AI - 4007
Applied Artificial Intelligence

Movie Recommendation System

22I-8803 Sahar Iqbal

Sabeeh Dayyan

Noor-ul-Iman

Table of Contents

1. Introduction

This report provides the reader with a detailed overview of the AI-driven Movie Recommendation System we have designed. This system will generate intelligent movie questions by integrating content-based filtering, sentiment analysis, mood-based clustering, as well as genetic algorithm optimization. The system will be accessed in multiple ways, namely a command-line interface (CLI), a Streamlit web interface, along with a React-based frontend, ensuring the customer gets a seamless user experience across all platforms.

2. Objectives

This system addresses the challenge faced by many users when choosing a movie after a long, tiring day. Instead of being able to relax immediately, people often spend a considerable amount of time searching for a film that aligns with their taste and mood. What some may perceive as a great movie, others might find mediocre.

The objectives of this system are to:

1. Develop a system that can recommend movies based on how similar the content is to another
2. Consider the user's mood as a factor for personalized filtering
3. Conduct a sentiment analysis on movie descriptions
4. Employ a Genetic Algorithm (GA) to optimize movie selection
5. Provide several ways of accessing the system for various kinds of users

3. Data Sources and Preparation

3.1 Datasets Used

The datasets used in this project are listed:

- tmdb_5000_movies.csv: Includes movie metadata
- tmdb_5000_credits.csv: Contains data about the cast and crew

3.2 Preprocessing Steps

1. The datasets are merged on the basis of the title column
2. overview and keywords are concatenated together for richer descriptions
3. Genre information will be transformed into one-hot encoded vectors
4. Numerical attributes, namely popularity, vote_average, and runtime are normalized to be used in GA

4. Backend Architecture (Python)

4.1 Tools and Libraries

1. pandas, numpy: For data manipulation and analysis
2. scikit-learn: Used for TF-IDF vectorization, cosine similarity computation, and KMeans clustering
3. TextBlob: Performs sentiment analysis.
4. random: Implements Genetic Algorithm operations.

4.3 Recommendation Engine

The recommendation engine starts with using TF-IDF to convert descriptions into vectors. Then, the cosine similarity will determine how related movies are. The indexing system is efficient and enables fast retrieval.

4.3 Sentiment Analysis

Every movie is assigned a sentiment according to the overview in the dataset. Sentiment is categorized by:

- Positive: Polarity > 0.2
- Neutral: -0.2 <= Polarity <= 0.2
- Negative: Polarity < 0.2

4.4 Genetic Algorithm Optimization

First, the genetic algorithm selects an optimal set of 5 movies. Then the fitness function will be weighted as:

- 40% normalized popularity
- 40% normalized vote average
- 20% normalized runtime

The genetic algorithm then incorporates elitism, crossover and mutation techniques.

5. Frontend (React)

The frontend has been developed in React. The user is prompted to input a movie title, and after processing the movie, the system will return recommended films. The frontend communicates with the backend using RESTful API. It incorporates technologies such as React.js, JSX and npm package manager.

6. Streamlit Interface

The Streamlit Interface will deliver a user-friendly interface for on-the-fly recommendations. It consists of two key sections: the top 5 recommendations based on similarity and sentiment, and the top 5 GA-optimized recommendations with supplementary data. It then displays sentiment classification for context.

7. Utility Modules (utils.py)

7.1 preprocess_genres()

- Converts genre strings into one-hot encoded format.
- Input for clustering algorithms like KMeans.

7.2 mood_genre_pref Mapping

- Maps moods to suitable genres:
    - Happy ↠ Comedy, Family, Adventure
    - Sad ↠ Drama, Romance
    - Excited ↠ Action, Thriller
    - Scared ↠ Horror, Mystery
    - Thoughtful ↠ Documentary, Drama
    - Fantasy ↠ Fantasy, Science Fiction

7.3 get_cluster_for_mood()

- Associates user mood with genre clusters from KMeans.
- Filters recommended movies based on predicted clusters.

8. CLI Application

The CLI will accept the movie title and the mood as parameters. It will output the top 5 content-based movie suggestions with sentiment labels and the top 5 GA-optimized suggestions with sentiment, popularity, rating, and runtime. The CLI can be used for rapid testing and debugging.

9. How to Run the System

9.1 Backend (Python)

Install required libraries:

```
pip install -r requirements.txt
```

Run Streamlit app:

```
streamlit run app.py
```

9.2 Frontend (React)

Navigate to project folder:

```
cd frontend
```

Install dependencies:

```
npm install
```

Start development server:

```
npm start
```

10. Evaluation and Testing

This program has been validated manually for:

- Recommendation accuracy using cosine similarity.
- Mood filtering efficacy via clustering validation.
- Sentiment label correctness.
- GA performance improvements through fitness score monitoring.

The fitness evolution is printed during GA execution for tracking.

11. Summary and Impact

This hybrid recommendation engine integrates:

- Content-based filtering for related movie discovery.
- Sentiment analysis for tonal alignment.
- Mood-driven clustering for contextual relevance.
- Genetic algorithm for multi-objective optimization.

The system enhances user experience by offering refined, personalized movie suggestions that go beyond basic recommendation methods.

12. Future Enhancements

- Add collaborative filtering to factor in peer preferences.
- Implement advanced NLP models like BERT for better sentiment and genre detection.
- Improve UI with charts, feedback forms, and mood selectors.
- Introduce user login, preferences, and viewing history.
- Use A/B testing to refine and evaluate recommendation strategies.