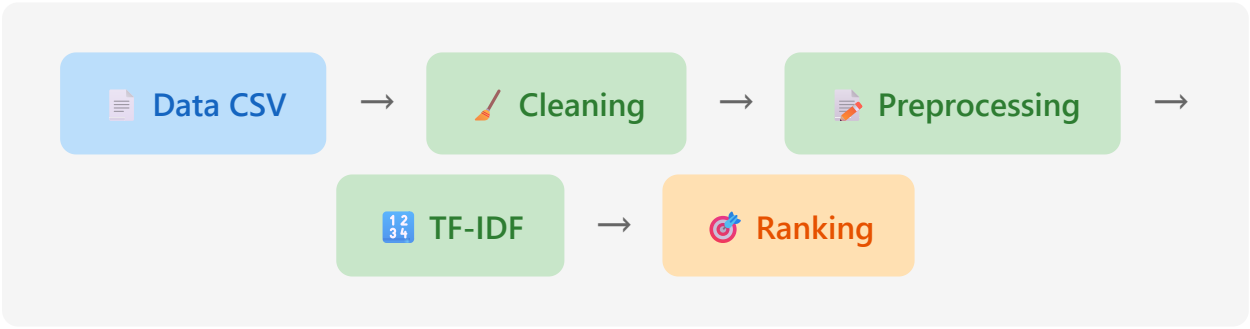
🌿 Panduan Content-Based Filtering dengan User Feedback Weighting (UFW)

File: indo_ecotourism_cbf_ufw_simplified.py

> **Apa itu CBF + UFW?**
> Sistem rekomendasi yang menyarankan destinasi wisata berdasarkan *konten/deskripsi* tempat, ditambah dengan *preferensi pengguna* dari feedback sebelumnya.

1. Gambaran Umum Sistem



> 💡 **Analogi:** Bayangkan kamu punya katalog wisata. CBF + UFW itu seperti **resepsionis hotel pintar** yang:
> 1. Membaca semua brosur wisata (TF-IDF)
> 2. Mengingat tempat yang kamu suka sebelumnya (User Feedback)
> 3. Menggabungkan keduanya untuk memberikan rekomendasi terbaik

2. Langkah 1: Load & Cleaning Data

Pertama, kita memuat data dari file CSV yang berisi informasi destinasi wisata.

```
# Load data
df = pd.read_csv("./eco_place.csv")

# Hapus duplikat
df = df.drop_duplicates().reset_index(drop=True)

# Parse harga (ambil angka pertama saja)
def parse_price_idr(x):
    if pd.isna(x):
        return np.nan
    s = str(x).lower()
    if "gratis" in s:
        return 0.0
    angka = re.findall(r"\d+", s)
    return float(angka[0]) if angka else np.nan
```

💡 **Analogi:** Seperti membersihkan lemari pakaian — kita buang baju duplikat dan rapikan label harga yang berantakan.

3. Langkah 2: Preprocessing Teks

Teks perlu dibersihkan agar komputer bisa memahaminya dengan baik:

- **Lowercase**: Ubah semua huruf jadi kecil
- **Tokenize**: Pecah kalimat jadi kata-kata
- **Remove Stopwords**: Buang kata-kata umum seperti "dan", "atau", "yang"

```
# Stopwords Indonesia
STOPWORDS_ID = {"ada", "adalah", "dan", "atau", "yang", ...}

def preprocess_text(text):
    text = str(text).lower()                    # lowercase
    tokens = re.findall(r"\w+", text)          # tokenize
    tokens = [t for t in tokens if t not in STOPWORDS_ID]
    return " ".join(tokens)
```

Sebelum	Sesudah

"Pantai yang Indah dan Menakjubkan di Bali"	"pantai indah menakjubkan bali"

4. Langkah 3: TF-IDF Vectorization

TF-IDF (Term Frequency - Inverse Document Frequency) mengubah teks menjadi angka yang bisa dihitung oleh komputer.

> **TF** = Seberapa sering kata muncul di dokumen
> **IDF** = Seberapa unik kata tersebut di seluruh dokumen

```
vectorizer = TfidfVectorizer(
    max_features=5000,      # Maksimal 5000 kata unik
    ngram_range=(1, 1),     # Hanya unigram (1 kata)
    min_df=2,               # Minimal muncul di 2 dokumen
    max_df=0.9,             # Maksimal 90% dokumen
    sublinear_tf=True,      # Gunakan log(1 + tf)
    norm="l2"               # Normalisasi vektor
)
tfidf_matrix = vectorizer.fit_transform(df["gabungan"])
```

> 💡 **Analogi:** TF-IDF seperti **sistem rating unik**:
> • Kata "pantai" di dokumen tentang pantai → skor tinggi (relevan)
> • Kata "dan" yang muncul di mana-mana → skor rendah (tidak unik)

5. Langkah 4: CBF + UFW Ranking

Ini adalah inti dari sistem rekomendasi. Skor dihitung dengan menggabungkan:

$$score(item) = cosine(query, item) + \alpha \times cosine(centroid\_liked, item)$$

Komponen	Penjelasan
cosine(query, item)	Seberapa mirip query dengan deskripsi item

centroid_liked	Rata-rata vektor dari item yang disukai user
α (alpha)	Bobot feedback (default: 0.6)

```python
def rank_cbf_ufw(query, liked_indices, alpha=0.6, topk=10):
    # 1. Transform query ke vektor
    query_vector = vectorizer.transform([preprocess_text(query)])

    # 2. Hitung similarity dengan semua item
    sim_query = cosine_similarity(query_vector, TFIDF_DENSE)[0]

    # 3. Hitung centroid dari liked items
    centroid = get_user_centroid(liked_indices)

    # 4. Gabungkan skor
    if centroid is None:
        scores = sim_query
    else:
        sim_centroid = cosine_similarity(centroid, TFIDF_DENSE)[0]
        scores = sim_query + alpha * sim_centroid

    # 5. Ambil top-k
    return np.argsort(-scores)[:topk].tolist()
```

💡 **Analogi:** Seperti **rekomendasi Netflix**:

- CBF = "Film ini mirip dengan yang kamu cari"
- UFW = "Kamu juga suka film sejenis ini sebelumnya"
- Gabungan = Rekomendasi yang lebih personal!

6. Langkah 5: Evaluasi Model

Kita mengukur seberapa bagus sistem rekomendasi bekerja dengan **metrik evaluasi**:

📊 Precision@K

Berapa persen dari Top-K rekomendasi yang benar-benar relevan?

```
Precision@10 = (jumlah item relevan di Top-10) / 10
```

Untuk single ground truth: 1/10 jika ditemukan, 0 jika tidak

🎯 Recall@K

Apakah item yang benar (ground truth) muncul di Top-K?

```
Recall@10 = 1 jika GT ada di Top-10, 0 jika tidak
```

Nilai tinggi = sistem berhasil menemukan item yang dicari

⏱️ Latency

Berapa lama waktu yang dibutuhkan untuk satu query?

```
Latency = waktu total / jumlah query (dalam ms)
```

Semakin kecil semakin bagus (cepat)

Contoh Hasil Evaluasi

Method	Precision@10	Recall@10	Latency (ms)
CBF+UFW (α=0.6)	0.0885	0.8846 (88.5%)	~8 ms

> **Interpretasi:** Recall 88.5% artinya dari 100 query, 88-89 query berhasil menemukan ground truth di Top-10. Ini adalah hasil yang **sangat bagus**!

7. Ringkasan Alur Sistem

1️⃣ **INPUT:** Query user + Liked items history

↓

2️⃣ **PREPROCESSING:** Lowercase → Tokenize → Remove Stopwords

↓

3️⃣ **VECTORIZATION:** TF-IDF mengubah teks → vektor numerik

↓

4️⃣ **SCORING:**

- Cosine similarity (query vs items)
- Cosine similarity (liked centroid vs items)
- Gabungkan dengan bobot α

↓

5️⃣ **OUTPUT:** Top-K rekomendasi destinasi wisata

📚 Dokumentasi ini dibuat sebagai panduan pembelajaran untuk memahami Content-Based Filtering dengan User Feedback Weighting

File: indo_ecotourism_cbf_ufw_simplified.py