

PlayEist Optimizer

Arkan Ramadhan Nugraha

241524033

Julian Dio Saputra

241524049

Background

Playlist musik seringkali memiliki transisi antar lagu yang kasar, dimana bisa terjadi perpindahan lagu dengan energi dan mood yang sangat berbeda. Hal ini dapat mengganggu pengalaman mendengarkan pengguna.

Sebagai contoh, terkadang dari lagu yang tenang bisa bertransisi menjadi lagu yang sangat energik sehingga terjadi "audio jumpscare"

Dampaknya alur playlist menjadi tidak Koheren dan pengalaman pengguna menurun

Problems

1

Proses Input manual

- Penambahan lagu spontan
- Berfokus pada koleksi bukan alur lagu

2

Sifat koleksi lagu

- genre yang beragam

3

Keterbatasan platform

- shuffle mode
- urutan tidak kontekstual

Algoritma internal



- fokus discovery
- metrik kemiripan sederhana

FOL akan solve bagaimana menentukan apakah transisi antar lagu dalam sebuah playlist bersifat harmonis atau kasar secara objektif dan konsisten, tanpa bergantung pada penilaian subjektif pengguna atau algoritma

Solution

1

Memformalkan aturan transisi lagu dengan First Order Logic menggunakan prolog sebagai inferensi dan mengubah penilaian subjektif sehingga membuat keputusan logis berbasis rule.

2

Hasil yang diharapkan adalah aplikasi dapat mendeteksi lagu ekstrem dan kasar sehingga tercipta rekomendasi urutan playlist yang optimal

Predicate and Rules

1. memiliki_atribut(<lagu>, <genre>, <tingkat energi>, <kecepatan tempo>)
2. jarak_graf(<lagu1>, <lagu2>, <nilai>)
3. beda_genre(<lagu1>, <lagu2>)
4. energi_kontras(<lagu1>, <lagu2>)
5. transisi_kasar(<lagu1>, <lagu2>)
6. transisi_harmonis(<lagu1>, <lagu2>)
7. butuh_lagu_bridge(<lagu1>, <lagu2>)
8. rekomendasi_urutan(<lagu1>, <lagu2>)

1. lagu_ekstrem(X) :- memiliki_atribut(X, _, sangat_tinggi, _) ; memiliki_atribut(X, _, rendah, _).
2. pembentuk_outlier(X) :- lagu_ekstrem(X).
3. wajib_kurasi_manual(X) :- pembentuk_outlier(X).
4. beda_genre(X, Y) :- memiliki_atribut(X, Genre1, _, _), memiliki_atribut(Y, Genre2, _, _), Genre1 \= Genre2.
5. energi_kontras(X, Y) :- memiliki_atribut(X, _, sangat_tinggi, _), (memiliki_atribut(Y, _, sedang, _) ; memiliki_atribut(Y, _, rendah, _)).
6. transisi_kasar(X, Y) :- energi_kontras(X, Y) ; (jarak_graf(X, Y, Jarak), Jarak > 0.7).
7. transisi_harmonis(X, Y) :- jarak_graf(X, Y, Jarak), Jarak <= 0.4.
8. butuh_lagu_bridge(X, Y) :- transisi_kasar(X, Y).
9. rekomendasi_urutan(X, Y) :- transisi_harmonis(X, Y).

```
● ● ●  
1 memiliki_atribut(beautiful_world, shoegaze, tinggi, sedang).  
2 memiliki_atribut(good_luck_babe, pop, tinggi, cepat).  
3 memiliki_atribut(russian_roulette, electronic, tinggi, cepat).  
4 memiliki_atribut(self_care, hip_hop, sedang, sedang).  
5 memiliki_atribut(tripping_wires, metal, sangat_tinggi, cepat).  
6  
7 % --- Fakta 11-15: Jarak Graf (Bobot Ketidakmiripan) ---  
8 % Semakin tinggi nilai, semakin tidak cocok (jauh)  
9 % Format: jarak_graf(Lagu1, Lagu2, Weight)  
10 jarak_graf(tripping_wires, self_care, 0.95).  
11 jarak_graf(tripping_wires, good_luck_babe, 0.8).  
12 jarak_graf(good_luck_babe, russian_roulette, 0.2).  
13 jarak_graf(beautiful_world, russian_roulette, 0.6).  
14 jarak_graf(self_care, good_luck_babe, 0.45).
```

Rule 3 Langkah

$\forall X \forall Y ((\text{memiliki_atribut}(X, g1, \text{ sangat_tinggi}, t1)$
 $\wedge (\text{memiliki_atribut}(Y, g2, \text{ sedang}, t2)$
 $\vee \text{memiliki_atribut}(Y, g3, \text{ rendah}, t3))) \rightarrow \text{energi_kontras}(X, Y)$

$\forall X \forall Y \forall N ((\text{energi_kontras}(X, Y) \vee (\text{jarak_graf}(X, Y, N) \wedge (N >$
 $0.7))) \rightarrow \text{transisi_kasar}(X, Y))$

$\forall X \forall Y (\text{transisi_kasar}(X, Y) \rightarrow \text{butuh_lagu_bridge}(X, Y))$

Mekanisme Unifikasi

Rule



```
1 % Rule 2 (Premis 2): Lagu ekstrem berpotensi menjadi outlier
2 % FOL: ∀X (lagu_ekstrem(X) → pembentuk_outlier(X))
3 pembentuk_outlier(X) :-
4   lagu_ekstrem(X).
5
```

Query ?- **pembentuk_outlier(tripping_wires)**

Melalui proses unifikasi, variabel X disubstitusikan dengan konstanta **tripping_wires**, sehingga diperoleh Most General Unifier (MGU): {X=**tripping_wires**}.

Selanjutnya, sistem memeriksa apakah **lagu_ekstrem(tripping_wires)** bernilai benar. Berdasarkan fakta atribut lagu:

memiliki_atribut(tripping_wires, metal, sangat_tinggi, cepat).

lagu tersebut memenuhi kriteria energi ekstrem, sehingga inferensi berhasil dan query dinyatakan TRUE.



Resolusi

Rule



```
1 % Rule 2 (Premis 2): Lagu ekstrem berpotensi menjadi outlier
2 % FOL: ∀X (lagu_ekstrem(X) → pembentuk_outlier(X))
3 pembentuk_outlier(X) :-  
4   lagu_ekstrem(X).
5
```

dapat ditulis dalam CNF sebagai:

$\neg \text{lagu_ekstrem}(X) \vee \text{pembentuk_outlier}(X)$

Untuk membuktikan query `pembentuk_outlier(tripping_wires)`, dilakukan negasi terhadap goal

$\neg \text{pembentuk_outlier}(\text{tripping_wires})$

Melalui proses resolusi antara klausa aturan dan fakta `lagu_ekstrem(tripping_wires)`, sistem menghasilkan klausa kosong. Klausa kosong ini menandakan bahwa kontradiksi telah tercapai, sehingga kesimpulan awal terbukti valid secara logika.



Hasie Uji

▼ Inferensi 1: Lagu Ekstrem

Tujuan: Rule 1 – Lagu dengan energi sangat tinggi atau rendah

`lagu_ekstrem(X)`

UJI INFERENSI 1

Ditemukan 1 hasil:

X	0	tripping_wires
---	---	----------------

▼ Inferensi 2: Pembentuk Outlier

Tujuan: Rule 2 – Lagu ekstrem berpotensi menjadi outlier

`pembentuk_outlier(X)`

UJI INFERENSI 2

Ditemukan 1 hasil:

X	0	tripping_wires
---	---	----------------

▼ Inferensi 3: Wajib Kurasi Manual

Tujuan: Rule 3 – Rantai inferensi (ekstrem → outlier → kurasi)

`wajib_kurasi_manual(X)`

UJI INFERENSI 3

Ditemukan 1 hasil:

X	0	tripping_wires
---	---	----------------

▼ Inferensi 4: Energi Kontras

Tujuan: Rule 5 – Perbedaan energi lagu signifikan

`energi_kontras(X, Y)`

UJI INFERENSI 4

Ditemukan 1 hasil:

X	0	tripping_wires	Y	self_care
---	---	----------------	---	-----------

Hasie Uji

▼ Inferensi 5: Transisi Kasar

Tujuan: Rule 6 – Energi kontras atau jarak graf besar

`transisi_kasar(X, Y)`

UJI INFERENSI 5

Ditemukan 3 hasil:

	X	Y
0	tripping_wires	self_care
1	tripping_wires	self_care
2	tripping_wires	good_luck_babe

▼ Inferensi 6: Butuh Lagu Bridge

Tujuan: Rule 8 – Kesimpulan dari transisi kasar

`butuh_lagu_bridge(X, Y)`

UJI INFERENSI 6

Ditemukan 3 hasil:

	X	Y
0	tripping_wires	self_care
1	tripping_wires	self_care
2	tripping_wires	good_luck_babe

▼ Inferensi 7: Transisi Harmonis

Tujuan: Rule 7 – Jarak graf kecil (transisi halus)

`transisi_harmonis(X, Y)`

UJI INFERENSI 7

Ditemukan 1 hasil:

	X	Y
0	good_luck_babe	russian_roulette

▼ Inferensi 8: Rekomendasi Urutan Playlist

Tujuan: Rule 9 – Pasangan lagu ideal

`rekомендации_urutan(X, Y)`

UJI INFERENSI 8

Ditemukan 1 hasil:

	X	Y
0	good_luck_babe	russian_roulette

Pada tahap ini dilakukan pengujian inferensi terhadap Knowledge Base menggunakan beberapa query utama. Hasil inferensi menunjukkan bahwa sistem mampu menarik kesimpulan kebijakan secara otomatis berdasarkan aturan FOL yang telah didefinisikan.

Uji Validitas Fallacy

Asumsi: Semua lagu bisa diurutkan atau di-shuffle secara bebas tanpa mempengaruhi kualitas playlist

Ini fallacy karena:

- Menggeneralisasi semua lagu
- Mengabaikan perbedaan atribut (energi, genre, jarak graf)



```
1 % Rule 2 (Premis 2): Lagu ekstrem berpotensi menjadi outlier
2 % FOL:  $\forall X$  (lagu_ekstrem( $X$ )  $\rightarrow$  pembentuk_outlier( $X$ ))
3 pembentuk_outlier( $X$ ) :-  
    lagu_ekstrem( $X$ ).
4
5
```



```
1 % FOL:  $\forall X, Y$  ( $\text{jarak}(X, Y) \leq 0.4 \rightarrow \text{transisi_harmonis}(X, Y)$ )
2 transisi_harmonis( $X$ ,  $Y$ ) :-  
    jarak_graf( $X$ ,  $Y$ , Jarak),
4     Jarak = $<= 0.4$ .
```



```
1 % FOL:  $\forall X, Y$  ( $\text{energi_kontras}(X, Y) \vee (\text{jarak}(X, Y) > 0.7 \rightarrow \text{transisi_kasar}(X, Y))$ )
2 transisi_kasar( $X$ ,  $Y$ ) :-  
    energi_kontras( $X$ ,  $Y$ );
4     (jarak_graf( $X$ ,  $Y$ , Jarak), Jarak > 0.7).
```

Ada lagu tertentu yang TIDAK boleh diperlakukan sama dengan lagu lain.

Hanya sebagian pasangan rekomendasi.

Ada pasangan lagu yang SECARA LOGIKA tidak layak disambung langsung.

Salah satu logical fallacy yang berhasil disanggah oleh Knowledge Base adalah generalization, yaitu asumsi bahwa semua lagu dapat diurutkan atau di-shuffle secara bebas tanpa mempengaruhi kualitas



Kesimpulan

- First Order Logic (FOL) mampu memformalkan kualitas transisi playlist secara objektif.
- Knowledge Base berhasil menyaring urutan lagu yang tidak valid secara logika.
- Sistem inferensi Prolog menghasilkan keputusan yang transparan dan dapat dijelaskan.

Rekomendasi Kebijakan

Larangan Shuffle Bebas pada Lagu Ekstrem

- Lagu dengan energi ekstrem ditandai sebagai outlier.
- Sistem menonaktifkan auto-shuffle untuk lagu tersebut.

(Mengatasi fallacy: "semua lagu bisa diperlakukan sama")

1

Penerapan Lagu Bridge pada Transisi Kasar

- Jika terdeteksi `transisi_kasar(X,Y)`, sistem wajib menyisipkan lagu bridge.
- Mencegah lonjakan energi yang mengganggu.

(Berbasis rule: `butuh_lagu_bridge(X,Y)`)

2

Rekomendasi Urutan Playlist Berbasis Bobot Graf

- Hanya pasangan lagu dengan $\text{jarak_graf} \leq 0.4$ yang direkomendasikan.
- Menghasilkan playlist yang lebih konsisten dan harmonis.

(Berbasis rule: `rekomendasi_urutan(X,Y)`)

3



Thank
You!

