COMP9727 Recommender Systems

Project Pitch and Design

Term 2, 2025

Student Name: Pan Lu

Student ID: z5530515

Table of Contents

1. Scope

1.1. Recommendation Domain & Target Users

I plan to design and implement a universal movie recommendation system for a wide range of movie enthusiasts. This system is not confined to a single type of film, but covers a wide range of categories from mainstream blockbusters to niche independent films. Its goal is to help users discover the works that best match their personal interests among the vast amount of film information.

I hope the system has good personalization capabilities and can make differentiated recommendations based on the interests of different users. For this purpose, I plan to integrate the following types of information as the basis for modeling:

- The user's viewing history (movies they have watched and rated)
- Users' search behaviors (search keywords, click behaviors)
- The user's browsing path (details of the movies visited, duration of stay)
- The user's preference tags or topic preferences (such as frequently watching suspense films or youth films)
- The user's registration information (such as age group, gender, region, if any
- Hot trends or time-sensitive content (such as recently released popular films)

I believe that relying solely on ratings or explicit feedback is not sufficient to fully capture users' interests. Therefore, I will also utilize users' implicit behaviors on the platform (such as clicking, browsing, and searching) as important signals to learn users' preferences from multiple dimensions.

In addition, considering that the platform will constantly attract new users, I also pay special attention to the issue of cold start for new users. When users register for the first time, I plan to design a guiding process to proactively show them a group of movies that are "well-received", "recently popular" or "highly representative in multiple categories", allowing them to make choices or rate them. This not only enhances the initial experience but also enables the model to quickly collect initial preference data, thus entering the personalized recommendation stage as soon as possible.

In conclusion, my recommendation system is targeted at all types of movie enthusiasts. It provides personalized recommendations through multi-dimensional modeling and also takes into account the cold start guidance for new users. It has scalability and practical application potential.

1.2. Recommendation Format & UI Platform

When designing the presentation form of the recommendation system, the first thing I considered was in what scenarios users would use this platform. Because movie recommendations are usually a "decision-making" process, I think users are more suitable to make choices on large-screen devices, such as viewing the recommendation list, browsing details and making choices through web pages or tablets. So I plan to set the main interface of the platform as the web version and make it compatible with the simple browsing function of mobile devices.

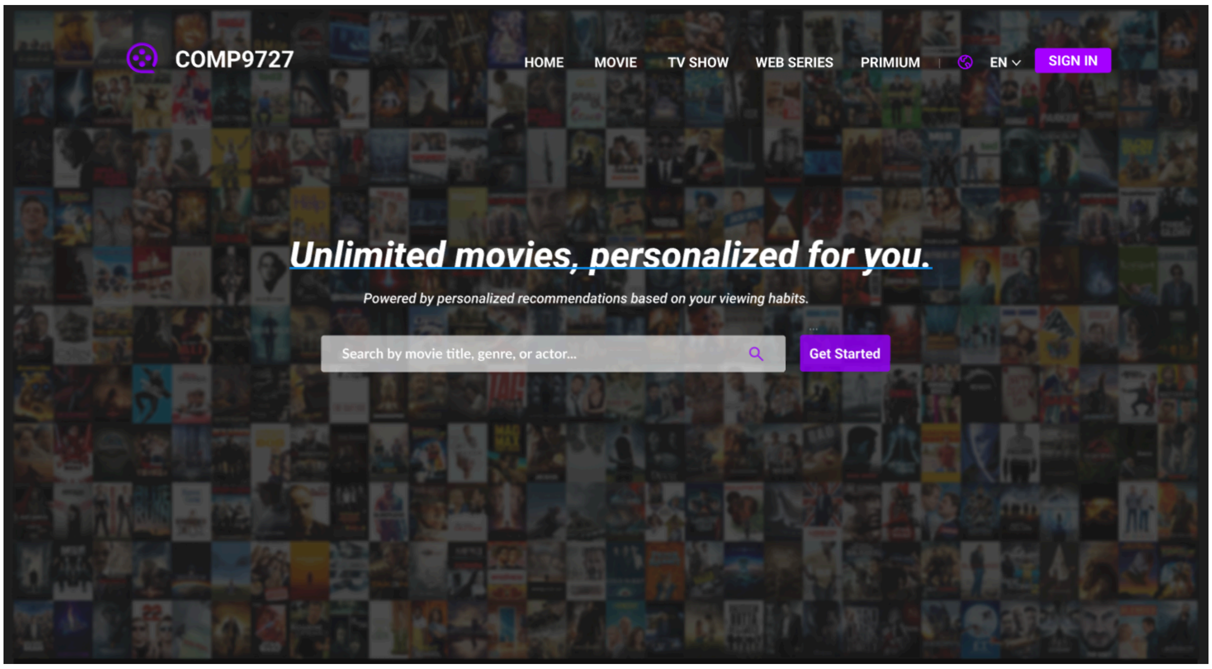
Regarding the recommended number of one-time displays, I believe that too few displays will leave users with no choice space, while too many displays will cause cognitive burden, especially on interfaces with high information density. Combining the practices of platforms like Netflix and Douban, I think it is a relatively appropriate range to recommend and display 5 to 10 movies each time. It can be grouped and displayed according to modules such as "Guess What You Like", "High Score Recommendation", and "Recent Hot Topics". Each group can slide horizontally, which not only saves space but also makes it convenient for users to quickly preview the content.

From the perspective of user interaction, I hope that the recommendation is not a "closed" list, but a design that can guide users to make feedback. For example, display the "Want to Watch/Not Interested" button when the mouse hovers over it, or record these behaviors after the user watches or rates. In this way, the system can continuously learn users' interests instead of making static recommendations all at once.

1.3. User Interface & Interaction Mockups

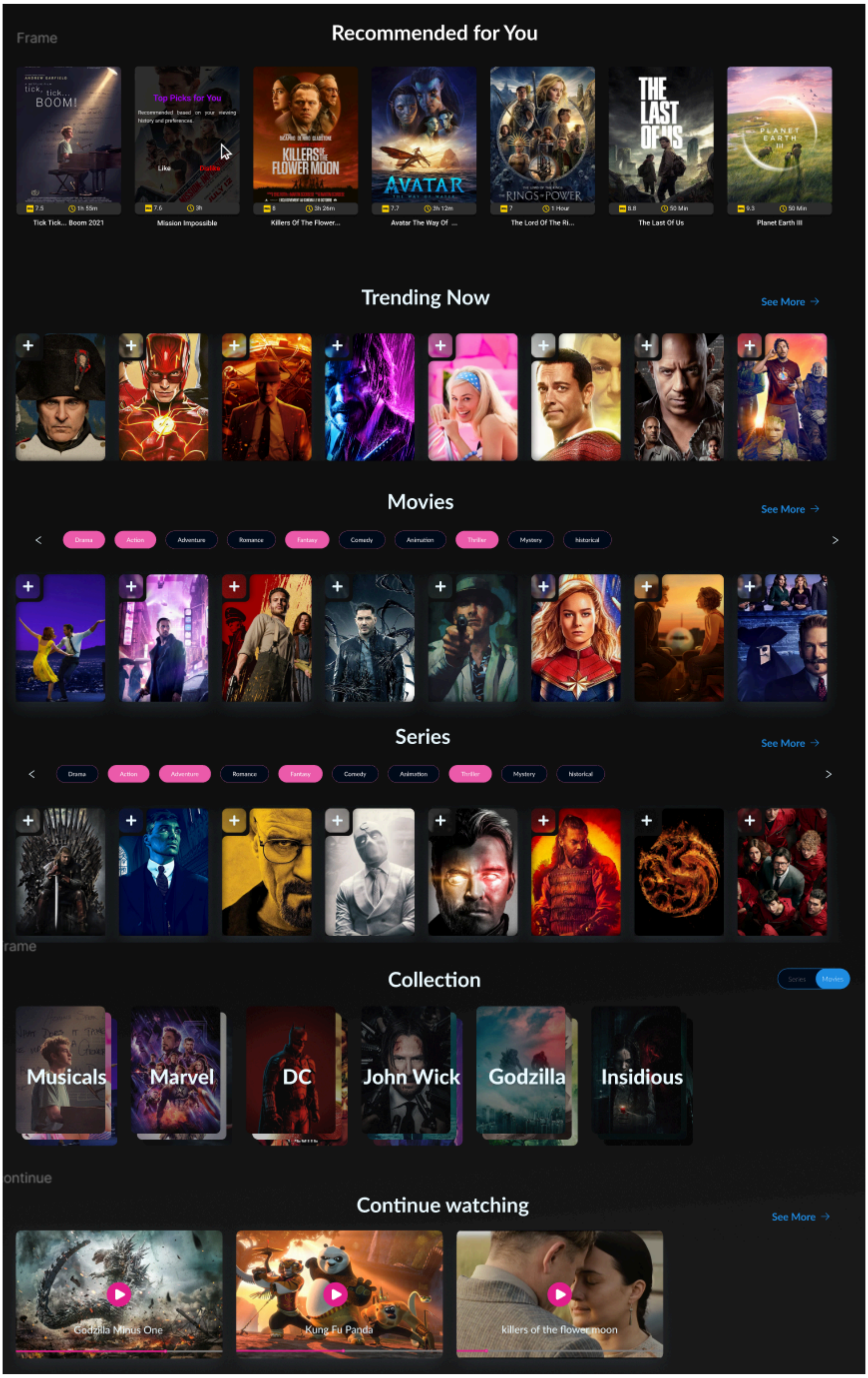
To visually demonstrate the interface of the recommendation system I envisioned and the way it interacts with users, I used **Figma** to design a complete static prototype. During the production process, I made use of some open-source Figma material components (such as movie cover templates, rating ICONS, character avatars, etc.) to enhance efficiency and ensure a consistent design style. The entire interface style draws on Netflix and IMDb's visual layout, while also incorporating my understanding of user behavior paths for simplification and adaptation.

First of all, in the home page section, I used a background image composed of movie posters spliced together as the visual base layer, and placed a welcome phrase "Unlimited movies, personalized for you." in the central area to highlight the personalized recommendation feature of the system. Below it is a search input box, which prompts users to search by keywords such as movie titles, genres or actors. This design is intended to support active exploratory interaction.

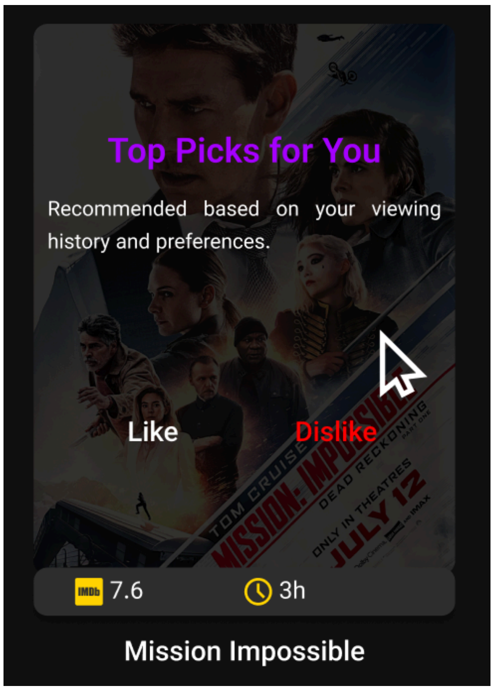

COMP9727
HOME
MOVIE
TV SHOW
WEB SERIES
PRIMIUM
EN
SIGN IN
Unlimited movies, personalized for you.
Powered by personalized recommendations based on your viewing habits.
Search by movie title, genre, or actor...
Get Started

When user scrolling down, users can see multiple recommendation modules. I have partitioned and displayed the content by recommendation logic and type, including:

- **Recommended for You:** Personalized recommendation results generated based on users' historical behaviors;
- **Trending Now:** Display the currently most popular, most-watched or highly-rated films;
- **Movies/Series:** Classified by content type, it supports further filtering (users can click on tags such as "Action", "Drama", etc. to filter content);
- **Collection:** Presenting specific themes or series (such as Marvel, DC, John Wick, etc.);
- **Continue Watching:** Records the videos that the user has not finished watching, making it convenient to continue watching.


Frame
Recommended for You
Top Picks for You
Recommended based on your viewing history and preferences.
Like
Dislike
7.5 1h 55m
Tick Tick... Boom 2021
7.6 3h
Mission Impossible
8 3h 26m
Killers Of The Flower...
7.7 3h 12m
Avatar The Way Of ...
7 1 Hour
The Lord Of The Ri...
8.8 50 Min
The Last Of Us
9.3 50 Min
Planet Earth III
Trending Now
See More
Movies
See More
Drama
Action
Adventure
Romance
Fantasy
Comedy
Animation
Thriller
Mystery
historical
Series
See More
Drama
Action
Adventure
Romance
Fantasy
Comedy
Animation
Thriller
Mystery
historical
rame
Collection
Series
Movies
Musicals
Marvel
DC
John Wick
Godzilla
Insidious
ontinue
Continue watching
See More
Godzilla Minus One
Kung Fu Panda
killers of the flower moon

In the design of the movie cards, I combined information such as ratings, duration, and types for display. When the user hovers the mouse over the card, the "Like" and "Dislike" operation buttons will be displayed, guiding the user to clearly express their preferences. Based on these feedbacks, I hope the system can continuously learn and adjust the recommendation strategy, thereby achieving truly dynamic and personalized recommendations.



To enhance the interactive experience, I have also incorporated some lightweight design details, such as the "+" button for collection, tag filtering, multi-module horizontal sliding browsing, and the "See More" control, etc. These designs aim to enable users to complete the exploration, screening and decision-making processes with a lower cognitive burden, while providing the system with more interaction data to assist in recommendation optimization.

Although this is a static prototype, I believe it has fully expressed my understanding and conception of the user interface of the recommendation system, and can provide a clear reference for subsequent system development and user experiments.

1.4. User Feedback Handling

When designing a recommendation system, I have always believed that user feedback is the core element for achieving personalization. Therefore, I consciously introduced multiple feedback mechanisms in the interface and interaction logic to collect users' explicit and implicit preference information, thereby continuously optimizing the recommendation effect.

Firstly, in terms of explicit feedback, I designed two buttons, "Like" and "Dislike", which will be displayed when the user hovers the mouse over the movie card. Users can actively click to indicate their preference or rejection for a certain movie. I chose this approach because it is lightweight enough. Users can complete the operation without jumping to the page, and the feedback results can be directly used to update the user profile or trigger model adjustments.

Secondly, I also attach great importance to the collection of implicit feedback. For instance, when a user clicks on a movie card, browses the detail page, plays a movie, collects a certain work, or even engages in search behavior and page dwell time, all these can be regarded as interest signals. I believe that incorporating these implicit behaviors into the modeling can effectively enhance the accuracy of recommendations, especially in real-world scenarios where users are reluctant to frequently click "like/Dislike".

To enhance the adaptability of the model, I plan to design the feedback mechanism to be continuously available and collected in real time. The system can record user behavior data in the background and update the recommendation model regularly, or trigger a local update after a user completes a feedback, maintaining the dynamics of the recommendation.

In addition, I am also considering collecting subjective feedback at the system level through questionnaires or simple rating mechanisms (such as "Does the recommendation match your interests?" or "Are you willing to continue using this system?") in future user experiments, so as to evaluate the effectiveness of the recommendation system at the overall experience level.

Overall, I hope that by combining the active guidance of explicit feedback with the natural recording of implicit feedback, the recommendation system can not only learn "what you like", but also gradually understand "why you like it", thereby enhancing the quality and credibility of personalized recommendations.

1.5. Dynamic Updates & Cold Start Strategy

In my opinion, an excellent recommendation system should not only be able to make reasonable recommendations in the initial stage, but more importantly, it should have the ability to learn dynamically and update continuously. Therefore, in the design, I clearly considered the update mechanism of the model and how to handle the cold start issue between users and items.

Firstly, in terms of the update strategy for the recommendation system, I plan to adopt a combined approach of "lightweight incremental update + periodic retraining". For explicit feedback (such as Like/Dislike), collection behavior, viewing records, etc., I hope the system can record them in real time in the background and update user profiles or rating matrices in batch processing or streaming mode, so as to quickly adjust personalized recommendation results. For more complex model parameters or recommendation algorithms, I tend to adopt a small-scale retraining once a week to balance the system response speed and computational cost.

Secondly, regarding the issue of user cold start, I adopted a guided interest collection mechanism: when new users register or log in for the first time, the system will proactively recommend a group of popular films covering various types and with high ratings, and guide users to select the content they are interested in (such as clicking to collect, marking "want to watch" or "Have watched"). In this way, I hope to quickly construct the initial user interest vector in the absence of historical behavior data, laying the foundation for subsequent recommendations.

For cold-start films (i.e., newly launched films on the platform that have no user interaction yet), I consider making recommendations based on content features, such as the film type, keywords, director, actor and other metadata. Based on this, I can embed the content into the existing collaborative filtering matrix or combine it with the hybrid model for initial push, so that the new film can enter the user's view as soon as possible.

1.6. Business Model & Revenue Possibilities

The recommendation system I envision can serve as the core recommendation engine of an independent content platform, or it can be embedded in existing video platforms, streaming media applications or film and television rating websites. By enhancing user retention rates and viewing experiences, it inherently possesses indirect commercial value.

Specifically, I think this system may bring about the following several potential profit opportunities:

Subscription-based platform recommendation engine (SaaS model)

I can provide this recommendation system as an embedded component for small and medium-sized streaming media platforms to use. The platform pays monthly based on the number of subscribers or the number of calls, and the system continuously provides personalized recommendation capabilities.

Bidding for advertising recommendations and film exposure positions

If the platform supports third-party advertising placement or the promotion of new movies, I can introduce advertising recommendation logic based on content relevance into the system.

By reasonably inserting "promotional videos" into the recommendation list and ensuring the quality of the recommendations, the system can support advertisers to pay by click or exposure.

Enhance the indirect value of user conversion

Even if the system itself does not directly generate profits, I believe it can effectively enhance the platform's key indicators such as "completion rate", "second view rate" and "user activity", thereby bringing more retention and more paid conversions to the platform. In this case, the recommendation system plays an indirect role in supporting product growth.

Preference insight analysis service based on user data

If the system integrates the user profiling analysis function in the future, it can also provide the platform with insights such as user interest clustering and behavior prediction, offering data support to the marketing operation team.

2. Datasets

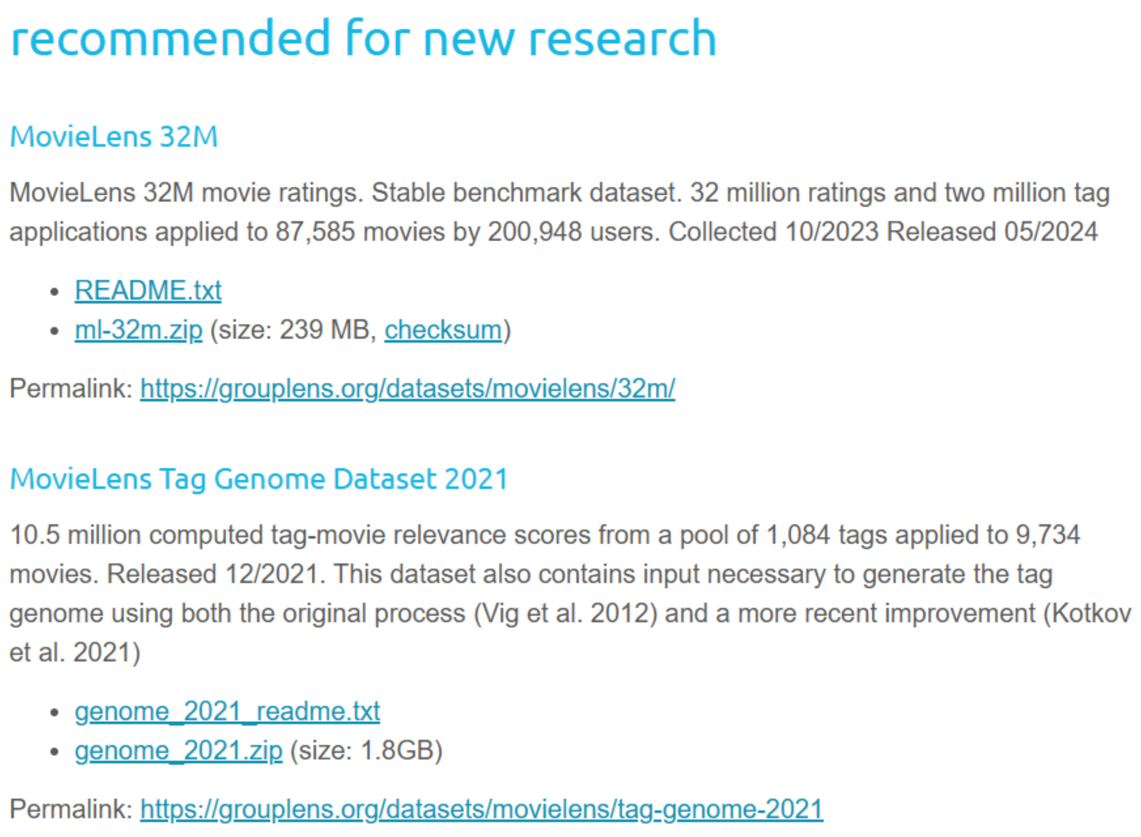
2.1. Data Source & Description

I choose to use the newly released MovieLens 32M dataset as the core training and evaluation data for this recommendation system. This dataset was released by the GroupLens research group in May 2024 and was collected in October 2023. It is currently one of the most complete and realistic public movie rating data.

The dataset contains 32,000,000 rating records, covering 200,948 users and 87,585 movies. Each rating record includes user ID, movie ID, rating value (1 to 5), and timestamp information. The accompanying movies.csv file provides information such as the movie title, release year and type. In addition, this version also contains two million tag application data, which can be used to enhance content modeling and cold start processing.

Compared with the classic MovieLens 25M dataset (released in 2019), the 32M dataset has seen significant growth in the number of users, the number of movies, and the number of reviews. It also covers more new movies in recent years and is closer to the current viewing trends. I believe this is a data source that better meets the requirements of modern recommendation systems and is particularly suitable for my current goal of building a universal movie recommendation platform.

The data set is available through GroupLens website (https://grouplens.org/datasets/movielens/) access, and release on Kaggle convenient for development. In this project, I plan to use the official website version as the citation source and conduct preprocessing and modeling through Kaggle data copies during actual processing.


recommended for new research

MovieLens 32M

MovieLens 32M movie ratings. Stable benchmark dataset. 32 million ratings and two million tag applications applied to 87,585 movies by 200,948 users. Collected 10/2023 Released 05/2024

- README.txt
- ml-32m.zip (size: 239 MB, checksum)

Permalink: https://grouplens.org/datasets/movielens/32m/

MovieLens Tag Genome Dataset 2021

10.5 million computed tag-movie relevance scores from a pool of 1,084 tags applied to 9,734 movies. Released 12/2021. This dataset also contains input necessary to generate the tag genome using both the original process (Vig et al. 2012) and a more recent improvement (Kotkov et al. 2021)

- genome_2021_readme.txt
- genome_2021.zip (size: 1.8GB)

Permalink: https://grouplens.org/datasets/movielens/tag-genome-2021

2.2. Data Fields & Usage

I will use the following core fields from the MovieLens 32M dataset to support the construction of the recommendation model and the establishment of user profiles:

UserId: A unique identifier representing the rating user.

I will use this as the primary key for user modeling to construct rows of user preference vectors or rating matrices, supporting collaborative filtering and personalized recommendations.

MovieId: A unique identifier representing a film.

I will use it to index the corresponding film content features (such as type, tag, etc.) and serve as the columns of the rating matrix. This field is also the core of item representation learning in content-enhanced models.

Rating: It is the user's rating of a certain movie, ranging from 1 to 5 points.

I will use this as a supervisory signal to train the recommendation model, evaluate the difference between the predicted score and the actual preference, and also to construct implicit feedback (for example, a score of ≥ 4 is regarded as positive feedback).

Timestamp: Record the timestamp of the score.

Although the main model may not rely on time features, I can use this field to identify the changing trends of user interests, filter recent viewing behaviors, or build training and test sets of time segments for time sensitivity analysis.

Title, genres: They are respectively the movie title and the genre it belongs to (e.g. "Action: Adventure: Sci-Fi").

I plan to use type information as content feature input to build hybrid models or assist in item modeling during the cold start phase. In addition, type tags can also be used for visual display and user filtering options.

Tag: It represents the keywords marked by the user for a certain movie.

I can regard these tags as more fine-grained content features to improve the interpretability of recommendations, and also construct Tag-Movie embedded representations in specific methods.

Through these fields, I can support collaborative filtering, content-based recommendation, and hybrid recommendation strategies that combine the two. Meanwhile, the simple structure and consistent naming of the fields are also conducive to the rapid prototyping of the model and the organization of the training and testing process.

2.3. Dataset Limitations

Although MovieLens 32M is one of the most comprehensive movie rating datasets currently available publicly, I am also aware that it still has some limitations, which may affect the generalization ability of the recommendation system or its actual deployment.

The coverage of user and behavioral data is limited

MovieLens is a dataset built on the ratings submitted by volunteer registered users, and its user behavior does not represent the complete user group on real commercial platforms such as Netflix or Disney+. In other words, it is more inclined towards the rating habits of active movie enthusiasts rather than including more diverse interactive behaviors such as clicking, browsing, and jumping out.

The dataset has been cleaned and lacks complexity

This dataset is a publicly released research version. The data structure is standardized, outliers have been removed, and the label classification has also been processed. Therefore, it does not contain the common data noise or inconsistent behavior in real scenarios, nor does it lack the semantic information for users to leave negative evaluations or comments, and thus cannot fully reflect the challenges faced by industrial-grade systems.

Lack of cold start information

Although the dataset contains complete data on movie types and user ratings, it does not include users' registration information (such as gender, age, and region) or social attributes, nor does it have metadata supplementation when new movies are launched. This makes me need to rely on external content modeling or hypothesized interaction behaviors when dealing with cold start issues.

It only includes scoring tasks and has a single assessment method

The main goal of MovieLens is to build predictive rating models, so most of the samples are constructed for rating prediction tasks. This imposes limitations on the model evaluation approach, making it unsuitable for verifying system-level recommendation quality metrics such as diversity and coverage, nor for fully simulating the interaction processes on real user interfaces.

Despite these limitations, I believe that MovieLens 32M still has extremely high research and development value at the current project stage. However, it is necessary to maintain a clear understanding of its limitations in the analysis and experimental design.

2.4. Dataset Justification & Relevance

The reason why I chose MovieLens 32M is that it highly aligns with the recommendation system goals I am currently designing in multiple dimensions.

Firstly, from the perspective of data scale and structure, MovieLens 32M offers over 32 million rating data, covering 200,000 users and more than 80,000 movies. It features an excellent user-item sparse matrix structure, making it suitable for building collaborative filtering and hybrid recommendation models. It also supports user interest modeling and clustering.

Secondly, this dataset provides clear structured fields such as rating timestamps, movie types, and tag information. These content features can be used to assist in modeling, cold start processing, or enhance the interpretability of recommendations, which is particularly suitable for the personalized recommendation + new user guidance mechanism I envisioned in my design.

Although this dataset itself was built for the rating prediction task and does not directly support a comprehensive assessment of all recommendation system metrics (such as coverage, diversity, etc.), I believe this does not prevent me from using it to design a representative recommendation process. As long as a reasonable evaluation plan is supplemented in the experimental design, it can still be used to reflect the effect of the model and the system's capacity.

In addition, this dataset has excellent open-source attributes and research background. It is officially released by GroupLens, with a stable structure and standard format. It is easy to integrate into the existing recommendation system toolchain and is also suitable for subsequent expansion and comparative experiments. I can independently build training sets, validation sets and user simulation interfaces without relying on third-party platforms or competing task structures.

Therefore, I believe that MovieLens 32M is currently the most suitable open-source dataset for building and evaluating a universal movie recommendation system, and it can support all my modeling and analysis goals in this project.

3. Methods

3.1. Recommend System Method Selected

In this project, I finally chose to adopt the Hybrid Recommendation strategy, combining the advantages of collaborative filtering and content-enhanced recommendation, in order to better meet the personalized recommendation needs of different users at different usage stages.

The reason why I do not use collaborative filtering alone is that although it can provide precise personalized recommendations when there is sufficient user behavior data, the collaborative filtering method will face obvious cold start problems when there is a lack of sufficient interaction data for new users or new movies. However, if one relies solely on content recommendations, although it can solve the problem of cold start, it is prone to the issue of "conservative recommendations" - users are always recommended to similar types of content that lacks diversity.

Therefore, I have decided to combine these two methods: for active users with historical behaviors, mainly relying on collaborative filtering to provide personalized recommendations; For new users or new movies lacking behavioral data, supplementary recommendations are made by using content information such as the type and tags of the movies.

In terms of specific implementation, I adopt a Weighted Hybrid strategy. In the recommendation scoring stage, I will respectively use the collaborative filtering model and the content recommendation model to score each candidate film, and then set a weight parameter α based on the completeness of the user's data (such as the number of ratings) to calculate the weighted average score:

The final recommendation score = α × collaborative filtering score + (1 - α) × content recommendation score

For users with a large number of behavioral records, I will give collaborative filtering a higher weight (such as $\alpha = 0.8$); For new users, they rely more on content recommendations (such as $\alpha = 0.3$). This integration approach not only ensures the personalization of recommendations but also enhances the system's adaptability to different user states.

I think this hybrid strategy is highly suitable for the universal movie recommendation platform I am currently designing. It can not only provide personalized experiences when user data is abundant, but also quickly offer reasonable recommendations during the cold start stage based on content features, balancing accuracy and practicality.

3.2. Method Justification

Before deciding to use a hybrid recommendation system, I carefully evaluated several mainstream recommendation methods and, in combination with the target scenarios of my current system, analyzed their respective advantages and disadvantages as well as applicability.

Collaborative Filtering

Collaborative filtering is one of the most widely used recommendation methods at present. It can recommend content that users may be interested in based on the similarity between their historical behaviors and those of others. It does not rely on the content information of the items themselves and builds the recommendation model entirely based on interaction data.

I think collaborative filtering is highly suitable for the "active users" group in my current system. Because once users have accumulated a certain number of rating records on the platform, the system can explore potential interests from the rating patterns between them and other similar users, thereby providing accurate and personalized recommendations.

What I am considering using is a collaborative filtering method based on matrix factorization, such as SVD (Singular Value Decomposition), which has good scalability and generalization ability under large-scale scoring data. Meanwhile, MovieLens 32M provides a timestamp field, which can also be expanded to a time-weighted dynamic matrix factorization model during the experimental stage in the future.

However, a significant drawback of collaborative filtering is the cold start problem. When a user is registering for the first time or a certain movie has not yet received sufficient ratings, the system has difficulty making recommendations based on behavioral data.

Content-based Recommendation

To address the limitations of collaborative filtering in cold start scenarios, I introduced a content-enhanced recommendation method, which utilizes metadata such as movie genres and tags to recommend films that are similar in content to users' preferences.

The advantage of this method lies in that even if the user's behavior is minimal, as long as he shows interest in a few movies, the system can recommend similar works based on their content features, thus avoiding the predicament of "no recommendations to give".

In addition, content recommendations are also conducive to enhancing the interpretability of the system. For example, I can display "Because you like Sci-Fi type movies" in the recommendation reason to enhance users' trust and the transparency of recommendations.

Of course, content recommendations also have limitations. It is prone to fall into the problem of "narrowing of interest", that is, the system constantly recommends the same type of topics, resulting in a decline in the diversity of recommendations. Therefore, I will not take content recommendation as the dominant strategy, but use it as a supplement when data is insufficient.

Method Combination

From the above analysis, it can be seen that neither a single collaborative filtering nor a content recommendation method can cover the recommendation needs throughout the entire user life cycle. Collaborative filtering is very powerful after users become active, but it is almost powerless during cold starts. Although content recommendations can fill the gap, they do not have the ability to be personalized in the long term.

Therefore, integrating these two methods into a fusion system is the optimal solution under the current platform structure. This decision is not only aimed at enhancing the quality of recommendations, but also at ensuring the continuity and robustness of the system during actual use, enabling new users to quickly receive recommendations and existing users to continuously receive precise push notifications.

3.3. Hybrid Recommendation System Design Details (Theoretically)

In order to effectively integrate collaborative filtering and content recommendation, I think a Weighted Hybrid Strategy is feasible. The outputs of the two methods are weighted and combined to obtain the final recommendation score.

Hybrid Recommendation Equation

For each candidate film i, the system will calculate respectively:

$Score_{CF}(i)$: The score calculated by the collaborative filtering model

$Score_{CB}(i)$: The score calculated by the content recommendation model

The final recommendation score is calculated by equation:

$$FinalScore_{CB}(i) \ = \boldsymbol{\alpha} * Score_{CF}(i) \ + (1 - \boldsymbol{\alpha}) * Score_{CB}(i)$$

$\boldsymbol{\alpha} \in [0,1]$ is an adjustable weight parameter, representing the proportion of collaborative filtering in the final score.

Weight distribution logic

I plan to dynamically adjust the value of α based on the number of rating records (or activity levels) of users on the platform:

User Type	**Number of Ratings**	**Recommendation Strategy**	**α Value**
New User (Cold-start)	No rating or less than 5	Mainly rely on content-based recommendation	**α** = 0.2
Regular User	Moderate number of rating	Combine behavior-based and content-based	**α** = 0.5
Active User	Sufficient rating history	Mainly rely on collaborative filtering	**α** = 0.8

If time permits, I also hope to explore the method of learning the α value through the model (such as training the α predictor with the sparsity of user embedding) in subsequent experiments. However, at the current stage, I will adopt static interval partitioning to reduce the implementation complexity.

Integrate the design process description

In Model training stage:

Train the collaborative filtering model using matrix factorization (such as SVD)

Based on movie types and tags, construct content feature vectors and calculate content similarity or predict preference scores

In Recommendation stage

Select the corresponding α based on the number of user behavior records

Calculate the collaborative filtering score and content score for each candidate film

Calculate the final score using the fusion formula

Sort and output the Top-N recommended list

Hybrid Recommendation Advantage

The reasonI chose this integration method because it has the following advantages:

Adaptive: The recommendation strategy can be adjusted according to the user's status to avoid a one-size-fits-all approach.

Strong controllability: As an explicit parameter, α is convenient for tuning and visualization in experiments

Simple structure and easy to implement: It does not rely on complex neural network structures and is suitable for the scope of course projects

Support for expansion: In the future, factors such as time, context, and behavioral intensity can be added to further enrich the integration logic

All in all, I chose the weighted hybrid fusion method because it strikes a good balance between implementation cost and flexibility. On the one hand, it can dynamically adjust the recommendation strategy based on the richness of user data, taking into account both the cold start issue of new users and the personalized demands of existing users. On the other hand, it has a clear structure, is easy to debug and evaluate, and is suitable for the implementation goal of my current universal movie recommendation system. Of course, it is not perfect. For instance, the reliance on weight Settings may affect the recommendation performance. However, I believe it can bring out the most core value of a recommendation system under the current task: the recommendations are both reasonable and highly adaptable.

4. Evaluation

4.1. Evaluation Objectives

I hope that through the assessment of this stage, I can have a comprehensive understanding of the performance of the recommendation system on two levels:

At the recommendation model level (based on historical data) : Evaluate the performance of the hybrid recommendation model I have constructed in terms of prediction accuracy, ranking effect, etc.

Overall level of the recommendation system (based on user experience) : Pay attention to the interaction feelings of users during the process of using the recommendation system, such as whether the recommendations are relevant, whether there is diversity, and whether users are willing to continue using them, etc.

I plan to use the methods of "offline metrics" and "simulated user study" respectively, and conduct evaluations by combining subjective and objective data to ensure both technical accuracy and user experience. The ultimate goal is to find a recommended solution that performs well under multiple evaluation criteria and is suitable for the actual use of the platform.

4.2. Offline Evaluation using Historical Data

To evaluate the performance of the recommendation model I designed on historical rating data, I plan to adopt a series of standard offline metrics to measure its accuracy and ranking effect in the "Top-N recommendation task".

Considering that users are more concerned about "whether the system can rank the movies I most want to watch at the Top" in actual use, I will mainly adopt the following several top-n ranking indicators:

Precision@K

Precision@K measures how many of the top K items in the recommended list are actually of interest to users. It reflects the hit rate of the system and is suitable for scenarios that emphasize "whether the recommendation accurately hits the user's preferences". The higher the precision, the more accurate the recommended content is, but it may lead to an overly concentrated recommendation range.

Recall@K

Recall@K measures how many of the items that the user is interested in are included in the top K recommended results. It reflects the coverage capability of the system and is suitable for measuring the completeness of recommendations during the cold start or new user phase.

I think these two metrics are suitable for measuring the "hit rate" and "coverage ability" of recommendations. Among them, Recall is more suitable for cold-start users, while Precision is more suitable for evaluating personalized recommendations from existing users.

NDCG@K (Normalized Discounted Cumulative Gain)

This indicator, based on the hit rate, takes into account the impact of the recommendation sequence on the user experience. The earlier an item is hit, the higher its contribution. NDCG integrates accuracy and ranking quality and is a key indicator for evaluating whether the ranking of recommendations is reasonable.

It is suitable for measuring whether users can quickly find the content they are interested in at the top few positions of the recommendation list.

MAP (Mean Average Precision)

It is used to evaluate the overall accuracy of recommendations and is particularly suitable for measuring the comprehensive performance of the model on all users during cross-validation.

RMSE (Root Mean Squared Error) [auxiliary indicator]

Although RMSE is more often used for score prediction tasks and my project mainly focuses on Top-N recommendations, I will still retain RMSE as a reference for the training loss index of the collaborative filtering model.

Evaluation method description

I will divide the historical scoring data into a training set and a test set by timestamp (for example, split by 80/20), and evaluate the real scoring behavior of each user in the test set. All indicators will be calculated on a per-user basis and the average value will be taken to avoid deviations caused by uneven data volume among users.

The reason why I chose these indicators is that they have been widely verified in the offline evaluation of recommendation systems, and at the same time, they can measure from different perspectives whether the model truly understands users' interests. I can also conduct fair comparisons of different model combinations (collaborative filtering/content recommendation/fusion) in subsequent experiments.

4.3. Metrics Selected and discuss trade-offs

After listing various evaluation indicators for recommendation systems earlier, I began to consider a more practical question: Which indicators are the most important for my project? By what criteria will I judge which model or recommendation strategy is better? First of all, I realized that different indicators reflect different aspects of the recommendation system - some emphasize "hit", some pay more attention to "ranking", and others are suitable for reference during the training process. Therefore, in this part, I attempt to compare their applicability and priority one by one from the perspective of my own project.

Precision@K vs Recall@K

These two indicators are often used together, but in fact, their evaluation perspectives are opposite. Precision@K is more like asking: "How much of what I recommend do users actually like?" It emphasizes the quality of the recommended list. Recall@K is more like asking: "How many things do I recommend that users like?" It emphasizes coverage.

In my project, users only see the first few recommendation results, so I care more about whether the system can provide highly relevant content within the limited recommendation space. Therefore, I prefer to use Precision@K as the preferred hit rate metric and Recall as an auxiliary observation metric during the cold start phase.

NDCG@K vs MAP@K

Both of these indicators can measure the overall performance of the recommendation list, but NDCG@K pay attention to the ranking position and is very sensitive to "relevant content at the front". MAP@K pays attention to average accuracy, but is not sensitive enough to the recommended sequence, and the explanation is not so intuitive.

I think when users are using a recommendation system, the order is a very crucial factor - users often only look at the first few items. If all the content I recommend is ranked at the back, even if it hits the mark, it doesn't make much sense. So I would choose NDCG@K as the main metric for ranking, which can better reflect the "practicality" of the recommendation system.

Limitations of RMSE

Although RMSE is a common training loss metric in collaborative filtering, I am well aware that it is not suitable for evaluating the final Top-K recommendation effect. A low RMSE of a model does not necessarily mean that the content it recommends is truly meaningful to users. Therefore, I will only use RMSE to determine whether the model converges during the model training stage, and will not use it as the basis for model selection.

The metric I final choice

I will not evaluate the model based on a single metric, but rather combine the performance of NDCG and Precision, and take into account the stability under different user types (new users and old users). Specifically, I will compare the average scores of different methods on NDCG and Precision. If a certain method is significantly better in terms of Recall, I will check whether it is suitable for cold-start users. If the results of the two groups are similar, I will combine the feedback from the user experiment as a tie-breaker.

In my opinion, the best system is not necessarily one that excels in a single indicator, but rather one that performs evenly in multiple dimensions and can adapt to different user states.

After comprehensive consideration, I have decided to adopt the following assessment plan:

The primary sorting metric: **NDCG@K**, which takes into account both hit and sorting order, is closest to the user experience.

Hit rate metric: **Precision@K**, measures the accuracy of the recommendation list.

Auxiliary indicators: **Recall@K** (for the new user stage), RMSE (for training process monitoring).

4.4. Computational Requirements of recommender system

Although the prototype stage of this project will not be deployed on a real online platform, I still hope that the recommendation system has good scalability and practical feasibility. Therefore, I also took into account the computing cost and dynamic update capability of the system.

Firstly, in the current design, the collaborative filtering method I adopt is based on SVD matrix factorization, while the content recommendation part is based on the preprocessed movie types and tag vectors. Both types of models can be trained offline locally and generate recommendation results when users visit, without relying on real-time inference or GPU acceleration. Therefore, at the current stage, the system does not require "near real-time recommendation capability".

However, to ensure that the system has the ability to dynamically adapt to changes in user interests, I retained the entry point for model updates in the design. For instance, I can retrain the collaborative filtering model using incremental rating data at regular intervals, or fine-tune the user vectors. For the content feature section, the tags and types themselves do not change frequently and do not need to be rebuilt frequently.

If the system is deployed to the actual platform in the future, I can further use user feedback (such as Like/Dislike) as fine-tuning signals and improve the accuracy of recommendations through a lightweight update mechanism. But at the current stage, I am more concerned about whether the model is structurally easy to retrain rather than real-time performance optimization.

I think that maintaining a controllable implementation cost while ensuring the recommendation effect is the most reasonable balance point at the current project stage.

4.5. Online Evaluation Simulated User Study

In addition to offline evaluations, I also plan to design a round of concise user experiments to assess the overall experience of the recommendation system in real-world usage scenarios. Considering that this project does not involve an official launch, I will collect subjective feedback by using a static UI prototype in conjunction with user simulation tasks.

Experimental design approach

I will invite several classmates/friends to act as simulation users and ask them to complete the following tasks through my interface prototypes (such as Figma mockup or interactive prototypes)

1. Browse the recommended list and select the movies that interest you
2. Perform feedback operations such as "Like/Dislike"
3. Score the recommendation quality, interface interaction and recommendation interpretability of the system

All processes will revolve around "movie recommendation decisions", without involving actual film screenings, and comply with the "informal user research" required by the course.

Types of feedback collected

I plan to collect the following feedback in the form of a "short Questionnaire" :

1. Is the recommendation relevant? Are you interested? (Relevance score: 1-5)
2. Are the recommendations diverse? Is it always similar content? (Diversity Score)
3. Are the reasons for the recommendation clear and easy to understand? Do you trust recommendations? (Interpretability)
4. Would you be willing to use such a system for a long time? (Acceptance Score)

In addition, I will also open up a free feedback item, allowing users to describe in writing the aspects they think the recommendation is "good" or "bad".

Although the sample size of these subjective evaluations is limited, they can help me determine whether the recommendation system has achieved the expected "perceived value of users" and serve as a qualitative guiding basis for future optimization. For instance, if multiple users all think that the recommendations are too monotonous, then I might need to introduce a diversity optimization module.

I think that this lightweight yet genuine user experiment can provide an important supplement to the system experience that offline metrics cannot measure, and at the same time, it is closer to the real goal that recommendation systems ultimately aim for users.