CineSuggest: Movie Recommender

Abstract:

CineSuggest is an advanced movie recommendation system designed to provide personalized movie recommendations based on user preferences and behavior. Leveraging collaborative filtering techniques and machine learning algorithms, CineSuggest analyzes user ratings, movie features, and historical data to generate accurate and relevant movie recommendations tailored to each user's unique tastes and interests.

Overview:

CineSuggest is developed using Python programming language and incorporates various libraries and tools, including Tkinter for the graphical user interface (GUI), SQLite for database management, and Pandas for data manipulation. The system operates by first loading movie ratings and details from CSV files into an SQLite database. It then utilizes collaborative filtering algorithms to calculate the similarity between movies and generate personalized recommendations for users.

Features:

1. Personalized Recommendations: CineSuggest offers personalized movie recommendations based on individual user preferences, viewing history, and ratings.

2. Advanced Algorithms: The system employs sophisticated collaborative filtering algorithms, including Pearson correlation coefficient, to analyze user behavior and generate accurate predictions.

3. Real-time Updates: CineSuggest continuously updates its recommendation engine based on new user ratings and movie additions, ensuring that recommendations remain relevant and up-to-date.

4. User-friendly Interface: The GUI provides an intuitive and user-friendly interface for users to input movie preferences, view recommendations, and explore new films effortlessly.

5. Data Integrity: CineSuggest ensures data integrity and reliability by leveraging SQLite for database management and Pandas for data manipulation, guaranteeing smooth operation and accurate results.

Usage:

1. Launching the Application: Open CineSuggest and navigate through the intuitive interface.

2. Entering Movie Preferences: Input your movie preferences, including movie IDs or titles, into the designated fields.

3. Generating Recommendations: Click the "Recommend" button to generate personalized movie recommendations based on your input.

4. Exploring Recommendations: Explore the recommended movies displayed in the results section and discover new films tailored to your tastes.

5. Enjoying Your Movie Experience: Sit back, relax, and enjoy watching movies recommended specifically for you by CineSuggest.

Future Enhancements:

1. Enhanced Recommendation Algorithms: Continuously improve recommendation algorithms to incorporate additional factors such as genre preferences, release dates, and user demographics.

2. User Interaction Features: Implement interactive features such as user ratings, feedback mechanisms, and user profiles to further personalize recommendations.

3. Integration with Streaming Platforms: Integrate CineSuggest with popular streaming platforms to provide seamless access to recommended movies directly within streaming services.

ML model:

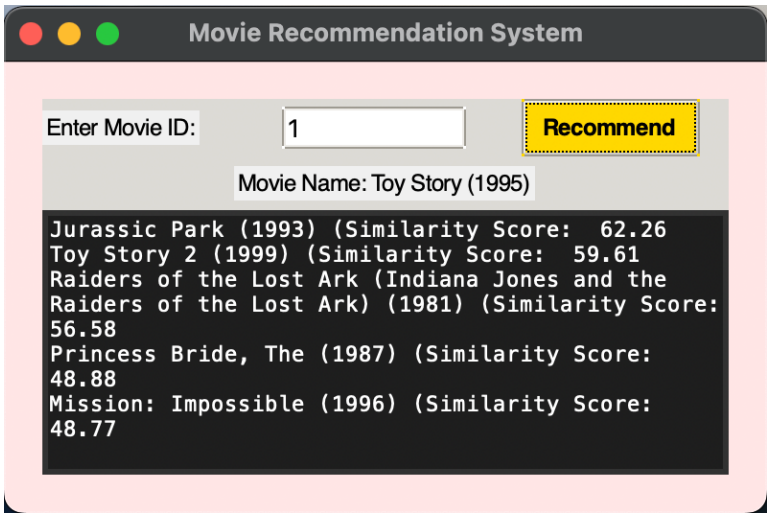
CineSuggest primarily utilizes memory-based collaborative filtering, specifically the Pearson correlation coefficient, to generate movie recommendations for users. This approach compares the ratings of movies by users and calculates how closely their ratings align, resulting in recommendations based on similarities between movies.

Python Code:

```
import tkinter as tk
from tkinter import ttk
import sqlite3
import pandas as pd

# Load data from CSV files
ratings_df = pd.read_csv('ratings.csv')
movies_df = pd.read_csv('movies.csv')

# Create SQLite connection and cursor
conn = sqlite3.connect(':memory:')  # Use in-memory database for demonstration, you can use a file-based da
cur = conn.cursor()

# Create tables and insert data
cur.execute('''CREATE TABLE ratings (
                user_id INTEGER,
                movie_id INTEGER,
                rating REAL,
                timestamp INTEGER
                )''')

ratings_df.to_sql('ratings', conn, if_exists='append', index=False)

cur.execute('''CREATE TABLE movies (
                movie_id INTEGER,
                title TEXT,
                genres TEXT
                )''')

movies_df.to_sql('movies', conn, if_exists='append', index=False)

# Commit changes
conn.commit()

def calculate_similarity(movie_id1, movie_id2):
    cur.execute('''SELECT r1.rating, r2.rating
                   FROM ratings r1
                   JOIN ratings r2 ON r1.user_id = r2.user_id
                   WHERE r1.movie_id = ? AND r2.movie_id = ?''', (movie_id1, movie_id2))
    rows = cur.fetchall()

    ratings1 = [row[0] for row in rows]
    ratings2 = [row[1] for row in rows]

    if not ratings1 or not ratings2:
        return 0, 0

    # Calculate Pearson correlation coefficient
    mean_rating1 = sum(ratings1) / len(ratings1)
    mean_rating2 = sum(ratings2) / len(ratings2)

    numerator = sum((x - mean_rating1) * (y - mean_rating2) for x, y in zip(ratings1, ratings2))
    denominator1 = sum((x - mean_rating1) ** 2 for x in ratings1)
    denominator2 = sum((y - mean_rating2) ** 2 for y in ratings2)

    if denominator1 == 0 or denominator2 == 0:
        return 0, 0

    correlation = numerator / ((denominator1 ** 0.5) * (denominator2 ** 0.5))

    return correlation, len(ratings1)  # Return correlation and number of ratings for normalization

def recommend_movies(movie_id, num_recommendations=5):
    cur.execute('''SELECT DISTINCT movie_id
                   FROM ratings
                   WHERE movie_id != ?''', (movie_id,))
    all_movies = cur.fetchall()

    similarity_scores = []
    for other_movie in all_movies:
        similarity = calculate_similarity(movie_id, other_movie[0])
        similarity_scores.append((other_movie[0], similarity[0] * similarity[1]))

    similarity_scores.sort(key=lambda x: x[1], reverse=True)
    recommendations = similarity_scores[:num_recommendations]

    recommended_movies = []
    for rec in recommendations:
        cur.execute('''SELECT title FROM movies WHERE movie_id = ?''', (rec[0],))
        movie_title = cur.fetchone()[0]
        recommended_movies.append((movie_title, rec[1]))

    return recommended_movies

def on_recommend():
    movie_id = int(movie_id_entry.get())
    recommendations = recommend_movies(movie_id)
    recommendation_text.delete(1.0, tk.END)  # Clear previous recommendations
    for movie, score in recommendations:
        recommendation_text.insert(tk.END, f"{movie} (Similarity Score: {score: .2f}\n")
```

```
# GUI setup
root = tk.Tk()
root.title("Movie Recommendation System")

# Customizing the style
style = ttk.Style()
style.theme_use("clam")
style.configure("TButton", foreground="black", background="#ffd700", font=("Helvetica", 12, "bold"))
style.configure("TLabel", foreground="black", background="#f0f0f0", font=("Helvetica", 12))
style.configure("TEntry", foreground="black", background="white", font=("Helvetica", 12))
style.configure("TText", foreground="black", background="white", font=("Helvetica", 12))

# Setting background color
root.configure(bg="#ffe6e6")

frame = ttk.Frame(root)
frame.grid(column=0, row=0, padx=20, pady=20)

movie_id_label = ttk.Label(frame, text="Enter Movie ID:")
movie_id_label.grid(column=0, row=0, sticky=tk.W)

movie_id_entry = ttk.Entry(frame, width=10)
movie_id_entry.grid(column=1, row=0)

recommend_button = ttk.Button(frame, text="Recommend", command=on_recommend)
recommend_button.grid(column=2, row=0)

recommendation_text = tk.Text(frame, wrap=tk.WORD, height=10, width=50)
recommendation_text.grid(column=0, row=1, columnspan=3, sticky=(tk.W, tk.E))

root.mainloop()

# Close connection
conn.close()
```

Snapshot of SQL Tables and GUI:



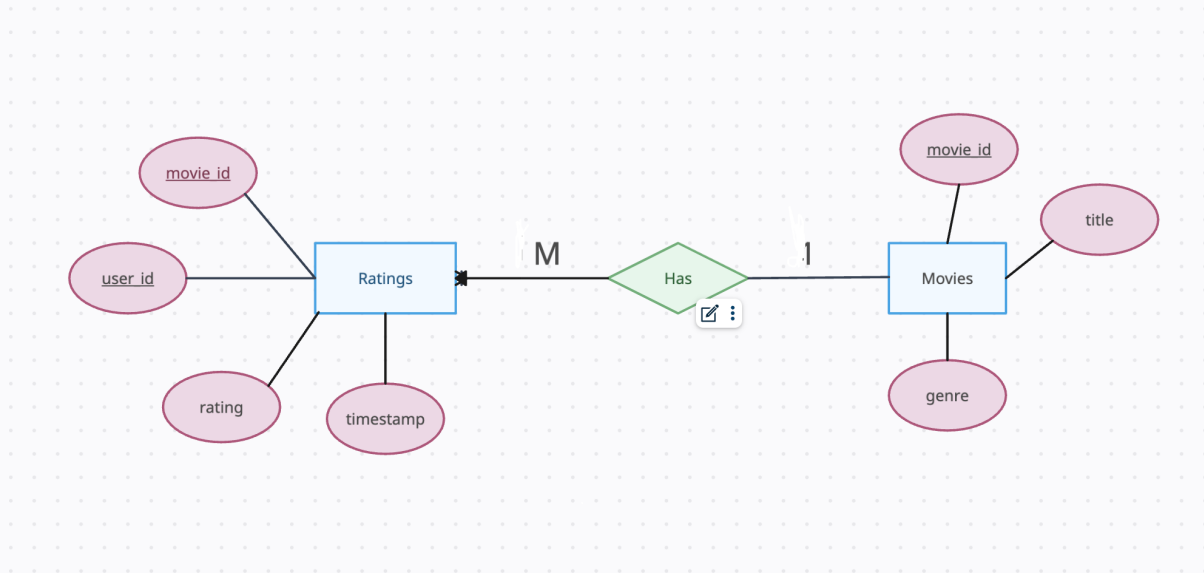
user_id	movie_id	rating	timestamp
1	16	4	1217897793
1	24	1.5	1217895807
1	32	4	1217896246
1	47	4	1217896556
1	50	4	1217896523
1	110	4	1217896150
1	150	3	1217895940
1	161	4	1217897864
1	165	3	1217897135

Ratings

movie_id	title	genres
1	title	Adventure|Animation|Children|
2	Jumanji (1995)	Adventure|Children|Fantasy
3	Grumpier Old Men (1995)	Comedy|Romance
4	Waiting to Exhale (1995)	Comedy|Drama|Romance
5	Father of the Bride Part II (1995)	Comedy
6	Heat (1995)	Action|Crime|Thriller
7	Sabrina (1995)	Comedy|Romance
8	Tom and Huck (1995)	Adventure|Children
9	Sudden Death (1995)	Action
10	GoldenEye (1995)	Action|Adventure|Thriller

Movies

ER Diagram:


movie id
user id
rating
timestamp
Ratings
M
Has
1
movie id
title
Movies
genre