- Assignment: 10.2 Recommender System
- Name: Barath Anandaraman
- Course: DSC630-T301
- Week10: Recommender Systems
- Date: 05/16/2025

Movie Recommender System using Content-Based and Collaborative Filtering

Techniques Used:

- Content-Based Filtering using TF-IDF on genres and tags
- Collaborative Filtering using Truncated SVD and Nearest Neighbors

---

Step1: Load the necessary packages

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.neighbors import NearestNeighbors
from sklearn.decomposition import TruncatedSVD
from sklearn.metrics.pairwise import cosine_similarity
```

Step2: Load and understand the data

In [9]:
```
# Load data
movies = pd.read_csv('ml-latest-small/movies.csv')
ratings = pd.read_csv('ml-latest-small/ratings.csv')
tags = pd.read_csv('ml-latest-small/tags.csv')
links = pd.read_csv('ml-latest-small/links.csv')
```

In [13]:
```python
print("*****Movies dataset Sample*****\n")
print(movies.head(3))
print("\n*****Ratings dataset Sample*****\n")
print(ratings.head(3))
print("\n*****Tags dataset Sample*****\n")
print(tags.head(3))
print("\n*****Links dataset Sample*****\n")
print(links.head(3))
```

```
*****Movies dataset Sample*****

   movieId                    title  \
0        1         Toy Story (1995)
1        2           Jumanji (1995)
2        3  Grumpier Old Men (1995)

                                        genres
0  Adventure|Animation|Children|Comedy|Fantasy
1                   Adventure|Children|Fantasy
2                               Comedy|Romance

*****Ratings dataset Sample*****

   userId  movieId  rating  timestamp
0       1        1     4.0  964982703
1       1        3     4.0  964981247
2       1        6     4.0  964982224

*****Tags dataset Sample*****

   userId  movieId              tag   timestamp
0       2    60756            funny  1445714994
1       2    60756  Highly quotable  1445714996
2       2    60756     will ferrell  1445714992

*****Links dataset Sample*****

   movieId  imdbId   tmdbId
0        1  114709    862.0
1        2  113497   8844.0
2        3  113228  15602.0
```

Key Columns:

- movies: movieId, title, genres

- ratings: userId, movieId, rating
- tags: userId, movieId, tag

Links dataset can be ignored as columns imdId and tmdbId are for referencing outer resources

---

Step3: Content - Based Recommender

3.1 Prepare Movie Content Genre + tag

In [24]:

```
# Aggregate tags and merge with genres
agg_tags = tags.groupby('movieId')['tag'].apply(lambda x: ' '.join(set(x))).reset_index()

# Merge with movie genres
movie_content = pd.merge(movies,agg_tags, on='movieId', how='left')
movie_content['tag'] = movie_content['tag'].fillna('')
movie_content['content'] = movie_content['genres'] + ' ' + movie_content['tag']
```

3.2 Vectorize using TfidfVectorizer and Apply Nearest Neighbors

TfidfVectorizer to turn movie content (genres + tags) into numerical vectors

Nearest Neighbor to find the most similar vectors with metric cosine as we are not working in Numerical or user behavior(ratings)

In [25]:

```
# TfidfVectorizer to turn movie content (genres + tags) into numerical vectors
tfidf = TfidfVectorizer(stop_words='english')
tfidf_matrix = tfidf.fit_transform(movie_content['content'])

# Build Nearest Neighbors model
nn_model = NearestNeighbors(metric='cosine', algorithm='brute')
nn_model.fit(tfidf_matrix)
```

Out[25]:

```
NearestNeighbors
NearestNeighbors(algorithm='brute', metric='cosine')
```

3.3 Use MovieId for indexing, to perform search by movieID based on title, and Build mappings

In [27]:

```
# Use MovieId for indexing and build title mappings
movie_content = movie_content.sort_values('movieId').reset_index(drop=True)
```

```python
movieId_to_index = {mid: idx for idx, mid in enumerate(movie_content['movieId'])}
index_to_movieId = {idx: mid for mid, idx in movieId_to_index.items()}
movieId_to_title = movies.set_index('movieId')['title'].to_dict()
title_to_movieId = {v: k for k,v in movieId_to_title.items()}
```

3.4 Create Recommender Function

In [35]:
```python
# Function to recommend top 10 titles based on movie title
def recommend_content_nn(title, top_n=10):
    if title not in title_to_movieId:
        return f"Movie '{title}' not found in."
    movie_id = title_to_movieId[title]
    if movie_id not in movieId_to_title:
        return f"Movie ID for '{title}' not found in content."
    idx=movieId_to_index[movie_id]
    distances, indices_nn = nn_model.kneighbors(tfidf_matrix[idx], n_neighbors=top_n+1)
    similar_ids = [index_to_movieId[i] for i in indices_nn[0][1:]]
    return [movieId_to_title[mid] for mid in similar_ids]
```

3.5 Try a Movie

In [29]:
```python
recommend_content_nn("Toy Story (1995)")
```

Out[29]:
```
["Bug's Life, A (1998)",
 'Guardians of the Galaxy 2 (2017)',
 'Moana (2016)',
 'Antz (1998)',
 "Emperor's New Groove, The (2000)",
 'Asterix and the Vikings (Astérix et les Vikings) (2006)',
 'The Good Dinosaur (2015)',
 'Adventures of Rocky and Bullwinkle, The (2000)',
 'Shrek the Third (2007)',
 'Monsters, Inc. (2001)']
```

In [53]:
```python
recommend_content_nn("Moana (2016)")
```

Out[53]:
```
["Emperor's New Groove, The (2000)",
 'Tale of Despereaux, The (2008)',
 'The Good Dinosaur (2015)',
 'Moana (2016)',
 'Adventures of Rocky and Bullwinkle, The (2000)',
 'Wild, The (2006)',
 'Shrek the Third (2007)',
 'Asterix and the Vikings (Astérix et les Vikings) (2006)',
 'Antz (1998)',
 'Monsters, Inc. (2001)']
```

Step4. Collaborative Filtering - Based Recommender

4.1 Prepare user-Movie Ratings matrix for collaborative filtering

In [30]:
```python
# Pivot ratings into user-movie matrix
user_movie_matrix = ratings.pivot_table(index='userId', columns = 'movieId', values='rating').fillna(0)

# Transpose to get movie-user matrix(rows: movies, columns: users)
movie_user_matrix = user_movie_matrix.T
```

4.2 Matrix Factorization using Truncated SVD and fit nearest neighbors on latent factors

- Matrix factorization to estimate user preferences and make recommendations
- Truncated SVD to reduce dimensionality and learn latent features of movies
- Nearestneighbors to find movies similar to a given movie in latent space

In [31]:
```python
# Reduce to 20 latent features
svd_model = TruncatedSVD(n_components=20,random_state=42)
movie_factors = svd_model.fit_transform(movie_user_matrix)
```

In [32]:
```python
# Fit Nearest Neighbors on reduced feature space
nn_model_cf = NearestNeighbors(metric='cosine', algorithm='brute')
nn_model_cf.fit(movie_factors)
```

Out[32]:
```
▾ NearestNeighbors
NearestNeighbors(algorithm='brute', metric='cosine')
```

4.3 Indexing by movieId

In [50]:
```
# Build index <-> movieId mappings
movie_ids = movie_user_matrix.index.tolist()
movieId_to_index_cf = {mid: idx for idx, mid in enumerate(movie_ids)}
index_to_movieId_cf = {idx: mid for mid, idx in enumerate(movie_ids)}
```

4.4 Build Recommendation Function for collaborative filtering

In [51]:
```
# Function to recommend top 10 titles based on movie title
def recommend_collaborative_nn(title, top_n=10):
    if title not in title_to_movieId:
        return f"Movie '{title}' not found in."
    movie_id = title_to_movieId[title]
    if movie_id not in movieId_to_index_cf:
        return f"Movie ID for '{title}' not found in ratings matrix."
    idx=movieId_to_index_cf[movie_id]
    distances, indices_nn = nn_model_cf.kneighbors([movie_factors[idx]], n_neighbors=top_n+1)
    similar_ids = [movie_ids[i] for i in indices_nn[0][1:]]
    return [movieId_to_title[mid] for mid in similar_ids if mid in movieId_to_title]
```

4.5 Test a movie

In [52]:
```
recommend_collaborative_nn("Toy Story (1995)")
```

Out[52]:
```
['Willy Wonka & the Chocolate Factory (1971)',
 'Back to the Future (1985)',
 'Home Alone (1990)',
 'Star Wars: Episode IV - A New Hope (1977)',
 'Groundhog Day (1993)',
 'Independence Day (a.k.a. ID4) (1996)',
 'Jurassic Park (1993)',
 'Babe (1995)',
 'Princess Bride, The (1987)',
 'Star Wars: Episode VI - Return of the Jedi (1983)']
```

In [54]:
```
recommend_collaborative_nn("Moana (2016)")
```

Out[54]:
```
['The Secret Life of Pets (2016)',
 'The Boss Baby (2017)',
 'Heat, The (2013)',
 'Pirates of the Caribbean: Dead Men Tell No Tales (2017)',
 'Home (2015)',
 'The Conjuring 2 (2016)',
 'What If (2013)',
 'They Came Together (2014)',
 'Motherhood (2009)',
 "Mom's Night Out (2014)"]
```

Step5. Interactive Recommendation System

In [60]:
```
def interactive_recommender():
    print("Welcome to the Movie Recommender System!")
    print("----------------------------------------")
    print("You can get recommendations using:")
    print("- Content-based Filtering (Genres and Tags)")
    print("- Collaborative Filtering (Ratings by similar users)")
    print("Type 'exit' anytime to quit.\n")

    while True:
        user_input = input("Enter a movie title").strip()
        if user_input.lower().strip() == 'exit':
            print("Thank you for using the Recommender. Goodbye!")
            break
        if user_input not in title_to_movieId:
            print("Movie not found. Please try again.")
            continue

        print("\nChoose Recommendation Type:")
        print("1 = Content-Based")
        print("2 = Collaborative Filtering")
        model_choice = input("Enter 1 or 2 to select Recommendation Type: ").strip()
        if model_choice == '1':
            print("\nContent-Based Recommendations:")
            print("---------------------------------")
            recommendations = recommend_content_nn(user_input)
        elif model_choice == '2':
            print("\nCollaborative Filtering Recommendations:")
            print("------------------------------------------")
            recommendations = recommend_collaborative_nn(user_input)
        else:
```

```python
            print("Invalid choice. Returning to main menu. \\n")
            continue

        for i, movie in enumerate(recommendations,1):
            print(f"{i}. {movie}")
        print("\n-------------------------------------------\n")
# Run the CLI
interactive_recommender()
```

```
Welcome to the Movie Recommender System!
----------------------------------------
You can get recommendations using:
- Content-based Filtering (Genres and Tags)
- Collaborative Filtering (Ratings by similar users)
Type 'exit' anytime to quit.

Choose Recommendation Type:
1 = Content-Based
2 = Collaborative Filtering
Content-Based Recommendations:
---------------------------------
1. Arrival, The (1996)
2. Day the Earth Stood Still, The (1951)
3. Men in Black (a.k.a. MIB) (1997)
4. Signs (2002)
5. Astronaut's Wife, The (1999)
6. Alien (1979)
7. Thing from Another World, The (1951)
8. My Stepmother Is an Alien (1988)
9. E.T. the Extra-Terrestrial (1982)
10. Total Recall (1990)

-------------------------------------------

Choose Recommendation Type:
1 = Content-Based
2 = Collaborative Filtering
```

```
Collaborative Filtering Recommendations:
-------------------------------------------
1. Mission: Impossible (1996)
2. Rock, The (1996)
3. Twister (1996)
4. Jurassic Park (1993)
5. Terminator 2: Judgment Day (1991)
6. Twelve Monkeys (a.k.a. 12 Monkeys) (1995)
7. Star Wars: Episode VI - Return of the Jedi (1983)
8. Toy Story (1995)
9. Star Wars: Episode IV - A New Hope (1977)
10. Dragonheart (1996)

-------------------------------------------

Thank you for using the Recommender. Goodbye!
```

In [ ]: