Create and compare movie recommendation systems

Task 1:

Creating a content based recommendation system:

For this , recommendations are given based on the cosine similarity between user vector and tfidf matrix.

Tfidf is the feature matrix in which features are created based on the movie genres (hence content based recommendation) .

User vector is a vector corresponding to every user containing information about the past movies the user has seen , the genre of those movies and rating given by the user to those movies. Overall this gives an idea of what genres of movies have been like by the user.

```
[5]: import pandas as pd
     from sklearn.feature_extraction.text import TfidfVectorizer
     from sklearn.metrics.pairwise import cosine_similarity
```

```
[6]: movies['clean_genres'] = movies['genres'].str.replace('\|', ' ', regex=True)
```

```
[7]: tfidf_vectorizer = TfidfVectorizer(stop_words='english')
     tfidf_matrix = tfidf_vectorizer.fit_transform(movies['clean_genres'])
```

```
[8]: # Create a matrix of zeros of shape (num_users, num_genres_features)
     user_profiles = pd.DataFrame(index=ratings['userId'].unique(), columns=tfidf_vectorizer.get_feature_names_out(), data=0.0)

     for index, row in ratings.iterrows():
         # Get the index of the movie in the original movies DataFrame
         movie_idx = movies.index[movies['movieId'] == row['movieId']].tolist()[0]

         # Add the weighted genres to the user's profile
         user_profiles.loc[row['userId']] += tfidf_matrix[movie_idx].toarray().flatten() * row['rating']

     # Normalize the user profiles
     user_profiles = user_profiles.div(user_profiles.sum(axis=1), axis=0)
```

Thus by calculating the cosine similarity between these two , the movies having largest similarity will be placed higher in recommendations.

```
[9]: def recommend_movies(user_id, user_profiles, tfidf_matrix, movies, top_n=10):
         # Compute cosine similarity between user profile and all movie genre vectors
         user_vector = user_profiles.loc[user_id].values.reshape(1, -1)
         cosine_sim = cosine_similarity(user_vector, tfidf_matrix)

         # Get indices of the top_n most similar movies
         top_movie_indices = cosine_sim.argsort().flatten()[-top_n:][::-1]

         # Fetch the movie titles based on the indices
         recommended_movies = movies.iloc[top_movie_indices]
         return recommended_movies[['title', 'genres']]
```

Example"

```
[19]: user_id = 1  # Assuming this is a valid user ID in your dataset
recommended_movies = recommend_movies(user_id, user_profiles, tfidf_matrix, movies, top_n=10)
print("Recommended movies for user", user_id, ":\n", recommended_movies)
```

```
Recommended movies for user 1 :
                                     title genres
16722  Brink of Life (Nära livet) (1958)  Drama
17575                        Aurora (2010)  Drama
50513             Closed For Winter (2009)  Drama
50510         An Ordinary Execution (2010)  Drama
17555              Chak De India! (2007)  Drama
17558             Cow, The (Gaav) (1969)  Drama
17560                  Local Color (1977)  Drama
17570                Iron Lady, The (2011)  Drama
50498              Untold Scandal (2003)  Drama
50496                     U-Carmen (2005)  Drama
```

```
[18]: user_id = 304  # Assuming this is a valid user ID in your dataset
recommended_movies = recommend_movies(user_id, user_profiles, tfidf_matrix, movies, top_n=10)
print("Recommended movies for user", user_id, ":\n", recommended_movies)
```

```
Recommended movies for user 304 :
                              title  \
55664      Red Peony Gambler (1968)
33940       Once Upon a Time (2008)
33107        Joseph Andrews (1977)
4850         Stunt Man, The (1980)
26854              Longshot (2001)
46316  Under New Management (2009)
62180       Thiruda Thirudi (2003)
53633          Rajathi Raja (1989)
10996             Wing Chun (1994)
47918             Ratchagan (1997)

                                                genres
55664             Action|Comedy|Drama|Romance|Thriller
33940  Action|Adventure|Comedy|Crime|Drama|Romance|Th...
33107   Action|Adventure|Comedy|Drama|Romance|Thriller
4850    Action|Adventure|Comedy|Drama|Romance|Thriller
26854       Action|Comedy|Crime|Drama|Romance|Thriller
46316       Action|Comedy|Crime|Drama|Romance|Thriller
62180                    Action|Comedy|Drama|Romance
53633                    Action|Comedy|Drama|Romance
10996                    Action|Comedy|Drama|Romance
47918                    Action|Comedy|Drama|Romance
```

[20]:

	action	adventure	animation	children	comedy	crime	documentary	drama	fantasy	fi	...	imax	listed	musical	mystery	noir	r
1	0.025210	0.061352	0.010406	0.015790	0.120035	0.055394	0.005220	0.343812	0.029497	0.026298	...	0.000000	0.000000	0.037605	0.023370	0.005723	(
2	0.113110	0.147385	0.026886	0.044459	0.087553	0.033578	0.000000	0.151848	0.061173	0.054548	...	0.019272	0.000000	0.021169	0.018371	0.000000	(
3	0.131121	0.083873	0.024032	0.021450	0.074023	0.074470	0.002058	0.092067	0.037217	0.096675	...	0.049452	0.000578	0.003656	0.032699	0.002977	(
4	0.149023	0.126126	0.040634	0.032669	0.098098	0.059669	0.012989	0.044566	0.044073	0.093294	...	0.048727	0.000000	0.012827	0.027356	0.000000	(
5	0.059014	0.077293	0.010885	0.028061	0.186324	0.066377	0.000000	0.154716	0.027965	0.042086	...	0.014589	0.000000	0.030488	0.039392	0.000000	(
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
7717	0.120567	0.133552	0.019622	0.019612	0.161468	0.092108	0.000000	0.046998	0.050525	0.106094	...	0.033087	0.000000	0.000000	0.010459	0.000000	(
7718	0.094736	0.059922	0.007722	0.009906	0.101160	0.094864	0.000000	0.141793	0.029766	0.053082	...	0.006523	0.000000	0.006022	0.048364	0.001800	(
7719	0.042021	0.057264	0.037311	0.009658	0.058161	0.072780	0.020114	0.156777	0.053462	0.078251	...	0.013287	0.000000	0.011106	0.093765	0.006074	(
7720	0.064718	0.129999	0.020587	0.083607	0.122615	0.058512	0.000000	0.087287	0.030412	0.051752	...	0.000000	0.000000	0.028048	0.015595	0.000000	(

This is a part of the user profile matrix , I have uploaded it to the github link.

Getting user profile matrix involved large computations , So I have used a Virtual Machine created using microsoft Azure for these computations.

<u>Creating a collaborative filtering based recommendation system</u>

This method involves recommending movies based on similar users.

Similar users are found based on the users who have given high rating to the same movie which the user for which recommendation is to be done. Top 10% movies (ones which have maximum similar users giving high rating are selected).

[10]:

```
similar_users = ratings[(ratings["movieId"] == movie_id) & (ratings["rating"] > 4)]["userId"].unique()
```

[11]:

```
similar_user_recs = ratings[(ratings["userId"].isin(similar_users)) & (ratings["rating"] > 4)]["movieId"]
```

[12]:

```
similar_user_recs = similar_user_recs.value_counts() / len(similar_users)

similar_user_recs = similar_user_recs[similar_user_recs > .10]
```

Apart from this , another data is collected, that is the movies which are liked by all the users. This is due to the fact that there are some movies which are liked by all the users , so they dont contribute to the similarity between two users.

[13]:

```python
all_users = ratings[(ratings["movieId"].isin(similar_user_recs.index)) & (ratings["rating"] > 4)]
```

[14]:

```python
all_user_recs = all_users["movieId"].value_counts() / len(all_users["userId"].unique())
```

[16]:

```python
rec_percentages
```

[16…

	similar	all
movieId		
89745	1.000000	0.040459
58559	0.573393	0.148256
59315	0.530649	0.054931
79132	0.519715	0.132987
2571	0.496687	0.247010
...	...	...
47610	0.103545	0.022770
780	0.103380	0.054723
88744	0.103048	0.010383
1258	0.101226	0.083887
1193	0.100895	0.120244

193 rows × 2 columns

This is an example for record percentage for a given movie_id = 89745.

So movies having more difference between ‘similar’ and ‘all’ values is a better recommendation.

Complete code in next page.

[20]:
```python
def find_similar_movies(movie_id):
    similar_users = ratings[(ratings["movieId"] == movie_id) & (ratings["rating"] > 4)]["userId"].unique()
    similar_user_recs = ratings[(ratings["userId"].isin(similar_users)) & (ratings["rating"] > 4)]["movieId"]
    similar_user_recs = similar_user_recs.value_counts() / len(similar_users)

    similar_user_recs = similar_user_recs[similar_user_recs > .10]
    all_users = ratings[(ratings["movieId"].isin(similar_user_recs.index)) & (ratings["rating"] > 4)]
    all_user_recs = all_users["movieId"].value_counts() / len(all_users["userId"].unique())
    rec_percentages = pd.concat([similar_user_recs, all_user_recs], axis=1)
    rec_percentages.columns = ["similar", "all"]

    rec_percentages["score"] = rec_percentages["similar"] / rec_percentages["all"]
    rec_percentages = rec_percentages.sort_values("score", ascending=False)
    return rec_percentages.head(10).merge(movies, left_index=True, right_on="movieId")[["score", "title", "genres"]]
```

+ Code + Markdown

[21]:
```python
import ipywidgets as widgets
from IPython.display import display

movie_name_input = widgets.Text(
    value='Toy Story',
    description='Movie Title:',
    disabled=False
)
recommendation_list = widgets.Output()

def on_type(data):
    with recommendation_list:
        recommendation_list.clear_output()
        title = data["new"]
        if len(title) > 5:
            results = search(title)
            movie_id = results.iloc[0]["movieId"]
            display(find_similar_movies(movie_id))

movie_name_input.observe(on_type, names='value')

display(movie_name_input, recommendation_list)
```

Results:

Movie Title: toy story

	score	title	genres
3021	18.841924	Toy Story 2 (1999)	Adventure|Animation|Children|Comedy|Fantasy
2264	8.210086	Bug's Life, A (1998)	Adventure|Animation|Children|Comedy
2669	6.868954	Iron Giant, The (1999)	Adventure|Animation|Children|Drama|Sci-Fi
14813	6.503216	Toy Story 3 (2010)	Adventure|Animation|Children|Comedy|Fantasy|IMAX
3650	6.272875	Chicken Run (2000)	Animation|Children|Comedy
1992	5.531892	Little Mermaid, The (1989)	Animation|Children|Comedy|Musical|Romance
1818	5.362941	Mulan (1998)	Adventure|Animation|Children|Comedy|Drama|Musi...
2895	5.349396	Who Framed Roger Rabbit? (1988)	Adventure|Animation|Children|Comedy|Crime|Fant...
0	5.287943	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
3082	5.283613	Galaxy Quest (1999)	Adventure|Comedy|Sci-Fi

Movie Title: jumanji

	score	title	genres
1	57.008249	Jumanji (1995)	Adventure|Children|Fantasy
156	18.757121	Casper (1995)	Adventure|Children
313	14.880390	Santa Clause, The (1994)	Comedy|Drama|Fantasy
578	9.382034	Home Alone (1990)	Children|Comedy
495	8.711980	Mrs. Doubtfire (1993)	Comedy|Drama
362	8.666058	Mask, The (1994)	Action|Comedy|Crime|Fantasy
2526	7.959267	Mummy, The (1999)	Action|Adventure|Comedy|Fantasy|Horror|Thriller
721	7.539414	Twister (1996)	Action|Adventure|Romance|Thriller
579	6.375923	Ghost (1990)	Comedy|Drama|Fantasy|Romance|Thriller
312	6.231007	Stargate (1994)	Action|Adventure|Sci-Fi