Sistem Rekomendasi Film(collaborative Filtering)

Percobaan kali membuat sistem rekomendasi film dengan metode *collaborative filtering* yang memanfaatkan *ratings* film dari tiap user untuk memprediksi *ratings* dari *user* lain yang nantinya menampilkan rekomendasi film lain. Tutorial pembuatan dan dataset sistem rekomendasi dapat diakses di link berikut:
https://www.youtube.com/watch?v=6N2vo3JZg2c&list=PL3VpLbLyLE56IVsWwn_rlv9IGGNiyab9e&t=4127s

https://github.com/GilangAgungS/Film-Recommendation-Tensor

Secara garis besar pembuatan sistem rekomendasi ini dibagi menjadi 4 tahap yaitu pengambilan dataset, memisahkan dataset menjadi data *train* dan data *test*, pembuatan model dengan Tensorflow, dan *training* model yang telah dibuat. Berikut adalah langkah-langkah dari pembuatan sistem rekomendasi film:

1. *Import Library* untuk Mengolah Dataset

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

2. *Load* Dataset “ratings.csv” dan “movies.csv”

```
ratings = pd.read_csv('ratings.csv')
ratings.sample(5)
```

✓ 0.8s

	movie_id	movie_title	movie_genres_str
152	153	Fish Called Wanda, A (1988)	["Children's"]
226	227	Star Trek VI: The Undiscovered Country (1991)	['unknown', 'Action', 'Sci-Fi']
1210	1211	Blue Sky (1994)	['Documentary', 'Romance']
1446	1447	Century (1993)	['Documentary']
1054	1055	Simple Twist of Fate, A (1994)	['Documentary']

```
movies = pd.read_csv('movies.csv')
movies.sample(5)
```

✓ 0.4s

	movie_id	movie_title	movie_genres_str
342	343	Alien: Resurrection (1997)	['unknown', 'Film-Noir', 'Sci-Fi']
470	471	Courage Under Fire (1996)	['Documentary', 'Western']
548	549	Rob Roy (1995)	['Documentary', 'Romance', 'Western']
483	484	Maltese Falcon, The (1941)	['Fantasy', 'Mystery']
1300	1301	Stripes (1981)	["Children's"]

3. Melihat beberapa informasi dari data yang digunakan

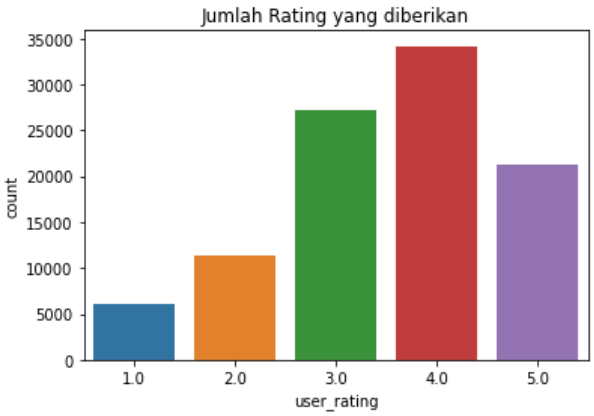
```
#melihat jumlah film dan user
jml_movie = movies['movie_id'].nunique()
jml_user = ratings['user_id'].nunique()
print("jumlah film:", jml_movie)
print("jumlah user:", jml_user)

#melihat jumlah rating yang diberikan
sns.countplot(data=ratings, x='user_rating').set_title('Jumlah Rating yang diberikan')
print('\njumlah rating yang diberikan:')
print(ratings['user_rating'].value_counts().sort_index(ascending=False))
```

Berikut adalah hasil dari kode di atas:

```
jumlah film: 1682
jumlah user: 943

jumlah rating yang diberikan:
5.0    21201
4.0    34174
3.0    27145
2.0    11370
1.0     6110
Name: user_rating, dtype: int64
```



Kebanyakan user memberikan *rating* 4 pada film yang ditonton

4. Memisahkan data menjadi data *train* dan data *test*

```
from sklearn.model_selection import train_test_split
data_train, data_test = train_test_split(ratings, test_size=0.2, random_state=42)

print("jumlah data train:", data_train.shape)
print("jumlah data test :", data_test.shape)
```

✓ 0.4s

```
jumlah data train: (80000, 3)
jumlah data test : (20000, 3)
```

Data dibagi dengan proporsi 20% untuk *testing* dan 80% untuk *training*, menggunakan random_state agar hasil *training* tidak berubah ketika dijalankan lagi.

5. Membuat model

Membuat input layer dengan ukuran 1

```
#import tensorflow
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, Embedding, Flatten, Dot

#membuat input layer dari movie dan user, ukuran masing-masing input adalah 1
movie_input = Input(shape=[1])
user_input = Input(shape=[1])
```

Membuat embedding layer

Embedding size/dimension dapat diatur sesuai keinginan, namun embedding size yang terlalu kecil menyebabkan model hanya menghafal data sehingga rekomendasi yang dimunculkan hanya itu-itu saja bahkan untuk *user* yang berbeda, sedangkan embedding size terlalu besar rawan terjadi overfitting.

```
embedding_size = 40
movie_embedding = Embedding(jml_movie+1, embedding_size)(movie_input)
user_embedding = Embedding(jml_user+1, embedding_size)(user_input)
```

Membuat flatten layer

Flatten layer berfungsi untuk mengubah data array multidimensi menjadi vektor 1 dimensi agar bisa digunakan untuk input dalam fully connected layer

```
#flatten layer bertujuan untuk mengubah bentuk data menjadi 1 dimensi
movie_flatten = Flatten()(movie_embedding)
user_flatten = Flatten()(user_embedding)
```

Membuat output layer

Output layer berfungsi untuk menghasilkan rekomendasi

```
#output layer bertujuan untuk menghasilkan nilai rating
output = Dot(axes=1)([movie_flatten, user_flatten])
```

Menyusun layer-layer menjadi model

Masukkan input layer yang akan dimasukkan dan output layernya

```
#membuat model
model = Model([movie_input, user_input], output)
```

Melihat rangkaian layer yang telah dibuat sebelumnya
Seluruh rangkaian layer sesuai dengan kode yang kita buat sebelumnya, mulai dari 2 input layer, 2 embedding layer dengan embedding size / dimension sebesar 40, flatten layer untuk mengubah data menjadi vektor 1 dimensi, dan diakhiri dengan output layer(Dot).

```
model.summary()
```
✓ 0.4s

```
Model: "model_36"
__________________________________________________________________________________________________
 Layer (type)                Output Shape              Param #   Connected to
==================================================================================================
 input_75 (InputLayer)       [(None, 1)]               0         []

 input_76 (InputLayer)       [(None, 1)]               0         []

 embedding_74 (Embedding)    (None, 1, 40)             67320     ['input_75[0][0]']

 embedding_75 (Embedding)    (None, 1, 40)             37760     ['input_76[0][0]']

 flatten_74 (Flatten)        (None, 40)                0         ['embedding_74[0][0]']

 flatten_75 (Flatten)        (None, 40)                0         ['embedding_75[0][0]']

 dot_37 (Dot)                (None, 1)                 0         ['flatten_74[0][0]',
                                                                  'flatten_75[0][0]']

==================================================================================================
Total params: 105,080
Trainable params: 105,080
Non-trainable params: 0
```

6. Training model
Konfigurasi hyperparameter dengan jenis optimizer ‘adam’ dan perhitungan loss dengan MSE.

```
from tensorflow.keras.optimizers import Adam
model.compile(optimizer='adam', loss='mse')
```

Artikel tentang optimizer ‘adam’:
https://medium.com/@saritilawah9/adam-optimizer-80cc267522af
Artikel tentang MSE(Mean Square Error):
https://dqlab.id/kriteria-jenis-teknik-analisis-data-dalam-forecasting

Melakukan training dengan x sebagai features yang berisi data_train, y sebagai target prediksi, validation_data berisi data test yang sebelumnya sudah dipisah, pelatihan model diulang sebanyak 20 kali(epochs) dengan 256 data dalam sekali training, verbose=1 artinya memunculkan proses training dan menunjukkan loss dalam tiap epochs.

```
#training model
history = model.fit(x=[data_train.movie_id, data_train.user_id], y=data_train.user_rating,
                    validation_data=([data_test.movie_id, data_test.user_id], data_test.user_rating),
                    epochs=20,
                    batch_size=256,
                    verbose=1)
```

Hasil:

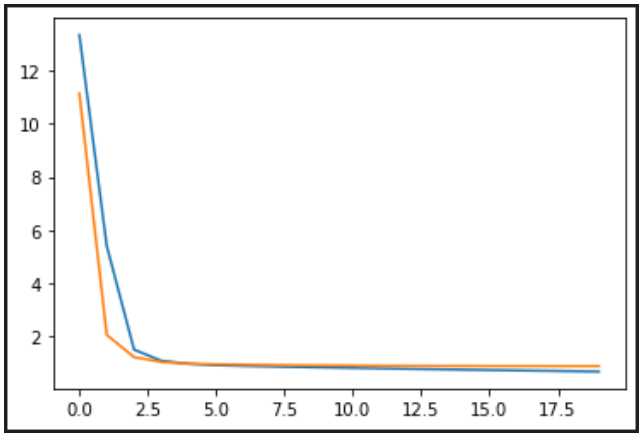
Dari hasil val_loss di epochs terakhir, kebanyakan loss berada di angka 0,8.

```
Epoch 1/20
313/313 [==============================] - 1s 2ms/step - loss: 13.3495 - val_loss: 11.1452
Epoch 2/20
313/313 [==============================] - 1s 2ms/step - loss: 5.3946 - val_loss: 2.0376
Epoch 3/20
313/313 [==============================] - 1s 2ms/step - loss: 1.4793 - val_loss: 1.1958
Epoch 4/20
313/313 [==============================] - 1s 2ms/step - loss: 1.0605 - val_loss: 1.0140
Epoch 5/20
313/313 [==============================] - 1s 2ms/step - loss: 0.9425 - val_loss: 0.9521
Epoch 6/20
313/313 [==============================] - 1s 2ms/step - loss: 0.8931 - val_loss: 0.9275
Epoch 7/20
313/313 [==============================] - 1s 2ms/step - loss: 0.8655 - val_loss: 0.9114
Epoch 8/20
313/313 [==============================] - 1s 2ms/step - loss: 0.8442 - val_loss: 0.9001
Epoch 9/20
313/313 [==============================] - 1s 2ms/step - loss: 0.8250 - val_loss: 0.8908
Epoch 10/20
313/313 [==============================] - 1s 2ms/step - loss: 0.8071 - val_loss: 0.8830
Epoch 11/20
313/313 [==============================] - 1s 2ms/step - loss: 0.7886 - val_loss: 0.8774
Epoch 12/20
313/313 [==============================] - 1s 2ms/step - loss: 0.7708 - val_loss: 0.8718
Epoch 13/20
...
Epoch 19/20
313/313 [==============================] - 1s 2ms/step - loss: 0.6662 - val_loss: 0.8583
Epoch 20/20
313/313 [==============================] - 1s 2ms/step - loss: 0.6521 - val_loss: 0.8584
```

Menampilkan grafik training(loss dan validation loss)

```
import matplotlib.pyplot as plt
losses = pd.DataFrame(history.history)
plt.plot(losses)
```

Grafik di bawah menunjukkan bahwa loss pada data training semakin mengecil, sedangkan loss pada data validasi pada epochs ke-19, jika epochs pada training ditambah maka berpotensi untuk overfitting(nilai loss pada data validasi semakin naik).



7. Mencoba prediksi ratings dengan model yang telah dibuat

```
def get_recommendations(user_id, movies, model):
  movies = movies.copy()
  user_ids = np.array([user_id] * len(movies))
  results = model([movies['movie_id'].values, user_ids]).numpy().reshape(-1)

  movies['predicted_rating'] = pd.Series(results)
  movies = movies.sort_values('predicted_rating', ascending=False)

  print(f'rekomendasi untuk user {user_id}')
  return movies

get_recommendations(712, movies, model).head(10)
```

Hasil:

```
rekomendasi untuk user 712
```

	movie_id	movie_title	movie_genres_str	predicted_rating
142	143	Sound of Music, The (1965)	['Musical']	4.990651
120	121	Independence Day (ID4) (1996)	['unknown', 'Sci-Fi', 'Western']	4.939096
401	402	Ghost (1990)	["Children's", 'Romance', 'Thriller']	4.935175
391	392	Man Without a Face, The (1993)	['Documentary']	4.919121
65	66	While You Were Sleeping (1995)	["Children's", 'Romance']	4.859806
719	720	First Knight (1995)	['unknown', 'Action', 'Documentary', 'Romance']	4.764417
691	692	American President, The (1995)	["Children's", 'Documentary', 'Romance']	4.763218
965	966	Affair to Remember, An (1957)	['Romance']	4.754288
738	739	Pretty Woman (1990)	["Children's", 'Romance']	4.748395
70	71	Lion King, The (1994)	['Adventure', 'Animation', 'Musical']	4.713131

8. Menyimpan Model

Simpan dengan format .h5

```
model.save('model.h5')
```

0.8s

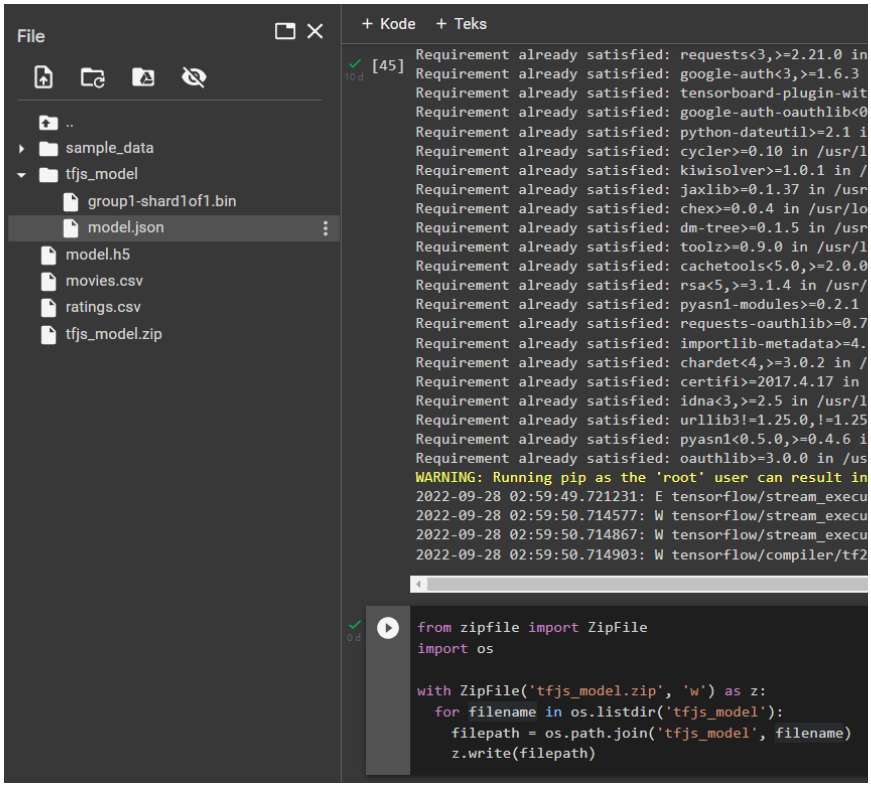
Convert model menjadi format .json, untuk mempermudah proses convert kita bisa upload file .ipynb dan dataset ke google colab, jalankan semua cell kemudian jalankan kode berikut

```
!pip install tensorflowjs
!tensorflowjs_converter --input_format=keras ./model.h5 ./tfjs_model
```

```
from zipfile import ZipFile
import os

with ZipFile('tfjs_model.zip', 'w') as z:
  for filename in os.listdir('tfjs_model'):
    filepath = os.path.join('tfjs_model', filename)
    z.write(filepath)
```

Berikut adalah hasilnya:



Kita bisa download file hasil convert ke memori local