MOVIE RECOMMENDATION SYSTEM

Disusun oleh:

Triska Paskah Uli Pangaribuan	11S17011	2017
Angel Alicya Lumania Napitupulu	11S17028	2017
Gracella Romauly Tambunan	11S17040	2017
Netty Togi Marito Siahaan	11S17045	2017

INSTITUT TEKNOLOGI DEL
LAGUBOTI
2019

MOVIE RECOMMENDATION SYSTEM WITH BIGGER DATASET

Movie Recommendation System adalah suatu system yang berfungsi untuk memberi saran kepada user tentang movie apa yang sebaiknya dipilih atau ditonton. Saran ini berkaitan dengan bermacam proses pengambilan keputusan.
Contoh mudahnya, seseorang menyukai film Lord of The Ring, kemudian ketika orang tersebut ingin mengetahui film yang sesuai dengan genre yang disukainya menggunakan movie recommendation system. Dalam pemrosesan, system kemudian menemukan bahwa orang-orang yang menyukai film Lord of The Ring juga menyukai film The Hobbit, Game of Throne, dan Seven Kingdoms. Dari hasil tersebut, maka system memutuskan bahwa ketiga film itulah yang akan direkomendasikan kepada orang tersebut.

Bagaiman cara system memberikan rekomendasi ?

1. Menghitung Nilai Similarity

   Dataset terdiri dari nama orang serta film-film rekomendasi mereka berdasarkan rating, nilai rating antara 0.0 – 5.0 . Nilai similarity digunakan untuk mengetahui seberapa sama dua objek . Ada dua nilai yang sering digunakan dalam menghitung nilai similarity, yaitu Euclidean Score dan Pearson Score.

   - **Euclidean Score**

     Euclidean Score menggunakan jarak Euclidean untuk menghitung score. Score Euclidean berkisar 0 – 1. Jika jarak Euclidean besar, maka score Euclidean nya rendah, dan menunjukkan kedua objek tidak mirip. Jarak Euclidean dan Euclidean Score berbanding terbalik.

     Pengimplementasian dalam code python:

```
# Compute the Pearson correlation score between user1 and user2
def pearson_score(dataset, user1, user2):
    if user1 not in dataset:
        raise TypeError('Cannot find ' + user1 + ' in the dataset')

    if user2 not in dataset:
        raise TypeError('Cannot find ' + user2 + ' in the dataset')
```

     ⇨ Jika user yang diinput tidak ada dalam dataset,maka muncul user tidak dapat ditemukan.

```
# Movies rated by both user1 and user2
common_movies = {}

for item in dataset[user1]:
    if item in dataset[user2]:
        common_movies[item] = 1

# If there are no common movies between the users,
# then the score is 0
if len(common_movies) == 0:
    return 0
```

⇨ Movie yang sama- sama diberi ratin oleh user1 dan user 2 dimasukkan ke common movies dengan nilai 1. Jika antar user tidak ada movie yang sama-sama diberi rating maka common movie diberi nilai 0.

Jarak Euclidean :

person×movie↦score

$$d(p_1, p_2) = \sqrt{\sum_{i \in \text{item}} (s_{p_1} - s_{p_2})^2}$$

Keterangan:

d = Distance (jarak)
p1 = User 1
p2 = User 2
sp1 = Score User 1
sp2 = Score User 2

Pengimplementasian Rumus pada code:

```
squared_diff = []

for item in dataset[user1]:
    if item in dataset[user2]:
        squared_diff.append(np.square(dataset[user1][item] -
dataset[user2][item]))

return 1 / (1 + np.sqrt(np.sum(squared_diff)))
```

⇨ jika user 1 dan user 2 berada pada dataset maka dapat dihitung jarak euclideannya.

Euclidean Score :

$$\frac{1}{1 + d(p_1, p_2)}$$

Keterangan :

d = Distance (jarak)
p1 = User 1
p2 = User 2

Pengimplementasian pada code:

```
return 1 / (1 + np.sqrt(np.sum(squared_diff)))
```

⇨ jika user 1 dan user 2 berada pada dataset maka dapat dihitung euclidean scorenya.

❖ **Pearson Score**

Pearson Score adalah ukuran korelasi antara dua objek. Pearson Score menggunakan kovarians antar dua objek dan standar deviasi masing-masing untuk menghitung skor. Nilai dapat berkisar dari -1 sampai dengan +1. Skor +1 menunjukkan bahwa objek sangat mirip, dimana skor -1 akan menunjukkan bahwa objek sangat berbeda. Skor 0 akan mengindikasikan hal tersebut tidak ada korelasi antara kedua objek.

Pengimplementasian dalam code python:

```
# Compute the Pearson correlation score between user1 and user2
def pearson_score(dataset, user1, user2):
    if user1 not in dataset:
        raise TypeError('Cannot find ' + user1 + ' in the dataset')

    if user2 not in dataset:
        raise TypeError('Cannot find ' + user2 + ' in the dataset')
```

⇨ Jika user yang diinput tidak ada dalam dataset,maka muncul user tidak dapat ditemukan.

Secara matematis, score dapat digambarkan sebagai:

$$Pearson(x,y) = \frac{\sum xy - \frac{\sum x \sum y}{N}}{\sqrt{(\sum x^2 - \frac{(\sum x)^2}{N})(\sum y^2 - \frac{(\sum y)^2}{N})}}$$

Keterangan:

x,y = data objek(movie)

N = Num rating = panjang dari common movies

Common movies = movie yang sama sama diberi rating oleh user dan diberi nilai 1

$\sum xy$ = jumlah perkalian antara movies dalam common rating

$\sum x$ = jumlah antara movies user 1

$\sum y$ = jumlah antara movies user 2

$\sum x^2$ = jumlah kuadrat antara movies user 1

$\sum y^2$ = jumlah kuadrat antara movies user 2

$(\sum x)^2$ = jumlah antara movies user 1 dikuadratkan

$(\sum y)^2$ = jumlah antara movies user 2 dikuadratkan

Pengimplementasian pada code:

```
    # Movies rated by both user1 and user2
    common_movies = {}

    for item in dataset[user1]:
        if item in dataset[user2]:
            common_movies[item] = 1

    # If there are no common movies between the users,
    # then the score is 0
    if len(common_movies) == 0:
        return 0


num_ratings = len(common_movies)

# If there are no common movies between user1 and user2, then the score is 0
if num_ratings == 0:
    return 0
```

⇨ Movie yang sama- sama diberi ratin oleh user1 dan user 2 dimasukkan ke common movies dengan nilai 1. Jika antar user tidak ada movie yang sama-sama diberi rating maka common movie diberi nilai 0, jika panjang common movie = 0 maka num_rating = 0 sehingga scorenya = 0

```
# Calculate the sum of ratings of all the common movies
user1_sum = np.sum([dataset[user1][item] for item in common_movies])
user2_sum = np.sum([dataset[user2][item] for item in common_movies])
```

⇨ Menghitung jumlah rating dari semua common movies

```
# Calculate the sum of squares of ratings of all the common movies
user1_squared_sum = np.sum([np.square(dataset[user1][item]) for item in
common_movies])
user2_squared_sum = np.sum([np.square(dataset[user2][item]) for item in
common_movies])
```

⇨ Menghitung jumlah kuadrat dari rating dari semua common movies

```
# Calculate the sum of products of the ratings of the common movies
sum_of_products = np.sum([dataset[user1][item] * dataset[user2][item] for
item in common_movies])
```

⇨ Menghitung jumlah rating dari perkalian antar semua movies dalam common movies

```
# Calculate the Pearson correlation score
Sxy = sum_of_products - (user1_sum * user2_sum / num_ratings)
Sxx = user1_squared_sum - np.square(user1_sum) / num_ratings
Syy = user2_squared_sum - np.square(user2_sum) / num_ratings
```

⇨ Menghitung nilai korelasi pearson

```
if Sxx * Syy == 0:
    return 0
```

⇨ Jika tidak ada deviasi maka scorenya = 0

```
return Sxy / np.sqrt(Sxx * Syy)
```

⇨ Menghitung nilai pearson

Membangun sistem movie recommendation

```
import json
import numpy as np

from compute_scores import pearson_score
```

```
# Get movie recommendations for the input user
def get_recommendations(dataset, input_user):
    if input_user not in dataset:
        raise TypeError('Cannot find ' + input_user + ' in the dataset')

    overall_scores = {}
    similarity_scores = {}

    for user in [x for x in dataset if x != input_user]:
        similarity_score = pearson_score(dataset, input_user, user)

        if similarity_score <= 0:
            continue

        filtered_list = [x for x in dataset[user] if x not in \
                dataset[input_user] or dataset[input_user][x] == 0]

        for item in filtered_list:
            overall_scores.update({item: dataset[user][item] *
similarity_score})
            similarity_scores.update({item: similarity_score})

    if len(overall_scores) == 0:
        return ['No recommendations possible']
```

⇨ Mendapatkan rekomendasi movie dari input user di dataset jika input user tidak ada maka akan menampilkan “user tidak ditemukan”, menghitung movie recommendation berdasarkan overall_scores dan similarity_score, jika overall_scores = 0, maka tidak ada movie recommendation untuk input user tersebut.

```
# Generate movie ranks by normalization
movie_scores = np.array([[score/similarity_scores[item], item]
        for item, score in overall_scores.items()])

# Sort in decreasing order
movie_scores = movie_scores[np.argsort(movie_scores[:, 0])[::-1]]

# Extract the movie recommendations
movie_recommendations = [movie for _, movie in movie_scores]

return movie_recommendations
```

⇨ Fungsi untuk mengenerate movie recommendation (dengan fugsi normalisasi) dan mengurutkannya berdasarkan rating tertinggi ke rating terendah dan menampilkannya dalam bentuk array

```
ratings_file = 'ratings.json'

with open(ratings_file, 'r') as f:
    data = json.loads(f.read())

user = input("\nInput user: ")
print("\nMovie recommendations for " + user + ":")
movies = get_recommendations(data, user)
for i, movie in enumerate(movies):
    print(str(i+1) + '. ' + movie)
```

⇨ mengambil dataset dari rating.json dan mengenerate movie recommendation berdasarkan input user.

Kesimpulan

Recommendation sistem dapat menjadio salah satu teknik dimana dapat menentukan rekomendasi kepada pengguna lewat kebiassan atau interaksi dengan pengguna lainnya maupun dengan beberapa item yang terkait dengan item yang digunakan oleh pengguna.