier	jeruk	pontianak	bas	fuzzyahp	salahuddin	sri	hartati	abstrak
5	sistem	dukung	putus	kelompok	tentu	layak	lokasi	mukim	mutammimul	ula	azhari	sn	abstrak	bagai
6	sistem	dukung	putus	untuk	pilih	budidaya	ikan	air	tawar	guna	aftopsis	hence	beedwel	lumentut
7	susun	notasi	musik	dengan	guna	onset	detection	pada	sinyal	audio	anindita	suryarasmi	reza	pulung
8	aplikasi	rekomendasi	tempat	makan	guna	algoritma	slope	one	pada	platform	android	dharma	pratama	seng
9	fuzzy	ahp	nonadditive	pada	putus	diri	klinik	salin	daerah	awaluddin	dan	r	wardoyo	abstrak

LIST OF WORDS IN EVERY DOCUMENT

209	210	211	212	213	214	215	216	217	218	219	220	221	222	223	224	225	226	227
nbil	putus	tempat	pkl	mahasiswa	prodi	teknik	komputer	politeknik	aceh	selatan	kata	kunci	spkk	ahp	borda	praktek	kerja	lapang
ne	None	None	None	None	None	None	None	None	None	None	None	None	None	None	None	None	None	None
em	dukung	putus	entropy	topsis	goal	programming	None	None	None	None	None	None	None	None	None	None	None	None
rah	rawan	pangan	kata	kunci	fuzzy	cmeans	fuzzy	takagi	sugeno	kang	rawan	pangan	cluster	None	None	None	None	None
ruk	pontianak	snibsn	None	None	None	None	None	None	None	None	None	None	None	None	None	None	None	None
ne	None	None	None	None	None	None	None	None	None	None	None	None	None	None	None	None	None	None
ng	akan	jalan	kata	kunci	ikan	air	tawar	analisis	finansial	topsis	spk	None	None	None	None	None	None	None
ne	None	None	None	None	None	None	None	None	None	None	None	None	None	None	None	None	None	None
ne	None	None	None	None	None	None	None	None	None	None	None	None	None	None	None	None	None	None
ne	None	None	None	None	None	None	None	None	None	None	None	None	None	None	None	None	None	None

8. **Membuka hasil Count Vectorizer**

   Melihat jumlah kemunculan kata di setiap dokumen.

```
print("COUNT VECTORIZER")
countVector = []
for i in range(n):
    countVectorizer(words, i, countVector)
pandas.DataFrame(countVector)
```

Didapat output sebagai berikut...

COUNT VECTORIZER

	sistem	dukung	putus	kelompok	pilih	tempat	pkl	mahasiswa	dengan	guna	metode	ahp	dan	borda	dirja	nur	ilham	sri	mulyana	abstral
0	5.0	4.0	8.0	5.0	3.0	10.0	10.0	11.0	1.0	4	4.0	4.0	6	4.0	1.0	1.0	1.0	1.0	1.0	:
1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	2.0	3	1.0	NaN	5	NaN	NaN	NaN	NaN	NaN	NaN	:
2	5.0	2.0	6.0	NaN	NaN	NaN	NaN	NaN	4.0	2	5.0	NaN	5	NaN	NaN	NaN	NaN	1.0	NaN	:
3	2.0	NaN	NaN	1.0	NaN	NaN	NaN	NaN	1.0	2	2.0	NaN	8	NaN	NaN	NaN	NaN	NaN	NaN	:
4	1.0	2.0	1.0	NaN	NaN	NaN	NaN	NaN	4.0	2	2.0	2.0	3	NaN	NaN	NaN	NaN	1.0	NaN	:
5	2.0	3.0	3.0	2.0	NaN	NaN	NaN	NaN	1.0	1	3.0	NaN	5	NaN	NaN	NaN	NaN	NaN	NaN	:
6	2.0	1.0	3.0	NaN	3.0	NaN	NaN	NaN	NaN	3	1.0	NaN	3	NaN	NaN	NaN	NaN	1.0	NaN	:
7	2.0	1.0	NaN	NaN	NaN	NaN	NaN	NaN	8.0	3	1.0	NaN	1	NaN	NaN	NaN	NaN	NaN	NaN	:
8	1.0	NaN	NaN	NaN	2.0	10.0	NaN	NaN	4.0	8	NaN	NaN	3	NaN	NaN	NaN	NaN	NaN	NaN	:
9	NaN	1.0	1.0	NaN	NaN	1.0	NaN	NaN	4.0	2	1.0	4.0	7	NaN	NaN	NaN	NaN	NaN	NaN	:

9. **Membuka hasil TF-IDF**

   Menampilkan hasil perhitungan TF-IDF

```
print("TF-IDF")
tf_idf = []
computeTFIDF(words, tf_idf)
pandas.DataFrame(tf_idf)
```

Didapat output sebagai berikut...

TF IDF

	sistem	dukung	putus	kelompok	pilih	tempat	pkl	mahasiswa	dengan	guna	metode	ahp	dan	borda	dirja
0	0.010965	0.012531	0.015949	0.027412	0.016447	0.020886	0.04386	0.04386	0.001512	0.005848	0.008772	0.017544	0.005721	0.04386	0.04386
1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	0.004130	0.005988	0.002994	NaN	0.006509	NaN	NaN
2	0.011574	0.006614	0.012626	NaN	NaN	NaN	NaN	NaN	0.006386	0.003086	0.011574	NaN	0.005032	NaN	NaN
3	0.004484	NaN	NaN	0.005605	NaN	NaN	NaN	NaN	0.001546	0.002990	0.004484	NaN	0.007799	NaN	NaN
4	0.002358	0.006739	0.002144	NaN	NaN	NaN	NaN	NaN	0.006506	0.003145	0.004717	0.009434	0.003076	NaN	NaN
5	0.004975	0.010661	0.006784	0.012438	NaN	NaN	NaN	NaN	0.001716	0.001658	0.007463	NaN	0.005408	NaN	NaN
6	0.004525	0.003232	0.006170	NaN	0.016968	NaN	NaN	NaN	NaN	0.004525	0.002262	NaN	0.002951	NaN	NaN
7	0.006098	0.004355	NaN	NaN	NaN	NaN	NaN	NaN	0.016821	0.006098	0.003049	NaN	0.001326	NaN	NaN
8	0.002632	NaN	NaN	NaN	0.013158	0.025063	NaN	NaN	0.007260	0.014035	NaN	NaN	0.003432	NaN	NaN
9	NaN	0.003925	0.002498	NaN	NaN	0.002616	NaN	NaN	0.007579	0.003663	0.002747	0.021978	0.008361	NaN	NaN

10. Membuka hasil Jaccard Similarity

Menampilkan hhasi perhitungan Jaccard Similarity.

```
print("JACCARD SIMILARITY")
jaccardSim = [[None for i in range(n)] for j in range(n)]
for i in range(n):
    for j in range(i,n):
        jaccardSim[i][j] = round(jaccard_similarity(words[i], words[j]), 3)
pandas.DataFrame(jaccardSim)
```

Didapat output sebagai berikut...

JACCARD SIMILARITY

	0	1	2	3	4	5	6	7	8	9
0	1.0	0.171	0.169	0.136	0.240	0.262	0.149	0.180	0.167	0.202
1	NaN	1.000	0.132	0.180	0.177	0.193	0.102	0.181	0.175	0.145
2	NaN	NaN	1.000	0.143	0.182	0.173	0.255	0.127	0.174	0.116
3	NaN	NaN	NaN	1.000	0.146	0.200	0.118	0.132	0.170	0.132
4	NaN	NaN	NaN	NaN	1.000	0.196	0.167	0.155	0.168	0.230
5	NaN	NaN	NaN	NaN	NaN	1.000	0.128	0.138	0.157	0.155
6	NaN	NaN	NaN	NaN	NaN	NaN	1.000	0.113	0.107	0.115
7	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1.000	0.141	0.147
8	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1.000	0.122
9	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1.000

11. Membuka hasil Cosine Similarity

Menampilkan hasil perhitungan Cosine Similarity.

```
print("COSINE SIMILARITY")
cosineSim = [[None for i in range(n)] for j in range(n)]
for i in range(n):
    for j in range(i,n):
        cosineSim[i][j] = cosine_similarity(words[i], words[j])
pandas.DataFrame(cosineSim)
```

Didapat output sebagai berikut...

COSINE SIMILARITY

	0	1	2	3	4	5	6	7	8	9
0	1.0	0.291461	0.290021	0.240370	0.390437	0.416025	0.262939	0.304604	0.286199	0.337171
1	NaN	1.000000	0.233186	0.306622	0.303221	0.325205	0.187542	0.306138	0.298295	0.254152
2	NaN	NaN	1.000000	0.250249	0.308822	0.294937	0.409988	0.225762	0.297174	0.208249
3	NaN	NaN	NaN	1.000000	0.254312	0.333333	0.211721	0.234672	0.293991	0.233786
4	NaN	NaN	NaN	NaN	1.000000	0.328145	0.286005	0.270292	0.291585	0.373988
5	NaN	NaN	NaN	NaN	NaN	1.000000	0.227404	0.244059	0.274391	0.268421
6	NaN	NaN	NaN	NaN	NaN	NaN	1.000000	0.206689	0.197797	0.206544
7	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1.000000	0.247541	0.256759
8	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1.000000	0.218411
9	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1.000000

KESIMPULAN

Text Similarity merupakah suatu proses untuk melihat kemiripan antar dokumen yang dapat dilakukan dengan menggunakan Text processing dan dengan menggunakan metode jaccard similarity dan cosine similarity. Antar dokumen dikatakan mirip jika nilai hasil perhitungan dengan jaccard similarity dan cosine similarity mendekati 1.

Dengan langkah-langkah yang kelompok kami lakukan untuk melakukan document similarity pada 10 absrak paper didapatkan kesimpulan bahwa kemiripan antar dokumen sangat kecil. Dibuktikan dari hasil akhir dengan perhitungan Jaccard Similarity dan Cosine Similarity, tidak ada hasil yang menunjukkan kemiripan lebih dari 0.5, sehingga dapat disimpulkan kemiripan antar dokumen sangat kecil.