Panduan Sistem Rekomendasi Content-Based Filtering + User Feedback Weighting

Dokumentasi Lengkap untuk Mahasiswa Pemula

📁 eco_recsys/cbf.py & ufw.py

---

1 Gambaran Umum Sistem

Apa itu Sistem Rekomendasi?
Sistem rekomendasi adalah program komputer yang membantu pengguna menemukan item yang mungkin mereka sukai, berdasarkan berbagai faktor seperti preferensi, perilaku, dan karakteristik item.

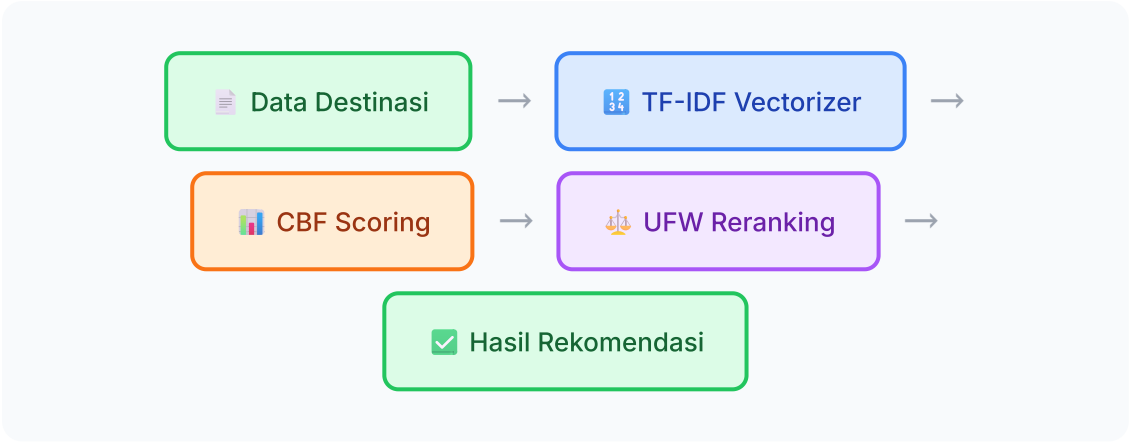
Sistem ini menggunakan dua pendekatan utama:

- **Content-Based Filtering (CBF)** — Merekomendasikan item berdasarkan kemiripan konten/deskripsi
- **User Feedback Weighting (UFW)** — Menyesuaikan ranking berdasarkan feedback user (Like/Skip)

🎯 **Analogi:** Bayangkan kamu punya teman yang tahu selera musikmu. Saat ada lagu baru, temanmu bisa bilang "Eh, ini cocok

> sama selera kamu!". Sistem rekomendasi bekerja seperti teman tersebut, tapi menggunakan algoritma matematika.

Alur Kerja Sistem



2 Struktur File Proyek

Berikut adalah file-file penting yang perlu kamu pahami:

```
📁 eco_recsys/ ← Package utama
  ├── cbf.py ← Algoritma Content-Based Filtering
  ├── ufw.py ← User Feedback Weighting
  ├── ui.py ← Komponen antarmuka Streamlit
  ├── data.py ← Loader untuk memuat artifacts
  ├── state.py ← Manajemen Like/Skip/Bookmark
  ├── utils.py ← Fungsi utilitas
  └── text.py ← Preprocessing teks

📁 artifacts/ ← Hasil training dari notebook
  ├── items.csv ← Data destinasi wisata
  ├── tfidf_matrix.npz ← Matriks TF-IDF
  ├── vectorizer.joblib ← TF-IDF vectorizer
  └── nbrs_cosine.joblib ← Index pencarian cepat

📄 app.py ← Aplikasi Streamlit utama
```

> 💡 **Tips:** Untuk memahami sistem ini, fokus pada dua file: `cbf.py` (bagaimana rekomendasi dibuat) dan `ufw.py` (bagaimana personalisasi bekerja).

3 Content-Based Filtering (cbf.py)

Apa itu Content-Based Filtering?
CBF adalah metode rekomendasi yang fokus pada *karakteristik/konten* dari item. Sistem menganalisis deskripsi, kategori, dan fitur item untuk menemukan item yang serupa.

🎯 **Analogi:** Seperti ketika kamu suka novel misteri karya Agatha Christie, sistem CBF akan merekomendasikan novel misteri lainnya karena memiliki "konten" yang mirip (genre misteri, gaya penulisan detektif, dll).

Langkah 1: TF-IDF (Text to Numbers)

TF-IDF (Term Frequency - Inverse Document Frequency) mengubah teks menjadi angka (vektor). Ini penting karena komputer tidak bisa langsung memproses teks.

Komponen	Penjelasan	Contoh
TF	Seberapa sering kata muncul dalam dokumen	"pantai" muncul 3× → TF tinggi
IDF	Seberapa jarang kata muncul di semua dokumen	"snorkeling" jarang → IDF tinggi
TF-IDF	TF × IDF = Skor kepentingan kata	Kata unik + sering = skor tinggi

🎯 **Analogi:** Bayangkan kamu membuat "sidik jari" untuk setiap destinasi. Kata-kata unik seperti "snorkeling" atau "pendakian"

> menjadi ciri khas yang membedakan satu destinasi dari lainnya.

Langkah 2: Cosine Similarity

Setelah setiap item diubah jadi vektor, kita mengukur "kemiripan" menggunakan **Cosine Similarity**:

$$\text{Cosine Similarity} = \cos(\theta) = (A \cdot B) / (|A| \times |B|)$$

Nilai: 0 (tidak mirip) sampai 1 (sangat mirip)

Langkah 3: MMR (Diversifikasi Hasil)

MMR (Maximal Marginal Relevance) menyeimbangkan *relevansi* dan *keragaman* hasil.

$$\text{MMR} = \lambda \times \text{Relevansi} - (1-\lambda) \times \text{Similarity dengan item terpilih}$$

λ tinggi = prioritas relevansi; λ rendah = prioritas keragaman

> 🎯 **Analogi:** Seperti memilih film untuk marathon: jika λ=0.9, kamu pilih film rating tertinggi meski genre sama. Jika λ=0.3, kamu pilih film beragam meski rating sedikit lebih rendah.

Kode: build_feed_cbf()

Ini adalah fungsi utama untuk membuat feed rekomendasi:

```
def build_feed_cbf(items, X, filters, top_n=12, ...):
    """
    Membangun feed rekomendasi untuk tab Feed.
    """

    # Step 1: Filter item berdasarkan kriteria user
    filter_mask = _mask_by_filters(items, filters)
    idx_all = np.arange(len(items))[filter_mask]
```

```
    # Step 2: Hapus item yang di-skip
    idx_all = _filter_blocked_items(idx_all, blocked_gids)

    # Step 3: Hitung skor dasar dari rating
    base_scores = _calculate_base_scores(items, idx_all)

    # Step 4: Pilih dengan MMR (relevan + beragam)
    selected_gids = mmr_select(idx_all, X, base_scores, top_n=top_n)

    # Step 5: Tambahkan item kejutan (serendipity)
    selected_gids = _add_serendipity_items(selected_gids, ...)

    return [Candidate(gid=g, base_score=score) for g in selected_gids]
```

Fungsi-Fungsi di cbf.py

Fungsi	Kegunaan
`_mask_by_filters()`	Filter item berdasarkan kategori, kota, harga
`_calculate_base_scores()`	Hitung skor dasar dari rating item
`mmr_select()`	Pilih item dengan algoritma MMR
`_add_serendipity_items()`	Tambahkan item "kejutan" untuk variasi
`build_feed_cbf()`	**UTAMA:** Bangun feed rekomendasi
`search_cbf()`	**UTAMA:** Cari berdasarkan query

4 User Feedback Weighting (ufw.py)

Apa itu User Feedback Weighting?
UFW menyesuaikan hasil rekomendasi berdasarkan *feedback langsung* dari pengguna. Jika pengguna Like atau Skip suatu item, sistem akan "belajar" untuk memperbaiki rekomendasi.

🎯 **Analogi:** Bayangkan pelayan restoran yang memperhatikan masakanmu. Jika kamu bilang "enak!" pada sup ayam, pelayan akan menawarkan menu sup lainnya. Jika kamu bilang "tidak suka" pada makanan pedas, pelayan tidak akan menawarkan menu pedas.

Komponen UFW

Parameter	Simbol	Fungsi
Like Boost	α (alpha)	Menaikkan skor item mirip dengan item yang di-Like
Skip Penalty	β (beta)	Menurunkan skor item yang sudah di-Skip
Category Boost	γ (gamma)	Menaikkan skor item dari kategori favorit

Rumus UFW

$$\text{Skor Akhir} = \text{Skor Dasar} + \alpha\times(\text{Similarity ke Like}) - \beta\times(\text{Skip}) + \gamma\times(\text{Kategori})$$

Item mirip Like naik ↑ | Item di-Skip turun ↓ | Kategori favorit bonus ↑

Langkah 1: Hitung Centroid Like

Centroid adalah "titik pusat" dari semua item yang di-Like. Ini mewakili "preferensi ideal" pengguna.

> 🎯 **Analogi:** Jika kamu Like 3 pantai yang berbeda, centroid adalah "pantai ideal" yang menggabungkan karakteristik ketiganya. Item baru yang mirip dengan centroid ini kemungkinan besar akan kamu sukai juga.

```
def _compute_centroid_dense(X, indices):
    """Menghitung centroid dari item yang di-Like."""
    if not indices:
        return None

    # Ambil vektor dari semua item yang di-Like
    liked_vectors = X[list(indices)]

    # Hitung rata-rata (mean) dari semua vektor
    centroid = liked_vectors.mean(axis=0)

    return centroid
```

Langkah 2: Hitung Similarity ke Centroid

```
def _compute_like_similarity(gids, X, centroid):
    """Hitung seberapa mirip setiap item dengan centroid."""

    if centroid is None:
        return np.zeros(len(gids))  # Tidak ada Like

    # Hitung cosine similarity ke centroid
    similarities = cosine_similarity(X[gids], centroid)

    # Normalisasi ke 0-1
    return normalize_minmax(similarities)
```

Langkah 3: Hitung Skor Akhir

```
for gid in gids:
    # Mulai dari skor dasar
    score = base_score

    # Boost dari similarity ke Like
    score += alpha * like_similarity

    # Penalty jika item di-Skip
    if gid in blocked:
        score -= beta

    # Bonus dari kategori favorit
    score += gamma * category_count
```

Contoh Dampak UFW

Item	Sebelum UFW	Setelah UFW	Alasan
Pantai A (di-Like)	0.90	1.20	↑ Boost dari Like
Pantai C	0.80	**1.30**	↑ Mirip dengan item Like
Gunung B (di-Skip)	0.85	0.15	↓ Penalty dari Skip

5 Alur Kerja Aplikasi

Tab Feed (Tanpa Query)

1. **Filter:** User pilih kategori/kota/harga → Item yang tidak sesuai dihapus
2. **Scoring:** Hitung skor dasar dari rating setiap item
3. **MMR:** Pilih item yang relevan DAN beragam
4. **Serendipity:** Tambahkan beberapa item "kejutan"
5. **UFW:** Jika aktif, sesuaikan ranking berdasarkan Like/Skip
6. **Display:** Tampilkan hasil dalam bentuk kartu

Tab Search (Dengan Query)

1. **Input:** User ketik kata kunci (misal: "pantai snorkeling")
2. **Preprocessing:** Query dibersihkan (lowercase, hapus stopwords)
3. **TF-IDF:** Query diubah menjadi vektor angka
4. **Similarity:** Cari item dengan vektor paling mirip
5. **MMR:** Pilih hasil yang relevan dan beragam
6. **UFW:** Sesuaikan ranking berdasarkan feedback user

> **Perbedaan Feed vs Search:**
> - **Feed:** Tidak perlu query, skor dasar dari rating
> - **Search:** Perlu query, skor dasar dari similarity dengan query

6 Cara Menjalankan Aplikasi

Instalasi

```
# Buka terminal, masuk ke folder proyek
cd "D:\Projek\Freelancer\cl25_fw - System Recommender CBF (DONE)"

# Install dependencies
pip install streamlit scikit-learn scipy pandas numpy joblib
```

Menjalankan

```
# Jalankan aplikasi Streamlit
streamlit run app.py

# Buka browser: http://localhost:8501
```

Menggunakan Aplikasi

1. **Tab Feed:** Lihat rekomendasi berdasarkan rating dan filter
2. **Tab Search:** Ketik kata kunci dan klik "Cari"
3. ⭐ **Like:** Tandai item yang kamu suka → Hasil akan dipersonalisasi
4. 🚫 **Skip:** Tandai item yang tidak menarik → Item diturunkan
5. 🔖 **Bookmark:** Simpan item favorit untuk dilihat nanti

7 Parameter yang Dapat Diatur

Parameter Feed

Parameter	Range	Penjelasan
Top-N	5 - 40	Jumlah item yang ditampilkan
MMR λ	0.0 - 1.0	0 = beragam, 1 = relevan
Batas per Kategori	0 - 6	Maks item per kategori
Serendipity %	0 - 30%	Item "kejutan" yang ditambahkan

Parameter UFW

Parameter	Default	Penjelasan
α (alpha)	0.6	Boost untuk item mirip Like (0-2)
β (beta)	0.7	Penalty untuk item Skip (0-2)
γ (gamma)	0.02	Bonus kategori favorit (0-0.2)

> 💡 **Tips: Tips Tuning:**
> - Hasil monoton? → Turunkan MMR λ atau tingkatkan batas kategori
> - UFW terlalu agresif? → Turunkan α dan β
> - Ingin lebih personal? → Tingkatkan α dan aktifkan UFW

EcoTourism Recommender — Content-Based Filtering with User Feedback Weighting

Dibuat untuk membantu mahasiswa memahami sistem rekomendasi