Item-Based Collaborative Filtering Recommender System

Z5007: Programming and Data Structures

Surabhi Gudla
Vineet Joshi

January 20, 2026

Outline

Introduction

- Online platforms contain a very large number of movies
- Users cannot manually explore all available options
- Recommender systems automatically suggest relevant movies
- This project builds a movie recommender from scratch

Why Item-Based Collaborative Filtering?

- Uses previous user ratings
- If users like two movies, the movies are related
- Movie similarity changes slowly over time
- More scalable than user-based methods

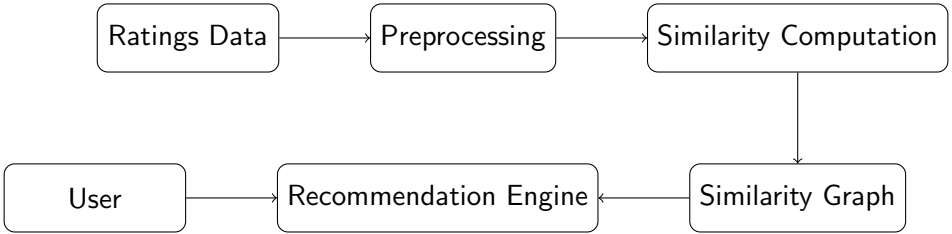
Dataset Used

- MovieLens dataset
- Contains user ratings for movies
- Each rating shows how much a user liked a movie

Dataset Statistics

- Users: about 610
- Movies: about 8,900
- Ratings: about 100,000
- Train–test split: 80% / 20%

System Pipeline



Pipeline Explanation

- First, ratings data is cleaned and split
- Movie similarities are computed offline
- Similar movies are stored in a graph
- Recommendations are generated when a user requests them

Data Structures Used

- Dictionaries to store user–movie ratings: Store all movies rated by each user
- Dictionaries to store movie–user ratings: Store all users who rated a movie
- Graph structure to store similar movies: Store strong similarity relationships between movies

Why These Data Structures?

- Fast access to ratings
- Efficient similarity lookup
- Scales well for large datasets

Recommendation Algorithm

- Compare movies rated by the same users
- Remove user rating bias using average rating
- Calculate similarity using cosine similarity

How Recommendations Are Generated

- Look at movies the user has rated
- Find similar movies from the graph
- Give higher score to more similar movies
- Recommend movies with highest scores

Program Output: Train–Test Split

```
Train size: 80764
Test size: 20072
Number of users: 610
Number of movies: 8985
```

Program Output: Top Similar movies

```
Sample movie ID: 1
Sample movie title: Toy Story (1995)

Top similar movies:
Similarity: 0.818 | Movie ID: 3967 | Title: Billy Elliot (2000)
Similarity: 0.813 | Movie ID: 1148 | Title: Wallace & Gromit: The Wrong Trousers (1993)
Similarity: 0.804 | Movie ID: 47610 | Title: Illusionist, The (2006)
Similarity: 0.803 | Movie ID: 345 | Title: Adventures of Priscilla, Queen of the Desert, The (1994)
Similarity: 0.792 | Movie ID: 81834 | Title: Harry Potter and the Deathly Hallows: Part 1 (2010)
```

Program Output: Recommendations

```
Top-N Recommendations:
Pineapple Express (2008) | Score: 5.0
Whiplash (2014) | Score: 5.0
Gravity (2013) | Score: 5.0
Little Women (1994) | Score: 5.0
Mary Shelley's Frankenstein (Frankenstein) (1994) | Score: 5.0
```

Program Output: Evaluation K=5

```
Evaluation Metrics:
{'Precision@K': 0.012, 'Recall@K': 0.006384, 'NDCG@K': 0.017896}
```

Program Output: Evaluation K=10

```
Evaluation Metrics:
{'Precision@K': 0.011, 'Recall@K': 0.011017, 'NDCG@K': 0.017343}
```

Evaluation Method

- Some ratings are hidden for testing
- Movies rated 4 or higher are treated as relevant
- We check if recommendations match test data

Evaluation Metrics

- Precision@K – correctness of recommendations
- Recall@K – coverage of relevant movies
- NDCG@K – quality of ranking

Evaluation Results

- Precision@5 = 0.012
- Recall@5 = 0.006384
- NDCG@5 = 0.017986

Why Are the Values Small?

- Large number of movies
- Each user rates only a few movies
- Top-5 recommendation is strict

Conclusion

- Built an item-based recommender from scratch
- Used only basic data structures and algorithms
- System produces meaningful recommendations

Future Work

- Combine content-based features
- Improve accuracy using hybrid models
- Handle new users better

Thank You