Peran Dekomposisi Matriks dalam Algoritma Rekomendasi Discover Weekly Spotify

Ahsan Malik Al Farisi, 13523074 [1,2]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13523074@itb.ac.id, [2]themalique1910@gmail.com

***Abstrak*—Pengalaman menikmati musik di era digital telah meningkat karena adanya sistem rekomendasi, seperti Discover Weekly Spotify. Fitur ini menggabungkan tiga algoritma utama, terutama *collaborative filtering* yang berperan dalam memberikan rekomendasi lagu berdasarkan kesamaan selera musik pengguna. Konsep *collaborative filtering* didasari oleh konsep aljabar linier yaitu dekomposisi matriks yang dipakai untuk memproyeksikan pola dari data pengguna. Penelitian ini menganalisis peran dekomposisi matriks dalam algoritma *collaborative filtering* Discover Weekly Spotify utamanya konsep *matrix factorization.* Penelitian ini menunjukkan akan bagaimana aplikasi dari dekomposisi matriks dapat dipakai untuk mencari pengguna dengan preferensi yang serupa dan memberikan rekomendasi yang akurat.**

***Kata Kunci*—algoritma rekomendasi, *collaborative filtering*, dekomposisi matriks, *discover weekly*, spotify.**

I. Pendahuluan

A. Latar Belakang

Industri musik merupakan salah satu industri yang terus berkembang terutama di era digital saat ini. Pada awal kemunculannya, musik biasa dimainkan dengan LP (Long Play) atau vinyl atau piringan hitam yang diputar atau dimainkan dengan suatu alat bernama turntable. Namun, seiring perkembangan zaman musik mulai dapat dimainkan melalui CD (Compact Disk) dan saat ini dapat dimainkan cukup dengan menggunakan HP (Handphone) atau telepon genggam dan memainkan file MP3 yang tersedia.

Pada awalnya orang dapat mencari lagu secara random di toko rekaman dan membeli salah satunya. Hal ini merupakan menjadi salah satu pengalaman yang berharga dalam menikmati suatu musik, karena kita dapat memilih suatu lagu yang baru yang kita tidak tau isi di dalamnya dan kemudian dapat mendengarkannya. Sedangkan, saat ini dengan musik yang mayoritas dinikmati melalui platform digital, seperti Youtube Music, Spotify, dan JOOX lagu yang dapat kita pilih tidak terhingga dan pengalaman dalam memilih lagu secara acak tidak lagi dapat dirasakan. Namun pemakaian platform digital ini dan juga dengan tersedianya internet banyak sekali hal yang dapat kita eksplorasi dalam meningkatkan pengalaman menikmati musik.

Salah satu cabang eksplorasi musik yang saat ini ada adalah algoritma untuk mencari lagu baru yang sesuai dengan selera kita atau *song recommendation system.* Dengan algoritma ini kita dapat mencari lagu baru lagi tanpa harus mencari atau memilih secara acak di toko rekaman. Namun, sekarang lagu yang mirip dengan selera musik kita dapat langsung disajikan di platform digital yang kita pilih dan dapat kita langsung dengarkan. Contohnya adalah fitur *discover weekly* dari Spotify yang memberikan lagu baru yang mirip dengan selera musik kita dan disajikan setiap minggunya pada hari senin.

Discover Weekly Spotify merupakan salah satu fitur yang menggabungkan 3 algoritma utama yaitu *collaborative filtering, natural language processing* (NLP), dan *audio models.* Hal utama yang akan dibahas pada penelitian ini adalah tentang algoritma yang dipakai dalam *collaborative filtering*. *collaborative filtering* adalah algoritma yang dapat mencari kemiripan antara satu data (vektor) pengguna dengan data (vektor) pengguna lainnya. Sehingga dengan pencarian tersebut akan didapatkan vektor yang searah (selera yang sama) dari suatu pengguna dan kemudian akan diberikan rekomendasi musik baru dari pengguna lain yang memiliki selera yang sama. Kemudian tak cukup dari situ, algoritma NLP di Spotify berfungsi untuk meningkatkan rekomendasi nya dengan menganalisis metadata dari lagu dan juga informasi akan lagu tersebut di internet. Terakhir adalah *audio models* yang menganalisis dari lagu, seperti temponya dan mood dari lagu tersebut.

Algoritma dari *collaborative filtering* ini merupakan implementasi dari salah satu konsep yang sesuai dengan mata kuliah Aljabar Linier dan Geometri IF 2123. Konsep tersebut adalah tentang dekomposisi matriks dan perannya dalam mencari kemiripan selera lagu. Dekomposisi matriks adalah peran inti dari mencari pola utama (proyeksi data) yang sesuai dengan tiap selera dari pengguna untuk kemudian dikalkulasikan jarak antara tiap pola atau tiap selera dari pengguna.

B. Tujuan Makalah

Tujuan dari makalah ini adalah untuk dapat memahami secara mendalam akan peran penting dari dekomposisi nilai matriks dalam meningkatkan pengalaman menikmati musik terkhusus pada fitur rekomendasi musik Discover Weekly Spotify yang

menggunakan *collaborative filtering* sebagai salah satu proses atau algoritmanya.

II. Dasar Teori

A. Dekomposisi Matriks

Dekomposisi matriks adalah memfaktorkan sebuah matriks, misalnya A, menjadi hasil kali dari sejumlah matriks lain, P_1, P_2, P_3, …, P_k seperti berikut [x].

$$A = P_1 x P_1 x \dots x P_k$$

Dekomposisi ini dilakukan dengan maksud untuk mengekstrak suatu informasi penting yang mewakili data suatu matriks. Selain itu, dekomposisi matriks juga dapat berfungsi untuk mengungkap pola penting yang tidak terlihat langsung, mengurangi dimensi data, dan juga agar dapat melakukan operasi matematika dengan efektif.

Terdapat beberapa metode dari dekomposisi matriks, seperti dekomposisi matriks LU, dekomposisi matriks QR, dekomposisi nilai singular atau Singular Value Decomposition (SVD), Principal Component Analysis (PCA), *matrix factorization*, dan masih banyak lagi.

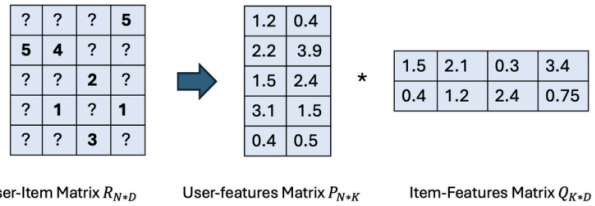
Contohnya adalah *matrix factorization* yang merupakan dekomposisi matriks R_{mxn} menjadi dua matriks dengan rank yang rendah P_{nxk} dan Q_{dxk} sehingga produk dari kedua matriks ini akan mengaproksimasi nilai orisinal matriksnya semirip mungkin, sehingga

$$P * Q^T \approx R$$

Misalkan sebuah matriks R_{MxN} dengan M merepresentasikan jumlah pengguna dan D merepresentasikan jumlah item (film). Tiap nilai matrix r_{ij} adalah sebuah *integer rating* dari skala 1 – 5 yang diberikan dari pengguna i kepada film j. Setelah dilakukan faktorisasi akan didapatkan dua matriks dengan rank lebih rendah, P_{NxK} yang merepresentasikan vektor laten berukuran K untuk tiap pengguna dan Q_{DxK} yang merepresentasikan vektor laten berukuran K tiap film. *Dot product* dari kedua matriks akan menghasilkan value yang mirip dengan mirip dengan value r_{ij}, sehingga dapat ditulis sebagai berikut.

$$\hat{r}_{ij} = p_i q_j^T = \sum_{k=1}^{K} p_{ik} q_{jk} \approx r_{ij}$$

Gambar 1. Aproksimasi nilai r_{ij} dari dot product matriks P_{NxK} dengan matriks Q_{DxK}
Sumber : builtin-matrix factorization



Gambar 2. Contoh dari faktorisasi matriks dengan angka yang dihasilkan secara random.
Sumber : builtin-matrixfactorization

B. Collaborative Filtering

Collaborative Filtering adalah salah satu tipe dari sistem rekomendasi yang merekomendasikan suatu barang berdasarkan kemiripan yang diukur antar pengguna dan/atau antar barang. Algoritma ini didasarkan dengan asumsi bahwa pengguna dengan selera yang sama akan memiliki pilihan barang yang sama. *Collaborative filtering* tidak memakai fitur dari data untuk mencari kemiripan dan merekomendasikannya (Content Based Information Retrieval), sedangkan kita mengklasifikasikan pengguna ke beberapa kelompok dengan tipe yang mirip untuk kemudian merekomendasikan tiap pengguna berdasarkan preferensi dari tiap kelompoknya.

Collaborative filtering pun terdiri dari dua jenis, yaitu pengguna based dan item based. pengguna based adalah mengukur kesamaan berdasarkan kesamaan antar pengguna dengan mengidentifikasi perilaku mirip, untuk kemudian merekomendasikan item yang disukai antar pengguna. Sedangkan item based berfokus pada kesamaan antar item dengan menganalisis hubungannya berdasarkan pengguna sebelumnya, untuk kemudian merekomendasikan item serupa yang relevan.

Contoh dari *collaborative filtering* adalah misal terdapat sistem rekomendasi film dan terdapat data yang mengandung pengguna sebagai barisnya dan tiap kolom merepresentasikan suatu film.

Users	Movie 1	Movie 2	Movie 3	Movie 4
User 1	5	4		5
User 2	4		3	
User 3		1		2
User 4	1	2		

Gambar 3. Representasi data skenario
Sumber : geeksforgeeks

Dari gambar di atas dapat dilihat bahwa pengguna 1 dan pengguna 2 memiliki *rating* yang mirip pada film pertama, dengan demikian film 4 dapat direkomendasikan kepada pengguna 2 dan film 3 dapat direkomendasikan kepada pengguna 1. Dari sini dapat dilihat juga bahwa pengguna 1 berkebalikan dengan pengguna 3 karena memiliki *rating* yang berkebalikan pada film 2. Dapat dikatakan juga bahwa pengguna 3 memiliki selera yang sama dengan pengguna 4 karena sama sama memberikan *rating* yang rendah untuk film 2, sehingga dapat dikatakan juga bahwa pengguna 3 tidak akan menyukai film 1 berdasarkan dari data pengguna 4.

Setelah didapatkan data di atas kita dapat menormalisasikan data dengan membulatkan rating yang kurang dari sama dengan tiga menjadi nol, dan di atasnya menjadi satu maka akan didapatkan data berikut.

Users	Movie 1	Movie 2	Movie 3	Movie 4
User 1	1	1		1
User 2	1		1	
User 3		0		0
User 4	0	0		

Gambar 4. Tabel Normalisasi Data
Sumber : geeksforgeeks

dari data di atas dapat kita lihat lebih jelas akan kemiripan antara pengguna 1 dengan 2, juga kemiripan antara pengguna 3 dan 4.

C. Perhitungan Kemiripan

Perhitungan kemiripan merupakan salah satu aspek penting dari suatu sistem rekomendasi. Hal ini terjadi karena dari data yang sudah didapat, misalnya data pengguna berupa vektor kita dapat menghitung kemiripannya satu pengguna dengan yang lain dengan metode perhitungan kemiripan. Contoh metode perhitungan yang dapat dipakai adalah euclidean distance dan cosine similarity. Euclidean distance adalah menghitung jarak antara satu titik dengan titik yang lain dalam satu garis di ruang euclides, dengan rumus sebagai berikut.

$$d(\mathbf{p}, \mathbf{q}) = \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}$$

Gambar 5. Persamaan Euclidean Distance
Sumber : gstatics

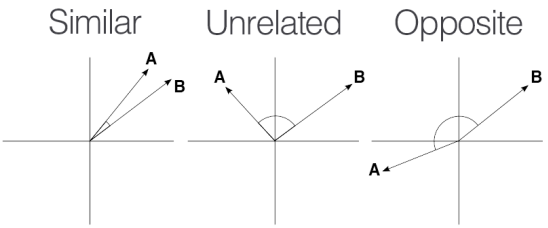
dengan demikian semakin jauh jarak suatu titik p dari q maka semakin tidak mirip kedua hal tersebut.

Selain itu terdapat juga metode cosine similarity atau mencari kemiripan dengan mencari sudut antara satu vektor dengan vektor yang lain. Semakin besar sudut cosine nya menunjukkan sudut yang semakin sempit antara dua pengguna, dengan demikian nilai kemiripannya akan semakin besar. Cosine similarity dapat dihitung dengan rumus berikut.

$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}}$$

Gambar 6. Persamaan Cosine Similarity
Sumber : towardsdatascience

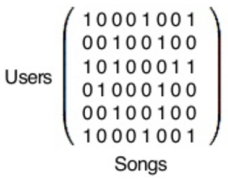
Adapun visualisasi dari cosine similarity adalah sebagai berikut.



Gambar 7. Visualisasi Cosine Similarity
Sumber : pyimagesearch

III. Penerapan Dekomposisi Matriks Dalam Discover Weekly Spotify

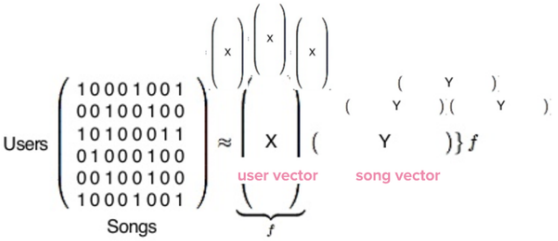
Algoritma dari Discover Weekly Spotify memanfaatkan properti dari matriks dengan tiap pengguna berperan sebagai satu baris dan tiap kolom adalah lagu yang dapat dimainkan. Sehingga akan terdapat matriks yang sangat besar yang mengandung 140 juta pengguna sebagai baris dan 30 juta lagu dalam data set spotify yang berperan sebagai kolom.



Gambar 8. Ilustrasi matriks hubungan antara pengguna dan lagu
Sumber : Linkedin - Discover Weekly Spotify Algorithm

Dari data di atas dapat kita demonstrasikan, misalnya penulis mendengarkan lagu Sialan - Juicy Lucy maka pada bagian r_{ij} akan bernilai 1 dengan i sebagai pengguna dan j sebagai kolom lagu. Nilai 1 menandakan bahwa pengguna telah mendengarkan lagu tersebut dan nilai 0 menunjukkan sebaliknya. Namun, pada kenyataannya Spotify tidak hanya menggunakan nilai 1 dan 0 tetapi memakai jumlah pemutaran lagu oleh pengguna. Sehingga jika saya memutar lagu yang sama sebanyak 20 kali maka pada bagian r_{ij} di matriks akan bernilai 20. Tentu saja akan terdapat banyak sekali nilai nol pada matriks tersebut karena banyaknya lagu tetapi beberapa nilai yang ada menjadi informasi penting yang dapat dianalisis lebih lanjut.

Kemudian dari data matriks yang sangat banyak dilakukan dekomposisi matriks dengan faktorisasi sehingga akan dihasilkan dua tipe vektor, yaitu X dan Y. Vektor X adalah vektor pengguna yang merepresentasikan selera dari seseorang dan vektor Y adalah vektor lagu dan merepresentasikan profile dari suatu lagu.



Gambar 9. Ilustrasi vektor X dan Y
Sumber : Linkedin - Discover Weekly Spotify Algorithm

Kedua vektor tersebut akan hanya berisi angka yang tidak memiliki makna banyak jika digunakan secara langsung, namun data tersebut dapat dibandingkan untuk mendapatkan hasil yang mendalam. Vektor dapat saling dibandingkan dengan menggunakan perhitungan kemiripan cosine. Misal selera pengguna atau vektor X yang dibandingkan menggunakan cosine similarity, kemudian akan didapatkan nilai kemiripan yang besar dengan vektor lain yang membentuk sudut gabung terkecil. Sehingga setelah mendapatkan selera pengguna lain yang mirip maka lagu dapat direkomendasikan misal antara lagu di pengguna A dapat direkomendasikan ke pengguna B dan juga sebaliknya lagu di pengguna B dapat direkomendasikan ke pengguna A. Metode membandingkan pengguna adalah salah satu tipe dari *collaborative filtering*, yaitu metode pengguna based. Sedangkan metode item based dapat dilakukan dengan memilih profil dari lagu atau vektor Y dan membandingkannya dengan lagu lain sehingga yang memiliki kemiripan terbesar dapat dikategorikan sebagai lagu dengan profil yang sama. Sehingga jika kita ingin mencari lagu yang sama kita dapat mencari vektor Y yang mirip dengan yang lain. Aplikasi dari kemiripan vektor tersebut termasuk dalam salah satu fitur dari Spotify yaitu Song Radio yang menampilkan lagu lain yang mirip dengan lagu yang kita pilih.

IV. Implementasi Dekomposisi Matriks dalam Algoritma Discover Weekly Spotify

A. Penjelasan Kode

```python
def matrix_factorization(matrix, num_users, num_items, latent_features=2,
                         learning_rate=0.001, regularization=0.05, num_ite
rations=100):
    """Faktorisasi matriks"""
    # Inisialisasi matriks pengguna dan item
    user_matrix = np.random.rand(num_users, latent_features)
    item_matrix = np.random.rand(num_items, latent_features)

    print("Start faktorisasi matriks")

    for iteration in range(num_iterations):
        for i in range(num_users):
            for j in range(num_items):
                if matrix[i, j] > 0:  # Perbarui entri yang tidak nol
                    prediction = np.dot(user_matrix[i, :], item_matrix[j,
:].T)
                    error = matrix[i, j] - prediction

                    for k in range(latent_features):
                        user_matrix[i, k] += learning_rate * (2 * error *
item_matrix[j, k] -
                                                              regularizatio
n * user_matrix[i, k])
                        item_matrix[j, k] += learning_rate * (2 * error *
user_matrix[i, k] -
                                                              regularizatio
n * item_matrix[j, k])

        # Batasi nilai untuk menghindari overflow
        user_matrix = np.clip(user_matrix, -10, 10)
        item_matrix = np.clip(item_matrix, -10, 10)

        # Cetak progres
        if iteration % 10 == 0:
            total_error = np.sum((matrix - np.dot(user_matrix, item_matri
x.T))**2)
            print(f"Iterasi {iteration}, Error: {total_error:.4f}")

    return user_matrix, item_matrix
```

Gambar 10. Implementasi dari dekomposisi matriks faktorisasi

Kode ini melakukan dekomposisi matriks dengan metode faktorisasi matriks dengan pendekatan *Gradient Descent*. Pertama kode menerima argumen berupa matriks pengguna item, jumlah pengguna, jumlah item, latent features atau jumlah pola tersembunyi yang ada pada data, learning rate atau seberapa besar pembaruan nilai pada matriks di tiap iterasi, regularization atau nilai yang berfungsi mengurangi overfitting, dan terakhir terdapat jumlah iterasi.

Pendekatan dari *Gradient Descent* adalah pertama dengan menginisialisasi matriks P (pengguna_matrix) dan Q (item_matrix) dengan nilai random, kemudian dia akan mengiterasi sesuai dengan jumlah argumen yang ada dengan memastikan hanya mengecek nilai matriks yang lebih dari nol. Kemudian fungsi tersebut akan memperbarui nilainya dengan cara mengalikan learning rate dengan mempertimbangkan nilai error yang ada dan menguranginya dengan nilai regularisasinya. Kemudian nilai dari tiap matriks akan dibatasi pada interval [-10,10] untuk menghindari underflow dan overflow. Setelah itu akan dihitung RMSE (Root Mean Square Error) tiap 10 iterasi. Dalam tiap iterasi error yang ada dikurangi dan pada akhirnya akan menuju nilai lokal yang optimal. Kemudian pada akhirnya fungsi akan mengembalikan pengguna matriks dan item matriks yang terdiri dari kumpulan vektor pengguna dan lagu. Fungsi ini dapat digunakan untuk memberikan rekomendasi berdasarkan hasil pengguna matriks dan item matriks.

Dalam mencari rekomendasi lagu pertama harus dilakukan inisialisasi matriks dan di dalamnya pencarian pengguna lain yang mirip cukup dilakukan dengan menggunakan fungsi cosine similarity.

```python
def __init__(self, matrix, user_matrix, item_matrix):
        self.matrix = matrix
        self.user_matrix = user_matrix
        self.item_matrix = item_matrix
        self.cos_sim = cosine_similarity(user_matrix)
```

Gambar 11. Inisialisasi fungsi untuk menganalisis vektor

Selain dari itu terdapat juga fungsi get_recommendation yang menerima argumen ID pengguna yang akan diberikan rekomendasi, Daftar pengguna yang mirip, dan jumlah yang akan dipilih dengan default valuenya nol.

```python
def _get_recommendations(self, user_id, similar_users, top_n=10):
        """Rekomendasi dengan skor yang dinormalisasi"""
        unrated_songs = np.where(self.matrix[user_id] == 0)[0]
        recommendations = []

        for song in unrated_songs:
            total_similarity = 0
            weighted_sum = 0
            supporters = []

            for similar_user in similar_users:
                if self.matrix[similar_user][song] > 0:
                    similarity = self.cos_sim[user_id][similar_user]
                    rating = self.matrix[similar_user][song]

                    # Akumulasi jumlah berbobot dan total kesamaan
                    weighted_sum += rating * similarity
                    total_similarity += similarity

                    supporters.append({
                        'user_id': similar_user,
                        'rating': rating,
                        'similarity': similarity
                    })

            if supporters:
                # Normalisasi skor dengan membaginya dengan total kesamaan
                normalized_score = (weighted_sum / total_similarity) if total_
similarity > 0 else 0

                recommendations.append({
                    'song_id': song,
                    'score': normalized_score,
                    'raw_score': weighted_sum,  # Simpan skor mentah untuk ref
erensi
                    'num_supporters': len(supporters),
                    'supporters': supporters
                })

        # Urutkan berdasarkan skor yang dinormalisasi dan kembalikan top N
        recommendations.sort(key=lambda x: x['score'], reverse=True)
        return recommendations[:top_n]
```

Gambar 12. Implementasi fungsi untuk mendapatkan rekomendasi lagu yang sesuai

Pertama akan dicari indeks lagu yang belum diberikan *rating* oleh pengguna atau yang memiliki nilai 0 dalam matriks dan juga dibuat array kosong untuk menyimpan rekomendasi. Selanjutnya akan dilakukan iterasi untuk lagu yang belum di-*rating* dan ada inisialisasi variabel lokal. Kemudian untuk tiap pengguna yang mirip akan dicek apakah pengguna memberikan *rating* pada lagu dan akan mengambil nilai kemiripan antara pengguna target dan pengguna yang mirip juga mengambil *rating* yang oleh pengguna yang mirip untuk lagu ini. Setelah itu akan dihitung jumlah bobotnya (*rating* * kesamaan) dan menambahkan nilai kemiripan ke total nya. Kemudian jika ada yang mendukung, data tersebut akan dimasukkan ke dalam array dan juga akan dihitung skor normalisasi dengan membagi jumlah berbobot dengan total kemiripan. Setelah didapatkan skornya maka akan dimasukkan ke dalam list rekomendasi dan akan disortir.

B. Hasil dan Pembahasan

Setelah dilakukan implementasi algoritma akan dilakukan pengetesan terhadap data dalam skala kecil yang terdiri dari 5 pengguna dan 10 lagu, dengan lagu 1-5 adalah lagu rock dan lagu 6-10 adalah lagu pop. Penilaian dari pengguna pun berada pada interval 1-5, dengan nilai tersebut menunjukkan kesukaan pengguna terhadap lagu dengan asumsi bahwa jika pengguna hanya mendengarkan sekali maka pengguna tidak suka dan jika pengguna mendengarkan sebanyak 5 kali maka berarti pengguna sangat suka akan lagu tersebut.

```python
ef create_test_data():
    """
    Membuat data uji dengan pola :
    - Pengguna 0 suka lagu rock (rating tinggi untuk lagu 0-4)
    - Pengguna 1 suka lagu rock (mirip dengan Pengguna 0)
    - Pengguna 2 suka lagu pop (rating tinggi untuk lagu 5-9)
    - Pengguna 3 memiliki selera campuran tetapi sedikit mirip dengan Pengguna
0
    - Pengguna 4 memiliki selera yang sangat berbeda
    """
    # Matriks kosong: 5 pengguna x 10 lagu
    matrix = np.zeros((5, 10))

    # Pengguna 0 (pengguna target) - Penggemar rock
    matrix[0] = [5, 5, 0, 5, 0, 1, 0, 1, 0, 1]  # Suka rock, tidak suka pop

    # Pengguna 1 - Mirip dengan Pengguna 0 (penggemar rock)
    matrix[1] = [5, 4, 5, 5, 5, 1, 1, 0, 1, 0]  # Selera sangat mirip dengan Pe
ngguna 0

    # Pengguna 2 - Selera berbeda (penggemar pop)
    matrix[2] = [1, 1, 0, 1, 0, 5, 5, 5, 5, 5]  # Selera berlawanan dengan Peng
guna 0

    # Pengguna 3 - Selera campuran tetapi cenderung ke rock
    matrix[3] = [4, 4, 0, 3, 0, 2, 3, 0, 2, 0]  # Agak mirip dengan Pengguna 0

    # Pengguna 4 - Selera sangat berbeda
    matrix[4] = [2, 1, 0, 1, 0, 5, 5, 5, 4, 5]  # Sangat berbeda dari Pengguna
0

    # Membuat metadata lagu
    songs = pd.DataFrame({
        'song_id': range(10),
        'genre': ['Rock', 'Rock', 'Rock', 'Rock', 'Rock',
                  'Pop', 'Pop', 'Pop', 'Pop', 'Pop'],
        'title': [f'Lagu Rock {i+1}' if i < 5 else f'Lagu Pop {i-4}'
                  for i in range(10)]
    })

    return matrix, songs
```

Gambar 13. Kode untuk membuat dataset

Berikut adalah implementasi test code implementasi yang terdapat fungsi tambahan untuk mengecek hasil analisis.

```python
def main():
    # Membuat data uji
    matrix, songs_df = create_test_data()
    print(matrix)

    # Menyimpan data uji ke CSV
    pd.DataFrame(matrix).to_csv('test_matrix.csv', index=True)

    # Mendapatkan dimensi matriks
    num_users, num_items = matrix.shape

    # Melakukan faktorisasi matriks
    user_matrix, item_matrix = matrix_factorization(matrix, num_users, num_item
s)

    print("User Matrix:")
    print(user_matrix)
    print()
    print("Item Matrix:")
    print(item_matrix)
    # Membuat analyzer
    analyzer = MusicTasteAnalyzer(matrix, user_matrix, item_matrix)

    # Memverifikasi rekomendasi
    verify_recommendations(matrix, user_matrix, item_matrix, analyzer, songs_d
f)
```

Gambar 14. Kode untuk mengetes dataset

Adapun contoh output dari fungsi tersebut seperti berikut.

```
User Matrix:
[[1.37250182 1.02974196]
 [1.22629695 1.87693538]
 [1.87216993 1.2794116 ]
 [1.42514069 1.25223284]
 [1.99351921 1.13247144]]

Item Matrix:
[[1.0065399  1.24010298]
 [0.82622102 1.13944324]
 [1.12901916 1.14560796]
 [1.14233662 0.81235126]
 [0.69046779 1.17026555]
 [1.47895266 0.47874938]
 [1.36227537 0.94482319]
 [1.48005897 0.95098666]
 [1.02796392 1.00987933]
 [1.38981292 1.25564545]]
```

Gambar 15. Output matrix yang terdiri dari pengguna

matriks dan item matriks

Setelah fungsi dijalankan akan menghasilkan output seperti berikut yang menunjukkan bahwa pengguna target (0) paling mirip dengan pengguna 1 dan pengguna 3 sesuai dengan data yang telah disusun.

```
1. Kesamaan Pengguna (seharusnya paling mirip dengan Pengguna 1, lalu Pengguna 3):
Pengguna 1: Kesamaan = 0.979
Lagu yang disukai bersama (rating >= 4): ['Lagu Rock 1', 'Lagu Rock 2', 'Lagu Rock 4']
Pengguna 3: Kesamaan = 0.885
Lagu yang disukai bersama (rating >= 4): ['Lagu Rock 1', 'Lagu Rock 2']
Pengguna 2: Kesamaan = 0.685
Lagu yang disukai bersama (rating >= 4): []
Pengguna 4: Kesamaan = 0.668
Lagu yang disukai bersama (rating >= 4): []
```

Gambar 16. Hasil kemiripan dengan pengguna lain

Selain itu didapatkan juga lagu yang direkomendasikan terhadap pengguna target sesuai dengan konsep *collaborative filtering* misalnya lagu yang paling direkomendasikan pertama adalah lagu rock 3 karena pengguna target memiliki kemiripan terbesar dengan pengguna 1 dan juga pengguna 1 telah mendengarkan lagu rock 3 sedangkan pengguna target belum. Hal yang sama terjadi juga pada lagu rock 5. Selanjutnya algoritma menyarankan lagu pop 2 dan lagu pop 4 karena pengguna target tidak menyukai lagu pop.

```
2. Rekomendasi Lagu (seharusnya lebih memilih lagu rock yang belum dirating):

Direkomendasikan: Lagu Rock 3 (Genre: Rock)
Skor: 5.000
Didukung oleh:
- Pengguna 1 memberi rating 5.0 (kesamaan: 0.979)

Direkomendasikan: Lagu Rock 5 (Genre: Rock)
Skor: 5.000
Didukung oleh:
- Pengguna 1 memberi rating 5.0 (kesamaan: 0.979)

Direkomendasikan: Lagu Pop 2 (Genre: Pop)
Skor: 3.233
Didukung oleh:
- Pengguna 1 memberi rating 1.0 (kesamaan: 0.979)
- Pengguna 3 memberi rating 3.0 (kesamaan: 0.885)
- Pengguna 2 memberi rating 5.0 (kesamaan: 0.685)
- Pengguna 4 memberi rating 5.0 (kesamaan: 0.668)

Direkomendasikan: Lagu Pop 4 (Genre: Pop)
Skor: 2.750
Didukung oleh:
- Pengguna 1 memberi rating 1.0 (kesamaan: 0.979)
- Pengguna 3 memberi rating 2.0 (kesamaan: 0.885)
- Pengguna 2 memberi rating 5.0 (kesamaan: 0.685)
- Pengguna 4 memberi rating 4.0 (kesamaan: 0.668)
```

Gambar 17. Hasil rekomendasi lagu kepada pengguna target

Terakhir didapatkan preferensi dari pengguna target yang menyukai rock dan tidak menyukai lagu pop dan telah sesuai dengan data yang ada.

```
3. Preferensi Dikenal Pengguna 0:
Lagu yang Sangat Disukai:
- Lagu Rock 1 (Rock): 5.0
- Lagu Rock 2 (Rock): 5.0
- Lagu Rock 4 (Rock): 5.0

Lagu yang Tidak Disukai:
- Lagu Pop 1 (Pop): 1.0
- Lagu Pop 3 (Pop): 1.0
- Lagu Pop 5 (Pop): 1.0
```

Gambar 18. Preferensi lagu pengguna target

C. Evaluasi Implementasi

Implementasi yang dilakukan sudah cukup menggambarkan akan cara kerja dari faktorisasi matriks dalam sistem rekomendasi Discover Weekly Spotify. Namun di dalamnya terdapat beberapa batasan dan kekurangan karena implementasi ini merupakan hal yang berguna untuk menggambarkan :

1. Tidak terdapatnya penanganan bias atau kecenderungan pengguna untuk memberikan *rating* tinggi atau rendah dan tidak ada perhitungan bias item untuk lagu yang secara umum populer atau tidak populer. Selain itu juga tidak ada penanganan bias temporal atau perubahan selera musik seiring berjalannya waktu.
2. Implementasi hanya menggunakan dataset kecil, sedangkan Spotify harus mengolah jutaan lagu dan pengguna. Selain itu tidak ada penanganan *cold-start problem* atau masalah yang muncul ketiak terdapat lagu baru dan belum memiliki pendengar manapun.
3. Tidak adanya optimasi untuk komputasi pada skala besar.
4. Tidak mempertimbangkan *latent feature* lain di dalam musik seperti urutan lagu, durasi, dan waktu mendengarkan.

V. Kesimpulan

Penelitian ini membahas peran dekomposisi matriks dalam algoritma rekomendasi lagu Discover Weekly Spotify. Dengan menggunakan konsep *matrix factorization,* sistem dapat mengidentifikasi pola-pola dalam data lagu yang besar, menghasilkan matriks pengguna dan matriks lagu. Selanjutnya, kemiripan antara vektor pengguna dihitung dengan *cosine similarity,* dengan hasil kemiripan tertinggi menunjukkan pengguna yang paling mirip dalam selera musik. Proses ini memungkinkan sistem untuk merekomendasikan lagu dari pengguna lain dengan selera musik yang mirip. Penelitian ini juga menunjukkan pentingnya dekomposisi matriks sebagai dasar dari algoritma sistem rekomendasi modern. Implementasi ini memberikan dampak yang signifikan terhadap algoritma sistem rekomendasi dan juga memperkaya pengalaman pengguna dalam menikmati musik digital secara lebih personal.

VI. Lampiran

Sumber kode implementasi dapat dilihat di Github Repository berikut :
https://github.com/hhansl/Implementasi-Makalah-Algeo

VI. Ucapan Terima Kasih

Saya ingin mengucapkan syukur kepada Tuhan YME. karena berkatnya makalah ini dapat selesai dengan baik. Selanjutnya saya ingin berterima kasih kepada pak Rinaldi Munir selaku dosen mata kuliah Aljabar Linier

dan Geometri IF 2123 yang telah memberikan pemahaman mendalam akan mata kuliah ini, sehingga saya dapat mengerjakan makalah ini dengan pemahaman penuh. Selain itu saya juga ingin berterima kasih kepada rekan saya di mata kuliah IF 2123 ini yang kerap saling mendukung dalam juga terutama dalam pembuatan makalah ini.

Referensi

[1] R. Munir. "Aljabar Geometri: Singular Value Decomposition Bagian 1," 2023. https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2023-2024/Algeo-21-Singular-value-decomposition-Bagian1-2023.pdf. [Diakses pada 28 Desember 2024].

[2] Alooba, "Matrix Decomposition," ALOOBA, 2023. https://www.alooba.com/skills/concepts/machine-learning/matrix-decomposition/. [Diakses pada 28 Desember 2024].

[3] V. Velardo. "Spotify's Discover Weekly Explained: Breaking from Your Music Bubble, or Maybe Not," Medium, 2024. https://medium.com/the-sound-of-ai/spotifys-discover-weekly-explained-breaking-from-your-music-bubble-or-maybe-not-b506da144123. [Diakses pada 28 Desember 2024].

[4] A. Tawanghar. "Spotify's Discover Weekly: How Machine Learning Finds Your Music," LinkedIn, 2024. https://www.linkedin.com/pulse/spotifys-discover-weekly-how-machine-learning-finds-your-tawanghar/. [Diakses pada 28 Desember 2024].

[5] GeeksforGeeks, "Collaborative Filtering in Machine Learning," GeeksforGeeks, 2024. https://www.geeksforgeeks.org/collaborative-filtering-ml/.[Diakses pada 28 Desember 2024].

[6] Google Developers, "Collaborative Filtering Basics," Google Developers, 2024. https://developers.google.com/machine-learning/recommendation/collaborative/basics. [Diakses pada 28 Desember 2024].

[7] A. Zonoozi, "Matrix Factorization for Recommender Systems," BuiltIn, 2024. https://builtin.com/articles/matrix-factorization. [Diakses pada 28 Desember 2024].

Pernyataan

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 27 Desember 2024

Ahsan Malik Al Farisi - 13523074