PRAKTIKUM 2

TEXT MINING

DIBUAT OLEH:

Nama	:	Fajar Krisna Jaya
NIM	:	162012133026
Tanggal Praktikum	:	23/09/2023

LAPORAN PRAKTIKUM:

1. **Lakukan seluruh percobaan pada modul ini dan berikan analisis yang kalian temukan**

NLTK :

```
import nltk

txt = 'In Brazil they drive on the right-hand side of the road. Has a large
coastline on the eastern side of South America'

from nltk.tokenize import word_tokenize
token = word_tokenize(txt)
token
```

```
['In',
 'Brazil',
 'they',
 'drive',
 'on',
 'the',
 'right-hand',
 'side',
 'of',
 'the',
 'road',
 '.',
 'Has',
 'a',
 'large',
 'coastline',
 'on',
 'the',
 'eastern',
 'side',
 'of',
 'South', 'America']
```

Analisis dan Pembahasan :

Kode tersebut mengimpor package Natural Language Toolkit (nltk) dan mengassign sebuah string teks ke variabel 'txt'. Kemudian, kode tersebut menggunakan fungsi 'word_tokenize' dari nltk untuk membagi teks tersebut menjadi kata-kata individual. Hasilnya disimpan dalam variabel 'token'.

```
from nltk.probability import FreqDist
fdist = FreqDist(token)
fdist
```

```
FreqDist({'the': 3, 'on': 2, 'side': 2, 'of': 2, 'In': 1, 'Brazil': 1, 'they': 1, 'drive': 1, 'right-hand': 1, 'road': 1, ...})
```

Analisis dan Pembahasan :

Kode tersebut mengimpor pustaka Natural Language Toolkit (nltk) dan menghitung distribusi frekuensi kata-kata dalam variabel 'token' menggunakan fungsi FreqDist. Hasilnya disimpan dalam variabel 'fdist'.

```
from nltk.probability import FreqDist
fdist = FreqDist(token)
fdist1 = fdist.most_common(10)
fdist1
```

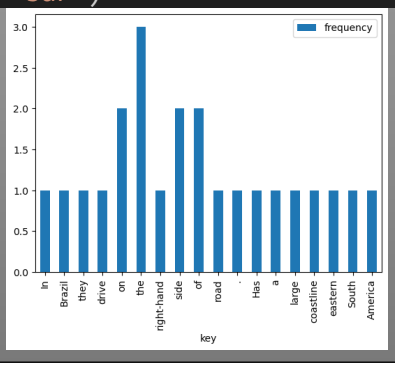
```
[('the', 3),
 ('on', 2),
 ('side', 2),
 ('of', 2),
 ('In', 1),
 ('Brazil', 1),
 ('they', 1),
 ('drive', 1),
 ('right-hand', 1),
 ('road', 1)]
```

Analisis dan Pembahasan :

Pertama, kode tersebut membuat objek FreqDist dari kata-kata dalam teks yang telah di-tokenize, yang disimpan dalam variabel 'fdist'. Kemudian, kode tersebut mengambil sepuluh kata yang paling sering

muncul dalam teks tersebut dengan menggunakan metode 'most_common(10)' dari objek FreqDist, dan hasilnya disimpan dalam variabel 'fdist1'. Hasil akhir berisi sepuluh kata yang paling sering muncul beserta jumlah kemunculannya dalam teks.

```
import pandas as pd
df_freq_tokens = pd.DataFrame.from_dict(FreqDist(token), orsient='index')
df_freq_tokens.columns = ['frequency']
df_freq_tokens.index.name = 'key'
df_freq_tokens.plot(kind = 'bar')
```



Analisis dan Pembahasan :

DataFrame 'df_freq_tokens' dibuat dengan menggunakan data dari FreqDist (distribusi frekuensi) yang ada dalam variabel 'token'. Kemudian, kolom DataFrame dinamai 'frequency' untuk mengindikasikan frekuensi kemunculan kata-kata. Indeks DataFrame ditetapkan sebagai 'key' untuk mengidentifikasi kata-kata. Lalu, dibuat bar plot untuk merepresentasikan masing-masing key dan frekuensinya.

```
from nltk import word_tokenize
from nltk.corpus import stopwords
a = set(stopwords.words('english'))
text = 'Cristiano Ronaldo was born on February 5, 1985, in Funchal, Madeira,
Portugal.'

text1 = word_tokenize(text.lower())
print(text1)
stopwords = [x for x in text1 if x not in a]
print(stopwords)
```

```
['cristiano', 'ronaldo', 'was', 'born', 'on', 'february', '5', ',', '1985',
',', 'in', 'funchal', ',', 'madeira', ',', 'portugal', '.']
['cristiano', 'ronaldo', 'born', 'february', '5', ',', '1985', ',', 'funchal',
                    ',', 'madeira', ',', 'portugal', '.']
```

Analisis dan Pembahasan :

Teks pada kode di atas melakukan beberapa tindakan terkait pemrosesan teks dalam bahasa Inggris. Pertama, teks di atas mengimport beberapa modul dari package milik Natural Language Toolkit (nltk), termasuk 'word_tokenize' dan 'stopwords'. Selanjutnya, kode menginisialisasi sebuah teks yang akan diproses. Kemudian, teks tersebut diubah menjadi huruf kecil dan di-tokenisasi menggunakan 'word_tokenize'. Hasil tokenisasi tersebut disimpan dalam variabel 'text1'. Selanjutnya, kode tersebut membuat daftar kata-kata tanpa stopwords dari 'text1', dan hasilnya disimpan dalam variabel 'stopwords'.

Hasil akhir adalah daftar kata-kata dari teks yang telah di-tokenisasi dan telah dihapus kata-kata stopwords dalam bahasa Inggris.

```
#contoh stemming di nltk
#lancaster
from nltk.stem.lancaster import LancasterStemmer
from nltk.stem.porter import PorterStemmer
from nltk.stem.snowball import  SnowballStemmer

S = "Presumably i would like to MultiPly my provision, saying tHat without
crYing"
print('Sentence: ',S)

stemmer_list = [LancasterStemmer,PorterStemmer,SnowballStemmer]
names = ['Lancaster','Porter','SnowBall']
for stemmer_name,stem in zip(names,stemmer_list):
    if stemmer_name == "SnowBall":
        st = stem("english")
    else:
        st = stem()
    print(stemmer_name,':',' '.join([st.stem(word) for word in S.split()]))
```

```
Sentence:  Presumably i would like to MultiPly my provision, saying tHat
without crYing
Lancaster : presum i would lik to multiply my provision, say that without cry
Porter : presum i would like to multipli my provision, say that without cri
 SnowBall : presum i would like to multipli my provision, say that without cri
```

Analisis dan Pembahasan :

Kode di atas menunjukkan contoh penggunaan stemming dengan tiga algoritma yang berbeda (Lancaster, Porter, dan Snowball) dalam Natural Language Toolkit (nltk). Teks awal yang akan di-stem adalah: "Presumably i would like to MultiPly my provision, saying tHat without crYing" Kemudian, kode tersebut melakukan stemming menggunakan ketiga algoritma tersebut pada teks di atas. Hasil stemming untuk setiap algoritma ditampilkan dengan menggabungkan kata-kata yang sudah di-stem dalam satu baris, dan nama algoritma juga ditampilkan.

```
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()

print('rocks :', lemmatizer.lemmatize('rocks'))
print('corpora :', lemmatizer.lemmatize('corpora'))
rocks : rock
corpora : corpus
```

Analisis dan Pembahasan :

Kode tersebut melakukan lemmatisasi pada dua kata, yaitu "rocks" dan "corpora". Hasil lemmatisasi sebagai berikut:

- Lemmatisasi kata "rocks" menghasilkan "rock".
- Lemmatisasi kata "corpora" menghasilkan "corpus".

```
from nltk import pos_tag
S = "i am currently learning NLP in English, but if possible to know NLP in
indonesian language too"

token = word_tokenize(S)
print(pos_tag(token))
[('i', 'NN'), ('am', 'VBP'), ('currently', 'RB'), ('learning', 'VBG'), ('NLP', 'NNP'), ('in', 'IN'), ('English', 'NNP'),
(',', ','), ('but', 'CC'), ('if', 'IN'), ('possible', 'JJ'), ('to', 'TO'), ('know', 'VB'), ('NLP', 'NNP'), ('in', 'IN'),
('indonesian', 'JJ'), ('language', 'NN'), ('too', 'RB')]
```

Analisis dan Pembahasan :

Kode di atas melakukan tokenisasi pada teks yang diberikan dan kemudian melakukan proses POS tagging (Part-of-Speech tagging) menggunakan nltk. Hasil POS tagging untuk kata-kata dalam teks tersebut akan

mengidentifikasi jenis kata (kata benda, kata kerja, kata sifat, dll.) Hasilnya akan menunjukkan kata-kata dalam teks dengan jenis kata yang sesuai, seperti kata benda (NN), kata kerja (VB), dll.

TEXTBLOB :

```
from textblob import TextBlob

T = " Hello, Mr. Man. He smiled !! This, i.e that, is it"
sentence_tokens = TextBlob(T).sentences

#Tokenisasi Kata
print(TextBlob(T).words)

#Tokenisasi Kalimat
print([str(sent) for sent in sentence_tokens])
```

```
['Hello', 'Mr', 'Man', 'He', 'smiled', 'This', 'i.e', 'that', 'is', 'it']
[' Hello, Mr. Man.', 'He smiled !!', 'This, i.e that, is it']
```

Analisis dan Pembahasan :

Kodingan tersebut yaitu melakukan 2 macam tokenisasi, yaitu tokenisasi kata dan tokenisasi kalimat. Hasil tokenisasi yaitu membagi teks menjadi kata-kata individual dan kalimat-kalimat berdasarkan tanda baca dan spasi

```
#Contoh Textblob Stemming & Lemmatizer
from textblob import Word
#Stemming
print("Stem :", Word("running").stem())

#Lemmatizer
print("Lemmatize :", Word("went").lemmatize("v"))
```

```
Stem : run
Lemmatize : go
```

Analisis dan Pembahasan :

Kode tersebut menghasilkan bentuk dasar (lemma) dari kata "running" dan "went" sesuai dengan jenis kata yang diinginkan ("v" untuk kata kerja dalam kasus lemmatisasi).

```
T = "Hello, Mr. Man. He smiled !! This, i.e that, is it"
for word, pos in TextBlob(T).tags:
    print(word, pos,end=", ")
```

```
Hello NNP, Mr. NNP, Man NNP, He PRP, smiled VBD, This DT, i.e NN, that IN,
is VBZ, it PRP,
```

Analisis dan Pembahasan :

Dalam kode di atas, pustaka TextBlob digunakan untuk melakukan POS tagging pada kata-kata dalam teks yang diberikan. Kode ini memberikan tanda jenis kata (POS tags) untuk setiap kata dalam teks, seperti kata benda (NN), kata kerja (VBD), kata sifat (JJ), dan sebagainya.

SASTRAWI :

```
from Sastrawi.StopWordRemover.StopWordRemoverFactory import
StopWordRemoverFactory
```

```
from nltk.tokenize import  word_tokenize
factory = StopWordRemoverFactory()
stopword = factory.create_stop_word_remover()
kalimat = 'Andi kerap melakukan transaksi rutin secara daring ataU online.
Menurut andi belanja online lebih praktis'
stop = stopword.remove(kalimat.lower())
print(stop)
```

```
andi kerap melakukan transaksi rutin daring online. andi belanja online lebih praktis
```

Analisis dan Pembahasan :

Kode tersebut menghilangkan kata-kata stopwords seperti "atau" dan "menurut" dari kalimat yang diberikan, dan mengonversi semua huruf menjadi huruf kecil untuk konsistensi.

```
#Lemmatizer dengan Sastrawi
from Sastrawi.Stemmer.StemmerFactory import StemmerFactory
stemmer = StemmerFactory().create_stemmer()

I = "Perayaan itu bebarengan dengan kita bepergian ke Makassar"
print(stemmer.stem(I))
print(stemmer.stem("Perayaan Bepergian Menyuarakan"))
```

```
raya itu bebarengan dengan kita pergi ke makassar
raya pergi suara
```

Analisis dan Pembahasan :

Kode di atas menggunakan pustaka Sastrawi untuk melakukan lemmatisasi pada kata-kata dalam bahasa Indonesia. Hasil tertera pada output di atas.

```
from matplotlib import pyplot as plt
from wordcloud import WordCloud

wordcloud = WordCloud(background_color='white').generate(text)

#plot the wordcloud
plt.figure(figsize = (12, 12))
plt.imshow(wordcloud)

#to remove the axis value
plt.axis('off')
plt.show()
```

Analisis dan Pembahasan :

Kode tersebut digunakan untuk membuat dan menampilkan *word cloud* dari teks yang diberikan. *Word cloud* adalah representasi visual dari kata-kata dalam teks, di mana kata-kata yang lebih sering muncul akan ditampilkan dengan ukuran yang lebih besar. Kode ini melakukan objek *word cloud* dengan latar belakang putih, mengatur ukuran tampilan, menampilkan *word cloud* tanpa sumbu, dan akhirnya menampilkan *word cloud* di layar.

CLUSTERING :

```
# representasi vektor dengan VSM-TFIDF
from sklearn.feature_extraction. text import TfidfVectorizer
from sklearn import cluster
tfidf_vectorizer = TfidfVectorizer (max_df=0.95, min_df=2)
X = tfidf_vectorizer.fit_transform(docs_clear)
print(X.shape)
k = 3
seed = 99 # Sembarang nilai untuk Random generator, mengapa penting? agar
ketika dijalankan ulang nilai randomnya te km = cluster.KMeans
(n_clusters=k, init=' random', max_iter=300, random_state = seed)
km = cluster.KMeans (n_clusters=k, init='random', max_iter=300, random_state
= seed)
km.fit(X)
#Hasil clusteringnya
C_km =km.predict(X)
C_km [:10]
```

```
(1653, 10697)
array([2, 2, 2, 2, 2, 2, 2, 2, 2, 2])
```

Analisis dan Pembahasan :

Kode tersebut digunakan untuk melakukan representasi vektor menggunakan metode VSM-TFIDF (Vector Space Model - Term Frequency-Inverse Document Frequency) pada dokumen yang telah diolah sebelumnya. Selanjutnya, kode ini mengubah dokumen yang telah dipreproses sebelumnya menjadi representasi vektor dengan memanggil metode 'fit_transform' pada objek TfidfVectorizer. Hasilnya disimpan dalam variabel 'X'.Kemudian, kode ini menentukan jumlah cluster 'k' yang diinginkan dan nilai

seed untuk generator angka acak supaya nanti bisa digunakan lagi dan dapat output sama. Setelah itu, kode melakukan proses clustering menggunakan algoritma K-Means dengan 'k' cluster menggunakan objek 'cluster.KMeans'. Hasil clustering disimpan dalam variabel 'C_km', dan kode menampilkan contoh hasil clustering untuk 10 dokumen pertama. Hasil akhir adalah daftar indeks cluster yang menunjukkan dokumen mana yang termasuk dalam setiap cluster. Misalnya, indeks 2 menunjukkan bahwa 10 dokumen pertama termasuk dalam cluster 2.

```
kmPP= cluster.KMeans (n_clusters=k, init='k-means++', max_iter=300,
tol=0.0001, random_state = seed)
kmPP.fit(X)
C_kmpp= kmPP.predict(X)
C_kmpp[:10]
```

```
array([2, 2, 2, 2, 2, 2, 2, 2, 2, 2])
```

Analisis dan Pembahasan :

Kode ini melanjutkan proses clustering dokumen menggunakan algoritma K-Means dengan inisialisasi 'k-means++' setelah sebelumnya menggunakan inisialisasi 'random'. Dengan parameter yang telah ditentukan seperti jumlah cluster, iterasi maksimum, toleransi, dan seed untuk angka acak, kode ini menghasilkan hasil clustering baru yang disimpan dalam variabel 'C_kmpp'. Contoh hasil clustering untuk 10 dokumen pertama dengan inisialisasi 'k-means++' ditampilkan dalam 'C_kmpp[:10]'.

```
import numpy as np
dbscan= cluster.DBSCAN (eps=0.5)
dbscan.fit(X)
C_db =dbscan. labels_.astype (int)
C_db [:10]
```

```
array([-1, -1, -1, -1, -1, -1, -1, -1, -1, -1])
```

Analisis dan Pembahasan :

Kode tersebut menerapkan algoritma DBSCAN (Density-Based Spatial Clustering of Applications with Noise) pada data yang direpresentasikan sebagai vektor menggunakan metode VSM-TFIDF. Dengan parameter `eps` yang mengontrol jarak maksimum antara data untuk membentuk kluster, DBSCAN berusaha untuk mengelompokkan data berdasarkan kepadatan, dan hasil clusteringnya disimpan dalam variabel `C_db`. Contoh hasil clustering untuk 10 data pertama menunjukkan label kluster, di mana label -1 mengindikasikan data yang dianggap sebagai "noise" atau bukan bagian dari kluster apa pun dalam algoritma DBSCAN.

```
from sklearn.metrics import silhouette_score as siluet
C = [C_km, C_kmpp, C_db]
for res in C:
    print(siluet (X, res), end=', ')
```

```
0.01858467601596153, 0.01858467601596153, -0.23697800236118444,
```

Analisis dan Pembahasan :

Kode tersebut menghitung dan mencetak nilai silhouette untuk tiga hasil clustering yang berbeda, yaitu dari algoritma K-Means dengan inisialisasi acak, K-Means dengan inisialisasi 'k-means++', dan DBSCAN. Silhouette adalah metrik yang mengukur kualitas kluster, dengan nilai yang lebih tinggi menunjukkan kluster yang lebih baik dalam memisahkan data.

```
from sklearn.metrics.cluster import homogeneity_score as purity
for res in C:
    print(purity (label, res), end=', ')
```

```
0.06952188099464576, 0.06952188099464576, 0.0015549989772357536,
```

Analisis dan Pembahasan :

Kode di atas menggunakan metrik homogeneity score (purity) dari scikit-learn untuk mengukur sejauh mana kluster yang dihasilkan konsisten dengan label sebenarnya (ground truth). Dalam loop, kode ini menghitung dan mencetak nilai purity untuk tiga hasil clustering yang berbeda (dari algoritma K-Means dengan inisialisasi acak, K-Means dengan inisialisasi 'k-means++', dan DBSCAN).

```
# Evaluasi eksternal NMI
from sklearn.metrics import normalized_mutual_info_score as NMI
for res in C:
    print (NMI (label, res), end=', ')
0.10920145484701148, 0.10920145484701148, 0.002704356633566744,
```

Analisis dan Pembahasan :

Sama seperti kodingan di atas, kode ini melakukan evaluasi terhadap 3 metode clustering. Bedanya adalah evaluasi kali ini menggunakan NMI.

2. Jelaskan perbedaan hasil dari Preprocessing menggunakan NLTK, TextBlob dan Sastrawi

```
# Teks contoh dalam bahasa Inggris
text = "This is an example of text preprocessing using TextBlob and \n /n NLTK.  !? This technique _ is used to clean text from stopwords. :)"
# Tokenisasi (TextBlob)
blob = TextBlob(text)
filtered_blob = [word for word in blob.words if word.lower() not in stopwords.words('english')]
# Tokenisasi (NLTK)
tokens = word_tokenize(text)
filtered_tokens = [word for word in tokens if word.lower() not in stopwords.words('english')]
# Hasil
print("TextBlob Result:")
print(filtered_blob)
print("\nNLTK Result:")
print(filtered_tokens)
TextBlob Result:
['example', 'text', 'preprocessing', 'using', 'TextBlob', 'n', 'NLTK', 'technique', 'used', 'clean', 'text', 'stopwords']

NLTK Result:
['example', 'text', 'preprocessing', 'using', 'TextBlob', '/n', 'NLTK', '.', '!', '?', 'technique', '_', 'used', 'clean', 'text', 'stopwords', '.', ':', ')']
```

Perbedaan utama adalah bahwa TextBlob secara otomatis menghapus tanda baca misal seperti titik (".") dari token, sedangkan NLTK mempertahankannya sebagai token terpisah.

```
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import PorterStemmer, LancasterStemmer, SnowballStemmer
from nltk.stem import WordNetLemmatizer
from textblob import TextBlob

# Teks contoh dalam bahasa Inggris
text = "This is an example of text preprocessing using TextBlob and NLTK. !?
\n This technique _ is used to clean text from stopwords. :)"

# Tokenisasi (NLTK)
tokens = word_tokenize(text)
filtered_tokens = [word for word in tokens if word.lower() not in
stopwords.words('english')]

# Stemming (Porter)
porter_stemmer = PorterStemmer()
stemmed_porter_tokens = [porter_stemmer.stem(word) for word in
filtered_tokens]

# Stemming (Lancaster)
lancaster_stemmer = LancasterStemmer()
stemmed_lancaster_tokens = [lancaster_stemmer.stem(word) for word in
filtered_tokens]

# Stemming (Snowball)
snowball_stemmer = SnowballStemmer('english')
stemmed_snowball_tokens = [snowball_stemmer.stem(word) for word in
filtered_tokens]

# Lemmatisasi (NLTK)
lemmatizer = WordNetLemmatizer()
lemmatized_tokens = [lemmatizer.lemmatize(word) for word in filtered_tokens]

# Stemming (TextBlob)
blob = TextBlob(text)
stemmed_textblob_tokens = [word.stem() for word in blob.words]

# Lemmatisasi (TextBlob)
lemmatized_textblob_tokens = [word.lemmatize() for word in blob.words]
```

```
# Hasil
print("Porter Stemming Result:")
print(stemmed_porter_tokens)
print("\nLancaster Stemming Result:")
print(stemmed_lancaster_tokens)
print("\nSnowball Stemming Result:")
print(stemmed_snowball_tokens)
print("\nNLTK Lemmatization Result:")
print(lemmatized_tokens)
print("\nTextBlob Stemming Result:")
print(stemmed_textblob_tokens)
print("\nTextBlob Lemmatization Result:")
print(lemmatized_textblob_tokens)
```

```
Porter Stemming Result:
['exampl', 'text', 'preprocess', 'use', 'textblob', 'nltk', '.', '!', '?',
'techniqu', '_', 'use', 'clean', 'text', 'stopword', '.', ':', ')']

Lancaster Stemming Result:
['exampl', 'text', 'preprocess', 'us', 'textblob', 'nltk', '.', '!', '?',
'techn', '_', 'us', 'cle', 'text', 'stopword', '.', ':', ')']

Snowball Stemming Result:
['exampl', 'text', 'preprocess', 'use', 'textblob', 'nltk', '.', '!', '?',
'techniqu', '_', 'use', 'clean', 'text', 'stopword', '.', ':', ')']

NLTK Lemmatization Result:
['example', 'text', 'preprocessing', 'using', 'TextBlob', 'NLTK', '.', '!',
'?', 'technique', '_', 'used', 'clean', 'text', 'stopwords', '.', ':', ')']

TextBlob Stemming Result:
['thi', 'is', 'an', 'exampl', 'of', 'text', 'preprocess', 'use', 'textblob',
'and', 'nltk', 'thi', 'techniqu', 'is', 'use', 'to', 'clean', 'text', 'from',
'stopword']
```

Kalau Stemmer Porter cenderung lebih konservatif, sedangkan untuk yang Snowball memiliki pendekatan yang lebih modern dan ekstensif. Lancaster cenderung memiliki pendekatan yang lebih agresif dalam mengubah kata-kata.

```
from Sastrawi.StopWordRemover.StopWordRemoverFactory import
StopWordRemoverFactory
from nltk.tokenize import  word_tokenize
factory = StopWordRemoverFactory()
stopword = factory.create_stop_word_remover()
kalimat = 'Saya mengikuti pelajaran mata kuliah data mining II di kelas ini!
:)'
stop = stopword.remove(kalimat.lower())
print(stop)
```

```
mengikuti pelajaran mata kuliah data mining ii kelas ini! :)
```

```
#Lemmatizer dengan Sastrawi
from Sastrawi.Stemmer.StemmerFactory import StemmerFactory
stemmer = StemmerFactory().create_stemmer()

I = 'Saya mengikuti pelajaran mata kuliah data mining II di kelas ini! :)'
print(stemmer.stem(I))
print(stemmer.stem("Perayaan Bepergian Menyuarakan"))
```

```
saya ikut ajar mata kuliah data mining ii di kelas ini
raya pergi suara
```

Sementara untuk Sastrawi, Bahasa yang digunakan adalah menggunakan bahasa Indonesia. Saat melakukan stemming dan lematizing, sastrawi juga menghilangkan tanda baca.

3. Crawling dataset dengan total 10 pada berbagai portal berita Nasional dengan kategori bebas namun wajib sama

```
#import news10.xlsx
df = pd.read_excel('news10.xlsx')
df.shape
```

```
(10, 3)
```

```
df
```

	title	url	content
0	Jawaban Gerindra saat GNPF-PA 212 Tak Lagi Ber...	https://news.detik.com/pemilu/d-6947541/jawaba...	Gerakan Nasional Pembela Fatwa Ulama (GNPF Ula...
1	Kaesang Beda Partai dengan Jokowi, Ganjar: Ya ...	https://nasional.kompas.com/read/2023/09/24/09...	JAKARTA, KOMPAS.com - Bakal capres dari PDI-P ...
2	Disambut Lautan Massa, Anies Ingin AMIN & Part...	https://kumparan.com/kumparannews/disambut-jut...	Lautan massa menyambut kedatangan pasangan bac...
3	Hubungan Mega-Jokowi Disorot usai Kaesang Gabu...	https://nasional.kompas.com/read/2023/09/23/17...	Politisi Partai Demokrasi Indonesia Perjuangan...
4	Prabowo Kenang Masa Digembleng Bersama SBY di ...	https://www.cnnindonesia.com/nasional/20230922...	Bakal capres Koalisi Indonesia Maju Prabowo Su...
5	Puji SBY-Jokowi, Prabowo Bakal Jor-joran Lanju...	https://www.viva.co.id/berita/politik/1639937-...	Bakal calon presiden dari Koalisi Indonesia Ma...
6	Ganjar soal Kaesang Gabung PSI: Semua Orang Pu...	https://news.detik.com/pemilu/d-6947647/ganjar...	Putra bungsu Presiden Joko Widodo (Jokowi), Ka...
7	Netizen Sebut Kaesang Seharusnya Gabung Partai...	https://news.republika.co.id/berita/s1e896409/...	ali Kota Solo Gibran Rakabuming beri komentar ...
8	Izin Minta Restu Gabung PSI ke Gibran, Kaesang...	https://www.jawapos.com/politik/013012804/izin...	Kaesang Pangarep mengaku mendapat respons nega...
9	Pengamat: Kaesang Gabung PSI Sinyal Jokowi Duk...	https://www.cnnindonesia.com/nasional/20230923...	Pengamat politik Ahmad Khoirul Umum menganggap...

Di antara sumber portal berita yang telah dilakukan crawling antara lain dari detik, Kompas, kumparan, viva, cnn, dan jawa pos. Berita tersebut memiliki kategori politik.

4. **Lakukan preprocessing yang sudah diajarkan pada modul ini (menggunakan salah satu library saja)**

```
# Import necessary libraries
import pandas as pd
import nltk
from nltk.corpus import stopwords
from Sastrawi.Stemmer.StemmerFactory import StemmerFactory
import re

# Create a Sastrawi stemmer
factory = StemmerFactory()
stemmer = factory.create_stemmer()

# Define the text preprocessing function
def preprocess_text(text):
    # Remove URLs
    text = re.sub(r'http\S+', '', text)

    # Convert text to lowercase
    text = text.lower()

    # Remove special characters, numbers, and punctuation
    text = re.sub(r'[^a-zA-Z\s]', '', text)

    # Tokenize the text
    words = text.split()

    # Remove Indonesian stopwords
    indonesian_stopwords = stopwords.words('indonesian')
    filtered_words = [word for word in words if word not in
indonesian_stopwords]

    # Apply stemming to each word
    stemmed_words = [stemmer.stem(word) for word in filtered_words]

    # Join the filtered words back into a single string
    processed_text = ' '.join(stemmed_words)

    return processed_text
df['content_preprocess'] = df['content'].apply(preprocess_text)
```

```
df['content_preprocess']
(10, 3)
0    gera nasional bela fatwa ulama gnpf ulama pa f...
1    jakarta kompascom capres pdip ganjar pranowo r...
2    laut massa sambut datang pasang bacapres bacaw...
3    politis partai demokrasi indonesia juang pdip ...
4    capres koalisi indonesia maju prabowo subianto...
5    calon presiden koalisi indonesia maju kim prab...
6    putra bungsu presiden joko widodo jokowi kaesa...
7    ali kota solo gibran rakabuming komentar tuju ...
8    kaesang pangarep aku respons negatif kakak gib...
9    amat politik ahmad khoirul anggap putus kaesan...
Name: content_preprocess, dtype: object
```

Kode di atas melakukan preprocessing dengan membuat function terlebih dahulu. Library yang digunakan yatu sastrawi. Berikut ini adalah urutan dari function preprocessing :

1. Menghapus URL dari teks menggunakan ekspresi reguler.

2. Mengubah teks menjadi huruf kecil untuk konsistensi.

3. Menghapus karakter khusus, angka, dan tanda baca, hanya menjaga karakter alfabet.

4. Tokenisasi teks menjadi kata-kata individual.

5. Menghapus stopwords dalam bahasa Indonesia menggunakan pustaka Sastrawi.

6. Melakukan stemming pada setiap kata menggunakan stemmer dari Sastrawi.

7. Menggabungkan kata-kata yang telah difilter dan di-stem kembali menjadi satu string tunggal.

Hasil dari preprocessing disimpan dalam kolom baru bernama "content_preprocess" dalam DataFrame .

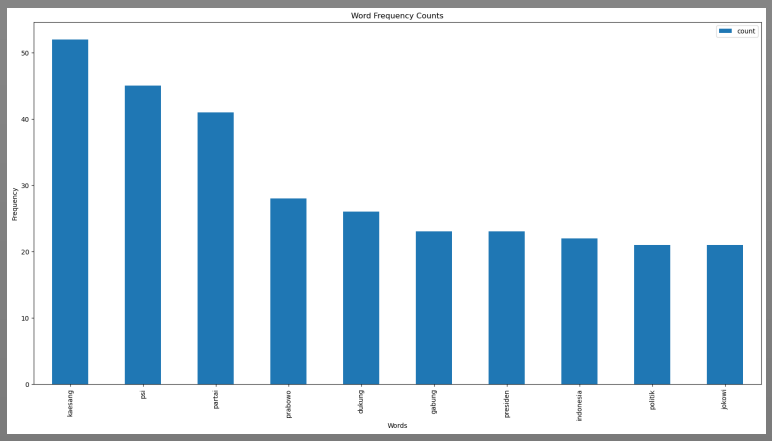
5. Buatlah wordcloud dan most common word barplot, interpretasikan hasilnya

```
from matplotlib import pyplot as plt
from wordcloud import WordCloud

# Combine all preprocessed text into a single string
text = ' '.join(df['content_preprocess'].tolist())

# Generate the wordcloud
wordcloud = WordCloud(background_color='white').generate(text)

# Plot the wordcloud
plt.figure(figsize=(12, 12))
plt.imshow(wordcloud)
plt.axis('off')
plt.show()
```

Analisis :

Dari hasil wordcloud di atas, bisa disimpulkan bahwa Kaesang, politik, psi, dan pdip menjadi kata yang paling sering muncul atau sedang hangat diberitakan oleh media. Hal ini juga sejalan dengan fakta bahwa per tanggal 24/09/2023, Kaesang Pangarep telah menyatakan diri untuk bergabung ke PSI daripada PDIP untuk mencalonkan diri menjadi Depok 1.

```python
import matplotlib.pyplot as plt
from collections import Counter

# Combine all preprocessed text into a single string
text = ' '.join(df['content_preprocess'].tolist())

# Count the frequency of each word
word_counts = Counter(text.split())

# Get the 10 most common words
most_common_words = word_counts.most_common(10)

# Convert the word counts to a pandas dataframe
df_word_counts = pd.DataFrame.from_dict(dict(most_common_words),
orient='index', columns=['count'])

# Sort the dataframe by count in descending order
df_word_counts = df_word_counts.sort_values(by='count', ascending=False)

# Plot the bar chart
df_word_counts.plot(kind='bar', figsize=(20,10))

# Set the title and axis labels
plt.title('Word Frequency Counts')
plt.xlabel('Words')
plt.ylabel('Frequency')
# Show the plot
plt.show()
```



Tampak juga pada bar plot bahwasanya kaesang, psi menempati 2 urutan terbanyak.

6. Lakukan clustering dengan menggunakan fitur TF-IDF

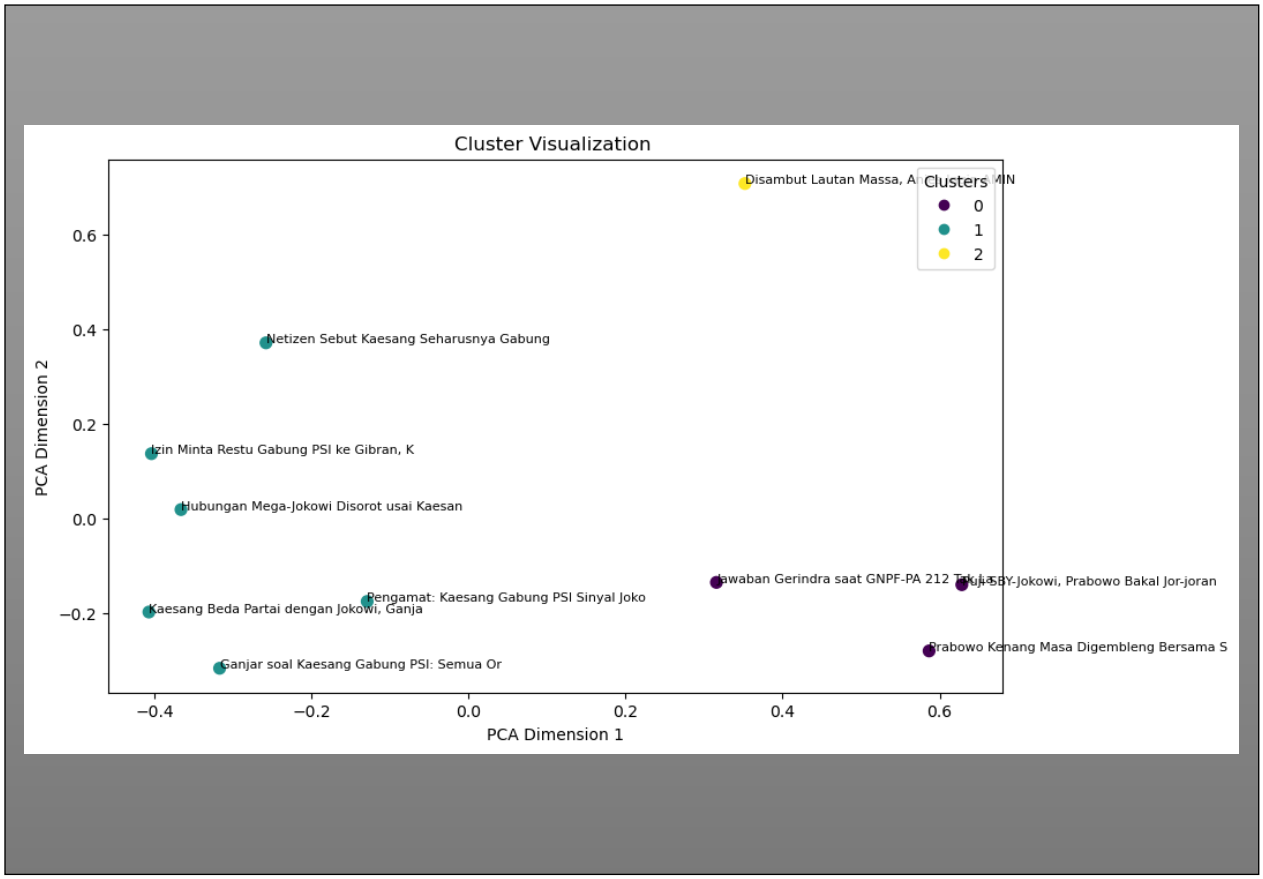
```
# representasi vektor dengan VSM-TFIDF
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn import cluster

# Create a TfidfVectorizer object
tfidf_vectorizer = TfidfVectorizer(max_df=0.95, min_df=2)

# Fit and transform the content column of the DataFrame
X = tfidf_vectorizer.fit_transform(df["content"])

# Print the shape of the resulting matrix
print(X.shape)

# Set the number of clusters
k = 3

# Set the random seed
seed = 99

# Create a KMeans object with the specified number of clusters and random seed
km = cluster.KMeans(n_clusters=k, init='random', max_iter=300,
random_state=seed)

# Fit the KMeans object to the data
km.fit(X)

# Get the cluster labels for each data point
C_km = km.predict(X)

# Print the first 10 cluster labels
print(C_km[:10])
```

```
(10, 311)
[1 0 1 0 1 1 0 2 2 0]
```

Tampak pada 10 klaster label bisa merepresentasikan klaster 0, 1, dan 2.

7. **Buat visualisasi clusternya dan lakukan interpretasi terhadap hasil tersebu**

```
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from sklearn.manifold import TSNE

# Apply PCA to reduce the dimensionality of the TF-IDF matrix
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X.toarray())

# Create a scatter plot of the clusters
plt.figure(figsize=(10, 6))
scatter = plt.scatter(X_pca[:, 0], X_pca[:, 1], c=C_km, cmap='viridis',
marker='o', s=50)

# Annotate each point with its corresponding truncated text title
max_title_length = 40
for i, txt in enumerate(df["title"]):
    truncated_txt = txt[:max_title_length] if len(txt) > max_title_length else
txt
    plt.annotate(truncated_txt, (X_pca[i, 0], X_pca[i, 1]), fontsize=8)
# Move the legend to a better position (e.g., upper right)
plt.legend(*scatter.legend_elements(), loc="upper right", title="Clusters")

plt.title('Cluster Visualization ')
plt.xlabel('PCA Dimension 1')
plt.ylabel('PCA Dimension 2')

plt.show()
```



Plot di atas merupakan bentuk visualisasi dua dimensi yang sebelumnya telah dilakukan PCA terlebih dahulu menggunakan number of components sebanyak 2. Bisa dilihat bahwa klaster 1 yang berwarna biru menunjukkan topik tentang PSI dan kaesang. Klaster ungu (0) menunjukkan tentang Gerindra dan prabowo. Sedangkan klaster 2 (kuning) menunjukkan tentang Cak Imin dan Anies.

8. Gunakan validasi menggunakan salah satu Davies-Bouldin index atau Silhouette score

```
#validate using silhouette score
from sklearn.metrics import silhouette_score as siluet
C = [C_km]
for res in C:
    print(siluet(X, res), end=', ')
```

```
0.12488082411605009,
```

Tampak bahwa nilai siluet adalah `0.12488082411605009` . Yang mana sebenarnya untuk interval siluet berada pada rentant -1 sampai dengan 1. Nilai klaster tersebut bisa disimpulkan bahwasanya data tidak tersebar dengan baik sehingga performa klaster kurang. Untuk memperbaiki bisa dilakukan dengan menambah data atau melakukan tuning. Namun menurut saya, apabila data hanya 10 maka tuning tidak berpengaruh signifikan untuk merubah performa.