In [1]:

```
"""
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)
"""

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)
```

In [2]:

```
#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.punctuatio

    #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
```

In [3]:

```
#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)
```

In [4]:

```
#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)
```

In [5]:

```
#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]
```

In [6]:

```
#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 = {}
```

In [7]:

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

In [8]:

```
#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
```

In [9]:

```
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)
```

```
LIST OF WORDS IN EVERY DOCUMENT
```

Out[9]:

	0	1	2	3	4	5	6	7	
0	sistem	dukung	putus	kelompok	pilih	tempat	pkl	mahasiswa	der
1	aplikasi	deteksi	dini	defisiensi	mineral	mikro	pada	manusia	
2	sistem	dukung	putus	alokasi	spare	part	rita	wiryasaputra	
3	terap	fcm	dan	tsk	untuk	tentu	cluster	rawan	par
4	sistem	dukung	putus	dalam	tentu	supplier	jeruk	pontianak	
5	sistem	dukung	putus	kelompok	tentu	layak	lokasi	mukim	mutamm
6	sistem	dukung	putus	untuk	pilih	budidaya	ikan	air	t
7	susun	notasi	musik	dengan	guna	onset	detection	pada	s
8	aplikasi	rekomendasi	tempat	makan	guna	algoritma	slope	one	p
9	fuzzy	ahp	nonadditive	pada	putus	diri	klinik	salin	da

In [10]:

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

```
COUNT VECTORIZER
```

Out[10]:

	sistem	dukung	putus	kelompok	pilih	tempat	pkl	mahasiswa	dengan	guna	metode
0	5.0	4.0	8.0	5.0	3.0	10.0	10.0	11.0	1.0	4	4.0
1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	2.0	3	1.0
2	5.0	2.0	6.0	NaN	NaN	NaN	NaN	NaN	4.0	2	5.0
3	2.0	NaN	NaN	1.0	NaN	NaN	NaN	NaN	1.0	2	2.0
4	1.0	2.0	1.0	NaN	NaN	NaN	NaN	NaN	4.0	2	2.0
5	2.0	3.0	3.0	2.0	NaN	NaN	NaN	NaN	1.0	1	3.0
6	2.0	1.0	3.0	NaN	3.0	NaN	NaN	NaN	NaN	3	1.0
7	2.0	1.0	NaN	NaN	NaN	NaN	NaN	NaN	8.0	3	1.0
8	1.0	NaN	NaN	NaN	2.0	10.0	NaN	NaN	4.0	8	NaN
9	NaN	1.0	1.0	NaN	NaN	1.0	NaN	NaN	4.0	2	1.0

In [11]:

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

```
TF-IDF
```

Out[11]:

	sistem	dukung	putus	kelompok	pilih	tempat	pkl	mahasiswa	dengan
0	0.010965	0.012531	0.015949	0.027412	0.016447	0.020886	0.04386	0.04386	0.001512
1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	0.004130
2	0.011574	0.006614	0.012626	NaN	NaN	NaN	NaN	NaN	0.006386
3	0.004484	NaN	NaN	0.005605	NaN	NaN	NaN	NaN	0.001546
4	0.002358	0.006739	0.002144	NaN	NaN	NaN	NaN	NaN	0.006506
5	0.004975	0.010661	0.006784	0.012438	NaN	NaN	NaN	NaN	0.001716
6	0.004525	0.003232	0.006170	NaN	0.016968	NaN	NaN	NaN	NaN
7	0.006098	0.004355	NaN	NaN	NaN	NaN	NaN	NaN	0.016821
8	0.002632	NaN	NaN	NaN	0.013158	0.025063	NaN	NaN	0.007260
9	NaN	0.003925	0.002498	NaN	NaN	0.002616	NaN	NaN	0.007579

In [12]:

```
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)
```

```
JACCARD SIMILARITY
```

Out[12]:

	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

In [13]:

```
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)
```

```
COSINE SIMILARITY
```

Out[13]:

	0	1	2	3	4	5	6	7	8	
0	1.0	0.291461	0.290021	0.240370	0.390437	0.416025	0.262939	0.304604	0.286199	0.337
1	NaN	1.000000	0.233186	0.306622	0.303221	0.325205	0.187542	0.306138	0.298295	0.254
2	NaN	NaN	1.000000	0.250249	0.308822	0.294937	0.409988	0.225762	0.297174	0.208
3	NaN	NaN	NaN	1.000000	0.254312	0.333333	0.211721	0.234672	0.293991	0.233
4	NaN	NaN	NaN	NaN	1.000000	0.328145	0.286005	0.270292	0.291585	0.373
5	NaN	NaN	NaN	NaN	NaN	1.000000	0.227404	0.244059	0.274391	0.268
6	NaN	NaN	NaN	NaN	NaN	NaN	1.000000	0.206689	0.197797	0.206
7	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1.000000	0.247541	0.256
8	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1.000000	0.218
9	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1.000