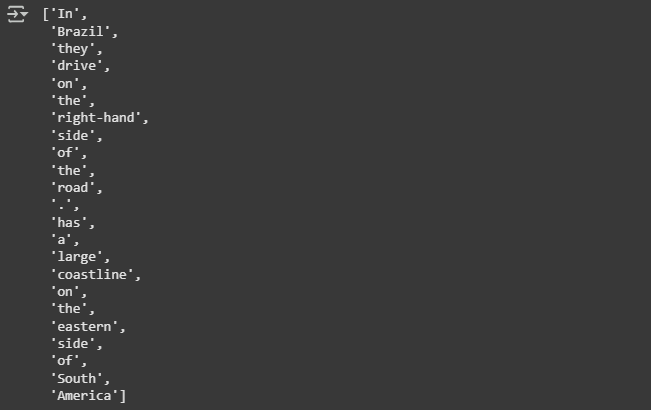
Saya melakukan import pada semua package/library yang dibutuhkan pada praktikum ini:

|  |
| --- |
| 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**

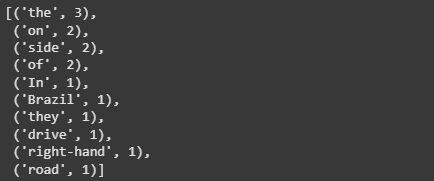
|  |
| --- |
| text = "In Brazil they drive on the right-hand side of the road. has a large coastline on the eastern side of South America"  token = word\_tokenize(text)  token |



|  |
| --- |
| fdist = FreqDist(token)  fdist |



|  |
| --- |
| fdist = FreqDist(token)  fdist1 = fdist.most\_common(10)  fdist1 |



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

* **Stopwords**

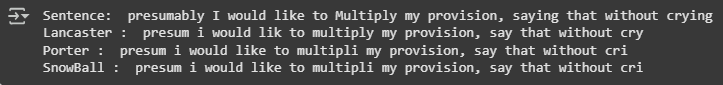
|  |
| --- |
| a = set(stopwords.words('english'))  text = "Cristiano Ronaldo was born on February 5, 1985, in Funchal, Madeira, Portugal."  text1 = word\_tokenize(text.lower())  stopwords\_removed = [x for x in text1 if x not in a]  print(stopwords\_removed) |



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())) |



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.

* **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')) |



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=', ') |



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) |

****

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")) |

****

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**

|  |
| --- |
| 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.pckl"  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'[^a-zA-Z0-9]', ' ', s) # menghapus simbol  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!') |

****

Analisis: Membersihkan 20newsgroup.pckl 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] |

****

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] |



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] |

****

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 |



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=', ') |



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=', ') |



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.

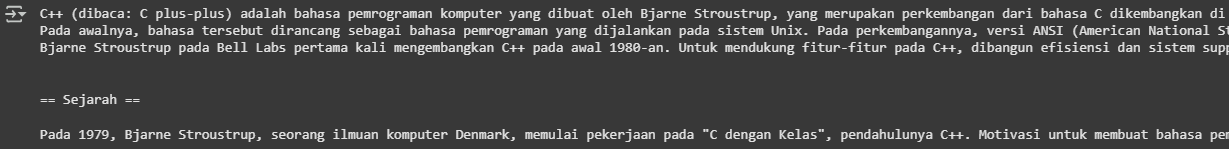
1. Jelaskan perbedaan hasil dari Preprocessing menggunakan NLTK, TextBlob dan Sastrawi dan berikan contohnya.

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.

1. 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]) |



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

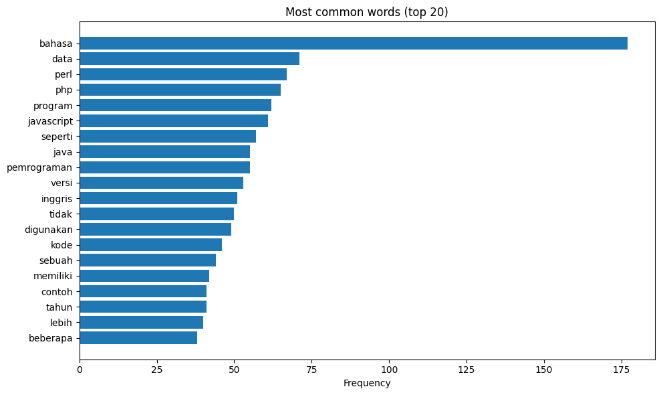
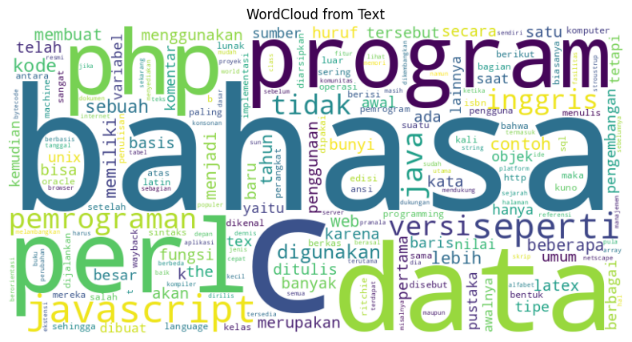
Saya menggunakan library TextBlob

|  |
| --- |
| 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) |



1. 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)')  plt.tight\_layout() |



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.

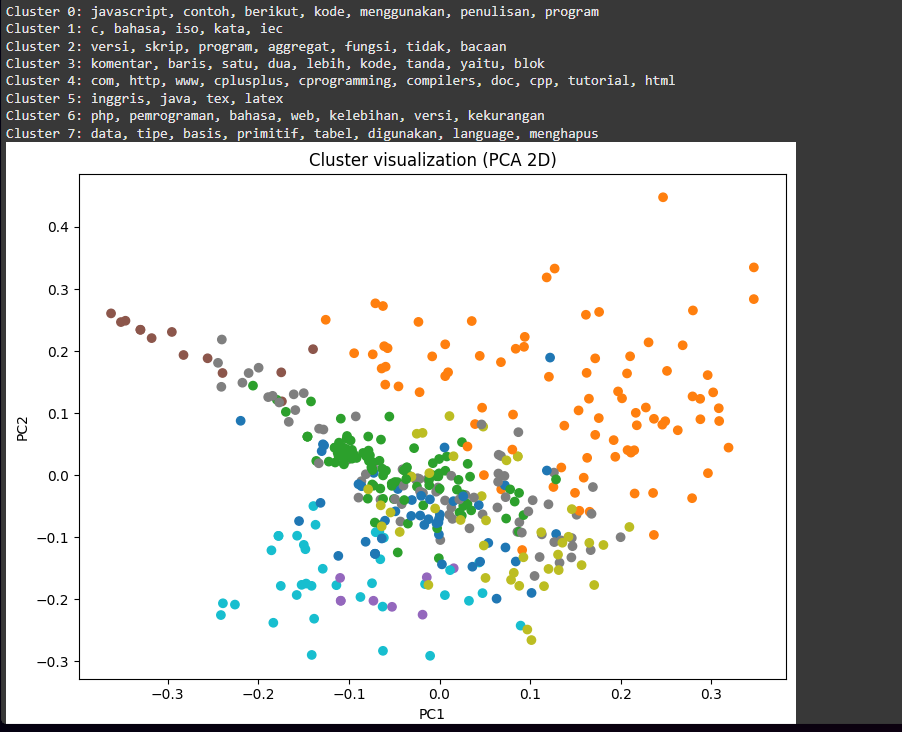
1. Lakukan clustering dengan menggunakan fitur TF-IDF

|  |
| --- |
| 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}') |



1. 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 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).

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

|  |
| --- |
| print('Silhouette score (final):', silhouette\_score(X, labels)) |



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.