

PRAKTEK TEXT MINING CLUSTERING

NAMA : HARVEST ECCLESIANO CHRIST WALUKOW
NIM : 164231104
MATA KULIAH : DATA MINING II

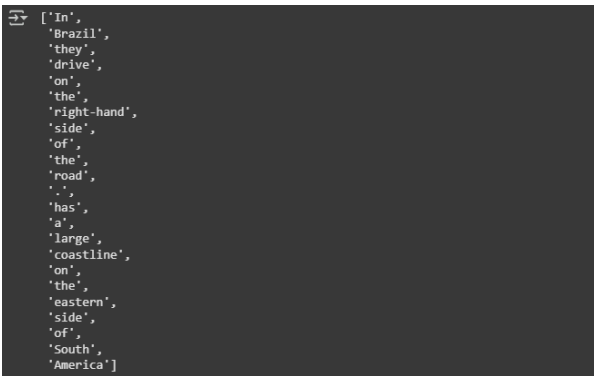
Saya melakukan import pada semua package/library yang dibutuhkan pada praktikum ini:

```
import re
from collections import Counter
import wikipedia
import matplotlib.pyplot as plt
import pandas as pd
from wordcloud import WordCloud
from sklearn.feature_extraction.text import TfidfVectorizer, CountVectorizer
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
from sklearn.metrics import silhouette_score
import nltk
from nltk.tokenize import word_tokenize
nltk.download('punkt')
nltk.download('punkt_tab')
nltk.download('averaged_perceptron_tagger')
nltk.download('wordnet')
nltk.download('stopwords')
from nltk.probability import FreqDist
```

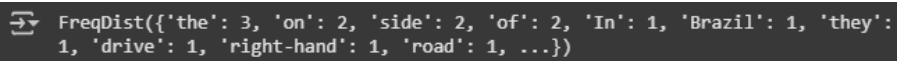
- 1. Lakukan seluruh percobaan pada modul ini dan berikan analisis yang kalian temukan
 - NLTK
 - Tokenisasi

```
text = "In Brazil they drive on the right-hand side of the road. has a large coastline on the eastern side of South America"

token = word_tokenize(text)
token
```



```
fdist = FreqDist(token)
fdist
```



```
fdist = FreqDist(token)
fdist1 = fdist.most_common(10)
```

PRAKTEK TEXT MINING CLUSTERING

NAMA : HARVEST ECCLESIANO CHRIST WALUKOW
NIM : 164231104
MATA KULIAH : DATA MINING II

```
fdist1
[('the', 3),
 ('on', 2),
 ('side', 2),
 ('of', 2),
 ('In', 1),
 ('Brazil', 1),
 ('they', 1),
 ('drive', 1),
 ('right-hand', 1),
 ('road', 1)]
```

Analisis: Kode tersebut memecah kalimat menjadi daftar kata dan tanda baca individual, lalu dihitung dan di sort banyak kemunculannya di chunk kode selanjutnya.

- Stopwords

```
a = set(stopwords.words('english'))
text = "Cristiano Ronaldo was born on February 5, 1985, in Funchal, Madeira, Portugal."
text1 = word_tokenize(text.lower())
stopwords_removed = [x for x in text1 if x not in a]
print(stopwords_removed)

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

Analisis: Kode ini mengubah text menjadi huruf kecil, lalu membuat daftar kata baru yang tidak termasuk kata-kata umum dalam bahasa Inggris (stopwords)

- Stemming

```
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(s) for s 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
```

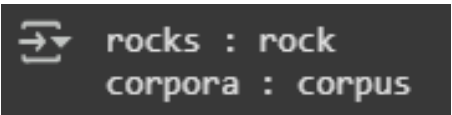
Analisis: Kode tersebut menerapkan tiga algoritma stemming yang berbeda (Lancaster, Porter, dan SnowBall) pada sebuah kalimat untuk membandingkan bagaimana setiap algoritma memotong kata-kata ke bentuk dasarnya.

NAMA : HARVEST ECCLESIANO CHRIST WALUKOW
NIM : 164231104
MATA KULIAH : DATA MINING II

- **Lemmatization**

```
lemmatizer = WordNetLemmatizer()

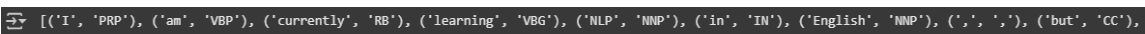
print("rocks :", lemmatizer.lemmatize("rocks"))
print("corpora :", lemmatizer.lemmatize("corpora"))
```



Analisis: Kode tersebut menggunakan lemmatizer WordNet untuk mengubah kata-kata ("rocks", "corpora") ke bentuk dasar ("rock", "corpus").

- **POS Tagging**

```
S = 'I am currently learning NLP in English, but if possible I want to know
NLP in Indonesian language too'
tokens = word_tokenize(S)
print(pos_tag(tokens))
```



Analisis: Memecahnya kalimat token, lalu melabeli setiap token dengan kategori gramatikalnya, seperti kata benda (NN), kata kerja (VB), atau kata sifat (JJ).

- **TextBlob**

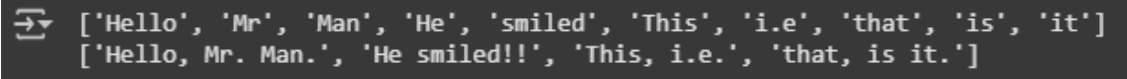
- **Tokenisasi**

```
# Contoh tokenisasi dengan 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])
```



Analisis: Menggunakan library TextBlob untuk menunjukkan cara memecah sebuah teks menjadi daftar kata-kata individual (word tokenization) dan juga menjadi daftar kalimat terpisah (sentence tokenization).

- **Stemming dan Lemmatization**

```
# Stemming
print("Stem: ", Word('running').stem())

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

PRAKTEK TEXT MINING CLUSTERING

NAMA : HARVEST ECCLESIANO CHRIST WALUKOW
NIM : 164231104
MATA KULIAH : DATA MINING II

```
➡ Stem: run  
Lemmatize: go
```

Analisis: Menggunakan TextBlob untuk melakukan stemming (memotong kata 'running' menjadi 'run') dan lemmatization (mengubah kata kerja 'went' ke bentuk dasarnya 'go').

- POS Tagging

```
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 DT, is VBZ, it PRP,
```

Analisis: Menggunakan TextBlob untuk melakukan stemming (memotong kata 'running' menjadi 'run') dan lemmatization (mengubah kata kerja 'went' ke bentuk dasarnya 'go').

- Sastrawi

- Stopwords

```
factory = StopWordRemoverFactory()  
stopword = factory.create_stop_word_remover()  
kalimat = "Andi kerap melakukan transaksi rutin secara daring atau online.  
Menurut Andi belanja online lebih praktis & murah."  
stop = stopword.remove(kalimat.lower())  
print(stop)
```

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

Analisis: Menggunakan library Sastrawi untuk mengambil sebuah kalimat berbahasa Indonesia dan menghapus kata-kata umum (stopwords).

- Stemming dan Lemmatization

```
# Lemmatizer dengan Sastrawi  
stemmer = StemmerFactory().create_stemmer()  
  
I = "perayaan itu berbarengan dengan saat kita bepergian ke Makassar"  
print(stemmer.stem(I))  
print(stemmer.stem("Perayaan Bepergian Menyuarakan"))
```

```
➡ raya itu bareng dengan saat kita pergi ke makassar  
raya pergi suara
```

Analisis: menggunakan stemmer dari library Sastrawi untuk mengubah kata-kata berimbuhan dalam bahasa Indonesia ("perayaan" atau "bepergian") ke bentuk kata dasarnya ("raya" atau "pergi").

- Worcloud

PRAKTEK TEXT MINING CLUSTERING

NAMA : HARVEST ECCLESIANO CHRIST WALUKOW
NIM : 164231104
MATA KULIAH : DATA MINING II

```
string = "wordcloud merupakan salah satu cara visualisasi deskriptif pada data teks  
sifatnya mirip dengan barplot frekuensi namun semakin besar frekuensi kata  
tersebut semakin besar ukuran kata tersebut dalam wordcloud"  
wordcloud = WordCloud(background_color="white").generate(string)  
  
# plot the wordcloud  
plt.figure(figsize = (12, 12))  
plt.imshow(wordcloud)  
  
# to remove the axis value  
plt.axis("off")  
plt.show()
```



- Clustering
- K-means


```
stop_words = set(stopwords.words('english'))  
src_path = "20newsgroup.pkl"  
with open(src_path, 'rb') as fin:  
    data = pickle.load(fin)  
  
docs = [doc for doc in data.data]  
label = data.target  
  
def preprocess(doc):  
    sents = word_tokenize(doc)  
    sents_tok = list() # tokenisasi kalimat  
    sents = [t for t in sents if t not in stop_words]  
    for s in sents:  
        s = s.strip().lower() # case folding dan menghilangkan new line  
        s = re.sub("\n", " ", s) # menggantikan \n dengan spasi  
        s = re.sub(r'[^\w\s-]', '', s) # menghapus simbol
```

PRAKTEK TEXT MINING CLUSTERING

NAMA : HARVEST ECCLESIANO CHRIST WALUKOW
NIM : 164231104
MATA KULIAH : DATA MINING II

```
s = re.sub(' +', ' ', s) # menghapus repetitive space
sents_tok.append(s)
return "".join(sents_tok)

docs_clear = list()
for d in docs:
    docs_clear.append(preprocess(d))
print('DONE!')
```


 DONE!

Analisis: Membersihkan 20newsgroup.pkl dengan menghilangkan stopwords, simbol, dan spasi berlebih, lalu mengubah semua huruf kecil.

```
# representasi vektor dengan VSM-TFIDF

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 tetap sama
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, 3306)
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0], dtype=int32)

Analisis: Dokumen yang sudah bersih diubah menjadi matriks angka menggunakan TF-IDF, lalu mengelompokkan dokumen-dokumen (1653 dokument) tersebut ke dalam 3 kategori berbeda dengan algoritma K-Means, sepuluh dokumen pertama semuanya masuk ke cluster 0.

```
# k-means++ clustering
# http://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html
from sklearn import cluster # Assuming cluster is already imported

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([0, 0, 0, 0, 0, 0, 0, 0, 0, 0], dtype=int32)

PRAKTEK TEXT MINING CLUSTERING

NAMA : HARVEST ECCLESIANO CHRIST WALUKOW
NIM : 164231104
MATA KULIAH : DATA MINING II

Analisis: Dengan inisialisasi K-means++, sepuluh dokumen pertama semuanya terkelompok ke dalam cluster 0.

- DB Scan

```
# http://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html

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

```
array([-1, -1, -1, -1, -1,  0, -1, -1, -1, -1])
```

Analisis: Menerapkan algoritma clustering DBSCAN, yang hasilnya mengidentifikasi sembilan dari sepuluh dokumen pertama sebagai noise atau outlier (label -1) dan hanya satu yang masuk ke cluster 0.

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

# NOTE: Silhouette coefficient hanya cocok untuk k-means
```

```
0.0701661665254299, 0.07176791258022419, -0.08587837330067329,
```

Analisis: Dua hasil pertama (K-Means) memiliki cluster yang valid namun tumpang tindih (skor positif kecil), sedangkan hasil ketiga (DBSCAN) sangat buruk (skor negatif)

```
from sklearn.metrics.cluster import homogeneity_score as purity

for res in C:
    print(purity(label, res), end=', ')

0.04335587574747806, 0.04389300205138689, 0.02630442528703482,
```

Analisis: Nilai homogeneity score (purity) dari hasil clustering dibandingkan dengan label aslinya, di mana angka yang kecil (misalnya 0.04 atau 0.02) berarti hasil cluster masih jauh dari label sebenarnya (kurang baik)

```
from sklearn.metrics import normalized_mutual_info_score as NMI

C = [C_km, C_kmpp, C_db]
for res in C:
    print(NMI(label, res), end=', ')

0.07352142821634576, 0.067727096875415, 0.03483558098524454,
```

PRAKTEK TEXT MINING CLUSTERING

NAMA : HARVEST ECCLESIANO CHRIST WALUKOW
NIM : 164231104
MATA KULIAH : DATA MINING II

Analisis: Membandingkan dua set label: label asli dari data (label) dan label hasil prediksi model Anda (res). Skor 1 berarti cocok sempurna, sementara skor 0 berarti tidak ada kecocokan sama sekali (hasilnya acak). Dari hasil ini, clustering yang dilakukan masih kurang baik.

2. Jelaskan perbedaan hasil dari Preprocessing menggunakan NLTK, TextBlob dan Sastrawi dan berikan contohnya.
Dari percobaan yang telah dilakukan perbedaan utama antara ketiga pustaka tersebut terletak pada bahasa yang didukung dan tujuannya. NLTK dan TextBlob dirancang untuk Bahasa Inggris; NLTK menawarkan kontrol yang lebih detail, sementara TextBlob menyediakan antarmuka yang lebih sederhana (syntax-nya lebih singkat). Sebaliknya, Sastrawi adalah pustaka khusus untuk Bahasa Indonesia.
3. Crawling dataset dengan total 10 pada berbagai judul artikel Wikipedia berdasarkan daftar topik sesuai dengan akhiran nim dan wajib dalam topik yang sama.
Saya mencoba untuk melakukan crawling pada artikel Wikipedia tentang 10 bahasa pemrograman.

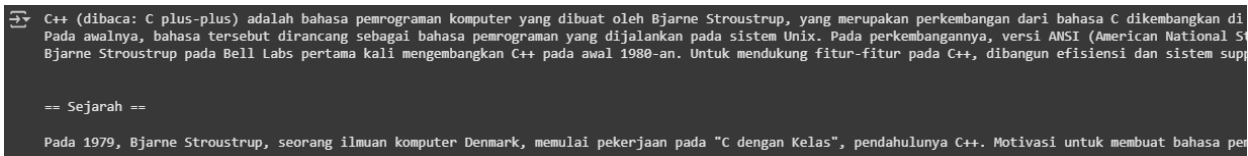
```
# Set bahasa ke Indonesia
wikipedia.set_lang("id")

# Ambil daftar 10 judul artikel dari halaman "Daftar istilah komputer"
page_list = ["C++", "C#", "C (bahasa pemrograman)", "Java (bahasa pemrograman)",
"Ruby", "SQL", "Perl", "LaTeX", "JavaScript", "PHP"]

# Variabel untuk menampung semua konten
all_content = ""

for title in page_list:
    try:
        page = wikipedia.page(title, auto_suggest=False)
        all_content += page.content + "\n\n"
    except wikipedia.exceptions.PageError:
        print(f'Halaman '{title}' tidak ditemukan.")
    except wikipedia.exceptions.DisambiguationError as e:
        print(f'Judul '{title}' ambigu. Pilih salah satu dari: {e.options}')

# Hasil semua konten dari 10 artikel
print(all_content[:2000])
```



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

PRAKTEK TEXT MINING CLUSTERING

NAMA : HARVEST ECCLESIANO CHRIST WALUKOW
NIM : 164231104
MATA KULIAH : DATA MINING II

Saya menggunakan library TextBlob

```
raw = all_content

# stopwords sederhana (indonesia + english common)
INDO_STOP =
set(['dan','atau','yang','dengan','adalah','untuk','di','ke','dari','ini','itu','pada','sebagai',
'oleh','juga','dapat','yg','file','nama','standar','per','lain','sistem','dalam'])

# lowercase
raw = raw.lower()

# textblob object
blob = TextBlob(raw)

# tokenisasi
words = blob.words

# filter stopwords + non-alphabetic
clean_text = " ".join([w for w in words if w.isalpha() and w not in INDO_STOP])

print(clean_text)
```

c dibaca c bahasa pemrograman komputer dibuat bjarne stroustrup merupakan perkembangan bahasa c dikembangkan bell labs dennis ritchie awal tahun bahasa me

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

```
wc = WordCloud(width=800, height=400, collocations=False,
background_color='white', stopwords=INDO_STOP).generate(clean_text)

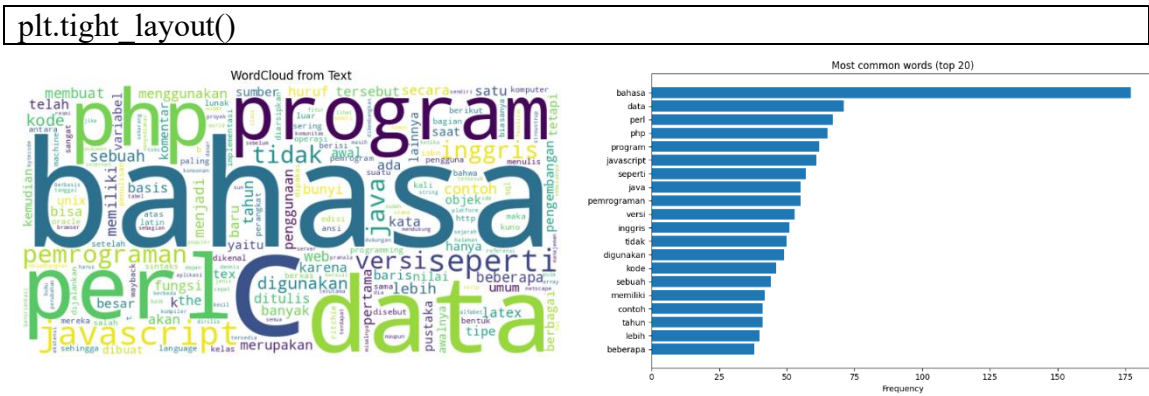
plt.figure(figsize=(10, 5))
plt.imshow(wc, interpolation='bilinear')
plt.axis("off")
plt.title('WordCloud from Text')
plt.show()

cv = CountVectorizer(token_pattern=r"\b\w+\b", stop_words=None)
Xc = cv.fit_transform([clean_text])
counts = Xc.toarray().flatten()
vocab = cv.get_feature_names_out()
counter = Counter(dict(zip(vocab, counts)))
filtered = {w: c for w, c in counter.items() if w not in INDO_STOP and len(w) > 2}
most_common = Counter(filtered).most_common(20)
mc_words, mc_counts = zip(*most_common)

plt.figure(figsize=(10,6))
plt.barh(mc_words[:-1], mc_counts[:-1])
plt.xlabel('Frequency')
plt.title('Most common words (top 20)')
```

PRAKTEK TEXT MINING CLUSTERING

NAMA : HARVEST ECCLESIANO CHRIST WALUKOW
NIM : 164231104
MATA KULIAH : DATA MINING II



Interpretasi: Hasil crawling dari 10 artikel bahasa pemrograman di Wikipedia menunjukkan bahwa teks sangat menekankan pada kata bahasa, data, dan program sebagai konsep inti, dengan nama-nama bahasa populer seperti PHP, Perl, Java, dan JavaScript muncul dominan.

6. Lakukan clustering dengan menggunakan fitur TF-IDF

```
paragraphs = [p.strip() for p in raw.split('\n') if len(p.strip()) > 20]
clean_paragraphs = [clean(p) for p in paragraphs]

vectorizer = TfidfVectorizer(token_pattern=r'\b\w+\b', stop_words=None,
                             max_df=0.9)
X = vectorizer.fit_transform(clean_paragraphs)

# pilih K terbaik dengan silhouette
n_docs = X.shape[0]
max_k = min(8, max(2, n_docs-1))
best_k = 2
best_score = -1
for k in range(2, max_k+1):
    km = KMeans(n_clusters=k, random_state=42, n_init=10)
    labels = km.fit_predict(X)
    try:
        score = silhouette_score(X, labels)
    except Exception:
        score = -1
    if score > best_score:
        best_score = score
        best_k = k

# fit final model
km = KMeans(n_clusters=best_k, random_state=42, n_init=20)
labels = km.fit_predict(X)
print(f'Dipilih k={best_k} dari silhouette score={best_score:.4f}')
```

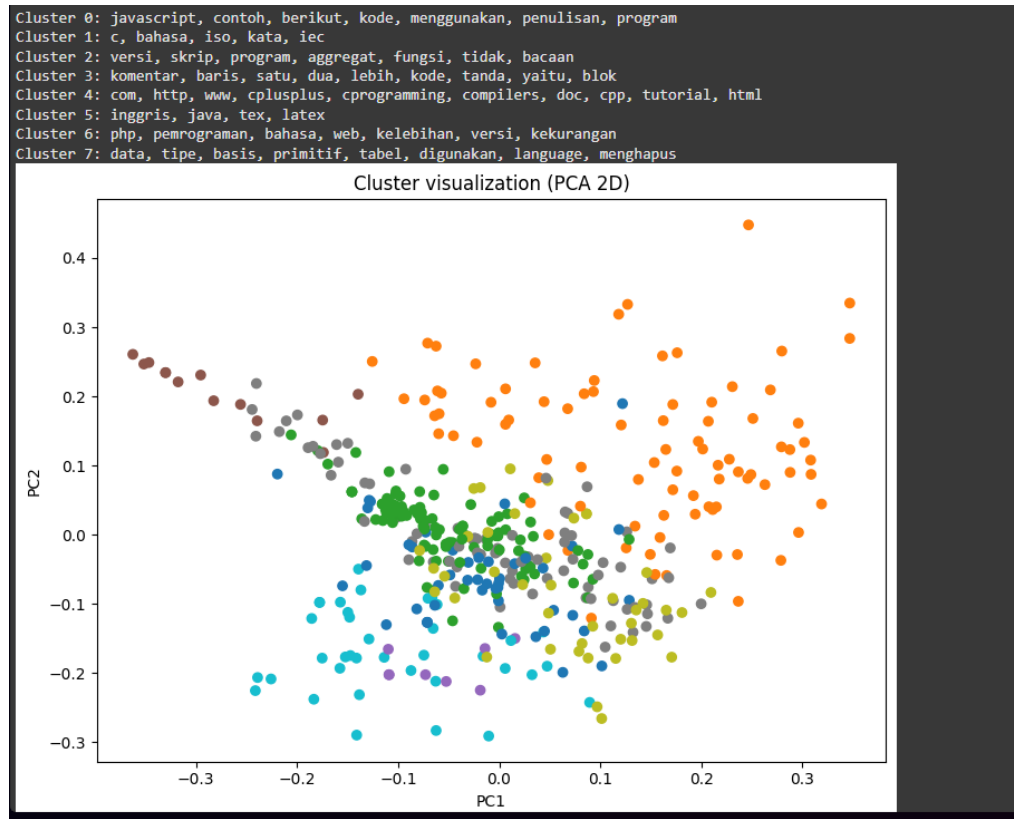
Dipilih k=8 dari silhouette score=0.0233

NAMA : HARVEST ECCLESIANO CHRIST WALUKOW
NIM : 164231104
MATA KULIAH : DATA MINING II

7. Buat visualisasi clusternya dan lakukan interpretasi terhadap hasil tersebut.

```
pca = PCA(n_components=2, random_state=42)
X2 = pca.fit_transform(X.toarray())
plt.figure(figsize=(8,6))
plt.scatter(X2[:,0], X2[:,1], c=labels, cmap='tab10')
plt.title('Cluster visualization (PCA 2D)')
plt.xlabel('PC1')
plt.ylabel('PC2')
plt.tight_layout()

# Cetak ringkasan tiap cluster
order_centroids = km.cluster_centers_.argsort()[:, :-1]
terms = vectorizer.get_feature_names_out()
for i in range(best_k):
    top_terms = [terms[ind] for ind in order_centroids[i, :10] if terms[ind] not in
INDO_STOP]
    print(f'Cluster {i}:', ', '.join(top_terms[:10]))
```



Interpretasi:

- Sebelumnya jumlah cluster (k) = 8 dipilih menggunakan silhouette score, tapi nilai skornya cukup rendah (0.0233).
- Silhouette score yang rendah artinya pemisahan antar cluster kurang jelas dan ada cukup banyak overlap antar kelompok. Dengan kata lain, meskipun ada pola tematik (misalnya cluster tentang PHP, JavaScript, C, data, komentar, dll.), batas antar cluster

PRAKTEK TEXT MINING CLUSTERING

NAMA : HARVEST ECCLESIANO CHRIST WALUKOW
NIM : 164231104
MATA KULIAH : DATA MINING II

tidak terlalu tegas karena banyak kata bersifat umum dan muncul di beberapa konteks sekaligus.

- Namun, meski kualitas clustering tidak optimal, hasilnya tetap membantu memberi gambaran bahwa teks bisa dikelompokkan menjadi beberapa tema besar: bahasa spesifik (C, PHP, Java, JavaScript, C++), standar/dokumen teknis, konsep kode (komentar, blok, skrip), dan konsep data (tipe, tabel, basis).

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

```
print('Silhouette score (final):', silhouette_score(X, labels))
```

```
➦ Silhouette score (final): 0.023282099214903505
```

Validasi pertama memilih k=8 dengan silhouette score rendah (0.0233), dan validasi kedua mengonfirmasi nilai yang sama. Artinya, meski jumlah cluster sudah divalidasi, kualitas pemisahannya lemah.