



Project → Kelompok 11



Clustering Lowongan Pekerjaan Berdasarkan
Keterampilan yang Dibutuhkan dan Tingkat
Pengalaman dengan web kitalulus.com





Anggota kelompok



- Shinta Usaila F. (23031554160)
- Ikhrima Atusifah. (23031554181)
- Arina Tri Yuni W. T. (23031554203)





Latar Belakang



Clustering lowongan pekerjaan berdasarkan keterampilan dan tingkat pengalaman bertujuan untuk mengelompokkan data pekerjaan agar pencari kerja dan perusahaan lebih mudah menemukan kecocokan. Metode seperti K-Means dan DBSCAN sering digunakan dalam proses ini. K-Means efektif untuk dataset terstruktur dengan jumlah cluster yang diketahui, sementara DBSCAN lebih fleksibel untuk data kompleks dan dapat mengidentifikasi outlier. Dengan menggabungkan kedua metode ini, pengelompokan dapat memberikan wawasan yang lebih dalam, membantu pencari kerja menemukan pekerjaan sesuai kompetensi, dan mendukung perusahaan dalam menyusun strategi perekrutan yang lebih tepat sasaran.

Selanjutnya!



Selanjutnya!

Tujuan

- Mempermudah pencari kerja menemukan lowongan yang sesuai keterampilan dan pengalaman mereka.
- Memudahkan perusahaan mengelompokkan jenis pekerjaan sesuai kebutuhan.
- Memanfaatkan metode K-Means dan DBSCAN untuk mengelompokkan data pekerjaan secara mudah dan akurat.

Manfaat

- Membantu pencari kerja menghemat waktu dalam mencari pekerjaan yang sesuai.
- Mengurangi risiko kesalahan dalam mencocokkan kandidat dengan posisi yang dibutuhkan.
- Memberikan gambaran yang jelas tentang tren keterampilan dan pengalaman yang banyak dicari di pasar kerja.

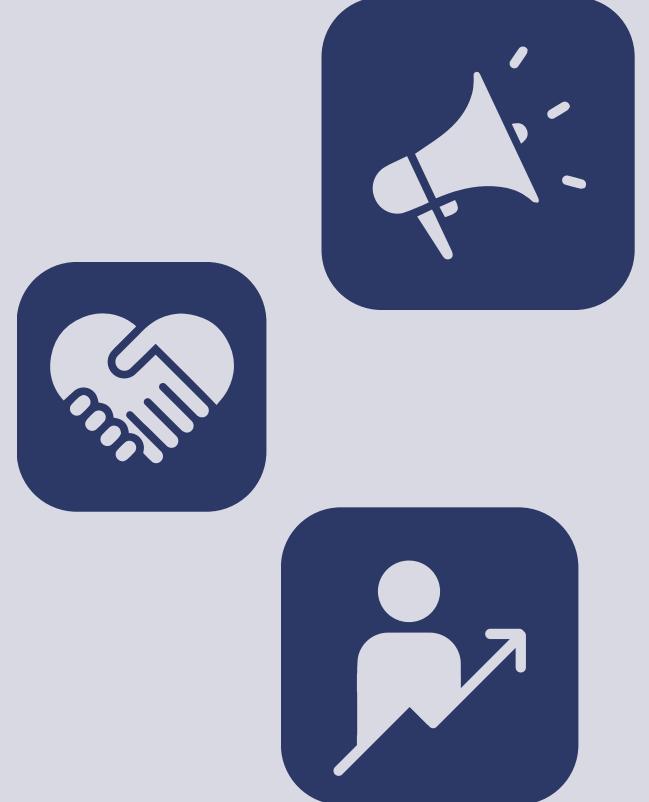




DataSet

Dataset yang digunakan dalam proyek ini yaitu data dari web yang akan di scraping secara manual berdasarkan posisi pekerjaan, Perusahaan yang mencari lowongan, lokasi, link, pengalaman serta keterampilan.

[https://www.kitalulus.com/lowongan?
job_functions=&sort_by=isHighlighted](https://www.kitalulus.com/lowongan?job_functions=&sort_by=isHighlighted)





ALUR DAN TAHAPAN PENGERJAAN

1. Tahap Scraping

```
import requests as rs

page = rs.get('https://www.kitalulus.com/lowongan?job_functions=&sort_by=isHighlighted')
page|
```

<Response [200]>

Langkah awal yakni melakukan Kode modul requests untuk melakukan permintaan HTTP GET ke URL https://www.kitalulus.com/lowongan?job_functions=&sort_by=isHighlighted. Permintaan ini bertujuan untuk mengakses data lowongan kerja dengan pengurutan berdasarkan parameter isHighlighted. Respons yang diterima disimpan dalam variabel page, dengan kode status HTTP 200 yang menunjukkan bahwa permintaan berhasil.

Selanjutnya!



Kampus
Merdeka
INDONESIA JAYA

UNESB
PTNBH
SATULANGKAHDIDEPAH

Selanjutnya!



```
from bs4 import BeautifulSoup
import csv
import time

# URL target
url = 'https://www.kitalulus.com/lowongan?job_functions=&sort_by=isHighlighted'

# Menyimpan hasil ke file CSV
csv_filename = 'job_listings.csv'
with open(csv_filename, mode='w', newline='', encoding='utf-8') as file:
    writer = csv.writer(file)
    # Menulis header
    writer.writerow(['posisi', 'tautan', 'perusahaan', 'lokasi', 'pengalaman', 'keterampilan'])

    job_data = []

    page = 1
    while len(job_data) < 200:
        # Mendapatkan halaman web
        response = requests.get(url + f"&page={page}")

        # Memeriksa apakah permintaan berhasil
        if response.status_code == 200:
            # Menggunakan BeautifulSoup untuk parsing HTML
            soup = BeautifulSoup(response.content, 'html.parser')

            # Mengambil data pekerjaan (disesuaikan dengan struktur halaman)
            job_elements = soup.find_all('div', class_='CardRectangleStyled__Container-sc-1lom4v1-0 iwlPnJ') # Sesuaikan dengan elemen asli
```

Kode ini menggunakan modul `requests`, `BeautifulSoup`, dan `csv` untuk melakukan web scraping dari situs `kitalulus.com` dan menyimpan data lowongan kerja ke file `job_listings.csv`. Skrip mengambil data seperti posisi, tautan, perusahaan, lokasi, pengalaman, dan keterampilan. Data diambil dari beberapa halaman dengan menambahkan parameter `page` pada URL. HTML halaman diproses dengan `BeautifulSoup`, dan elemen data diekstrak sesuai struktur situs. Proses ini diulang hingga terkumpul 200 entri, lalu data disimpan ke file CSV.





Kampus
Merdeka
INDONESIA JAYA

UNESR
PTNBH
SATULANGKAHDIDEPAH

Selanjutnya!



```
for job in job_elements:
    posisi = job.find('h3').get_text().strip() # Judul pekerjaan
    perusahaan = job.find('p').get_text().strip()
    tautan = 'https://www.kitalulus.com' + job.find('a')['href'] # Link ke pekerjaan

    job_response = requests.get(tautan)
    if job_response.status_code == 200:
        job_soup = BeautifulSoup(job_response.content, 'html.parser')

        # Mengambil lokasi
        lokasi_element = job_soup.find('p', class_='TextStyled__Text-sc-18vo2dc-0 kaIrvs')
        lokasi = lokasi_element.get_text().strip() if lokasi_element else 'Tidak tersedia'

        pengalaman_elements = job_soup.find_all('p', class_='TextStyled__Text-sc-18vo2dc-0 kaIrvs')

        # Inisialisasi default
        pengalaman = 'Tidak tersedia'

        # Periksa apakah elemen ke-9 ada
        if len(pengalaman_elements) > 9:
            if "pengalaman" in pengalaman_elements[9].get_text().lower():
                # Jika elemen ke-9 berisi kata "pengalaman", ambil elemen ke-10
                pengalaman = pengalaman_elements[10].get_text().strip() if len(pengalaman_elements) > 10 else 'Tidak tersedia'
            else:
                # Jika tidak, gunakan elemen ke-9
                pengalaman = pengalaman_elements[9].get_text().strip()

        keterampilan = job_soup.find('div', class_='VacancyDescriptionStyled__Wrapper-sc-13uwryz-2 cDTmUf').text
```

Kode ini mengekstrak data pekerjaan dari situs web dalam loop. Setiap elemen pekerjaan diproses untuk mengambil judul, perusahaan, dan tautan dari halaman utama. Tautan tersebut digunakan untuk mengakses halaman detail, lalu data lokasi, pengalaman, dan keterampilan diekstrak. Jika data tidak ditemukan, diberi nilai "Tidak tersedia." Semua data kemudian disiapkan untuk disimpan ke file CSV.





2. Tahap preprocessing

Kode ini memproses teks dengan langkah-langkah utama: mengunduh data NLTK seperti 'wordnet' dan 'stopwords', menginisialisasi PorterStemmer dan WordNetLemmatizer, serta membuat daftar stopwords gabungan (Indonesia, Inggris, dan tambahan khusus). Fungsi preprocessing membersihkan teks dengan memisahkan kata tanpa spasi, menghapus karakter non-alfabet, mengubah teks ke huruf kecil, menghapus stopwords, dan menerapkan stemming serta lemmatization. Namun, ada kesalahan sintaks seperti regex dan fungsi yang perlu diperbaiki agar kode bisa dijalankan.

```
import csv
import re
import nltk
from nltk.stem import PorterStemmer
from nltk.stem import WordNetLemmatizer
from nltk.corpus import stopwords

# Mengunduh data yang diperlukan dari nltk
nltk.download('wordnet')
nltk.download('cmw-1.4')
nltk.download('stopwords')

# Inisialisasi Stemmer dan Lemmatizer
stemmer = PorterStemmer()
lemmatizer = WordNetLemmatizer()

# Stopwords bahasa Indonesia dan Inggris
stop_words = set(stopwords.words('indonesian')) | set(stopwords.words('english'))
# Tambahkan kata-kata tambahan yang ingin dihapus
additional_stop_words = {'Tidak'}
stop_words.update(additional_stop_words)

# Fungsi preprocessing untuk pengalaman dan keterampilan
def preprocess_text(text):
    # Tambahkan spasi antara kata yang tidak memiliki spasi (misalnya "2Tahun" menjadi "2 Tahun")
    text = re.sub(r'(?<=[a-zA-Z])(?=\d)|(?<=\d)(?=[a-zA-Z])|(?<=[a-z])(?=[A-Z])', ' ', text)
    # Hilangkan karakter non-alfabet kecuali angka
    text = re.sub(r'[^a-zA-Z0-9\s]', '', text)
    # Ubah ke huruf kecil
    text = text.lower()
    # Tokenisasi teks
    words = text.split()
    # Hilangkan stop words
    words = [word for word in words if word not in stop_words]
    # Lakukan stemming dan lemmatization pada setiap kata
    words = [stemmer.stem(word) for word in words]
    words = [lemmatizer.lemmatize(word) for word in words]
    # Gabungkan kembali kata-kata yang relevan
    return ' '.join(words)
```





```
# Nama file CSV awal dan hasil
input_csv = 'job_listings.csv' # File CSV yang sudah ada
output_csv = 'job_listings_preprocessed.csv' # File CSV dengan data yang diproses

# Membaca data dari file CSV awal
with open(input_csv, mode='r', encoding='utf-8') as infile, open(output_csv, mode='w', newline='', encoding='utf-8') as outfile:
    reader = csv.DictReader(infile)
    fieldnames = reader.fieldnames # Ambil header dari file awal
    writer = csv.DictWriter(outfile, fieldnames=fieldnames)

    # Menulis header ke file output
    writer.writeheader()

    # Memproses setiap baris
    for row in reader:
        # Preprocess pengalaman
        if 'pengalaman' in row and row['pengalaman']:
            row['pengalaman'] = preprocess_text(row['pengalaman'])
        else:
            row['pengalaman'] = 'Tidak tersedia'

        # Preprocess keterampilan
        if 'keterampilan' in row and row['keterampilan']:
            row['keterampilan'] = preprocess_text(row['keterampilan'])
        else:
            row['keterampilan'] = 'Tidak tersedia'

        # Menulis baris yang telah diproses ke file baru
        writer.writerow(row)

print(f"Data hasil preprocessing berhasil disimpan ke {job_listings_preprocessed.csv}")
```

Kode membaca file CSV `job_listings.csv`, memproses kolom pengalaman dan keterampilan dengan fungsi `preprocess_text`, lalu menyimpan hasilnya ke `Job_listings_preprocessed.csv` . Jika data tidak ada, diisi "Tidak tersedia." Koreksi sintaks diperlukan agar berfungsi.



3. Tahap Feature Engineering



```
● import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from gensim.models import Word2Vec, FastText

# Membaca data CSV hasil preprocessing
input_csv = 'job_listings_preprocessed.csv'
df = pd.read_csv(input_csv)

# Kolom target untuk feature engineering
pengalaman_column = 'pengalaman'
keterampilan_column = 'keterampilan'

# Mengisi nilai kosong dengan string kosong untuk kolom pengalaman dan keterampilan
df[pengalaman_column] = df[pengalaman_column].fillna('')
df[keterampilan_column] = df[keterampilan_column].fillna('')

# 1. Bag of Words (BoW) untuk Pengalaman dan Keterampilan
def extract_bow(texts):
    vectorizer = CountVectorizer()
    bow_matrix = vectorizer.fit_transform(texts)
    bow_df = pd.DataFrame(bow_matrix.toarray(), columns=vectorizer.get_feature_names_out())
    return bow_df

# 2. TF-IDF untuk Pengalaman dan Keterampilan
def extract_tfidf(texts):
    vectorizer = TfidfVectorizer()
    tfidf_matrix = vectorizer.fit_transform(texts)
    tfidf_df = pd.DataFrame(tfidf_matrix.toarray(), columns=vectorizer.get_feature_names_out())
    return tfidf_df

# 3. Word2Vec untuk Pengalaman dan Keterampilan
def extract_word2vec(texts):
    tokenized_texts = [text.split() for text in texts]
    w2v_model = Word2Vec(sentences=tokenized_texts, vector_size=100, window=5, min_count=1, workers=4)
    w2v_features = [w2v_model.wv[text].mean(axis=0) if len(text) > 0 else [0]*100 for text in tokenized_texts]
    w2v_df = pd.DataFrame(w2v_features)
    return w2v_df

# 4. FastText untuk Pengalaman dan Keterampilan
def extract_fasttext(texts):
    tokenized_texts = [text.split() for text in texts]
    ft_model = FastText(sentences=tokenized_texts, vector_size=100, window=5, min_count=1, workers=4)
    ft_features = [ft_model.wv[text].mean(axis=0) if len(text) > 0 else [0]*100 for text in tokenized_texts]
    ft_df = pd.DataFrame(ft_features)
    return ft_df
```

Kode ini mengekstrak fitur teks dari data pengalaman dan keterampilan menggunakan teknik NLP seperti Bag of Words (BoW), TF-IDF, Word2Vec, dan FastText. Data diimpor sebagai DataFrame dan kolom yang kosong diisi dengan nilai default. BoW menghitung frekuensi kata menggunakan CountVectorizer, sementara TF-IDF memberi bobot kata berdasarkan frekuensi relatif. Word2Vec dan FastText menghasilkan vektor numerik untuk setiap dokumen, dengan FastText lebih unggul dalam menangkap informasi sub-kata. Representasi ini digunakan untuk analisis atau tugas prediksi.



Kode ini menghitung fitur teks untuk kolom pengalaman dan keterampilan menggunakan Bag of Words (BoW), TF-IDF, Word2Vec, dan FastText. Untuk setiap kolom, BoW merepresentasikan teks berdasarkan frekuensi kata, TF-IDF menghitung bobot relatif kata, sementara Word2Vec dan FastText menghasilkan vektor numerik berdasarkan hubungan semantik. Hasil dari setiap teknik kemudian ditampilkan untuk analisis lebih lanjut.



www.sadaa23A.com

Selanjutnya!

```
# Proses Feature Engineering untuk Pengalaman
print("Menghitung Bag of Words (BoW) untuk Pengalaman...")
bow_pengalaman = extract_bow(df[pengalaman_column])

print("Menghitung TF-IDF untuk Pengalaman...")
tfidf_pengalaman = extract_tfidf(df[pengalaman_column])

print("Menghitung Word2Vec untuk Pengalaman...")
word2vec_pengalaman = extract_word2vec(df[pengalaman_column])

print("Menghitung FastText untuk Pengalaman...")
fasttext_pengalaman = extract_fasttext(df[pengalaman_column])

# Proses Feature Engineering untuk keterampilan
print("Menghitung Bag of Words (BoW) untuk Keterampilan...")
bow_keterampilan = extract_bow(df[keterampilan_column])

print("Menghitung TF-IDF untuk Keterampilan...")
tfidf_keterampilan = extract_tfidf(df[keterampilan_column])

print("Menghitung Word2Vec untuk Keterampilan...")
word2vec_keterampilan = extract_word2vec(df[keterampilan_column])

print("Menghitung FastText untuk Keterampilan...")
fasttext_keterampilan = extract_fasttext(df[keterampilan_column])

# Menampilkan hasil untuk pengalaman dan keterampilan
print("\nHasil Feature Engineering untuk Pengalaman:")
print("Bag of Words (BoW) Pengalaman:\n", bow_pengalaman.head())
print("TF-IDF Pengalaman:\n", tfidf_pengalaman.head())
print("Word2Vec Pengalaman:\n", word2vec_pengalaman.head())
print("FastText Pengalaman:\n", fasttext_pengalaman.head())

print("\nHasil Feature Engineering untuk Keterampilan:")
print("Bag of Words (BoW) Keterampilan:\n", bow_keterampilan.head())
print("TF-IDF Keterampilan:\n", tfidf_keterampilan.head())
print("Word2Vec Keterampilan:\n", word2vec_keterampilan.head())
print("FastText Keterampilan:\n", fasttext_keterampilan.head())
```



4. Tahap Clustering

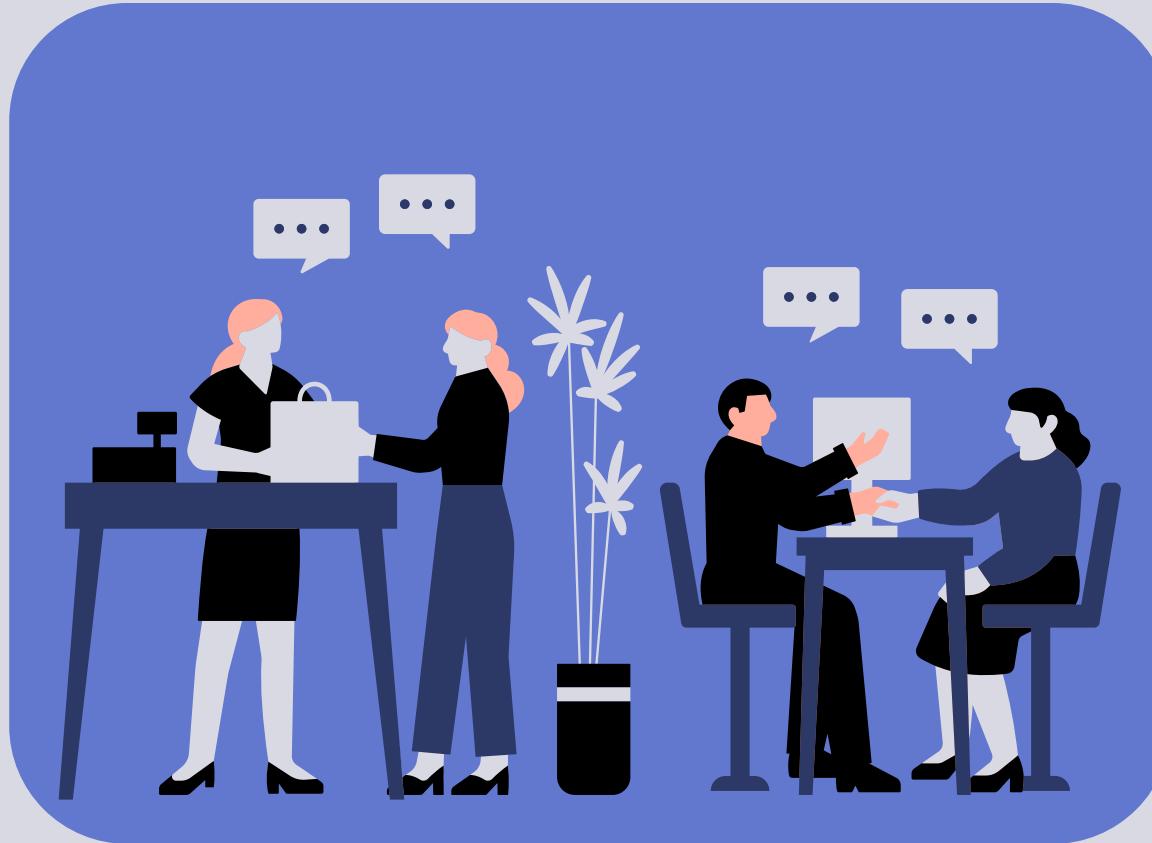
Proyek ini menggunakan metode clustering dengan algoritma K-means dan DBSCAN untuk mengelompokkan lowongan pekerjaan berdasarkan keterampilan dan tingkat pengalaman. Sebelum membuat code clustering install terlebih dulu library fasttext agar bisa melanjutkan ke code selanjutnya.

```
[ ] !pip install fasttext
```

Perintah `pip install fasttext` digunakan untuk menginstal pustaka FastText, FastText adalah sebuah model pembelajaran mesin yang digunakan untuk memproses teks, terutama dalam tugas-tugas pemrosesan bahasa alami (NLP). FastText dapat digunakan untuk berbagai aplikasi seperti klasifikasi teks, analisis sentimen, dan representasi kata (word embeddings).

a. Menentukan silhouette score dan Davies-Bouldin Index

Sebelum menyimpan hasil clustering dan memvisualisasikannya, akan dihitung terlebih dahulu silhouette score dan Davies-Bouldin Index untuk menentukan hasil yang terbaik untuk clustering.





Kampus
Merdeka
INDONESIA JAYA

UNESR
PTNBH
#SATULANGKAHDIDEPAK



Kode ini mulai dengan memproses data menggunakan pandas dan numpy, kemudian mengekstrak fitur teks seperti BoW, TF-IDF, Word2Vec, dan FastText dari kolom combined_features. Fitur yang sudah diekstrak kemudian di-scale menggunakan StandardScaler. Setelah itu, dua metode clustering diterapkan: KMeans dan DBSCAN. Pada KMeans, jumlah klaster ditentukan dan dihitung nilai silhouette score untuk mengevaluasi hasil clustering. Sedangkan pada DBSCAN, clustering dilakukan dengan parameter eps dan min_samples, dan juga dihitung silhouette score untuk menilai kualitas clustering.



```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.cluster import KMeans, DBSCAN
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import silhouette_score
from sklearn.decomposition import PCA
from gensim.models import Word2Vec, FastText

df['combined_features'] = df['pengalaman'] + ' ' + df['keterampilan']

# Ekstraksi fitur untuk kolom gabungan
bow_combined = extract_bow(df['combined_features'])
tfidf_combined = extract_tfidf(df['combined_features'])
word2vec_combined = extract_word2vec(df['combined_features'])
fasttext_combined = extract_fasttext(df['combined_features'])

# Fungsi untuk clustering K-means
def kmeans_clustering(features, n_clusters=3):
    scaler = StandardScaler()
    scaled_features = scaler.fit_transform(features)
    kmeans_model = KMeans(n_clusters=n_clusters, random_state=42)
    cluster_labels = kmeans_model.fit_predict(scaled_features)
    silhouette_avg = silhouette_score(scaled_features, cluster_labels)
    return cluster_labels, silhouette_avg

# Fungsi untuk clustering DBSCAN
def dbscan_clustering(features, eps=0.5, min_samples=5):
    scaler = StandardScaler()
    scaled_features = scaler.fit_transform(features)
    dbscan_model = DBSCAN(eps=eps, min_samples=min_samples)
    cluster_labels = dbscan_model.fit_predict(scaled_features)
    unique_labels = list(set(cluster_labels))
    cluster_labels = [unique_labels.index(label) if label != -1 else -1 for label in cluster_labels]
    if len(set(cluster_labels)) > 1:
        silhouette_avg = silhouette_score(scaled_features, cluster_labels)
    else:
        silhouette_avg = -1 # Jika hanya ada satu cluster, silhouette score tidak dapat dihitung
    return cluster_labels, silhouette_avg
```



```
D # Clustering untuk fitur gabungan
kmeans_tfidf_combined_labels, kmeans_tfidf_combined_silhouette = kmeans_clustering(tfidf_combined, 3)
dbSCAN_tfidf_combined_labels, dbSCAN_tfidf_combined_silhouette = dbSCAN_clustering(tfidf_combined, eps=0.8, min_samples=5)

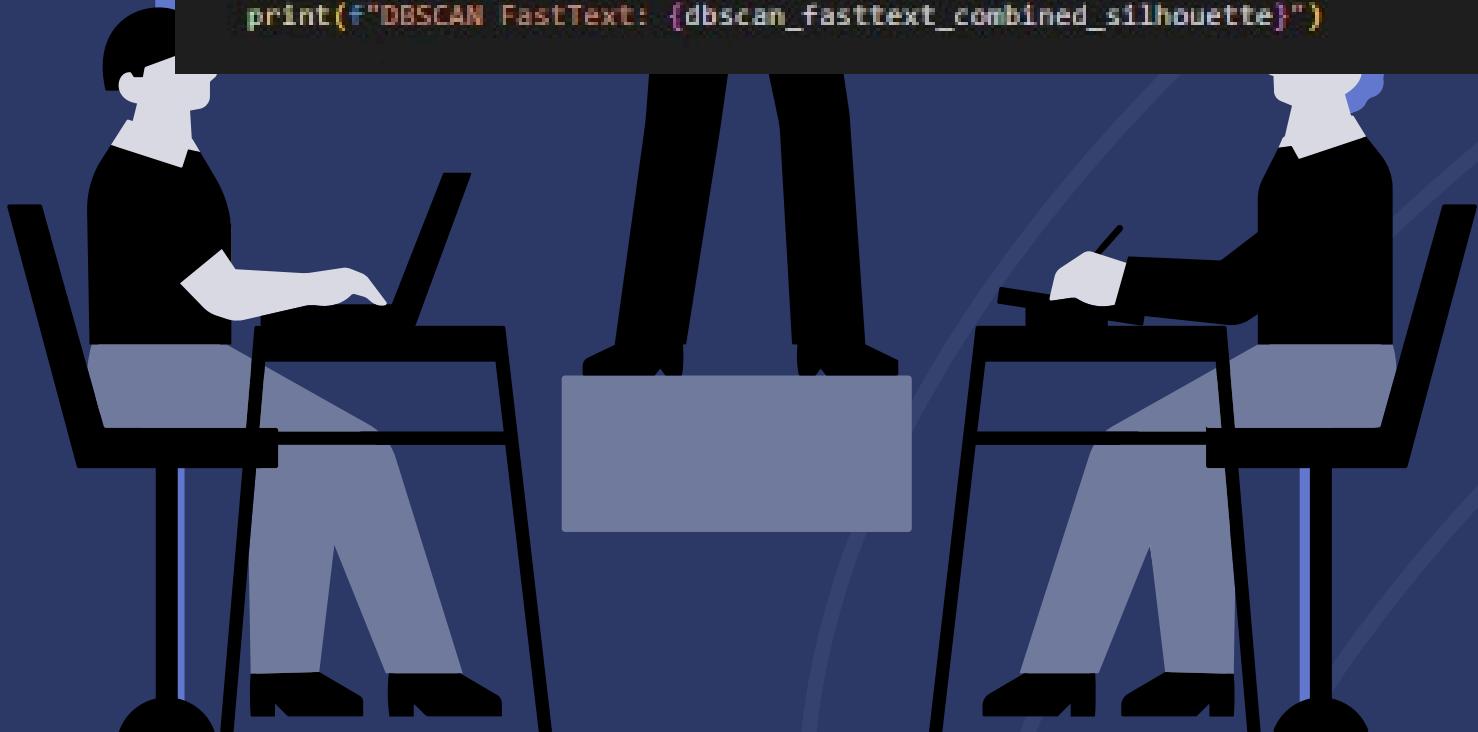
kmeans_bow_combined_labels, kmeans_bow_combined_silhouette = kmeans_clustering(bow_combined, 3)
dbSCAN_bow_combined_labels, dbSCAN_bow_combined_silhouette = dbSCAN_clustering(bow_combined, eps=0.8, min_samples=5)

kmeans_word2vec_combined_labels, kmeans_word2vec_combined_silhouette = kmeans_clustering(word2vec_combined, 3)
dbSCAN_word2vec_combined_labels, dbSCAN_word2vec_combined_silhouette = dbSCAN_clustering(word2vec_combined, eps=0.8, min_samples=5)

kmeans_fasttext_combined_labels, kmeans_fasttext_combined_silhouette = kmeans_clustering(fasttext_combined, 3)
dbSCAN_fasttext_combined_labels, dbSCAN_fasttext_combined_silhouette = dbSCAN_clustering(fasttext_combined, eps=0.8, min_samples=5)

# Menampilkan skor Silhouette
print("\nSilhouette Score untuk fitur gabungan (KMeans):")
print(f"KMeans TF-IDF: {kmeans_tfidf_combined_silhouette}")
print(f"KMeans BoW: {kmeans_bow_combined_silhouette}")
print(f"KMeans Word2Vec: {kmeans_word2vec_combined_silhouette}")
print(f"KMeans FastText: {kmeans_fasttext_combined_silhouette}")

print("\nSilhouette Score untuk fitur gabungan (DBSCAN):")
print(f"DBSCAN TF-IDF: {dbSCAN_tfidf_combined_silhouette}")
print(f"DBSCAN BoW: {dbSCAN_bow_combined_silhouette}")
print(f"DBSCAN Word2Vec: {dbSCAN_word2vec_combined_silhouette}")
print(f"DBSCAN FastText: {dbSCAN_fasttext_combined_silhouette}")
```



Kode ini melakukan clustering pada fitur gabungan yang diekstraksi menggunakan teknik seperti TF-IDF, BoW, Word2Vec, dan FastText dengan menggunakan metode KMeans dan DBSCAN. Setiap teknik ekstraksi fitur diuji dengan KMeans untuk 3 klaster, sementara DBSCAN diuji dengan parameter eps dan min_samples. Setelah itu, silhouette score dihitung untuk mengevaluasi kualitas setiap klaster. Hasilnya, skor silhouette untuk masing-masing teknik dan metode clustering ditampilkan untuk analisis lebih lanjut.





Selain menggunakan silhouette score evaluasi hasil clustering juga dihitung menggunakan Davies-Bouldin Index untuk membandingkan hasil perhitungan.

Penjelasan tentang code ini hampir sama dengan code silhouette score, hanya berbeda di defini fungsinya saja. jika silhouette score maka fungsi yang didefinisikan adalah fungsi silhouette score, sedangkan jika Davies-Bouldin Index maka menggunakan fungsi Davies-Bouldin Index.

```
from sklearn.metrics import davies_bouldin_score
from sklearn.decomposition import PCA

# Fungsi untuk clustering K-means
def kmeans_clustering(Features, n_clusters=3):
    scaler = StandardScaler()
    scaled_features = scaler.fit_transform(Features)
    kmeans_model = KMeans(n_clusters=n_clusters, random_state=42)
    cluster_labels = kmeans_model.fit_predict(scaled_features)
    davies_bouldin_avg = davies_bouldin_score(scaled_features, cluster_labels) if len(set(cluster_labels)) > 1 else -1
    return cluster_labels, davies_bouldin_avg

# Fungsi untuk clustering DBSCAN
def dbscan_clustering(Features, eps=0.3, min_samples=3):
    scaler = StandardScaler()
    scaled_features = scaler.fit_transform(Features)
    dbscan_model = DBSCAN(eps=eps, min_samples=min_samples)
    cluster_labels = dbscan_model.fit_predict(scaled_features)
    unique_labels = list(set(cluster_labels))
    cluster_labels = [unique_labels.index(label) if label != -1 else -1 for label in cluster_labels]
    if len(set(cluster_labels)) > 1:
        davies_bouldin_avg = davies_bouldin_score(scaled_features, cluster_labels)
    else:
        davies_bouldin_avg = -1 # Jika hanya ada satu cluster, Davies-Bouldin score tidak dapat dihitung
    return cluster_labels, davies_bouldin_avg

data = pd.read_csv('/content/job_listings_preprocessed.csv')
df = pd.DataFrame(data)

df['combined_features'] = df['pengalaman'] + ' ' + df['keterampilan']

# Clustering untuk fitur gabungan
kmeans_tf_idf_combined_labels, kmeans_tf_idf_combined_davies_bouldin = kmeans_clustering(tfidf_combined, 3)
dbscan_tf_idf_combined_labels, dbscan_tf_idf_combined_davies_bouldin = dbscan_clustering(tfidf_combined, eps=0.3, min_samples=3)

kmeans_bow_combined_labels, kmeans_bow_combined_davies_bouldin = kmeans_clustering(bow_combined, 3)
dbscan_bow_combined_labels, dbscan_bow_combined_davies_bouldin = dbscan_clustering(bow_combined, eps=0.3, min_samples=3)

kmeans_word2vec_combined_labels, kmeans_word2vec_combined_davies_bouldin = kmeans_clustering(word2vec_combined, 3)
dbscan_word2vec_combined_labels, dbscan_word2vec_combined_davies_bouldin = dbscan_clustering(word2vec_combined, eps=0.3, min_samples=3)

kmeans_fasttext_combined_labels, kmeans_fasttext_combined_davies_bouldin = kmeans_clustering(fasttext_combined, 3)
dbscan_fasttext_combined_labels, dbscan_fasttext_combined_davies_bouldin = dbscan_clustering(fasttext_combined, eps=0.3, min_samples=3)

# Menampilkan skor Davies-Bouldin
print("\nDavies-Bouldin Index untuk fitur gabungan (KMeans):")
print(f"KMeans TF-IDF: ({kmeans_tf_idf_combined_davies_bouldin})")
print(f"KMeans BoW: ({kmeans_bow_combined_davies_bouldin})")
print(f"KMeans Word2Vec: ({kmeans_word2vec_combined_davies_bouldin})")
print(f"KMeans FastText: ({kmeans_fasttext_combined_davies_bouldin})")

print("\nDavies-Bouldin Index untuk fitur gabungan (DBSCAN):")
print(f"DBSCAN TF-IDF: ({dbscan_tf_idf_combined_davies_bouldin})")
print(f"DBSCAN BoW: ({dbscan_bow_combined_davies_bouldin})")
print(f"DBSCAN Word2Vec: ({dbscan_word2vec_combined_davies_bouldin})")
print(f"DBSCAN FastText: ({dbscan_fasttext_combined_davies_bouldin})")
```



5. Visualisasi Hasil Clustering

```
► # Fungsi untuk menampilkan hasil clustering menggunakan diagram batang
def plot_cluster_distribution(labels, title):
    unique, counts = np.unique(labels, return_counts=True)
    plt.figure(figsize=(8, 4))
    plt.bar(unique, counts, tick_label=unique)
    plt.xlabel('Cluster Labels')
    plt.ylabel('Number of Samples')
    plt.title(title)
    plt.show()

# Visualisasi Hasil Clustering
print("\nVisualisasi Distribusi Cluster:")

plot_cluster_distribution(kmeans_word2vec_combined_labels, "KMeans Word2Vec Fitur Gabungan")
plot_cluster_distribution(dbSCAN_word2vec_combined_labels, "DBSCAN Word2Vec Fitur Gabungan")
plot_cluster_distribution(kmeans_fasttext_combined_labels, "KMeans FastText Fitur Gabungan")
plot_cluster_distribution(dbSCAN_fasttext_combined_labels, "DBSCAN FastText Fitur Gabungan")
```

Disini kita mengambil word2vec dan fasttext sebab dari pengamatan kelompok kami kedua clustering yang paling efisien.

Fungsi `plot_cluster_distribution` memvisualisasikan hasil clustering sebagai diagram batang, menerima parameter `labels` (label cluster untuk tiap sampel) dan `title` (judul grafik). Fungsi ini menghitung jumlah sampel di setiap cluster menggunakan `np.unique` lalu memplotnya dengan `plt.bar`. Sumbu x menunjukkan label cluster, sumbu y jumlah sampel, dan grafik diberi label serta judul untuk memperjelas.

Fungsi ini digunakan untuk membandingkan distribusi cluster dari KMeans dan DBSCAN berdasarkan fitur Word2Vec dan FastText. Visualisasi membantu memahami persebaran sampel dalam cluster serta mengevaluasi hasil clustering dari tiap metode.



6. Menyimpan Data Hasil Clustering

```
df['KMeans_Word2Vec_Labels'] = kmeans_word2vec_combined_labels
df['DBSCAN_Word2Vec_Labels'] = dbscan_word2vec_combined_labels
df['KMeans_FastText_Labels'] = kmeans_fasttext_combined_labels
df['DBSCAN_FastText_Labels'] = dbscan_fasttext_combined_labels

# Menyimpan dataframe dengan label cluster ke file CSV baru
output_csv = 'job_listings_with_clusters_combined.csv'
df.to_csv(output_csv, index=False)
print(f"Hasil clustering disimpan ke {output_csv}")

# Menampilkan beberapa baris hasil clustering
print("\nDataFrame dengan label cluster:")
df.head()
```

Untuk menyimpan hasil clustering dari word2vec dan fasttext dari code sebelumnya. Hasil clustering akan disimpan dengan nama job_listings_with_clusters_combined.csv.



Analisis Hasil

Berdasarkan penggeraan projek disini ada beberapa hal yang dapat dianalisis.



1. Proses scraping

Scraping 200 data membutuhkan waktu yang lebih lama dibandingkan 5-10 data , ini terjadi karena beberapa faktor yaitu, jumlah permintaan http yang banyak (terlihat disini jumlah permintaan ada 200 permintaan), jeda antar permintaan (ada penundaan untuk setiap halaman selama 1 detik), proses membuka halaman detail (untuk mengambil data keterampilan dan pengalaman), serta waktu yang dibutuhkan untuk mengolah data.

2. Pada tahap preprocessed

Untuk analisis hasil preprocessed dari kelompok kami tidak ada analisa khusus, untuk yang yang umum bisa dilihat pada penjelasan alur dan tahapan.



3. Tahap future engineering

Pada tahap feature engineering, awalnya kelompok kami hanya berencana menggunakan TF-IDF. Namun, atas saran dosen, kami menambahkan Word2Vec, BoW, dan FastText. Hasilnya, variasi ini sangat membantu, terutama dalam clustering, di mana Word2Vec dan FastText memberikan hasil yang lebih baik dibandingkan TF-IDF dan BoW.



4. Tahap Clustering

Pada tahap clustering, ada dua analisis utama:

1. Perbedaan Hasil KMeans dan DBSCAN

KMeans menghasilkan jumlah cluster sesuai yang kita tentukan (contoh: 3 cluster), sedangkan DBSCAN bergantung pada nilai eps dan min_samples. Jika eps terlalu kecil, hanya titik-titik yang sangat dekat dikelompokkan, menghasilkan banyak cluster kecil atau noise. Jika min_samples terlalu rendah, lebih banyak cluster terbentuk; jika terlalu tinggi, jumlah cluster berkurang. Dalam proyek ini, DBSCAN menghasilkan 26 cluster dengan eps 0.3 dan min_samples 3.

2. Perbedaan Silhouette Score

KMeans memiliki silhouette score rendah, sedangkan DBSCAN sangat tinggi. Ini karena KMeans membagi data ke jumlah cluster tetap, sedangkan DBSCAN lebih fleksibel, menyesuaikan dengan kepadatan dan bentuk data, sehingga menghasilkan cluster yang lebih sesuai.



Terimakasih Banyak!..

Karena
Sudah
Menyimak.

