THE UNIVERSITY OF NEW SOUTH WALES

The School of Computer Science and Engineering

COMP9727: PROJECT DESIGN

Shoudi Huang

z5605172@ad.unsw.edu.au

Date of submission: 6 July 2025

1. Scope

1.1. Domain

This project focuses on a **movie recommender system** designed to help individual viewers discover films aligned with their personal tastes. By leveraging users' past movie preferences and film attributes such as genres, release years, and other metadata, the system will generate a short list of highly relevant movie suggestions on demand.

1.2. Intended Users

The recommender system is intended for ***individual movie viewers*** looking for personalized movie suggestions. In a real scenario, this could be any movie enthusiast or subscriber of a movie service who interacts with a recommendation tool. For this project's scope, the "users" will be simulated by the student and the user profile data. We assume each user seeks movie recommendations tailored to their taste. The system is designed as a single-user application, meaning it serves one user at a time in an isolated session. The focus is on personal usage, a user will query the system for movie recommendations and provide feedback (ratings) on those recommendations.

1.3. User Interaction and Interface

Users will interact with the movie recommender through a **web-based interface**. A simple responsive page will display a Top-N list of movie suggestions (e.g. 10 recommendations per view) to avoid overwhelming the user. Each recommendation entry will include the film's title, release year, genre(s) and possibly a brief description. Controls will allow users to:

- Request fresh recommendations.
- Provide feedback by clicking a star-rating widget beneath each movie.

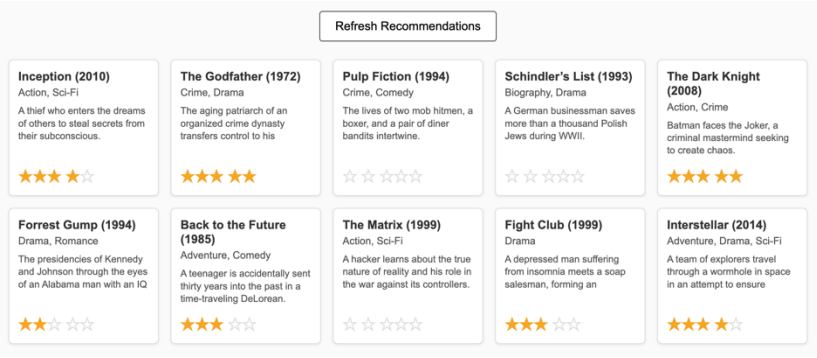

Figure 1.1: Sample web interface mockup

1.3.1. Simulating User Interactions

Although this interface will not be implemented in the project, the user interactions will be simulated though the user study. Participants will engage in a **think-aloud** protocol in which they review a table of N movies (title, release year, genres), select the most appealing movies while verbalizing their reasoning, and assign it a rating. After each selection, the facilitator updates a mock user profile with the chosen film and rating and presents a new tailored table of N recommendations. This cycle repeats for multiple rounds, allowing us to observe how recommendations adapt to incremental feedback.

1.4. Dynamic Model Updates

Upon each user rating, the system should immediately incorporate the new value into the user's profile and, at the next recommendation request, recomputes the Top-N list using the updated profile. This on-demand recalculation, whether by adjusting the user's preference vector in a neighborhood-based model or re-scoring items via content features, ensures that each successive recommendation set reflects the most recent feedback. Although the update mechanism is modest in scope (e.g. incremental neighbor reweighting), it demonstrates real-time personalization.

1.5. Cold-Start Strategy

1.5.1. New User Cold-Start

For new users with no rating history, the system can initially present a Top-N list of globally popular movies until sufficient feedback is gathered. Alternatively, it may prompt users to rate a small set of sample films. This dual strategy ensures early recommendations remain relevant while bootstrapping a personalized user profile.

1.5.2. New Item Cold-Start

For new items with no rating history, the system can adopt a **hybrid strategy**: It will apply **collaborative filtering** to "warm" movies that have accrued sufficient ratings. And fall back on **content-based filtering**, using metadata to recommend "cold" titles to users whose profiles indicate affinity for similar attributes. This approach ensures both established and novel movies are suggested with appropriate relevance.

1.6. Potential Business Model

- **Subscription:** Integrate the recommender into a paid service, either as a monthly-subscription app or a premium feature of an existing movie-club platform, to enhance engagement and retention through personalized suggestions.
- **Affiliate Marketing:** Provide referral links (e.g., “Watch on Netflix” or “Rent on Amazon”) alongside recommendations. Revenue accrues via commissions when users follow links to purchase or rent content.
- **Advertising:** In a free version, feature sponsored titles or banner ads within the recommendation interface.

2. Dataset

The **MovieLens32M** dataset (ml-32m) is selected for this movie recommender system, which containing 32,000,204 ratings and 2,000,072 tag applications across 87,585 movies, provided by 200,948 users from 1995 to 2023. These data were collected on the MovieLens platform and have been anonymized and pre-processed. Users included in the dataset each rated at least 20 movies, ensuring a substantial activity profile for each user. The dataset is organized into four interrelated CSV files:

userId	movieId	# rating	# timestamp
1	17	4.0	944249077
1	25	1.0	944250228
1	29	2.0	943230976
1	30	5.0	944249077
1	32	5.0	943228858

Figure 2.1: ratings.csv

- **ratings.csv**: Contains one entry per rating with fields *userId, movieId, rating, timestamp*. Ratings are explicit feedback on a 0.5–5.0 star scale (half-star increments). This file provides ~32 million user-movie ratings (the core interactions for collaborative filtering). The *timestamp* (in Unix seconds) indicates when each rating was made, enabling temporal analysis or time-based splits if needed.

movieId	title	genres
1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
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

Figure 2.2: movies.csv

- **movies.csv**: Contains one row per movie (approximately 87k movies) with *movieId, title, genres*. The *title* includes the movie name and year of release, and *genres* is a pipe-separated list of zero or more genres (chosen from 19 predefined genres such as Drama, Comedy, Documentary). This file provides essential item metadata. For example, it allows grouping movies by genre and including content features in the model.

userId	movieId	tag	timestamp
22	26479	Kevin Kline	1583038886
22	79592	misogyny	1581476297
22	247150	acrophobia	1622483469
34	2174	music	1249808064
34	2174	weird	1249808102

Figure 2.3: tags.csv

- **tags.csv**: Contains ~2 million records of user-provided annotations, with fields *userId, movieId, tag, timestamp*. Each entry is a free-text *tag* (a keyword or short phrase) that a user assigned to a movie, with a timestamp. Tags serve as rich descriptive metadata, reflecting nuanced attributes or themes. They can be used to augment content-based representations of movies or to infer user preferences beyond ratings.

movieId	imdbId	tmdbId
1	0114709	862
2	0113497	8844
3	0113228	15602
4	0114885	31357
5	0113041	11862

Figure 2.4: links.csv

- **links.csv**: Contains *movieId, imdbId, tmdbId* for each movie, linking MovieLens entries to the corresponding identifiers on IMDb and The Movie Database (TMDb). While not directly used for modeling, this file enables fetching additional information (plots, cast, etc.) from third-party sources if needed to enrich the recommendation system.

2.1. Suitability (Quality & Quantity)

This dataset's scale and richness make it well-suited for developing a robust recommendation model. The large number of ratings (~32m) provides high quantity, ensuring that the model can learn fine-grained preference patterns. Each of the ~200k users has a relatively dense profile, which helps in capturing individual tastes with less cold-start sparsity.

In terms of data quality, every movie in the dataset has at least one rating or tag, ensuring both comprehensive collaborative coverage and high data quality. Ratings on a 0.5–5.0 scale from 1995–2023 capture genuine, temporally diverse user preferences. With ~87 k films across all major genres and over 100 k unique user-generated tags, the dataset's rich

metadata and broad item diversity support both collaborative and hybrid recommendation approaches.

2.2. Core Fields for Modelling

The core model will leverage the *userId, movieId, rating* columns from **ratings.csv** as its primary collaborative signal, treating explicit ratings as targets for prediction or as implicit positives in latent-factor methods. We will augment this with *genre* metadata from **movies.csv**, encoding each film's genre vector to compute content similarities or enforce genre diversity in a hybrid framework. User *tags* from **tags.csv** will further enrich item and user feature representations by capturing thematic or mood-based preferences. Finally, *timestamps* will support temporal train–test splits and, if desired, time-aware modeling. Collectively, these fields enable a hybrid recommender that combines collaborative and content-based signals for robust personalization.

2.3. Dataset Limitations

Despite its strengths, the MovieLens32M dataset has some limitations, especially as a competition-style dataset, which must be acknowledged to set proper expectations:

2.3.1. Subset of Users/Interactions

The dataset includes only users with ≥20 ratings, thereby excluding casual or infrequent users—this increases data density but biases the model toward active profiles. It also covers exclusively MovieLens interactions up to 2023, omitting external or more recent data and thus constraining the system's representativeness.

2.3.2. Pre-processed and Cleaner than Real-World Data

The dataset is pre-processed: anonymized IDs, complete interaction records, and standardized metadata. Which eliminates missing values, duplicates, and operational noise (e.g. implicit interactions or input errors). While this clarity benefits experimental control, it also omits demographic or contextual attributes and real-world irregularities, potentially limiting model generalization to raw, noisy production data.

2.3.3. Static Snapshot & Rating-Focused Task

MovieLens32M dataset represents a static snapshot of explicit user-movie ratings, structured for offline evaluation on a predetermined train/test split. This narrow focus may encourage overfitting to the dataset's idiosyncratic distribution and rating patterns. It also excludes implicit feedback signals (e.g. clicks, view durations), lacks temporal dynamics beyond simple timestamps. Consequently, models optimized on this data may

achieve strong offline accuracy yet fail to generalize when deployed in live, production environments characterized by continuous data streams and real-world noise.

3. Method(s)

To tackle the movie recommendation problem and the MovieLens32M dataset, a combination of collaborative filtering and content-based techniques, along with a hybrid strategy that combines their strengths was proposed. Each method is chosen for its suitability to our user base and data, as well as its feasibility to implement and evaluate within the project timeframe.

3.1. Collaborative Filtering Approaches

3.1.1. Memory-based Collaborative Filtering

We will implement a **neighbourhood-based collaborative filtering** method, specifically an ***item-item*** collaborative filtering algorithm. In item-based CF, we compute similarities between items based on user ratings and recommend movies like those a user has highly rated. This approach is useful for its scalability and speed, since the expensive step of finding similar item pairs is done offline, and online recommendation simply looks up precomputed similar items, making it efficient even for very large datasets. The MovieLens32M data is substantial, so the ability to precompute an item similarity matrix offline is advantageous for performance.

Item-based CF also can perform well even for users with few ratings by leveraging item similarities, which addressing the cold-start scenario to some extent. We will likely use **cosine similarity** or **Pearson correlation** on item rating vectors to identify nearest neighbours for each movie, and then predict user ratings based on a weighted average of a user's ratings on similar items.

For comparison, we may also consider a ***user-user*** collaborative filtering algorithm, but user-based CF is less scalable on large data, so our focus will be on the item-based technique which offers better computational performance for our dataset size.

3.1.2. Model-based Collaborative Filtering

To further improve accuracy, we propose using a **matrix factorization** method as our model-based CF approach. Matrix factorization (e.g. **Singular Value Decomposition**) transforms the user-item rating matrix into latent feature factors, which can generalize and predict missing ratings effectively. For our project, a factorization-based recommender (such as using the **SVD** or **ALS** algorithm) is suitable because the MovieLens32M data is sparse and high-dimensional. A latent factor model can capture

underlying user preferences and item attributes in a compact way. By using matrix factorization, we aim to maximize the predictive performance of the recommender system, since these models can better handle data sparsity and usually achieve lower error rates than memory-based CF. We will compare the results of the matrix factorization model with the item-based CF to evaluate how much accuracy gain is obtained by the more sophisticated model-based approach.

3.2. Content-Based Filtering Approach

In parallel with collaborative methods, we will implement a content-based recommender that exploits movie meta-data (e.g. *genres, tags*) from the MovieLens32M dataset. We will derive **item profiles** from the metadata, encoding each movie's *genre* indicators and weighted term-frequency vectors of user-generated *tags*, and construct **user profiles** by aggregating the feature vectors of movies each individual has highly rated. Recommendations are generated by computing similarity (e.g. **cosine similarity**) between a user's profile vector and candidate movie vectors, thus directly matching items to a user's demonstrated content preferences without relying on other users' data.

This content-based approach leverages the dataset's rich *genre* and *tag* information to capture thematic affinities and inherently mitigates the item cold-start problem, since new or sparsely rated films can be recommended based on their descriptive attributes. It can also serve as a complementary component in our **hybrid system**, improving coverage and personalization alongside collaborative filtering methods.

3.3. Hybrid Recommender Strategy

Considering the strengths and weaknesses of collaborative and content-based systems, we propose a **hybrid recommender system** that combines both methods to achieve better overall performance. In our design, the hybrid will ensure that recommendations remain accurate and personalized even in situations where one technique alone would fail (such as new users or new items). Specifically, we plan to explore either a **weighted hybrid** or a **switching hybrid** strategy.

3.3.1. Weighted hybrid

For weighted hybrid, the system will combine scores from collaborative filtering (e.g. matrix factorization or item-based CF) and content-based filtering via a tunable weighted average. We will adjust weights, which favoring CF for established users and content for cold-start scenarios, based on validation performance. This hybrid strategy leverages

CF's predictive accuracy while maintaining content-driven recommendations when ratings are sparse.

3.3.2. Switching hybrid

For switching hybrid, the system dynamically selects between content-based and collaborative filtering based on data availability. It defaults to content-based recommendations for new users or obscure movies lacking ratings, and to collaborative filtering for long-standing users and popular items with rich rating histories. This rule-based approach ensures that each algorithm operates in scenarios where it is most effective and directly addressing cold-start challenges.

3.4. Other Considerations

Alternative paradigms also been reviewed, such as **knowledge-based systems**, which rely on explicit domain rules, and **context-aware** or **sequence-aware** methods, which exploit situational or temporal signals. However, given the rich ratings and metadata in MovieLens32M and the absence of detailed contextual attributes, we concentrate on collaborative filtering, content-based filtering, and their hybridization. Timestamps remain available for future time-aware extensions.

4. Evaluation

4.1. Metrics for Recommendation Model

To assess the recommendation model on historical data, we will use several **top-N metrics** (computed per user and averaged across users):

- **Precision@N**: measures the proportion of the top-N recommended items that are actually relevant
- **Recall@N**: measures the proportion of all relevant items that are successfully included in the top-N recommendations
- **F1@N**: the harmonic mean of Precision@N and Recall@N, which captures their balance.

We will also use ranking metrics like **Normalized Discounted Cumulative Gain (NDCG)**, which accounts for the position of relevant items in the ranked list (giving higher weight to relevant items appearing earlier). Additionally, a **mean average precision (MAP)** may be reported to summarize precision across recall levels on a per-user basis. These metrics reflect how well the model predicts and ranks relevant items for each user.

4.2. Metrics for Recommender System

In the simulated think-aloud study, we can approximate real-world engagement of the recommender system by measuring the following metrics:

- **Selection Rate**: measure the proportion of recommended items that participants choose to rate or "click," analogous to click-through rate.
- **Positive Feedback Rate**: the fraction of selected recommendations that receive favourable ratings (e.g., 4–5 stars), serving as a proxy for user satisfaction.
- **Decision Latency**: the time interval from recommendation display to participant selection, which indicates perceived relevance and ease of choice

Together, these simulated metrics focus on user's feedback, selection behaviours, rating outcomes, and decision times, providing insight into the compelling and relevance of the recommender system without a live deployment.

4.3. Metric Trade-offs and Model Selection

Different metrics emphasize distinct aspects of recommendation quality. Precision@N rewards accuracy in the top-N list but may overlook relevant items outside those N, whereas Recall@N ensures broader coverage at the cost of possibly diluting the top slots. F1@N balances precision and recall but does not account for the positional value of an item. Which is an insight captured by NDCG@N, weights higher-rank hits more heavily. MAP@N further summarizes rank performance across all recall levels.

In selecting the "best" model, we will designate a NDCG@N as primary metric, since our interface shows N recommendations and values rank position, and require competing models to meet minimum thresholds on Precision@N and Recall@N. Models will be compared on the primary metric first, if two models achieve statistically indistinguishable NDCG@N, we will choose the one with higher F1@N to ensure balanced relevance and coverage.

4.4. Computational and Real-Time Considerations

In order to validate real-time responsiveness and update costs of the recommender system, we can create a **local performance benchmark** that mimics production workloads.

4.4.1. Microbenchmark Top-N Retrieval

- Implement the final recommendation pipeline (e.g. loading precomputed item similarities or latent factors, merging scores, selecting top-N).
- Then measure the average and worst-case latency for generating recommendations on a representative sample of users. Ensure these times remain under the target (e.g. 200 ms).

4.4.2. Simulate Incremental Updates

- Script a small sequence of "new" ratings and measure the time required to incorporate them into the data structures (e.g. updating user factors, similarity caches).
- Run this simulation periodically (e.g. 100, 1000 or 10000 updates) to estimate the cost of daily or hourly refreshes.

4.5. User Study Design

4.5.1. Think-aloud User Study

A **think-aloud user study** can be conducted using a mockup of the recommender interface. Participants will be shown a table of N movies (title, year, genres), then verbalize their reasoning as they select and rate the most appealing movies. After each selection, the facilitator will update the **mock user profile** with the new rating and present an updated table of N personalized recommendations. This iterative cycle of review, selection, rating, and profile update, will repeat for several rounds. We will capture **verbal transcripts**, **selection rate**, **positive feedback rate** and **decision latencies**. Then conclude with a brief questionnaire. Although no live interface is implemented, this protocol effectively simulates real user interactions to evaluate the system's adaptability and user experience.

4.5.2. Pairwise List Comparison

Besides, we can generate recommendation lists for the same user profile using different algorithms (e.g. item-based CF vs hybrid). Present these lists side-by-side to participants and ask them to indicate which list they would prefer and why. By repeating this across multiple profiles and algorithms, we collect preference data (percentage of times one algorithm's list is chosen) and qualitative justifications, revealing which approach users find more relevant and intuitive.