

**LAPORAN HASIL PROYEK
PEMROSESAN TEXT**



OLEH KELOMPOK 11:

Shinta Usaila Farachin : 23031554160
Ikhrima Atusifah : 23031554181
Arina Tri Yuni W. T. : 23031554203

Kelas 2023A

UNIVERSITAS NEGERI SURABAYA
2024

CLUSTERING LOWONGAN PEKERJAAN BERDASARKAN KETERAMPILAN YANG DIBUTUHKAN DAN TINGKAT PENGALAMAN

Latar Belakang

Clustering lowongan pekerjaan berdasarkan keterampilan yang dibutuhkan dan tingkat pengalaman adalah proses pengelompokan data pekerjaan menjadi kelompok-kelompok tertentu berdasarkan kesamaan kriteria. Proyek ini bertujuan untuk membantu pencari kerja menemukan pekerjaan yang sesuai dengan keterampilan dan pengalaman mereka, sekaligus membantu perusahaan menyusun kategori pekerjaan yang lebih terstruktur. Dengan meningkatnya volume data lowongan pekerjaan, analisis manual menjadi sulit dilakukan, sehingga pendekatan berbasis algoritma seperti K-Means dan DBSCAN menjadi solusi yang efektif.

Metode K-Means adalah algoritma clustering yang membagi data ke dalam sejumlah cluster yang telah ditentukan sebelumnya, berdasarkan kesamaan jarak antar data. Metode ini cocok digunakan untuk dataset yang terstruktur dengan baik dan memiliki jumlah cluster yang dapat diperkirakan. Sementara itu, DBSCAN (Density-Based Spatial Clustering of Applications with Noise) adalah metode clustering berbasis kepadatan yang mampu menangani data dengan distribusi yang tidak merata dan dapat mengidentifikasi data yang tidak termasuk dalam cluster mana pun (outlier). Kombinasi kedua metode ini memungkinkan analisis yang lebih komprehensif, karena K-Means dapat memberikan hasil yang cepat dan efisien untuk data terstruktur, sedangkan DBSCAN lebih fleksibel dalam menangani data yang kompleks dan tidak terstruktur.

Dengan menerapkan kedua metode ini, hasil clustering dapat memberikan wawasan yang lebih dalam tentang pengelompokan lowongan pekerjaan. Informasi ini bermanfaat bagi pencari kerja dalam memilih pekerjaan yang sesuai dengan kompetensi mereka dan bagi perusahaan dalam menyusun strategi perekrutan yang lebih tepat sasaran. Proyek ini juga mendukung pengembangan platform pencarian kerja yang lebih cerdas dan adaptif terhadap kebutuhan pasar tenaga kerja.

Tujuan

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

Manfaat

1. Membantu pencari kerja menghemat waktu dalam mencari pekerjaan yang sesuai.
2. Mengurangi resiko kesalahan dalam mencocokkan kandidat dengan posisi yang dibutuhkan.

3. 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

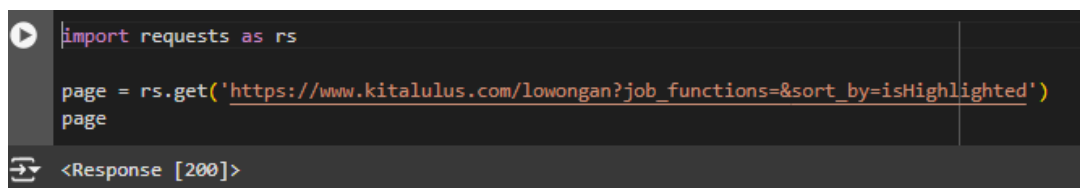
ALUR DAN TAHAPAN Pengerjaan

Ada beberapa tahapan yang dikerjakan dalam proyek ini yaitu:

1. Tahap Scraping

Pada tahap ini data dari web akan di scraping secara manual berdasarkan posisi pekerjaan, Perusahaan yang mencari lowongan, lokasi, link, pengalaman serta keterampilan. Langkah awal yakni melakukan code requests untuk melakukan permintaan pada link:

https://www.kitalulus.com/lowongan?job_functions=&sort_by=isHighlighted



```
import requests as rs

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

<Response [200]>

Permintaan ini bertujuan untuk mengakses data lowongan kerja dengan pengurutan berdasarkan parameter semua pengerjaan. Respons yang diterima disimpan dalam variabel page, dengan kode status HTTP 200 yang menunjukkan bahwa permintaan berhasil. Kemudian baru dilanjutkan code untuk scraping.

```

import requests
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-11om4v1-0 iwlPnJ3') # Sesuaikan dengan elemen asli

        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')


```

Kode tersebut menggunakan modul requests, BeautifulSoup, dan csv untuk melakukan web scraping dari situs kitalulus.com dan menyimpan data lowongan kerja ke dalam file CSV. URL target ditentukan dalam variabel url, dan hasil scraping disimpan dalam file bernama job_listings.csv. File CSV ini berisi header seperti posisi, tautan, perusahaan, lokasi, pengalaman, dan keterampilan. Skrip menggunakan loop untuk mengambil data dari beberapa halaman dengan menambahkan parameter page pada URL. Permintaan HTTP GET dilakukan menggunakan request, dan responnya diproses dengan BeautifulSoup untuk parsing HTML. Elemen data pekerjaan di ekstrak berdasarkan struktur HTML situs (ditentukan oleh kelas element). Data yang diperoleh disimpan ke dalam file CSV hingga jumlah data mencapai 200 entri.

```

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 kaIrsrv')
        lokasi = lokasi_element.get_text().strip() if lokasi_element else 'Tidak tersedia'

        pengalaman = 'Tidak tersedia'
        pengalaman_elements = job_soup.find_all('p', class_='TextStyled__Text-sc-18vo2dc-0 kaIrsrv')

        for i, elem in enumerate(pengalaman_elements):
            if "pengalaman" in elem.get_text().strip().lower():
                if i+1 < len(pengalaman_elements):
                    pengalaman = pengalaman_elements[i+1].get_text().strip()
                    break

        keterampilan_element = job_soup.find('div', class_='VacancyDescriptionStyled__Wrapper-sc-13uwtyz-2 cDTmUf')
        keterampilan = keterampilan_element.get_text().strip() if keterampilan_element else 'Tidak tersedia'
    else:
        lokasi = 'Tidak tersedia'
        pengalaman = 'Tidak tersedia'
        keterampilan = 'Tidak tersedia'

    job_data.append([posisi, tautan, perusahaan, lokasi, pengalaman, keterampilan])

```

```

# Tampilkan jumlah data yang sudah di-scrape
print(f"Data yang telah di-scrape: {len(job_data)}")

# Berhenti jika sudah mendapatkan 50 data
if len(job_data) >= 200:
    break

# Menambahkan jeda agar tidak terlalu sering melakukan permintaan
time.sleep(1)
page += 1
else:
    print("Gagal mendapatkan halaman:", response.status_code)
    break

# Menulis data
writer.writerow(job_data)

print(f"Data berhasil disimpan ke {csv_filename}")

```

Kode ini adalah bagian dari proses web scraping untuk mengekstrak data pekerjaan dari situs web. Dalam loop, setiap elemen pekerjaan diproses untuk mengambil informasi seperti posisi, perusahaan, tautan ke pekerjaan, lokasi, pengalaman, dan keterampilan. Berikut adalah langkah-langkahnya dalam paragraf:

Untuk setiap elemen pekerjaan (`job_elements`), kode mengekstrak judul pekerjaan dari elemen `

`, nama perusahaan dari elemen ` `, dan tautan dari atribut `href` di elemen ``. Kemudian, tautan ini digunakan untuk mengirim permintaan HTTP GET, dan responsnya di-parse menggunakan `BeautifulSoup`. Lokasi pekerjaan diambil dari elemen tertentu dengan kelas `TextStyled__Text-sc-18vo2dc-0`, dengan nilai default "Tidak tersedia" jika elemen tidak ditemukan. Data pengalaman pekerjaan diekstrak dari daftar elemen serupa, memeriksa apakah elemen ke-9 berisi kata "pengalaman". Jika ditemukan, elemen ke-10 digunakan, atau elemen terakhir diambil sebagai alternatif. Keterampilan pekerjaan diekstrak dari elemen dengan kelas spesifik, dan jika tidak ditemukan, maka ditulis "Tidak tersedia". Semua data yang diekstrak ini siap dimasukkan ke dalam file CSV.

Outputnya:

posisi	tautan	perusahaan	lokasi	pengalaman	1 to 10 of 200 entries Filter keterampilan
(Khusus Pengguna Iphone) Gabung Freelance Misi Seru Rating & Review WFH Raih Reward Berlimpah 80.000 (UR) - DL	https://www.kitalulus.com/lowongan/detail/seri-iv-misi-seru-freelance-with-dapatkan-reward-10-v4fb	Freelance MisiSeru - Kita Lulus	Kota Banjarmasin, Kalimantan Selatan	Tidak ada ketentuan	Administrasi
(Khusus Pengguna Iphone) Hiring Buzzer Misi Seru Freelance Online WFH Dapatkan Special Reward hingga 80.000 (UR) -DL	https://www.kitalulus.com/lowongan/detail/misi-seru-freelance-dari-rumah-dapetin-reward-500-v0tz	Freelance MisiSeru - Kita Lulus	Kabupaten Jember, Jawa Timur	Tidak ada ketentuan	Administrasi
Work from home freelance admin Reward Hingga Rp 25.000 (Q) - RLY	https://www.kitalulus.com/lowongan/detail/freelance-work-from-home-misi-seru-admin-rt-oywo	Freelance MisiSeru - KitaLulus	Jakarta Selatan, DKI Jakarta	Tidak ada ketentuan	Administrasi
Business Development Associate B2B (Yogyakarta)	https://www.kitalulus.com/lowongan/detail/business-development-sales-yogyakarta-zcyt	KitaLulus	Kota Yogyakarta, DI Yogyakarta	Minimal 1 Tahun	AdministrasiKeterampilan KomunikasiKeterampilan PenjualanSpreadsheet
Akuntan Pajak	https://www.kitalulus.com/lowongan/detail/akuntan-pajak-7yqk	PT Palar Persada Sejahtera	Kabupaten Gresik, Jawa Timur	Minimal 2 Tahun	Akurasi PajakAnalisis PajakAnalisis Data PajakHukum PajakKebijakan Pajak
Teknisi Body Repair & Body Painting Alat Berat	https://www.kitalulus.com/lowongan/detail/teknisi-body-repair-body-painting-alat-berat-btcm	PT. Eka Nusa Global (ENG)	Kota Padang, Sumatra Barat	Minimal 3 Tahun	Body PaintingPemeliharaan Sistem PengelasanPengelasan Kendaraan
COMMUNITY OFFICER- KAB GARUT	https://www.kitalulus.com/lowongan/detail/community-officer-kab-garut-3d49	BTPN Syariah	Kabupaten Garut, Jawa Barat	Tidak ada ketentuan	Keterampilan Komunikasi
COMMUNITY OFFICER (KUNINGAN, JABAR)	https://www.kitalulus.com/lowongan/detail/community-officer-kuningan-jabar-ldvk	BTPN Syariah	Kabupaten Kuningan, Jawa Barat	Tidak ada ketentuan	AdministrasiKeterampilan Komunikasi
Community Officer - Penempatan	https://www.kitalulus.com/lowongan/detail/community-officer-		Kabupaten	Tidak ada	

2. Tahap Pre-Processing

```

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('omw-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)

```

Kode ini bertujuan untuk memproses teks dengan beberapa langkah utama: mengunduh data yang diperlukan dari NLTK, seperti kamus 'wordnet' dan daftar 'stopwords'. Setelah itu, inisialisasi dilakukan untuk 'stemmer' (PorterStemmer) dan 'lemmatizer' (WordNetLemmatizer), serta pembuatan daftar 'stopwords' gabungan dari bahasa Indonesia dan Inggris, termasuk tambahan kata-kata khusus yang ingin dihapus. Fungsi preprocessing ini dirancang untuk membersihkan teks, memisahkan kata tanpa spasi yang benar, menghapus karakter non-alfabet, mengubah teks menjadi huruf kecil, menghapus 'stopwords', dan menerapkan 'stemming' serta 'lemmatization' untuk

menghasilkan teks yang siap analisis. Namun, terdapat beberapa kesalahan sintaks pada kode, seperti format regex dan penulisan fungsi yang perlu diperbaiki agar dapat dijalankan.

```
# 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("Data hasil preprocessing berhasil disimpan ke (job_listings_preprocessed.csv)")
```

Kode ini memproses data dari file CSV bernama `job_listings.csv`, kemudian menyimpan hasil yang telah diproses ke file baru bernama `Job_listings_preprocessed.csv`. Proses dimulai dengan membaca data CSV awal menggunakan `csv.DictReader`, lalu membuat file output dengan header yang sama menggunakan `csv.DictWriter`.

Setiap baris data diproses dengan memeriksa kolom pengalaman dan keterampilan. Jika data ada di kolom tersebut, teksnya akan diproses menggunakan fungsi `preprocess_text`. Jika tidak ada data, nilai "Tidak tersedia" akan ditambahkan sebagai pengganti. Data yang telah diproses kemudian ditulis ke file baru dengan header yang sama. Namun, terdapat beberapa kesalahan pada kode, termasuk penulisan parameter fungsi dan penempatan tanda kutip. Koreksi sintaksis diperlukan agar kode dapat berjalan.

Outputnya:

posisi	tautan	perusahaan	lokasi	pengalaman	keterampilan
Misi Seru Gratis Freelance Reward 10.000-30.000 - kz1 (KR)	https://www.kitalulus.com/lowongan/detail/misi-seru-gratis-freelance-reward-1000030000-kz1-qhv	Freelance MisiSeru - Kitalulus	Kota Tangerang Selatan, Banten	keterangan	administrasi
Misi Seru 10.000-15.000 Pembuatan Akun, Freelance dari rumah - kz1 (KR)	https://www.kitalulus.com/lowongan/detail/misi-seru-1000015000-pembuatan-akun-freelance-v8o	Freelance MisiSeru - Kitalulus	Kota Bekasi, Jawa Barat	keterangan	administrasi
Freelance Misi Seru Gratis Reward 10.000-30.000 - kz1 (KR)	https://www.kitalulus.com/lowongan/detail/freelance-misi-seru-gratis-reward-1000030000-kz1-rnd	Freelance MisiSeru - Kitalulus	Kabupaten Bandung Barat, Jawa Barat	keterangan	administrasi
Petugas Kebersihan / Cleaning Service	https://www.kitalulus.com/lowongan/detail/petugas-kebersihan-cleaning-service-tjd	Planet Soccer Indonesia	Kota Samarinda, Kalimantan Timur	minim 2	keterampilan kebersihan
Steward / Dishwasher	https://www.kitalulus.com/lowongan/detail/steward-dishwasher-cyzb	PT Margo Ambawang Food	Jakarta Utara, DKI Jakarta	minim 1	berorientasi pelanggan keterampilan kebersihan keterampilan masakan tradisional
Tutor Skripsi WFH	https://www.kitalulus.com/lowongan/detail/tutor-skripsi-wfh-8keb	PT. Cepat Lulus Utama Edukasi	Kota Malang, Jawa Timur	keterangan	keterampilan menui laporan
SPG&SPB	https://www.kitalulus.com/lowongan/detail/spgsb-o2b2	BM Group	Kota Surabaya, Jawa Timur	keterangan	adaptasi bahasa tubuh analasi penjualan
Salesman	https://www.kitalulus.com/lowongan/detail/salesman-kndc	Maka Motors	Jakarta Utara, DKI Jakarta	keterangan	interperson keterampilan komunikasi negosiasi pemasaran
WAREHOUSE STAFF EKSPEDISI KOSAMBI TANGERANG	https://www.kitalulus.com/lowongan/detail/warehouse-staff-ekspedisi-kosambi-tangerang-kbw0	PT Personel Aiah Daya Tbk	Kabupaten Serang, Banten	minim 1	analasi logistik pemahaman sistem manajemen gudang
Bancassurance Financial Advisor	https://www.kitalulus.com/lowongan/detail/bancassurance-financial-advisor-ofmk	Rajawali Berdikari Indonesia	Kota Denpasar, Bali	minim 1	keuangan industri perbankan

3. Tahap Feature Engineering

```

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

# Memload 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 di atas digunakan untuk untuk mengekstrak fitur teks dari data pengalaman dan keterampilan menggunakan berbagai teknik pemrosesan bahasa alami (NLP). Data diimpor dari file CSV ke dalam Data Frame menggunakan pandas, kemudian kolom pengalaman dan keterampilan diisi dengan nilai kosong untuk menangani data yang hilang. Teknik pertama adalah *Bag of Words (BoW)*, yang menggunakan *CountVectorizer* untuk mengubah teks menjadi matriks berdasarkan frekuensi kemunculan kata. Selanjutnya, *TF-IDF* digunakan melalui *TfidfVectorizer* untuk memberi bobot lebih tinggi pada kata yang sering muncul di dokumen tertentu namun jarang dalam dokumen lainnya. Teknik lainnya adalah *Word2Vec*, yang menghasilkan vektor numerik dengan mempelajari hubungan semantik antar kata dalam teks, diikuti oleh *FastText* yang serupa namun lebih unggul dalam menangkap informasi sub-kata. Semua teknik ini menghasilkan representasi numerik dari teks, yang dapat digunakan untuk berbagai tugas NLP seperti analisis, klasifikasi, atau prediksi.


```

# 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())

```

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.

Outputnya:

```

Hasil Feature Engineering untuk Pengalaman:
Bag of Words (BoW) Pengalaman:
ketentuan  minim
0          1      0
1          1      0
2          1      0
3          1      0
4          0      1
TF-IDF Pengalaman:
ketentuan  minim
0          1.0    0.0
1          1.0    0.0
2          1.0    0.0
3          1.0    0.0
4          0.0    1.0
Word2Vec Pengalaman:
0 0.000095 0.003077 -0.006813 -0.001375 0.007669 0.007346 -0.003673
1 0.000095 0.003077 -0.006813 -0.001375 0.007669 0.007346 -0.003673
2 0.000095 0.003077 -0.006813 -0.001375 0.007669 0.007346 -0.003673
3 0.000095 0.003077 -0.006813 -0.001375 0.007669 0.007346 -0.003673
4 -0.004577 0.001951 0.005149 0.007379 -0.000923 -0.006645 0.003785

0 0.002643 -0.008317 0.006205 ... -0.004509 0.005702 0.009180 -0.004100
1 0.002643 -0.008317 0.006205 ... -0.004509 0.005702 0.009180 -0.004100
2 0.002643 -0.008317 0.006205 ... -0.004509 0.005702 0.009180 -0.004100
3 0.002643 -0.008317 0.006205 ... -0.004509 0.005702 0.009180 -0.004100
4 0.007514 -0.003930 -0.004970 ... 0.001360 -0.000693 0.002837 -0.003831

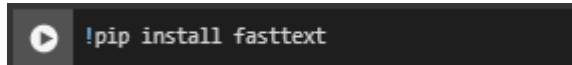
0 0.007965 0.005375 0.005879 0.000513 0.008213 -0.007019
1 0.007965 0.005375 0.005879 0.000513 0.008213 -0.007019
2 0.007965 0.005375 0.005879 0.000513 0.008213 -0.007019
3 0.007965 0.005375 0.005879 0.000513 0.008213 -0.007019
4 0.003456 0.003864 -0.001790 -0.004720 -0.004383 0.005452

[5 rows x 100 columns]
FastText Pengalaman:
0 0.000705 -0.000745 -0.000144 -0.001721 -0.001118 0.000932 -0.000283

```

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

A screenshot of a terminal window with a dark background. On the left, there is a play button icon. To its right, the text `!pip install fasttext` is displayed in a light blue monospace font.

Perintah ``pip install fasttext`` digunakan untuk menginstal pustaka FastText yang dikembangkan oleh Facebook AI Research. 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. Berikut adalah codenya

```

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

```

```

# 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 dimulai dengan mengolah data menggunakan `pandas` dan `numpy`, kemudian mengaplikasikan metode ekstraksi fitur teks seperti BoW, TF-IDF, Word2Vec, dan FastText pada kolom `combined_features`. Data fitur yang telah diekstrak kemudian di-scale menggunakan `StandardScaler`. Selanjutnya, dua metode clustering diterapkan: `KMeans` dan `DBSCAN`. Untuk KMeans, jumlah kluster ditentukan dan silhouette score dihitung untuk menilai kualitas clustering. Sedangkan pada DBSCAN, clustering dilakukan dengan parameter `eps` dan `min_samples`, dan

juga dihitung silhouette score untuk evaluasi hasil clustering. Kemudian 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.

Outputnya:

```
Silhouette Score untuk fitur gabungan (KMeans):  
KMeans TF-IDF: 0.183412174520924  
KMeans BoW: 0.19093673179195825  
KMeans Word2Vec: 0.18847201764583588  
KMeans FastText: 0.21779808402061462  
  
Silhouette Score untuk fitur gabungan (DBSCAN):  
DBSCAN TF-IDF: 0.999999988142394  
DBSCAN BoW: 0.999999986354721  
DBSCAN Word2Vec: 1.0  
DBSCAN FastText: 1.0
```

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

```

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_tfidf_combined_labels, kmeans_tfidf_combined_davies_bouldin = kmeans_clustering(tfidf_combined, 3)
dbscan_tfidf_combined_labels, dbscan_tfidf_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_tfidf_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_tfidf_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}")

```

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

outputnya

```

Davies-Bouldin Index untuk fitur gabungan (KMeans):
KMeans TF-IDF: 1.0762729052661018
KMeans Bow: 1.0374386033688137
KMeans Word2Vec: 1.8978111201158285
KMeans FastText: 1.6931020939420132

Davies-Bouldin Index untuk fitur gabungan (DBSCAN):
DBSCAN TF-IDF: 3.8858733684187146e-08
DBSCAN Bow: 4.1057079319600106e-08
DBSCAN Word2Vec: 1.9724684947768685e-07
DBSCAN FastText: 2.090153387772489e-07

```

b. Visualisasi hasil clustering

Berikut adalah code visualisasi clustering. Disini yang diambil menggunakan word2vec dan fasttext sebab dari pengamatan kelompok kami kedua clustering yang paling efisien.

```
# 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()

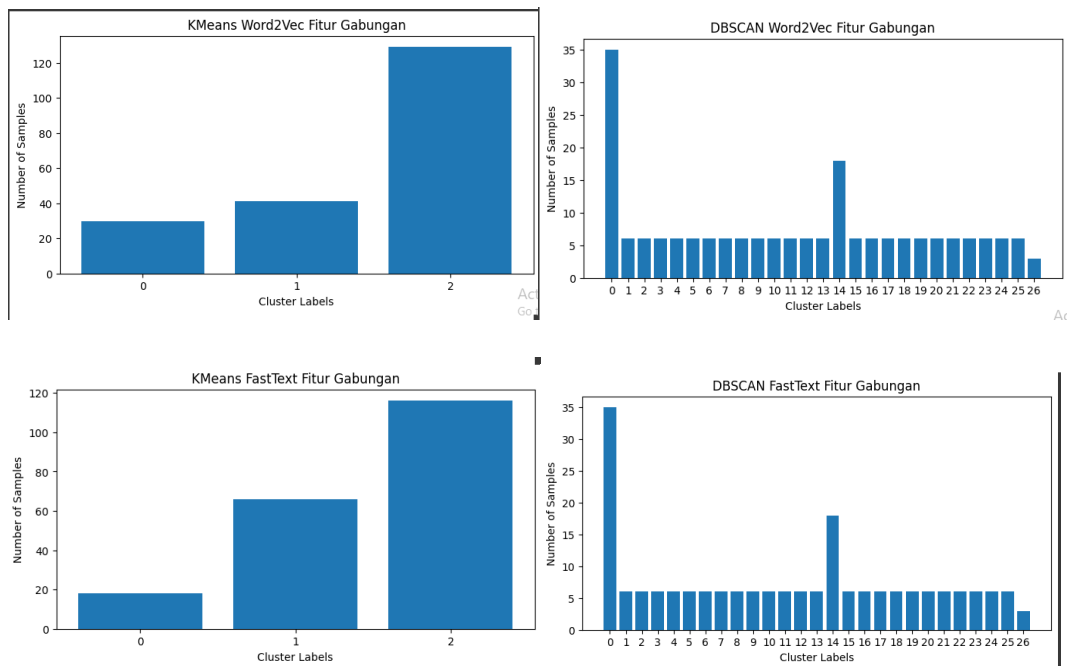
# Visualisasi Hasil Clustering
print("\nVisualisasi Distribusi Cluster:")

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")
```

Kode ini berfungsi untuk memvisualisasikan hasil clustering dalam bentuk diagram batang (bar chart) menggunakan data yang telah dikelompokkan oleh algoritma clustering. Fungsi `plot_cluster_distribution` menerima dua parameter utama: `labels`, yang berisi label cluster untuk setiap sampel, dan `title`, yang menetapkan judul grafik. Pertama-tama, fungsi ini menghitung jumlah sampel di setiap cluster dengan menggunakan `np. unique` untuk mengidentifikasi nilai unik dalam label serta menghitung frekuensinya. Setelah itu, fungsi `plt. bar` dipakai untuk membuat diagram batang, di mana sumbu x menunjukkan label cluster, sementara sumbu y menunjukkan jumlah sampel pada setiap cluster. Agar visualisasi lebih jelas, label pada sumbu x dan y serta judul grafik ditambahkan, dan grafik ditampilkan dengan `plt. show()`.

Dalam tahap visualisasi hasil clustering, fungsi `plot_cluster_distribution` dipanggil untuk menggambarkan distribusi cluster dari empat kombinasi yang berbeda: KMeans dan DBSCAN, masing-masing menggunakan fitur gabungan dari Word2Vec dan FastText. Setiap pemanggilan fungsi ini menghasilkan grafik yang menunjukkan bagaimana sampel data tersebar ke dalam cluster-cluster yang terbentuk oleh metode clustering yang diterapkan. Visualisasi ini memungkinkan kita membandingkan jumlah sampel di setiap cluster dan mengevaluasi perbedaan hasil clustering tergantung pada algoritma dan fitur yang digunakan. Dengan demikian, grafik ini memberikan gambaran yang jelas tentang efektivitas serta distribusi cluster yang dihasilkan oleh tiap metode clustering.

Outputnya



c. Menyimpan data hasil clustering

Code selanjutnya adalah untuk menyimpan hasil clustering dari word2vec dan fasttext dari code sebelumnya. Hasil clustering akan disimpan dengan nama `job_listings_with_clusters_combined.csv`.

```
[29] 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()
```

Outputnya

Hasil clustering disimpan ke job_listings_with_clusters_combined.csv

DataFrame dengan label cluster:

	posisi	tautan	perusahaan	lokasi	pengalaman	keterampilan	combined_features	KMeans_Word2Vec_Labels	DBSCAN_Word2Vec_Labels	KMeans_Fas
0	(Khusus Pengguna Iphone) Hiring Buzzer Misi Se...	https://www.kitalulus.com/lowongan/detail/misi...	Freelance MisiSeru - Kita Lulus	Kota Depok, Jawa Barat	ketentuan	administrasi	ketentuan administrasi	1	0	
1	(Bagi Pengguna Iphone) Peluang Freelance Work ...	https://www.kitalulus.com/lowongan/detail/free...	Freelance MisiSeru - Kita Lulus	Kabupaten Bandung Barat, Jawa Barat	ketentuan	administrasi	ketentuan administrasi	1	0	
2	(Misi Seru Khusus Iphone) Freelance WFH Buzzer...	https://www.kitalulus.com/lowongan/detail/misi...	Freelance MisiSeru - Kita Lulus	Kabupaten Tangerang, Banten	ketentuan	administrasi	ketentuan administrasi	1	0	

d. Visualisasi pca

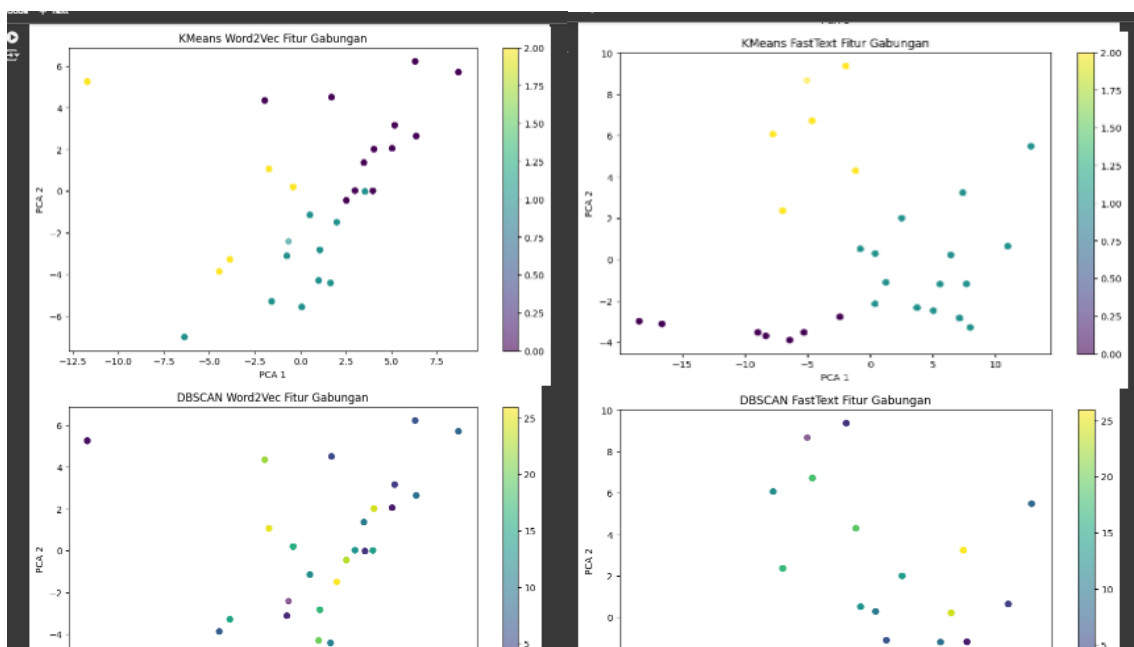
```
# Visualisasi PCA
def plot_pca(features, cluster_labels, title):
    scaler = StandardScaler()
    scaled_features = scaler.fit_transform(features)
    pca = PCA(n_components=2)
    pca_result = pca.fit_transform(scaled_features)
    plt.figure(figsize=(10, 5))
    plt.scatter(pca_result[:, 0], pca_result[:, 1], c=cluster_labels, cmap='viridis', marker='o', alpha=0.6)
    plt.title(title)
    plt.xlabel('PCA 1')
    plt.ylabel('PCA 2')
    plt.colorbar()
    plt.show()

# Visualisasi PCA
print("\nVisualisasi PCA untuk fitur gabungan:")

# Fitur gabungan
plot_pca(word2vec_combined, kmeans_word2vec_combined_labels, "KMeans Word2Vec Fitur Gabungan")
plot_pca(word2vec_combined, dbscan_word2vec_combined_labels, "DBSCAN Word2Vec Fitur Gabungan")
plot_pca(fasttext_combined, kmeans_fasttext_combined_labels, "KMeans FastText Fitur Gabungan")
plot_pca(fasttext_combined, dbscan_fasttext_combined_labels, "DBSCAN FastText Fitur Gabungan")

# Menyimpan hasil clustering ke file CSV
df['kmeans_word2vec_combined_cluster'] = kmeans_word2vec_combined_labels
df['dbscan_word2vec_combined_cluster'] = dbscan_word2vec_combined_labels
df['kmeans_fasttext_combined_cluster'] = kmeans_fasttext_combined_labels
df['dbscan_fasttext_combined_cluster'] = dbscan_fasttext_combined_labels
```

Output



ANALISIS HASIL Pengerjaan

Berdasarkan pengerjaan projek disini ada beberapa hal yang dapat dianalisis.

1. Proses scraping

Proses scraping butuh waktu yang lebih banyak untuk mengscraping 200 data daripada hanya 5 atau 10 data. setelah dianalisis ternyata hal ini terjadi karena beberapa faktor yaitu, jumlah permintaan http yang banyak (terlihat disini jumlah permintaan ada 200 permintaan), jeda antar permintaan (dalam code ada penundaan untuk setiap halaman selama 1 detik), pengambilan data dari halaman detail (hal ini bisa dilihat ketika akan mengambil data keterampilan dan pengalaman yang harus mengklik lowongan kerja terlebih dahulu), yang terakhir adalah karena dari pengolahan data itu sendiri.

2. Pada tahap preprocessed

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

3. Tahap future engineering

Pada tahap future engineering, sebelumnya dari kelompok kami hanya berniat menggunakan tfidf, tetapi dari dosen pengampu ingin kami lebih bervariasi menggunakan word2vec, bow dan fasttext. Dan ternyata setelah di variasi hasil yang diberikan dari word2vec, bow dan fasttext sangat berguna untuk tahap selanjutnya. Sebagai contoh, ketika hasil clustering diberikan ternyata hasil word2vec dan fasttext lebih bagus hasilnya daripada menggunakan tfidf atau bow.

4. Tahap Clustering

Pada tahap clustering ada beberapa analisa yang dapat diambil. Pertama, hasil kmeans dan dbscan ternyata tidak sama. Untuk hasil kmeans kita bisa menentukan hasil clustering yang kita mau contohnya disini kita hanya menggunakan 3 cluster maka yang muncul nanti hanya ada 3 cluster. Akan tetapi untuk dbscan tidak, untuk hasil dbscan hasil cluster tidak bisa kita tentukan sendiri sebab hasil dbscan ditentukan oleh nilai eps dan min_samplenya. Jika nilai eps terlalu kecil, hanya titik-titik yang sangat dekat satu sama lain yang akan dikelompokkan bersama, sehingga menghasilkan banyak cluster kecil, dan mungkin juga banyak titik yang dianggap sebagai noise (label -1). Jika min_sample terlalu rendah, algoritma mungkin akan membentuk lebih banyak cluster karena lebih sedikit titik yang dibutuhkan untuk membentuk cluster. Sebaliknya, jika nilai ini terlalu tinggi, DBSCAN mungkin tidak dapat

menemukan banyak cluster. Disini di proyek kita terdapat 26 cluster dbscan dengan eps 0.3 dan min_sample 3.

Kedua, hasil silhouette score untuk kmeans dan dbscan sangat berbanding terbalik. Hasil cluster kmeans sangatlah rendah sedangkan untuk dbscan sangat tinggi. Setelah dianalisa hal ini terjadi karena Kmeans secara langsung membagi data ke dalam cluster yang telah ditentukan sebelumnya, sementara DBSCAN memiliki fleksibilitas untuk menyesuaikan dengan bentuk dan kepadatan data, sehingga menghasilkan cluster yang lebih sesuai dengan karakteristik data yang ada.

REFERENSI

Referensi 1

Sone, P. P. (2020). Cluster-Based Job Matching System (Doctoral dissertation, MERAL Portal).

https://scholar.google.com/scholar?hl=id&as_sdt=0%2C5&q=Density-based+clustering+and+skill+classification+in+job+postings&btnG=#d=gs_qabs&t=1731976223778&u=%23p%3DtHNfDyczQA4J

Referensi 2

Mahajan, S., & Kumar, P. (2014). Implementation of Various Clustering Techniques.

https://scholar.google.com/scholar?start=30&q=Comparative+analysis+of+K+Means+and+BSCAN+for+job+market+analysis&hl=id&as_sdt=0,5#d=gs_qabs&t=1731976871267&u=%23p%3DJiyVVXKNMLAJ

Referensi 3

Lukauskas, M., Šarkauskaitė, V., Pilinkienė, V., Stundžienė, A., Grybauskas, A., & Bruneckienė, J. (2023). Enhancing skills demand understanding through job ad segmentation using NLP and clustering techniques. *Applied sciences*, 13(10), 6119.

<https://www.mdpi.com/2076-3417/13/10/6119>

KONTRIBUSI MASING-MASING KELOMPOK

1. Arina Tri Yuni Wahyuning Tiyas (23031554203)

- Membuat kode utama untuk fitur ekstraksi dan clustering.
- Melakukan analisis hasil clustering dan visualisasi data.

2. Shinta Usaila Farachin (23031554160)

- Memperbaiki error yang ditemukan selama pengembangan dan melakukan debugging kode.
- Membuat presentasi powerpoint untuk memaparkan hasil progress project

3. Ikhrima Atusifah (23031554181)

- . - Membantu menyempurnakan dan mengoptimalkan algoritma clustering.
- Membuat laporan proyek secara rinci, termasuk penjelasan metode dan hasil.