CODE REPRODUCTION

untuk memenuhi Ujian Akhir Semester

Mata Kuliah Komputasi Lanjut dan Big Data

Dosen Pengampu:

Prof Alhadi Bustaman, Ph.D.

Risman Adnan, M.Si.

disusun oleh:

Elizabeth Lilies Megawati 2206014731

Program Magister Matematika

Fakultas Matematika dan Ilmu Pengetahuan Alam

Universitas Indonesia

2022

Daftar Isi

Daftar Gambar

1. Pendahuluan

Collaborative filtering adalah proses memprediksi melalui mesin rekomendasi. Mesin rekomendasi menganalisis informasi tentang pengguna dengan selera yang sama untuk menilai kemungkinan individu target akan menikmati sesuatu, seperti video, buku, atau item. Penyaringan kolaborative juga digunakan untuk memilih konten dan iklan untuk individu di media sosial. (TechTarget)

Cara kerja teknik ini adalah dengan memanfaatkan data pada komunitas dengan cara mencari kemiripan antar pengguna, yaitu mengasumsikan bahwa pengguna yang memiliki preferensi serupa di masa lalu cenderung memiliki preferensi yang sama di masa depan. Dalam menghasilkan rekomendasi, sistem memerlukan data untuk mendapatkan ide preferensi pengguna, dimana nantinya akan dibuat suatu rekomendasi berdasarkan preferensi tersebut.
Terdapat dua teknik pada *memory-based collaboratuve filtering*, yaitu: (builtin)

a. *User-based,* yaitu mengukur kesamaan antara target pengguna dan pengguna lain
b. *Item-based,* yaitu mengukur kesamaan antara item yang diberi rating atau berinteraksi dengan pengguna target dan item lainnya.

Selanjutnya dalam dokumentasi ini akan dijelaskan secara detail repositori berjudul ”*review-on-collaborating-filtering”* yang dapat diakses pada https://github.com/nzhinusoftcm/review-on-collaborative-filtering.

Repositori tersebut menyajikan implementasi komprehensif dari sistem rekomendasi penyaringan kolaboratif, dari penyaringan kolaboratif berbasis memori hingga algoritma *machine learning* yang lebih canggih. Hal tersebut diawali dengan menerapkan dasar-dasar algoritma penyaringan kolaboratif seperti pengarsipan kolaboratif berbasis pengguna (*user-based collaborative filtering*) yang juga dikenal sebagai penyaringan kolaboratif antar pengguna (*user-to-user collaborative filtering*) dan penyaringan kolaboratif berbasis item (*item-based* atau *item-to-item collaborative filtering*). Repositori tersebut juga menyajikan model seperti Dekomposisi Nilai Singular (SVD), Faktorisasi Matriks (MF), Faktorisasi Matriks Non Negatif (NMF) dan Faktorisasi Matriks yang Dapat Dijelaskan (EMF)

Topik yang dicakup dalam repositori adalah:

1. Eksplorasi Data
2. Penyaringan Kolaborasi berbasis Memori
3. Pengurangan Dimensi
4. Perbandingan Performa

2. Pembahasan

2.1. Eksplorasi Data

Data yang digunakan dalam repositori ini dapat diakses menggunakan kode berikut:

```
import os

if not (os.path.exists("recsys.zip") or os.path.exists("recsys")):
   !wget https://github.com/nzhinusoftcm/review-on-collaborative-
filtering/raw/master/recsys.zip
   !unzip recsys.zip
```

Sebagai contoh, dipilih dataset mlLatestSmall yang terdapat dua dataframe pada file,yaitu:

a. data ‘ratings’ sebanyak 100836 sampel yang terdiri dari variabel ‘userid’, ‘itemid’, ‘rating’, dan ‘timestamp’; serta tidak ada data yang hilang.
b. data ‘movies’ sebanyak 9742 sampel yang terdiri dari variabel ‘itemid’, ‘title’, dan ‘genres’; serta tidak ada data yang hilang

Kedua dataframe tersebut dapat digabung menjadi satu dataframe karena masing-masing dataset memiliki satu variabel yang sama, yaitu ‘itemid’, Namun, karena sampel pada dataframe ‘ratings’, tidak sama dengan dataframe ‘movies’, maka akan terdapat *missing values.*

	userid	itemid	rating	timestamp
0	1	1	4.0	964982703
1	1	3	4.0	964981247
2	1	6	4.0	964982224
3	1	47	5.0	964983815
4	1	50	5.0	964982931

```
RangeIndex: 100836 entries, 0 to 100835
Data columns (total 4 columns):
 #   Column     Non-Null Count   Dtype
---  ------     --------------   -----
 0   userid     100836 non-null  int64
 1   itemid     100836 non-null  int64
 2   rating     100836 non-null  float64
 3   timestamp  100836 non-null  int64
dtypes: float64(1), int64(3)
memory usage: 3.1 MB
```

Gambar 1. DataFrame ‘ratings’ dari dataset *mlLatestSmall*

	itemid	title	genres
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
1	2	Jumanji (1995)	Adventure|Children|Fantasy
2	3	Grumpier Old Men (1995)	Comedy|Romance
3	4	Waiting to Exhale (1995)	Comedy|Drama|Romance
4	5	Father of the Bride Part II (1995)	Comedy

```
RangeIndex: 9742 entries, 0 to 9741
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   itemid  9742 non-null   int64
 1   title   9742 non-null   object
 2   genres  9742 non-null   object
dtypes: int64(1), object(2)
```

Gambar 1. DataFrame ‘movies’ dari dataset *mlLatestSmall*

2.2. *Memory-Based Collaborative Filtering*

Sebelumnya telah dijelaskan bahwa *memory-based collaborative filtering* terdiri dari dua teknik, yaitu *user-based* dan *item-based.* Kedua algoritma tersebut berjalan dalam tiga tahap, yaitu: komputasi kesamaan (antara pengguna atau item), prediksi rating (menggunakan rating pengguna atau item serupa), dan rekomendasi Top-N.

2.2.1. *User-Based Collaborative Filtering*

Algoritma penyaringan kolaboratif berbasis pengguna dapat merekomendasikan item berdasarkan kesamaan pengguna. Penyaringan kolaboratif berbasis pengguna membuat rekomendasi berdasarkan interaksi *user-item* di masa lalu. Asumsi di balik algoritma ini adalah bahwa pengguna serupa menyukai item serupa. (medium)

Ide pada algoritma ini adalah dengan menjadikan u sebagai pengguna yang direncanakan untuk membuat rekomendasi, sedemikian hingga: 1) mencari pengguna lain yang sebelumnya memberi rating yang mirip dengan pengguna u; 2) menggunakan rating mereka pada item lain untuk memprediksi apa yang saat ini pengguna suka.

Algoritma penyaringan kolaboratif berbasis pengguna dalam repositori yang dibahas dalam laporan ini memiliki langkah-langkah sebagai berikut.

1) Mengidentifikasi G_u, himpunan k users yang mirip dengan pengguna aktif u
   Mengidentifikasi himpunan G_u dari pengguna paling banyak k yang mirip.G_u adalah grup pengguna yang mirip dengan pengguna aktif u. Kemiripan antara pengguna u dan v dapat dihitung dengan perhitungan kemiripan *cosine*:

$$w_{u,v} = \frac{\overrightarrow{r_u}.\overrightarrow{r_v}}{\left|\left|\overrightarrow{r_u}\right|\right|_2 * \left|\left|\overrightarrow{r_v}\right|\right|_2} = \frac{\sum_{i \epsilon I} r_{u,i} r_{v,i}}{\sqrt{\sum_{i \epsilon I}\left(r_{u,i}\right)^2}\sqrt{\sum_{i \epsilon I}\left(r_{v,i}\right)^2}}$$

…(1)

$w_{u,v}$ adalah tingkat kemiripan antara pengguna u dan v. Formula tersebut dihitung untuk semua $v \epsilon U$, dimana U adalah humpunan semua pengguna.

Pada repositori ini:

a. Pertama, dibuat terlebih dahulu model *nearest neighbour* dengan sklearn melalui fungsi create_model(). Fungsi ini membuat dan menyesuaikan sebuah model nearest neighbors dengan rating pengguna. Kemudian dapat dipilih cosine or euclidian berbasis metrik similarity. n_neighbors=21 didefinisikan sebagai banyaknya neighbors yang kembali yang diperoleh dari G_u yang memuat 20 pengguna yang mirip ditambahkan dengan pengguna aktif u. Setiap baris r_u dari matriks rating R merepresentasikan rating dari pengguna u pada semua item dari database. Untuk rating yang hilang, diganti dengan 0.0.
b. Fungsi nearest_neighbors()mengembalikan *similarities,* yaitu jarak antara *neighbors* dari referensi pengguna; dan mengembalikan *neighbors,* yaitu *neighbors* dari referensi pengguna dalam urutan kemiripan secara menurun.
c. Memanggil fungsi create_model() dan nearest_neighbors()untuk membuat model k-NN dan menghitung *nearest neighbors* untuk pengguna tertentu.

2) Mencari kandidat item

Mencari himpunan C kandidat item, yang dibeli oleh grup dan bukan dibeli oleh pengguna aktif u. Kandidat item harus menjadi item yang paling sering dibeli oleh grup.

Pada repositori ini:

Fungsi find_candidate_items()dibuat untuk mencari item yang dibeli oleh pengguna-pengguna yang mirip beserta frekuensinya. Frekuensi item dalam himpunan C dapat dihitung dengan menghitung frekuensi kejadian sebenarnya dari item tersebut. Pada fungsi ini , didalamnya akan memuat: Gu_items, yaitu item permintaan G_u dalam urutan frekuensi yang menurun, active_items, yaitu item yang sudah dibeli oleh pengguna aktif, dan candidates, yaitu item permintaan G_u yang tidak dibeli oleh pengguna aktif u.

3) Prediksi Rating

Mengagregat peringkat pengguna di G_u untuk membuat prediksi bagi pengguna u pada item yang belum dia beli. Beberapa pendekatan agregasi yang sering digunakan seperti rata-rata, jumlah bobot, jumlah bobot yang disesuaikan. Dengan menggunakan penjumlahan bobot, prediksi rating pengguna u pada item i dinotasikan $\hat{r}_{u,i}$ yang dihitung sebagai berikut:

$$\hat{r}_{u,i} = \overline{r_u} + \frac{\sum_{v \in G_u}\left(r_{v,i} - \overline{r_v}\right).w_{u,v}}{\sum_{v \in G_u}|w_{u,v}|}$$

…(2)

Rating dari pengguna yang mirip ditimbang berdasarkan kemiripan yang sesuai dengan pengguna aktif. Penjumlahan dibuat pada semua pengguna yang memberi rating pada item i. Pengurangan rata-rata rating pengguna $\overline{r_v}$ mengkompensasi perbedaan dalam penggunaan skala peringkat oleh pengguna karena beberapa pengguna akan cenderung memberikan peringkat lebih tinggi daripada yang lain. Prediksi ini dibuat untuk semua item $i \in C$ yang tidak dibeli oleh pengguna u.

Pada repositori ini:
Akan diprediksi skor apa yang pengguna aktif u akan berikan untuk 30 kandidat item teratas.
Untuk memprediksi skor u pada kandidat item i, dibutuhkan:

a. Kemiripan antara u dan semua neighborsnya $v \epsilon G_u$ yang memberikan rating item i Fungsi nearest_neighbors()mengembalikan pengguna yang mirip dari seorang pengguna yang sesuai dengan kemiripannya.
b. Normalisasi rating semua $v \in G_u$ pada item i yang didefinisikan $r_{v,i} - \overline{r_v}$.

Kemudian, menghitung rata-rata rating pada setiap pengguna dan menormalisasi rating untuk setiap item. DataFrame mean berisi rata-rata rating pada setiap pengguna. Dengan rata-rata rating dari setiap pengguna, dapat ditambahkan kolom norm_rating ke DataFrame ratings yang dapat diakses untuk membuat prediksi.
Fungsi predict dibutuhkan untuk memprediksi rating antara pengguna u dan item i. Kemudian panggil formula prediksi yang didefinisikan pada persamaan (2). Selanjutnya, dibuat prediksi rating untuk pengguna yang diberikan pada setiap item pada himpunan kandidat item miliknya.

4) Top-N rekomendasi
Rekomendasi N teratas diperoleh dengan memilih N item yang paling memberikan kepuasan kepada pengguna menurut prediksi

Pada repositori ini:
Fungsi user2userRecommendation()membaca prediksi untuk pengguna yang diberikan dan mengembalikan daftar item dalam urutan menurun dari rating yang diprediksikan.

	userid	itemid	predicted_rating	title
0	212	483	4.871495	Casablanca (1942)
1	212	357	4.764547	One Flew Over the Cuckoo's Nest (1975)
2	212	50	4.660002	Star Wars (1977)
3	212	98	4.613636	Silence of the Lambs, The (1991)
4	212	64	4.550733	Shawshank Redemption, The (1994)
5	212	194	4.522336	Sting, The (1973)
6	212	174	4.521300	Raiders of the Lost Ark (1981)
7	212	134	4.414819	Citizen Kane (1941)
8	212	187	4.344531	Godfather: Part II, The (1974)
9	212	196	4.303696	Dead Poets Society (1989)
10	212	523	4.281802	Cool Hand Luke (1967)
11	212	216	4.278246	When Harry Met Sally... (1989)
12	212	100	4.260087	Fargo (1996)
13	212	168	4.206139	Monty Python and the Holy Grail (1974)
14	212	435	4.122984	Butch Cassidy and the Sundance Kid (1969)
15	212	135	4.115228	2001: A Space Odyssey (1968)
16	212	83	4.106995	Much Ado About Nothing (1993)
17	212	69	4.086366	Forrest Gump (1994)
18	212	70	4.086328	Four Weddings and a Funeral (1994)
19	212	275	3.985037	Sense and Sensibility (1995)
20	212	153	3.981619	Fish Called Wanda, A (1988)
21	212	514	3.956640	Annie Hall (1977)
22	212	521	3.937792	Deer Hunter, The (1978)
23	212	97	3.906106	Dances with Wolves (1990)
24	212	173	3.879325	Princess Bride, The (1987)
25	212	660	3.847897	Fried Green Tomatoes (1991)
26	212	215	3.709920	Field of Dreams (1989)
27	212	258	3.583718	Contact (1997)
28	212	202	3.508617	Groundhog Day (1993)
29	212	237	3.039041	Jerry Maguire (1996)

Gambar 3. Hasil prediksi rating dataset *movielens 100k* dengan algoritma *user-based CF*

Penyaringan kolaboratif berbasis pengguna yang dijalankan pada repositori ini memiliki kekurangan, diantaranya sebagai berikut:

1) Ketersebaran
   Secara umum, pengguna berinteraksi dengan kurang dari 20% item. Hal tersebut menyebabkan matriks rating menjadi sangat jarang. Pada movielens-100k berisi rating 100k dari 943 pengguna pada 1682 item. Presentasi *sparsity* dalam hal tersebut adalah sekita 94%. Sistem rekomendasiberbasis algoritma *nearest neighbors* mungkin tidak dapat membuat rekomendasi item lain untuk pengguna tertentu. Akibatnya, akurasi rekomendasi mungkin buruk.
2) Stabilitas rating pengguna
   Saat pengguna memberi rating dan menilai ulang item, vektor peringkat mereka akan berubah seiring dengan kemiripan dengan pengguna lain. Lingkurang pengguna tidak hanya ditentukan oleh peringkat mereka tetapi juga oleh peringkat pengguna lain, sehingga lingkungan mereka dapat berubah sebagai akibat dari peringkat baru yang diberikan oleh setiap pengguna dalam sistem.
3) Skalabilitas
   Karena rating pengguna tidak stabil, sulit untuk menemukan pengguna serupa sebelumnya. Karena hal itu, sebagian besar sistem *collaborative filtering* berbasis pengguna menemukan lingkungan setiap kali prediksi atau rekomendasi diperlukan. Namun ini adalah perhitungan besar yang tumbuh dengan jumlah pengguna dan jumlah item. Dengan jutaan pengguna dan item, sistem rekomendasi berbasis *typical web* yang menjalankan algoritma yang ada akan mengalami masalah skalabilitas yang serius.

2.2.2. *Item-Based Collaborative Filtering*

Penyaringan kolaboratif berbasis item adalah sistem rekomendasi untuk menggunakan kesamaan antar item menggunakan peringkat oleh pengguna. Asumsi mendasar algoritma ini adalah pengguna memberikan rating serupa untuk item serupa. (towardsdatascience)

Ide dalam algoritma ini adalah dengan menjadikan u sebagai pengguna aktif dan i sebagai item yang direferensikan, sedemikian hingga: 1) jika u menyukai item yang mirip dengan i, maka u mungkin akan menyukai item i; 2) jika u tidak menyukai item yang mirip dengan i, maka u juga akan tidak menyukai item i.

Algoritma penyaringan kolaboratif berbasis item dalam repositori yang dibahas dalam laporan ini memiliki langkah-langkah sebagai berikut.

1) Mencari kemiripan untuk setiap item
   Mengidentifikasi k item yang paling mirip untuk setiap item dalam katalog dan catatan kemiripan yang sesuai. Untuk menghitung kemiripan antar dua item, dapat digunakan *Adjusted Cosine Similarity* yang telah terbukti efisien daripada ukuran *basic Cosine similarity* yang digunakan untuk kolaboratif berbasis pengguna. Jarak *Adjusted Cosine* antara dua item i dan j dihitung sebagai berikut

$$w_{i,j} = \frac{\sum_{u \in U}\left(r_{u,i} - \overline{r_u}\right)\left(r_{u,j} - \overline{r_u}\right)}{\sqrt{\sum_{u \in U}\left(r_{u,i} - \overline{r_u}\right)^2}\sum_{u \in U}\left(r_{u,j} - \overline{r_u}\right)^2} \quad \ldots(3)$$

   $w_{i,j}$ adalah tingkat kemiripan antara item i dan j. Bentuk ini dihitung untuk semua pengguna $u \in U$, dimana u adalah himpunan pengguna yang merating item i dan j. Notasi $S^{(i)}$ melambangkan himpunan k item yang paling mirip dengan item i

Pada repositori ini:

Untuk menghitung kemiripan antara dua item i dan j, dibutuhkan: 1) mencari semua pengguna yang merating kedua item tersebut; 2) menormalisasi rating mereka pada item i dan j; 3) mengaplikasikan metrik *cosine* pada rating yang dinormalisasi untuk menghitung kemiripan antara item i dan j.

Fungsi normalize() memproses rating dataframe untuk menormalisasi rating semua pengguna. Pada setiap rating yang sudah dinormalisasi, setiap item direpresentasikan oleh suatu vektor rating yang dinormalisasi. Kemudian membangun dan menyesuaikan model $k - NN$ menggunakan sklearn.

Kemiripan antar item dapat dihitung dengan jarak *Cosine* atau *Euclidian.* Kelas *NearestNeighbors* dari library sklearn menyederhanakan perhitungan neighbors. Hanya dibutuhkan penentuan metrik yang akan digunakan untuk menghitung kemiripan. Metode create_model membuat model kNN dan mengikuti metode nearest_neighbors menggunakan model yang dibuat ke item kNN. Hal tersebut mengembalikan *nearest neighbors* yang memuat: similarities (*array* numpy berbentuk (n, k)); dan neighbors (*array numpy* berbentuk (n, k)); dimana n adalah total item dan k adalah banyak neighbors yang kembali, ditentukan saat membuat model kNN.

Dalam konteks penyaringan kolaboratif berbasis item, *adjusted cosine similarity* telah terbukti lebih efisien dari pada *cosine* atau jarak *euclidian.* Formula untuk menghitung bobot *adjusted cosine* antara dua item i dan j adalah persamaan (3). Karena *library* sklearn tidak secara langsung mengimplementasikan metrik *adjusted cosine similarity,* maka untuk mengimplementasikannya digunakan adjusted_cosine, dengan beberapa fungsi bantuan, yaitu: 1) save_similarities untuk menyimpan kesamaan yang dihitung untuk penggunaan yang terlambat karena perhitungan *adjusted cosine similarity* memakan waktu sekitar 5 menit untuk kumpulan dari ml100k; 2) load_similarities memuat kemiripan yang disimpan; 3) cosine, yaitu cosinus jarak antara dua vektor.

Fungsi adjusted_cosine dipanggil untuk menghitung dan menyimpan kesamaan item dan neighbors berdasarkan metrik *adjusted cosinus.* Selanjutnya, *nb_items* dan *similarity, neighbors* dibatalkan untuk menghitung *adjusted cosine* di antara semua item. Neighbors dan similarities dimunculkan dalam bentuk array numpy, yang masing-masing entri adalah daftar 20 neighbors dengan kemiripan yang sesuai.

2) Rekomendasi N teratas untuk pengguna tertentu

Untuk menghasilkan rekomendasi N teratas untuk pengguna u yang sudah membeli himpunan I_u item, berikut ini langkah yang diperlukan:

a. Mencari himpunan C kandidat item dengan mengambil gabungan semua $S^{(i)}, \forall i \in I_u$ dan memindahkan setiap item dalam himpunan I_u.

$$C = \bigcup_{i \in I_u} \{S^{(i)}\} \backslash \mathrm{I_u} \quad \ldots(4)$$

b. $\forall c \in C,$ kemiripan antara c dan himpunan I_u dapat dihitung dengan formula sebagai berikut

$$w_{c,I_u} = \sum_{i \in I_u} w_{c,i}, \forall c \in C \quad \ldots(5)$$

c. Mengurutkan item di C dalam urutan menurun dari $w_{c,I_u}, \forall c \in C$, dan mengembalikan N item pertama sebagai daftar rekomendasi N teratas.

Sebelum mengembalikan N item teratas sebagai daftar rekomendasi N teratas, dapat dibuat prediksi pengguna u yang akan diberikan untuk setiap item dalam daftar rekomendasi N teratas, mengatur ulang daftar rating yang diprediksikan dalam urutan menurun dan mengembalikan daftar yang diatur ulang sebagai daftar rekomendasi akhir. Prediksi rating untuk penyaringan kolaboratif berbasis item dapat dihitung dengan formula berikut:

$$\hat{r}_{u,i} = \frac{\sum_{i \epsilon S^{(i)}} r_{u,j} w_{i,j}}{\sum_{j \epsilon S^{(i)}} |w_{i,j}|} \quad \ldots(6)$$

Pada repositori ini:

Untuk mencari kandidat item untuk pengguna u, dibutuhkan fungsi candidate_items() untuk:

a. Mencari himpunan I_u item yang sudah dirating oleh pengguna u
b. Mengambil gabungan item yang mirip sebagai C untuk semua item pada I_u
c. Mengecualikan dari himpunan C semua item di I_u untuk menghindari merekomendasikan item pengguna yang sudah dibeli

Selanjutnya untuk mencari kemiripan antara setiap kandidat item dan himpunan I_u digunakan fungsi similarity_with_Iu(), kemudian untuk meranking kandidat item berdasarkan kemiripannya dengan I_u digunakan fungsi rank_candidates().

Untuk membuat rekomendasi N teratas untuk pengguna yang diperlukan, dibuat fungsi topn_recommendation(). Dengan hasil sebagai berikut.

	itemid	similarity_with_Iu	title
0	1356	52.867173	Ed's Next Move (1996)
1	1189	50.362199	Prefontaine (1997)
2	1516	31.133267	Wedding Gift, The (1994)
3	1550	31.031738	Destiny Turns on the Radio (1995)
4	1554	27.364494	Safe Passage (1994)
5	1600	27.287712	Guantanamera (1994)
6	1223	26.631850	King of the Hill (1993)
7	1388	26.624397	Gabbeh (1996)
8	766	26.590175	Man of the Year (1995)
9	691	26.461802	Dark City (1998)
10	1378	25.787842	Rhyme & Reason (1997)
11	1664	25.327445	8 Heads in a Duffel Bag (1997)
12	1261	24.785660	Run of the Country, The (1995)
13	1123	24.524028	Last Time I Saw Paris, The (1954)
14	1538	24.492453	All Over Me (1997)
15	1485	24.345312	Colonel Chabert, Le (1994)
16	1450	24.262120	Golden Earrings (1947)
17	909	23.357301	Dangerous Beauty (1998)
18	359	22.973658	Assignment, The (1997)
19	1369	22.710078	Forbidden Christ, The (Cristo proibito, Il) (1...
20	1506	22.325504	Nelly & Monsieur Arnaud (1995)
21	1537	22.061914	Cosi (1996)
22	1474	21.877034	Nina Takes a Lover (1994)
23	1467	21.861203	Saint of Fort Washington, The (1993)
24	1255	21.750924	Broken English (1996)
25	1499	21.529748	Grosse Fatigue (1994)
26	1466	21.063269	Margaret's Museum (1995)
27	1448	20.846909	My Favorite Season (1993)
28	927	20.730153	Flower of My Secret, The (Flor de mi secreto, ...
29	1375	20.627152	Cement Garden, The (1993)

Gambar 4 Tingkat kemiripan pengguna dataset *movielens 100k* dengan algoritma i*tem-based CF*

Dataframe tersebut merepresentasikan daftar rekomendasi teratas beserta penggunanya. Item tersebut diurutkan berdasarkan kemiripan dengan I_u dalam urutan menurun. Item yang rekomendasikan adalah yang paling mirip dengan himpunan I_u item yang sudah dibeli oleh pengguna.

Sebelum merekomendasikan daftar sebelumnya ke pengguna, dapat dilakukan lebih jauh untuk memprediksi rating yang akan diberikan pengguna untuk setiap item tersebut, mengurutkan prediksi dalam urutan menurun dan mengembalikan daftar yang disusun ulang sebagai daftar rekomendasi N teratas yang baru.

Rating yang diprediksikan $\hat{r}_{u,i}$ untuk pengguna u pada item i diperoleh dari mengagregasi rating yang diberikan oleh u pada item yang mirip dengan u menggunakan formula persamaan (6). Selanjutnya digunakan fungsi topn_prediction() untuk memprediksi rating pengguna yang akan diberikan pengguna ke daftar N teratas sebelumnya dan mengembalikan daftar yang diatur ulang (dalam urutan prediksi menurun) sebagai daftar N teratas yang baru. Berikut ini hasil prediksi rating N teratas.

	itemid	similarity_with_Iu	title	prediction
7	1388	26.624397	Gabbeh (1996)	4.666667
18	359	22.973658	Assignment, The (1997)	4.600000
4	1554	27.364494	Safe Passage (1994)	4.500000
14	1538	24.492453	All Over Me (1997)	4.500000
27	1448	20.846909	My Favorite Season (1993)	4.490052
29	1375	20.627152	Cement Garden, The (1993)	4.333333
26	1466	21.063269	Margaret's Museum (1995)	4.271915
2	1516	31.133267	Wedding Gift, The (1994)	4.000000
23	1467	21.861203	Saint of Fort Washington, The (1993)	4.000000
21	1537	22.061914	Cosi (1996)	4.000000
10	1378	25.787842	Rhyme & Reason (1997)	4.000000
19	1369	22.710078	Forbidden Christ, The (Cristo proibito, Il) (1...	4.000000
3	1550	31.031738	Destiny Turns on the Radio (1995)	3.777778
1	1189	50.362199	Prefontaine (1997)	3.666528
20	1506	22.325504	Nelly & Monsieur Arnaud (1995)	3.610294
15	1485	24.345312	Colonel Chabert, Le (1994)	3.610294
11	1664	25.327445	8 Heads in a Duffel Bag (1997)	3.610294
9	691	26.461802	Dark City (1998)	3.610294
6	1223	26.631850	King of the Hill (1993)	3.610294
5	1600	27.287712	Guantanamera (1994)	3.610294
17	909	23.357301	Dangerous Beauty (1998)	3.500000
12	1261	24.785660	Run of the Country, The (1995)	3.333333
24	1255	21.750924	Broken English (1996)	3.265749
13	1123	24.524028	Last Time I Saw Paris, The (1954)	3.200000
16	1450	24.262120	Golden Earrings (1947)	3.142978
22	1474	21.877034	Nina Takes a Lover (1994)	3.000000
8	766	26.590175	Man of the Year (1995)	3.000000
0	1356	52.867173	Ed's Next Move (1996)	2.280926
28	927	20.730153	Flower of My Secret, The (Flor de mi secreto, ...	1.665010
25	1499	21.529748	Grosse Fatigue (1994)	1.122032

Gambar 5 Hasil prediksi rating dataset *movielens 100k* dengan algoritma *item-based CF*

Gambar 4 dan gambar 5 diurutkan dengan cara yang berbeda. Gambar 1 disusun menurut kemiripan pengguna, sedangkan gambar 5 disusun menurut prediksi yang dibuat pengguna. Ketika membuat prediksi untuk pengguna u pada item i, pengguna u mungkin tidak memberi peringkat k pada item yang paling mirip dengan i. Hal tersebut menganggap rata-rata rating u sebagai nilai prediksi.

Keuntungan algoritma *item-based* dibandingkan algoritma *item-based,* yaitu:

1) Stabilitas

Rating item lebih stabil daripada rating pengguna. Rating baru pada item tidak mungkin mengubah kesamaan antara dua item secara signifikan, terutama ketika item memiliki banyak rating.

2) Skalabilitas
Dengan rating item yang stabil, masuk akan untuk melakukan perhitungan kemiripan antar item dalam matriks *item-item similarity* (kemiripan antar produk dapat dihitung secara offline). Hal ini akan mengurangi masalah skalabilitas dari algoritma.

2.3. Pengurangan Dimensi

2.3.1. *Singular Value Decomposition* (SVD)

Karena tingkat ketersebaran yang tinggi dari matriks peringkat R, pemfilteran kolaboratif berbasis pengguna dan berbasis item memiliki ketersebaran dan skalabilitas data. Hal ini menyebabkan pemfilteran kolaboratif tersebut menjadi kurang efektif dan sangat mempengaruhi kinerja mereka. Untuk mengatasi masalah sparsity tingkat tinggi, diusulkan untuk mengurangi dimensi rating R menggunakan algoritma Singular Value Decomposition (SVD)

SVD memfaktorkan matriks rating $R_{m\times n}$ menjadi tiga matriks P, Σ, dan Q sebagai berikut

$$R = P\Sigma \mathrm{Q}^{\mathrm{T}} \quad \ldots(7)$$

dengan P dan Q adalah dua matriks ortogonal masing-masing berukuran $m \times \hat{k}$ dan $n \times \hat{k}$ dan Σ adalah matriks diagonal berukuran $\hat{k} \times \hat{k}$ ($\hat{k}$ adalah rank matriks R) memiliki semua nilai singular rating matriks R sebagai entri diagonalnya.

Setelah memilih k, dimensi faktor yang akan merepresentasikan pengguna dan item, matriks Σ dapat dipotong dengan hanya mempertahankan k nilai singular terbesar untuk menghasilkan Σ_{k} dan mengurangi matriks P dan Q yang sesuai untuk mendapatkan P_k dan Q_k. Matriks rating kemudian akan diestimasi sebagai $R_k = P_k \Sigma_{\mathrm{k}} \mathrm{Q}_{\mathrm{k}}^{\mathrm{T}}$

Setelah matriks tersebut diketahui, matriks tersebut dapat digunakaa untuk prediksi rating dan rekomendasi N teratas. $P_k \Sigma_{\mathrm{k}}^{\frac{1}{2}}$ mewakili ruang laten pengguna dan $\Sigma_{\mathrm{k}}^{\frac{1}{2}} \mathrm{Q}_{\mathrm{k}}^{\mathrm{T}}$ ruang laten item. Prediksi rating untuk pengguna u dan i dilakukan dengan formula:

$$\hat{R}_{u,i} = \left[P_k \Sigma_{\mathrm{k}}^{\frac{1}{2}}\right]_u \left[\Sigma_{\mathrm{k}}^{\frac{1}{2}} \mathrm{Q}_{\mathrm{k}}^{\mathrm{T}}\right]_i \quad \ldots(8)$$

Sebelum mengaplikasikan SVD, penting untuk melengkapi (mengisi) *missing value* rating matriks R. Rata-rata rating item dapat menjadi nilai default yang berguna. Menambahkan normalisasi peringkat dengan mengurangi rata-rata rating pengguna atau prediktor garis dasar lainnya dapat meningkatkan akurasi.

Langkah-langkah dalam algoritma SVD adalah sebagai berikut:
1) Memfaktorkan normalisasi matriks rating R_{norm} untuk memperoleh P, Σ dan Q
2) Mengurangi Σ ke dimensi k untuk memperoleh Σ_{k}
3) Menghitung akar dari Σ_{k} untuk memperoleh $\Sigma_{\mathrm{k}}^{\frac{1}{2}}$
4) Menghitung matriks resultan $P_k \Sigma_{\mathrm{k}}^{\frac{1}{2}}$ dan $\Sigma_{\mathrm{k}}^{\frac{1}{2}} \mathrm{Q}_{\mathrm{k}}^{\mathrm{T}}$ yang akan digunakan untuk menghitung nilai rekomendasi untuk sembarang pengguna dan item.

Pada repositori ini:

itemid	1	2	3	4	5	6	7	8	9	10	...	193565	193567	193571	193573	193579	193581	193583	193585	193587	193609
userid																					
1	4.0	NaN	4.0	NaN	NaN	4.0	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
5	4.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
606	2.5	NaN	NaN	NaN	NaN	NaN	2.5	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
607	4.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
608	2.5	2.0	2.0	NaN	NaN	NaN	NaN	NaN	NaN	4.0	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
609	3.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	4.0	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
610	5.0	NaN	NaN	NaN	NaN	5.0	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

Gambar 6. Matriks rating dataset *mlLatestSmall*

Dapat diamati bahwa matriks rating memiliki banyak *missing value.* Karena algoritma SVD mengharuskan semua input dalam matriks harus didefinisikan, maka rating yang *missing* diinisialisasi dengan rata-rata produk yang menghasilkan kinerja yang lebih baik dibandingkan dengan rata-rata pengguna atau bahwa inisialisasi nol.

Selanjutnya mengurangkan setiap rata-rata rating pengguna yang sesuai untuk menormalisasi data. Hal ini bertujuan untuk meningkatkan akurasi model.

itemid	1	2	3	4	5	6	7	8	9	10	...	193565	193567	193571	193573	193579	193581	193583	193585	193587	193609
userid																					
1	-0.366379	-0.934561	-0.366379	-2.009236	-1.294951	-0.366379	-1.181194	-1.491379	-1.241379	-0.870167	...	-0.866379	-1.366379	-0.366379	-0.366379	-0.866379	-0.366379	-0.866379	-0.866379	-0.866379	-0.366379
2	-0.027346	-0.516458	-0.688660	-1.591133	-0.876847	-0.002197	-0.763091	-1.073276	-0.823276	-0.452064	...	-0.448276	-0.948276	0.051724	0.051724	-0.448276	0.051724	-0.448276	-0.448276	-0.448276	0.051724
3	1.485033	0.995921	0.823718	-0.078755	0.635531	1.510181	0.749288	0.439103	0.689103	1.060315	...	1.064103	0.564103	1.564103	1.564103	1.064103	1.564103	1.064103	1.064103	1.064103	1.564103
4	0.365375	-0.123737	-0.295940	-1.198413	-0.484127	0.390523	-0.370370	-0.680556	-0.430556	-0.059343	...	-0.055556	-0.555556	0.444444	0.444444	-0.055556	0.444444	-0.055556	-0.055556	-0.055556	0.444444
5	0.363636	-0.204545	-0.376748	-1.279221	-0.564935	0.309715	-0.451178	-0.761364	-0.511364	-0.140152	...	-0.136364	-0.636364	0.363636	0.363636	-0.136364	0.363636	-0.136364	-0.136364	-0.136364	0.363636
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
606	-1.157399	-0.225581	-0.397784	-1.300256	-0.585971	0.288679	-1.157399	-0.782399	-0.532399	-0.161187	...	-0.157399	-0.657399	0.342601	0.342601	-0.157399	0.342601	-0.157399	-0.157399	-0.157399	0.342601
607	0.213904	-0.354278	-0.526481	-1.428953	-0.714668	0.159982	-0.600911	-0.911096	-0.661096	-0.289884	...	-0.286096	-0.786096	0.213904	0.213904	-0.286096	0.213904	-0.286096	-0.286096	-0.286096	0.213904
608	-0.634176	-1.134176	-1.134176	-0.777033	-0.062747	0.811903	0.051009	-0.259176	-0.009176	0.865824	...	0.365824	-0.134176	0.865824	0.865824	0.365824	0.865824	0.365824	0.365824	0.365824	0.865824
609	-0.270270	0.161548	-0.010655	-0.913127	-0.198842	0.675808	-0.085085	-0.395270	-0.145270	0.729730	...	0.229730	-0.270270	0.729730	0.729730	0.229730	0.729730	0.229730	0.229730	0.229730	0.729730
610	1.311444	-0.256738	-0.428941	-1.331413	-0.617127	1.311444	-0.503371	-0.813556	-0.563556	-0.192344	...	-0.188556	-0.688556	0.311444	0.311444	-0.188556	0.311444	-0.188556	-0.188556	-0.188556	0.311444

Gambar 7. Matriks rating dataset *mlLatestSmall* yang sudah dinormalisasi

Selanjutnya menyandikan pengguna dan id item sedemikian hingga nilai pengguna berkisar 0 sampai 909 dan nilai item berkisar 0 sampai 9723. Setelah data rating dinormalisasi dan nilai yang hilang diisi, algoritma SVD dapat diterapkan. Untuk menjalankan algoritma SVD, beberapa fungsi yang dibutuhkan adalah sebagai berikut:

1) fit(): untuk menghitung svd dari matriks rating dan menyimpan matriks yang dihasilkan $P, S,$ dan Qh (Q transpose) sebagai atribut kelas SVD
2) predict(): untuk menggunakan matriks P, Q, dan Qh untuk membuat rating prediksi untuk pengguna u pada item i. Perhitungan dilakukan pada nilai userid dan itemid yang disandikan. Nilai prediksi adalah perkalian titik antara baris u^{th} dari $P.\sqrt{S}$ dan kolom i^{th} dari $\sqrt{S}.Qh$. Karena rating dinormalisasi sebelum menerapkan SVD, nilai prediksi juga akan dinormalisasi. Jadi untuk mendapatkan nilai prediksi akhir, harus ditambahkan nilai prediksi rata-rata rating pengguna u
3) recommend(): untuk menggunakan matriks P, S, dan Qh untuk membuat rekomendasi untuk pengguna yang diberikan. Item yang direkomendasikan adalah produk yang tidak diberi peringkat oleh pengguna dan mendapat skor tinggi menurut model SVD

Setelah itu, membuat model SVD dan memberikan nilai rata-rata pengguna, serta menyesuaikan model dengan matriks rating yang dinormalisasi R. Setelah model diterapkan, beberapa prediksi untuk pengguna dapat dibuat menggunakan fungsi predict dari kelas SVD.

	userid	itemid	rating	timestamp
0	1	1	4.0	964982703
1	1	3	4.0	964981247
2	1	6	4.0	964982224
3	1	47	5.0	964983815
4	1	50	5.0	964982931
5	1	70	3.0	964982400
6	1	101	5.0	964980868
7	1	110	4.0	964982176
8	1	151	5.0	964984041
9	1	157	5.0	964984100

```
prediksi untuk userid=1 dan itemid=1 : 3.9999999999999996
prediksi untuk userid=1 dan itemid=3 : 4.00000000000000036
prediksi untuk userid=1 dan itemid=6 : 3.99999999999999867
prediksi untuk userid=1 dan itemid=47 : 5.0
prediksi untuk userid=1 dan ItemId = 50: 4.9999999999999964
Prediksi untuk userid = 1 dan itemId = 70: 2.9999999999981
Prediksi untuk UserID
= 1 dan itemID = 101: 5.000000000099999999999999999999999999.0099
Prediksi = 1 dan itemID = 110: 3.99999999999999
Predikerer A) untuk userid=1 dan itemid=157 : 5.000000000000028
```

Gambar 8. Hasil prediksi rating dataset *mlLatestSmall* dengan algorritma SVD

Kesalahan prediksi kurang dari 0,00001

Selanjutnya fungsi recommend membuat rekomendasi untuk pengguna tertentu.

	itemid	predictions	title	genres
0	148	5.0	Awfully Big Adventure, An (1995)	Drama
1	6086	5.0	I, the Jury (1982)	Crime|Drama|Thriller
2	136445	5.0	George Carlin: Back in Town (1996)	Comedy
3	6201	5.0	Lady Jane (1986)	Drama|Romance
4	2075	5.0	Mephisto (1981)	Drama|War
5	6192	5.0	Open Hearts (Elsker dig for evigt) (2002)	Romance
6	117531	5.0	Watermark (2014)	Documentary
7	158398	5.0	World of Glory (1991)	Comedy
8	6021	5.0	American Friend, The (Amerikanische Freund, De...	Crime|Drama|Mystery|Thriller
9	136556	5.0	Kung Fu Panda: Secrets of the Masters (2011)	Animation|Children
10	136447	5.0	George Carlin: You Are All Diseased (1999)	Comedy
11	136503	5.0	Tom and Jerry: Shiver Me Whiskers (2006)	Animation|Children|Comedy
12	134095	5.0	My Love (2006)	Animation|Drama
13	3851	5.0	I'm the One That I Want (2000)	Comedy
14	136469	5.0	Larry David: Curb Your Enthusiasm (1999)	Comedy
15	158882	5.0	All Yours (2016)	Comedy|Drama|Romance
16	134004	5.0	What Love Is (2007)	Comedy|Romance
17	67618	5.0	Strictly Sexual (2008)	Comedy|Drama|Romance
18	3567	5.0	Bossa Nova (2000)	Comedy|Drama|Romance
19	158027	5.0	SORI: Voice from the Heart (2016)	Drama|Sci-Fi
20	59814	5.0	Ex Drummer (2007)	Comedy|Crime|Drama|Horror
21	5745	5.0	Four Seasons, The (1981)	Comedy|Drama
22	118894	5.0	Scooby-Doo! Abracadabra-Doo (2010)	Animation|Children|Mystery
23	5746	5.0	Galaxy of Terror (Quest) (1981)	Action|Horror|Mystery|Sci-Fi
24	118834	5.0	National Lampoon's Bag Boy (2007)	Comedy
25	3940	5.0	Slumber Party Massacre III (1990)	Horror
26	95311	5.0	Presto (2008)	Animation|Children|Comedy|Fantasy
27	3496	5.0	Madame Sousatzka (1988)	Drama
28	156025	5.0	Ice Age: The Great Egg-Scapade (2016)	Adventure|Animation|Children|Comedy
29	2196	5.0	Knock Off (1998)	Action

Gambar 9. Hasil prediksi rating pengguna tertentu pada dataset *mlLatestSmall* dengan algoritma SVD

Dapat dilihat bahwa 30 item teratas memiliki prediksi rating yang setara untuk pengguna 1.

SVD dapat diterapkan untuk meningkatkan pemfilteran kolaboratif berbasis pengguna dan item. Dibandingkan menghitung kesamaan antara rating pengguna atau item, kita dapat merepresentasikan pengguna dan item dengan faktor laten terkait yang diekstrak dari algoritma SVD.

2.3.2. *Matrix Factorization* (MF)

Algoritma faktorisasi matriks adalah varian dari SVD atau dikenal dengan sebutan Regularized SVD dengan menggunakan pengoptimal *Gradient Descent* untuk mengoptimalkan fungsi nilai saat *training* model.

Dataset movielen lasted small memiliki rating 100k dengan $m = 610$ pengguna pada $n = 9724$ item, sehingga dibentuk matriks rating $m \times n$. Fakta bahwa biasanya pengguna berinteraksi dengan kurang dari 1% item menyebabkan matriks peringkat R menjadi sangat jarang. Sebagai contoh, berikut ini tingkat *sparsity* dari dataset movielen lasted small:

$$sparsity = 100 - \frac{total\#ratings}{m \times n} \quad \ldots(9)$$

$$sparsity = 100 - \frac{100000}{610 \times 9724} = 98{,}3\% \quad \ldots(10)$$

Hasil perhitungan tersebut menunjukkan bahwa pengguna berinteraksi dengan kurang dari 2 % item. Untuk mengurangi dimensi matriks R, Faktorisasi Matriks memetakan pengguna dan item ke ruang faktor laten gabungan dari dimensi k sedemikian hingga interaksi pengguna-item dimodelkan sebagai produk dalam ruang tersebut. Kemudian Faktorisasi Matriksdekomposisi R menjadi dua matriks sebagai berikut:

$$R = Q^T P \quad \ldots(11)$$

dimana $P \in \mathbb{R}^{m \times k}$ merepresentasikan faktor laten dari pengguna dan $Q \in \mathbb{R}^{n \times k}$ adalah faktor laten dari item. Setiap baris dari P, misalkan $p_u \in \mathbb{R}^k$ menotasikan selera pengguna u dan setiap $q_i \in \mathbb{R}^k$ fitur item i. *Dot product* antara p_u dan q_i akan menjadi prediksi rating dari pengguna u dan item i:

$$\hat{r}_{u,i} = q_i^T p_u \quad \ldots(12)$$

Untuk mempelajari faktor laten p_u dan q_i, sistem meminimalkan *regularized squared error* pada himpunan rating yang diketahui. Fungsi nilai J didefinisikan:

$$J = \frac{1}{2} \sum_{(u,i) \in \kappa} \left(r_{u,i} - q_i^T p_u\right)^2 + \lambda \left(\left|\left|p_u\right|\right|^2 + \left|\left|q_i\right|\right|^2\right) \quad \ldots(13)$$

dimana κ adalah himpunan dari pasangan (u, i) yang mana $r_{u,i}$ diketahui (himpunan *training*), dan λ adalah parameter *regularizer.*

Untuk meminimalkan fungsi nilai J, algoritma faktorisasi matriks memprediksikan $\hat{r}_{u,i}$ untuk setiap kasus *training* yang diberikan ($r_{u,i}$ diberikan), dan menghitung *associated error* yang didefinisikan *Mean Absolute Error (MAE):*

$$e_{u,i} = \left|r_{u,i} - q_i^T p_u\right| \quad \ldots(14)$$

Error keseluruhan E didefinisikan:

$$E = \frac{1}{M} \sum_{(u,i)\in\kappa} e_{u,i}$$

…(15)

Dimana M adalah banyaknya contoh. Peraturan diperbarui untuk parameter p_u dan q_i yang didefinisikan sebagai berikut:

$$q_i \leftarrow q_i - \alpha \frac{\partial}{\partial p_u} J_{u,i},$$

…(16)

$$p_u \leftarrow p_u - \alpha \frac{\partial}{\partial p_u} J_{u,i}$$

…(17)

Dimana α adalah *learning rate* dan $\frac{\partial}{\partial p_u} J_{u,i}$ adalah turunan parsial dari fungsi nilai J terhadap p_u. Turunan parsial tersebut menghitung sejauh mana p_u berkontribusi terhadap kesalahan total. Berikut ini hasil dari turunan parsial persamaan 13

$$\frac{\partial}{\partial p_u} J_{u,i} = -e_{u,i}.p_u + \lambda.q_i$$

…(18)

Sehingga aturan pembaruan sebagai berikut

$$q_i \leftarrow q_i + \alpha\left(e_{u,i}.p_u - \lambda.q_i\right)$$

…(19)

$$p_u \leftarrow p_u + \alpha\left(e_{u,i}.p_u - \lambda.q_i\right)$$

…(20)

Berikut ini langkah-langkah dalam algoritma Faktorisasi Matriks:

1) Menginisialisasi P dan Q dengan nilai random,
2) Untuk setiap contoh *training* $(u, i) \in \kappa$ dengan rating yang bersesuaian $r_{u,i}$
   - Menghitung persamaan (12)
   - Menghitung error: persamaan (14)
   - Memperbarui p_u dan q_i dengan persamaan (19) dan (20)
3) Mengulang langkah 2 sampai parameter optimal tercapai

Pada repositori ini:

Faktor laten P dan Q digunakan untuk membuat prediksi dan rekomendasi. Berikut ini prediksi peringkat untuk pengguna 1 pada item 1 yang rating kebenarannya $r = 5.0$

	userid	itemid	rating	rating_mean	norm_rating
0	1	1	5	4.188679	0.811321
1	1	48	5	4.188679	0.811321
2	1	145	5	4.188679	0.811321
3	1	254	4	4.188679	-0.188679
4	1	514	5	4.188679	0.811321

```
4.188679  + MF.predict(userid= 1 , itemid= 1 )  # menambahkan rata-rata karena harus menggunakan r atinng yang dinormalkan untuk pelatihan
```

```
4.188679163563357
```

Gambar 10. Hasil prediksi rating dataset *movielens 1M* dengan MF

2.3.3. *Non Negative Matrix Factorization* (NMF)

Dengan model Faktorisasi Matriks, faktor laten P dan Q tidak dapat ditafsirkan karena mengandung nilai negatif atau positif yang berubah-ubah, sehingga membuat model Faktorisasi Matriks menjadi tidak dapat dijelaskan. Algoritma NMF adalah varian dari Faktorisasi Matriks yang menghasilkan faktor laten yang dapat dijelaskan untuk P dan Q dengan membatasi nilainya dalam rentang [0,1], sehingga memungkinkan interpretasi probabilitas dari faktor-faktor laten.

Sama seperti Faktorisasi Matriks, NMF memfaktorkan matriks peringkat R menjadi dua matriks:

$$R = PQ^T \quad \ldots(21)$$

Dengan NMF:

- Faktor laten merepresentasikan grup pengguna yang membagi selera yang sama
- Nilai $P_{u,l}$ merepresentasikan probabilitas bahwa pengguna u milik grup l pengguna
- Nilai $Q_{l,i}$ merepresentasikan probabilitas bahwa pengguna di grup l menyukai item i

Dengan jarak *Euclidian,* fungsi objektif NMF didefinisikan sebagai berikut

$$J = \frac{1}{2} \sum_{(u,i)\in\kappa} \left|\left|R_{u,i} - P_u Q_i^T\right|\right|^2 + \lambda_p \left|\left|P_u\right|\right|^2 + \lambda_Q \left|\left|Q_i\right|\right|^2 \quad \ldots(22)$$

Tujuannya adalah meminimalkan nilai fungsi J dengan mengoptimalkan parameter P dan Q, dengan λ_P dan λ_Q parameter regularizer.

Untuk aturan pembaruan perkalian P dan Q adalah sebagai berikut.

$$P \leftarrow P.\frac{RQ}{PQ^TQ} \quad \ldots(23)$$

$$Q \leftarrow Q.\frac{R^TP}{QP^TP} \quad \ldots(24)$$

Namun, karena R adalah *sparse matrix,* dibutuhkan pembaruan setiap P_u terhadap rating yang tersedia dari pengguna u. Pada hal yang sama, diperlukan pembaruan Q_i terhadap rating yang ada dari item i, sedemikian hingga:

$$P_{u,k} \leftarrow P_{u,k}.\frac{\sum_{i\in I_u} Q_{i,k} \cdot r_{u,i}}{\sum_{i\in I_u} Q_{i,k} \cdot \hat{r}_{u,i} + \lambda_P |I_u| P_{u,k}} \quad \ldots(25)$$

$$Q_{i,k} \leftarrow Q_{i,k}.\frac{\sum_{i\in I_u} P_{u,k} \cdot r_{u,i}}{\sum_{i\in I_u} P_{u,k} \cdot \hat{r}_{u,i} + \lambda_Q |U_i| Q_{i,k}} \quad \ldots(26)$$

dimana:

- $P_{u,k}$ adalah faktor laten ke-k dari P_u
- $Q_{i,k}$ adalah faktor laten ke-k dari Q_i
- I_u adalah item yang dirating oleh pengguna u
- U_i adalah himpunan pengguna yang memberi rating pada item i

2.3.4. *Explainable Matrix Factorization* (EMF)

NMF memperkenalkan kemampuan menjelaskan ke MF dengan membatasi nilai-nilai P dan Q di [0,1], sehingga hal ini dianggap sebagai model *inner* yang dapat dijelaskan. Dimungkinkan untuk menyuntikkan informasi eksternal ke dalam model untuk memberikan penjelasan. Inilah yang dilakukan oleh algoritma *Explainable Matrix Factorization* (EMF) yang menghitung skor yang dapat dijelaskan dari kemiripan pengguna atau item yang diambil dari penyaringan kolaboratif berbasis pengguna atau berbasis item.
Cara untuk mengukur kemampuan *explainability,* yaitu:

- Menggunakan distribusi rating dalam lingkungan pengguna aktif
- Jika banyak *neighbors* telah menilai item yang direkomendasikan, maka ini dapat memberikan dasar untuk menjelaskan rekomendasi tersebut, dengan menggunakan mekanisme penjelasan gaya lingkungan

Sebuah item i dianggap *explainable* untuk pengguna u jika sejumlah besar neighborsnya menilai item i. Skor *explainability* E_{ui} adalah persentase neighbors dari pengguna u yang telah merating i.

$$E_{ui} = \frac{\left|N_k^{(i)}(u)\right|}{|N_k(u)|} \quad \ldots(27)$$

dimana $N_k(u)$ adalah himpunan k neighbors terdekat dari pengguna u dan $N_k^{(i)}(u)$ adalah himpunan neighbors dari pengguna u yang telah merating item i. Namun, skor *explainability* di atas ambang optimal θ yang diterima.

$$W_{u,i} = \begin{cases} E_{ui}, & \text{jika } E_{ui} > \theta \\ 0, & \text{jika } E_{ui} \leq \theta \end{cases} \quad \ldots(28)$$

Dengan memasukkan bobot *explainability* pada algoritma *training,* fungsi objektif baru yang akan diminimalkan pada himpunan rating yang diketahui diformulasikan:

$$J = \sum_{(u,i)\in\kappa} \left(R_{ui} - \hat{R}_{ui}\right)^2 + \frac{\beta}{2}\left(\left||P_u|\right|^2 + \left||Q_i|\right|^2\right) + \frac{\lambda}{2}(P_u - Q_i)^2 W_{ui}, \quad \ldots(29)$$

Dimana $\frac{\beta}{2}\left(\left||P_u|\right|^2 + \left||Q_i|\right|^2\right)$ adalah istilah regularisasi L_2 yang telah diberi bobot oleh koefisien β, dan λ adalah koefisien regularisasi *explainability* yang mengontrol kelancaran representasi baru dan pertukaran antara *explainability* dan akurasi. Jika item i *explainable* untuk pengguna u, kemudian representasinya di ruang laten, Q_i dan P_u harus berdekatan satu sama lain. Penurunan gradien stokastik dapat digunakan untuk mengoptimalkan fungsi objektif.

$$P_u \leftarrow P_u + \alpha\left(2\left(R_{u,i} - P_u Q_i^T\right)Q_i - \beta P_u - \lambda(P_u - Q_i)W_{ui}\right) \quad \ldots(30)$$

$$Q_i \leftarrow Q_i + \alpha\left(2\left(R_{u,i} - P_u Q_i^T\right)P_u - \beta Q_i + \lambda(P_u - Q_i)W_{ui}\right) \quad \ldots(31)$$

Pada repositori ini:

Hasil prediksi yang diperoleh:

	itemid	predictions	title	genres
0	3460	4.364036	Hillbillys in a Haunted House (1967)	Comedy
1	701	4.324177	Daens (1992)	Drama
2	3057	4.307404	Where's Marlowe? (1999)	Comedy
3	2214	4.304979	Number Seventeen (1932)	Thriller
4	1145	4.299559	Snowriders (1996)	Documentary
5	2258	4.292125	Master Ninja I (1984)	Action
6	3353	4.281912	Closer You Get, The (2000)	Comedy|Romance
7	868	4.278937	Death in Brunswick (1991)	Comedy
8	826	4.269901	Diebinnen (1995)	Drama
9	3305	4.266769	Bluebeard (1944)	Film-Noir|Horror
10	2619	4.265997	Mascara (1999)	Drama
11	763	4.264092	Last of the High Kings, The (a.k.a. Summer Fli...	Drama
12	1852	4.262517	Love Walked In (1998)	Drama|Thriller
13	642	4.260353	Roula (1995)	Drama
14	682	4.258829	Tigrero: A Film That Was Never Made (1994)	Documentary|Drama
15	792	4.253339	Hungarian Fairy Tale, A (1987)	Fantasy
16	1316	4.252915	Anna (1996)	Drama
17	3228	4.245526	Wirey Spindell (1999)	Comedy
18	853	4.240745	Dingo (1992)	Drama
19	3172	4.238188	Ulysses (Ulisse) (1954)	Adventure
20	2254	4.238008	Choices (1981)	Drama
21	2503	4.234547	Apple, The (Sib) (1998)	Drama
22	2905	4.224974	Sanjuro (1962)	Action|Adventure
23	744	4.224278	Brothers in Trouble (1995)	Drama
24	757	4.224226	Ashes of Time (1994)	Drama
25	858	4.223665	Godfather, The (1972)	Action|Crime|Drama
26	789	4.220788	I, Worst of All (Yo, la peor de todas) (1990)	Drama
27	3748	4.216508	Match, The (1999)	Comedy|Romance
28	790	4.216455	An Unforgettable Summer (1994)	Drama
29	745	4.215986	Close Shave, A (1995)	Animation|Comedy|Thriller

Gambar 11. Hasil prediksi rating dataset *Movielens 1M* dengan EMF

2.4. Perhitungan Performa

Berikut ini adalah tabel perbandingan Mean Absolute Error pada masing-masing algoritma dan dataset.

Metric	Dataset	User-based	Item-based
Euclidean	ML-100k	0.81	0.83
Euclidean	ML-1M	0.81	0.82
Cosine	ML-100k	0.75	0.51
Cosine	ML-1M	0.73	0.42

Gambar 12. Perbandingan MAE antara algoritma Used-based dan item-based

Preprocessing	Dataset	MF	NMF	EMF
Raw data	ML-100k	1.497	0.951	0.797
Raw data	ML-1M	1.482	0.9567	0.76
Normalized data	ML-100k	0.828	---	0.783
Normalized data	ML-1M	0.825	---	0.758

Gambar 13. Perbandingan MAE antara algoritma MF, NMF, dan EMF

3. Kesimpulan

a. Collaborative filtering adalah proses memprediksi melalui mesin rekomendasi. Mesin rekomendasi menganalisis informasi tentang pengguna dengan selera yang sama untuk menilai kemungkinan individu target akan menikmati sesuatu, seperti video, buku, atau item.

b. Terdapat dua teknik pada *memory-based collaboratuve filtering*, yaitu: (builtin)
   - *User-based,* yaitu mengukur kesamaan antara target pengguna dan pengguna lain
   - *Item-based,* yaitu mengukur kesamaan antara item yang diberi rating atau berinteraksi dengan pengguna target dan item lainnya.

c. Beberapa model algoritma yang dapat mengurangi dimensi, diantaranya:
   - Dekomposisi Nilai Singular (SVD),
   - Faktorisasi Matriks (MF)
   - Faktorisasi Matriks Non Negatif (NMF)
   - Faktorisasi Matriks yang Dapat Dijelaskan (EMF)

Daftar Pustaka

Builtin. *What Is Collaborative FilteringL A Simple Introduction*. Diakses dari https://builtin.com/data-science/collaborative-filtering-recommender-system tanggal 2 Januari 2023

Medium. *Recommendation System: User-Based Collaborative Filtering*. Diakses dari https://medium.com/grabngoinfo/recommendation-system-user-based-collaborative-filtering-a2e76e3e15c4 tanggal 2 Januari 2023

TechTarget. *Definition Collaborative filtering*. Diakses dari https://www.techtarget.com/whatis/definition/collaborative-filtering tanggal 2 Januari 2023

TowardsDataScience. *Item-Based Collaborative Filtering in Python*. Diakses dari https://towardsdatascience.com/item-based-collaborative-filtering-in-python-91f747200fab tanggal 2 Januari 2023

Wenga, Carmel. 2021. *Review on Collaborative Filtering*. Diakses dari https://github.com/nzhinusoftcm/review-on-collaborative-filtering tanggal 2 Januari 2023