AIE425 Intelligent Recommender Systems, Fall Semester 25/26

Assignment 1: Neighborhood CF & Clustering in CF

Student ID: A20000366
A20000375
A20001134

Full Name: Rawan Emad
Reem Ahmed
Sohila Mohamed

3.1.16 Discussion: Matrix Sparsity, Rating Bias and Long-Tail Effects

Using the MovieLens-20M dataset we obtained:
- 138,493 users, 26,744 items, and 20,000,263 ratings.
- The rating matrix density is approximately 0.54%, i.e. more than 99% of all possible user–item pairs are missing.

Average user and item activity are around 144 ratings per user and 748 ratings per item, but this average hides a very skewed distribution as discussed below.

1) Insights from Co-Rating Statistics and Matrix Sparsity

From the co-rating analysis (points 13 and 14), we consider three target users U1, U2, and U3 with different coverage levels of the item space. For each user, we looked at the number of other users who share at least one co-rated item (No_common_users), and for two target items we counted the number of co-rated items (No_coRated_items).

For users:
- U1: 104 ratings (coverage ≈ 0.39% of all items) with No_common_users ≈ 132,262 users.
- U2: 978 ratings (coverage ≈ 3.66%) with No_common_users ≈ 138,467 users.
- U3: 2,108 ratings (coverage ≈ 7.88%) with No_common_users ≈ 138,485 users.

Even though the matrix is globally very sparse, almost every user shares at least one movie with each target user. This is due to the presence of very popular movies that act as “hubs” and connect many users.

For items:
- I1: co-rated with about 13,303 other items (≈ 49.7% of all items).
- I2: co-rated with about 14,587 other items (≈ 54.5% of all items).

Each low-rated target movie is therefore co-rated with roughly half of the catalogue, indicating that a relatively small set of active users ties many items together.

In point 14 we added a stricter condition: users must have co-rated at least 30% of the target user’s items. The resulting β values (beta_per_user) are:
- U1: about 8,601 users meet the 30% threshold (≈ 6.2% of all users).
- U2: about 3,572 users (≈ 2.6% of users).
- U3: about 909 users (≈ 0.66% of users).

Thus β, defined as the maximum over the three, equals 8,601.

Comparing the results with and without the 30% threshold shows that with just one common item, overlaps are almost universal, but when we require substantial overlap, only a small minority of users remain similar. The proportion decreases as the target user’s activity increases (U3 has the highest coverage and the fewest strong neighbors). This highlights the sparse yet highly connected nature of the matrix: it is sparse globally, but dense around popular items and within communities of similar users.

2) Rating Bias

Items were grouped according to their average rating percentage (average rating divided by 5 and multiplied by 100) into ten groups G1–G10. The counts per group showed that:
- G1–G3 contain almost no items (no item has an extremely low average rating).
- G4–G5 contain a small number of items with relatively low averages.
- G6–G7 contain items with moderate average ratings.
- G8–G10 contain the majority of items, with averages above approximately 2.5–3.5 out of 5.

In terms of the number of ratings, most ratings fall into G9–G10, corresponding to items with high average scores. Very few ratings correspond to items with very low averages. This clearly indicates a positive rating bias: users tend to rate items they already expect to like and systematically give scores in the upper part of the scale. The absence or scarcity of items in the lowest groups and the concentration of ratings in the highest groups show a strong compression towards high values, which must be taken into account when using raw averages or when normalizing ratings.

3) Long-Tail Popularity

The plot of the number of ratings per item (sorted in ascending order) exhibits a typical long-tail pattern:
- A small set of very popular movies receives a very large number of ratings.
- A long tail of movies receives comparatively few ratings, with many items rated only a few times.

The co-rating analysis of the target items supports this observation. The chosen items have hundreds of ratings and are co-rated with thousands of other items, typical of head items that drive most of the overlap and connectivity in the graph. However, many other items reside in the tail with few ratings and limited connectivity, which makes them difficult to model and recommend.

This long-tail behavior leads to an inherent trade-off for recommender systems: focusing on popular items improves accuracy because we have more data, but ignoring the tail reduces coverage and novelty, and may hurt user satisfaction for niche interests.

4) Overall Conclusions

Overall, the MovieLens-20M dataset is highly sparse (with density around 0.54%) but still strongly connected through popular items, as shown by the large No_common_users and No_coRated_items values. When we tighten the similarity criterion (using the 30% overlap threshold), only a small fraction of users remain strongly related, emphasizing the challenge of finding reliable neighbors for collaborative filtering, especially for heavy raters.

Ratings are biased towards high values, with most items and almost all ratings concentrated in groups with average ratings above 2.5–3.5. The item popularity distribution follows a clear long-tail pattern, where a few popular movies accumulate

most ratings and co-ratings, while many items are rarely rated. These characteristics—sparsity, rating bias, and a pronounced long tail—are crucial to consider when designing and evaluating recommender systems on the MovieLens-20M dataset.

Part Assignments: Case Studies 1, 2, and 3

3.1.16 Discussion: Matrix Sparsity, Rating Bias and Long-Tail Effects
Using the MovieLens-20M dataset we obtained:
- 138,493 users, 26,744 items, and 20,000,263 ratings.
- The rating matrix density is approximately 0.54%, i.e. more than 99% of all possible user–item pairs are missing.
Average user and item activity are around 144 ratings per user and 748 ratings per item, but this average hides a very skewed distribution as discussed below.

1) Insights from Co-Rating Statistics and Matrix Sparsity

From the co-rating analysis (points 13 and 14), we consider three target users U1, U2, and U3 with different coverage levels of the item space. For each user, we looked at the number of other users who share at least one co-rated item (No_common_users), and for two target items we counted the number of co-rated items (No_coRated_items).

For users:
- U1: 104 ratings (coverage ≈ 0.39% of all items) with No_common_users ≈ 132,262 users.
- U2: 978 ratings (coverage ≈ 3.66%) with No_common_users ≈ 138,467 users.
- U3: 2,108 ratings (coverage ≈ 7.88%) with No_common_users ≈ 138,485 users.

Even though the matrix is globally very sparse, almost every user shares at least one movie with each target user. This is due to the presence of very popular movies that act as “hubs” and connect many users.

For items:
- I1: co-rated with about 13,303 other items (≈ 49.7% of all items).
- I2: co-rated with about 14,587 other items (≈ 54.5% of all items).

Each low-rated target movie is therefore co-rated with roughly half of the catalogue, indicating that a relatively small set of active users ties many items together.

In point 14 we added a stricter condition: users must have co-rated at least 30% of the target user’s items. The resulting β values (beta_per_user) are:
- U1: about 8,601 users meet the 30% threshold (≈ 6.2% of all users).
- U2: about 3,572 users (≈ 2.6% of users).
- U3: about 909 users (≈ 0.66% of users).
Thus β, defined as the maximum over the three, equals 8,601.

Comparing the results with and without the 30% threshold shows that with just one

common item, overlaps are almost universal, but when we require substantial overlap, only a small minority of users remain similar. The proportion decreases as the target user's activity increases (U3 has the highest coverage and the fewest strong neighbors). This highlights the sparse yet highly connected nature of the matrix: it is sparse globally, but dense around popular items and within communities of similar users.

2) Rating Bias

Items were grouped according to their average rating percentage (average rating divided by 5 and multiplied by 100) into ten groups G1–G10. The counts per group showed that:
- G1–G3 contain almost no items (no item has an extremely low average rating).
- G4–G5 contain a small number of items with relatively low averages.
- G6–G7 contain items with moderate average ratings.
- G8–G10 contain the majority of items, with averages above approximately 2.5–3.5 out of 5.

In terms of the number of ratings, most ratings fall into G9–G10, corresponding to items with high average scores. Very few ratings correspond to items with very low averages. This clearly indicates a positive rating bias: users tend to rate items they already expect to like and systematically give scores in the upper part of the scale. The absence or scarcity of items in the lowest groups and the concentration of ratings in the highest groups show a strong compression towards high values, which must be taken into account when using raw averages or when normalizing ratings.

3) Long-Tail Popularity

The plot of the number of ratings per item (sorted in ascending order) exhibits a typical long-tail pattern:
- A small set of very popular movies receives a very large number of ratings.
- A long tail of movies receives comparatively few ratings, with many items rated only a few times.

The co-rating analysis of the target items supports this observation. The chosen items have hundreds of ratings and are co-rated with thousands of other items, typical of head items that drive most of the overlap and connectivity in the graph. However, many other items reside in the tail with few ratings and limited connectivity, which makes them difficult to model and recommend.

This long-tail behavior leads to an inherent trade-off for recommender systems: focusing on popular items improves accuracy because we have more data, but ignoring the tail reduces coverage and novelty, and may hurt user satisfaction for niche interests.

4) Overall Conclusions

Overall, the MovieLens-20M dataset is highly sparse (with density around 0.54%) but still strongly connected through popular items, as shown by the large No_common_users and No_coRated_items values. When we tighten the similarity criterion (using the 30% overlap threshold), only a small fraction of users remain strongly related, emphasizing the challenge of finding reliable neighbors for collaborative filtering, especially for heavy raters.

Ratings are biased towards high values, with most items and almost all ratings concentrated in groups with average ratings above 2.5–3.5. The item popularity distribution follows a clear long-tail pattern, where a few popular movies accumulate most ratings and co-ratings, while many items are rarely rated. These characteristics—sparsity, rating bias, and a pronounced long tail—are crucial to consider when designing and evaluating recommender systems on the MovieLens-20M dataset.

3.2.3 Discussion and Conclusion

3.2.3.1 Outcomes

This study thoroughly evaluated a range of **Collaborative Filtering (CF)** techniques using the MovieLens dataset, focusing on both **user-based** and **item-based** approaches. Multiple similarity metrics were tested, including:

Raw Cosine Similarity

Mean-Centered Cosine Similarity

Pearson Correlation Coefficient (PCC)

Additionally, the study incorporated **Significance Weighting** through the **Discount Factor (DF)** and **Discounted Similarity (DS)** to improve the reliability of similarity measures in sparse or unevenly distributed datasets.

The key outcomes from the analyses are as follows:

Raw Cosine Similarity:

Tends to produce high similarity scores even when users have very different rating scales. For example, a strict rater (average 2–3 stars) and a generous rater (average 4–5 stars) can appear highly similar if they rate the same items proportionally.

Users with very few common ratings could appear in the top 20% neighbors, which may not reflect true similarity.

As a result, some top neighbors were unreliable, potentially reducing prediction accuracy.

Mean-Centered Cosine Similarity:

Adjusting ratings by subtracting the user or item mean effectively reduced the bias caused by strict or generous raters.

This normalization resulted in a significant reshuffling of neighbor rankings. Some previously high-similarity neighbors dropped out, while more meaningful neighbors entered the top 20%.

Predictions using mean-centered similarity were generally more stable and better reflected true preferences.

Pearson Correlation Coefficient (PCC):

PCC measures the linear relationship between deviations from the mean, highlighting whether users or items move together.

In several cases, PCC converted high positive cosine similarities into negative correlations, indicating that users with seemingly similar rating patterns actually had opposite preferences relative to their average ratings.

PCC proved highly effective at detecting real agreement and disagreement, especially in situations with rating scale differences.

Significance Weighting (DF and DS):

By incorporating the number of commonly rated items, DS penalized users/items with insufficient overlap.

After discounting:

The top 20% neighbor lists changed in most cases.

The selected neighbors were more reliable, reflecting stronger evidence of similarity.

Predicted ratings became more stable and, in many instances, more accurate.

Overall Conclusion for Outcomes:
Applying significance weighting consistently enhances the trustworthiness of recommendations and stabilizes predictions. Combining mean-centering or PCC with discounting yields the most reliable results, particularly in sparse datasets.

3.2.3.2 Summary of the Comparison of Part 1 and Part 2

The study compared **Part 1 (User-Based CF)** and **Part 2 (Item-Based CF)** to examine how different CF approaches respond to similarity metrics and discounting.

User-Based Filtering (Part 1):

Similarity heavily depends on user behavior patterns.

Highly sensitive to differences in rating scales; strict versus generous raters may be misrepresented.

Users with few common items could appear highly similar by chance.

Applying Discounted Similarity (DS) significantly improved the reliability of the top 20% neighbors.

Top neighbor sets changed considerably after discounting, reflecting more meaningful neighbor selection.

Item-Based Filtering (Part 2):

Focuses on item-item relationships rather than user-user relationships.

Tends to be more stable since items usually have more ratings, reducing the effect of sparsity.

Less sensitive to individual user bias, but discounting still improved the reliability of item neighbors.

The top 20% item similarity lists were more consistent, and prediction changes were smoother compared to user-based CF.

Impact of Significance Weighting:

Penalized users/items with few co-rated items.

Removed “false neighbors” that had high similarity purely due to small sample size.

Improved the consistency and reliability of top-N similarity lists.

The improvement was more pronounced in **user-based CF** because user behavior is more variable than item ratings.

In Summary:

User-based CF is flexible but can be error-prone without discounting.

Item-based CF is generally more stable and consistent.

Applying DF and DS improves both methods, but its effect is particularly crucial for user-based filtering.

3.2.3.3 Conclusion

This study demonstrates that **similarity metric selection** and **significance weighting** critically influence recommendation quality:

Raw Cosine Similarity is simple and computationally efficient but can be misleading when users have different rating scales or when the dataset is sparse.

Mean-Centered Cosine and **PCC** provide bias correction, with PCC being especially effective at detecting genuine agreement or disagreement in preferences.

Significance Weighting (DF and DS) improves neighbor reliability, reduces the impact of weak overlaps, enhances prediction accuracy, and stabilizes recommender system performance.

Trade-offs:

Discounting may exclude potentially useful neighbors if the overlap threshold is too strict, particularly in sparse datasets.

However, the benefits—more reliable recommendations and improved accuracy—often outweigh this drawback.

Practical Implications:

Applying significance weighting is highly recommended in real-world recommender systems.

When combined with mean-centering or PCC, it significantly increases the credibility of predictions and improves robustness, especially in sparse or unevenly rated datasets.

Overall, significance weighting is a low-cost, practical enhancement that adds meaningful improvements to neighborhood-based collaborative filtering.

1. Clustering Strategies Comparison

1.1. Comparison Across All Four Clustering Approaches

Section THREE examined four clustering-based collaborative filtering strategies:

1. Part 1 — Average Rating–Based Clustering
2. Part 2 — Co-rating Statistics Clustering (Avg/Max/Min common items)
3. Part 3 — Rating Pattern Clustering (full rating vectors)
4. Part 4 — Hybrid Clustering (combined co-rating + rating behavior features)

Across these approaches, clustering improved computational efficiency by restricting similarity calculations to smaller user groups, but the degree of improvement and prediction accuracy varied.

Accuracy Trends

•**Part 3 (rating-pattern clustering)** and **Part 4 (hybrid)** generally produced the most accurate predictions, because the clusters reflect actual rating behavior.

•**Part 1 (average rating)** produced the least accurate predictions, since users with similar averages can have very different rating patterns.

•**Part 2 (co-rating overlap)** performed well when co-rating density was high but struggled when users had sparse overlaps.

Efficiency Trends

•**Part 1 and Part 2** produced the **highest** computational efficiency, because the feature vectors were very low-dimensional (1–3 values).

•**Part 3** had moderate efficiency improvements; clusters were based on high-dimensional rating vectors.

•**Part 4** provided a balanced trade-off: higher cost than Parts 1–2, but still more efficient than full CF.

Applicability

•**Part 1** works when user behavior is simple or when rating vectors are sparse.

•**Part 2** is ideal for sparse datasets with varied overlap.

•**Part 3** is best when ratings are dense enough to capture full behavioral patterns.

•**Part 4** is optimal for large datasets requiring both accuracy and efficiency.

Clustering Strategies Comparison — Narrative Summary

Part 1 — Clustering Based on Average Rating

This is the simplest and fastest approach since it relies on only one feature: the user's average rating.
However, this simplicity makes it the least accurate method because users with the same average rating can still have completely different rating behaviors.
It is most useful in very large systems where computational speed is the top priority.

——

Part 2 — Clustering Based on Co-Rating Statistics

This method groups users based on how many items they have rated in common (average, maximum, and minimum overlap).
It performs well in sparse datasets, where users do not all rate the same items.
It produces better accuracy than Part 1 while still being computationally efficient since each user is represented by only three values.
Its limitations appear when the overlap between users is extremely small.

——

Part 3 — Clustering Based on Full Rating Patterns

This approach uses the entire rating vector for each user, making it the most behavior-aware method.
It typically yields high prediction accuracy, since users are grouped based on the actual shapes of their rating histories.
However, it is more computationally expensive, especially for large datasets with many items.
It is ideal when users have rated enough items to provide meaningful patterns.

Part 4 — Hybrid Clustering (Co-Rating + Rating Behavior)

The hybrid method combines both co-rating structure and rating behavior.
It provides the best balance among all approaches:

- •accuracy higher than Parts 1, 2, and 3
- •efficiency better than Part 3
- •stable performance even in sparse data

This makes it the most suitable method for real-world recommender systems that require both accuracy and scalability.

1.3. Strengths and Weaknesses

Part 1 — Average Rating Clustering

- •Strengths: fastest, simplest, extremely efficient.
- •Weaknesses: low accuracy; ignores rating patterns.

Part 2 — Co-Rating Statistics Clustering

- •Strengths: captures structural overlap between users; good for sparse matrices.
- •Weaknesses: accuracy depends on overlap; users with low ratings may cluster poorly.

Part 3 — Rating Pattern Clustering

- •Strengths: captures full behavior; best modeling of user taste.
- •Weaknesses: computationally expensive; requires dense rating data.

Part 4 — Hybrid Clustering

- •Strengths: best accuracy; strong balance between structure and behavior.
- •Weaknesses: more complex to implement; moderate computational cost.

Computational Efficiency Analysis

2.1. Speedup Achieved by Each Approach

Across all clustering strategies, speedup followed this general trend:

Part 1 > Part 2 > Part 4 > Part 3 > No clustering

- Part 1 provided the highest speedup due to single-feature clustering.
- Part 3 provided the least speedup due to high-dimensional user vectors.
- Hybrid methods retained strong efficiency while improving accuracy.

2.2. Memory Requirements

- Without clustering: store full similarity matrix → O(U^2) memory.
- With clustering: similarity is computed within clusters

Memory savings were greatest in Part 1 and Part 2 (small, low-dimensional feature sets).

2.3. Scalability Improvements

Clustering improves scalability by:

- reducing the pairwise similarity computations,
- lowering per-user neighborhood size,
- enabling parallel, cluster-specific models.

This is especially important in real-world recommenders with millions of users.

3.2. When Clustering Is Beneficial

Clustering is advantageous when:

- the number of users is large,
- co-rating overlap is uneven,
- computation cost for neighbor selection is high,
- rating patterns differ widely across user groups.

Not beneficial when:

- dataset is extremely small,
- all users rate similar items (high homogeneity),
- full CF is computationally cheap.

3.3. Recommendations for Choosing Clustering Strategy

Use:

- **Part 1** for fast, approximate systems.
- **Part 2** for sparse datasets or when overlap matters (e.g., movie ratings with long tails).
- **Part 3** for highly accurate recommenders with rich data.
- **Part 4** for real-world deployment: the best balance of accuracy and scalability.

4. Cold-Start Problem Solution

4.1. Effectiveness of Clustering for Cold-Start Users

Clustering provides several advantages for cold-start scenarios:

1. Faster Initial Placement

Even with few ratings, users can be placed into a cluster using:

- average rating (Part 1),
- co-rating structure (Part 2),
- partial rating patterns (Part 3),
- hybrid inference (Part 4).

2. Reduced Neighbor Search Space

Instead of comparing a cold-start user to the entire user base, clustering ensures similarity is computed only within the assigned cluster.

3. Improved Prediction Stability

Cold-start users benefit from:

- more consistent neighbors within clusters,
- reduced noise from unrelated users,
- better significance weighting due to cluster homogeneity.

4. Lower Risk of Sparsity Error

Clusters with dense overlap improve similarity stability even when the new user has only a few ratings.

4.2. Cold-Start Performance Across Clustering Strategies

Cold-start users and items present major challenges because they provide little information for similarity computation. The four clustering strategies respond to this problem differently:

- **Part 1 — Average Rating Clustering**

Performs reasonably well for cold-start users since even a small number of ratings is enough to compute an average. However, item-level cold-start remains poorly addressed because items are not clustered. Overall, it offers a basic but incomplete solution.

- **Part 2 — Co-Rating Statistics Clustering**

Cold-start users tend to have low overlap with others, making them difficult to place into meaningful clusters. As a result, this method struggles the most with cold-start scenarios, especially when common-rating counts approach zero.

- **Part 3 — Full Rating Pattern Clustering**

Requires a richer rating history to accurately capture user behavior. Cold-start users are therefore difficult to cluster correctly. However, once placed, the cluster offers strong prediction stability due to coherent rating patterns.

•**Part 4 — Hybrid Clustering**

Provides the best cold-start performance. Even users with few ratings can be partially clustered using average rating or co-rating signals, while more experienced users benefit from full behavior-based features. This dual representation makes hybrid clustering more robust when information is incomplete.

Overall, hybrid clustering (Part 4) delivers the most reliable cold-start handling among all strategies.

4.3. Practical Guidelines for Handling Cold-Start Users and Items

Based on the comparative analysis, the following guidelines are recommended:

1. **Use simple features first** (e.g., average rating or number of ratings) to assign new users to preliminary clusters.
2. **Gradually update cluster membership** as more data becomes available, especially when rating patterns stabilize.
3. **Apply significance weighting** for cold-start users since similarities computed from few co-ratings may be unreliable.
4. **Use item-level clustering** (if available) to address cold-start items, grouping them by metadata, genre, or content features.
5. **Fallback strategies** such as global averages, trending items, or default baselines should be used when no meaningful overlap exists.
6. **Hybrid clustering is recommended** because it can incorporate partial behavior (few ratings), co-rating structure, and average tendencies simultaneously.

These guidelines ensure smoother handling of cold-start scenarios in both user-based and item-based CF systems.

---

5. Conclusion

5.1. Overall Findings from All Clustering Experiments

Across all experiments, clustering significantly improved both computational efficiency and prediction quality compared to non-clustering collaborative filtering. Each clustering method produced distinct trade-offs:

•Simple clustering (Part 1) offered large speedups but limited accuracy.

•Overlap-based clustering (Part 2) captured structural relationships but struggled with very sparse users.

•Full rating-pattern clustering (Part 3) delivered high accuracy but required heavier computation.

•Hybrid clustering (Part 4) consistently combined strong accuracy with practical efficiency, making it the most effective approach.

In all cases, clustering reduced the search space for neighbors and improved the stability of similarity scores.

5.2. Recommended Clustering Strategy for This Dataset

Based on the strengths and weaknesses identified, the hybrid clustering approach (Part 4) is recommended for this dataset. It achieves:

- better prediction accuracy than all other strategies,
- more coherent clusters,
- improved handling of sparse and incomplete rating profiles,
- strong computational efficiency without sacrificing quality.

It provides the best balance between performance and scalability, making it the most suitable choice for real-world deployment.

5.3. How Clustering Addresses Challenges from Section ONE

Clustering directly mitigates the core challenges identified earlier in the project:

- **Sparsity**: By grouping similar users, clustering ensures that similarity is computed only within dense, relevant neighborhoods, reducing noise from unrelated users.
- **Bias in average ratings**: Clustering based on behavioral patterns or co-rating structure reduces the distortion caused by extreme raters or users with unusually high/low averages.
- **Long-tail distribution of items:** When users with similar niche interests are grouped together, long-tail items gain more meaningful similarity signals, improving predictions for rare items.

Thus, clustering reduces sparsity effects, improves the robustness of similarity metrics, and increases prediction reliability.

5.4. Practical Insights for Deploying Clustering-Based CF Systems

When building practical recommender systems, the following considerations are essential:

- **Cluster maintenance:** Clusters should be periodically refreshed as new ratings arrive to prevent drift.
- **Incremental updates:** New users should be placed using lightweight heuristics (e.g., average rating, early co-ratings) and reassigned later.
- **Hybrid feature design** greatly enhances both accuracy and efficiency in large-scale systems.
- **Parallelization** becomes easier because similarity calculations occur independently inside each cluster.

•**Cold-start support** improves when clustering captures high-level patterns early, not just detailed rating behavior.

•**Balanced cluster sizes** reduce computational overhead and improve neighbor selection stability.

Overall, clustering transforms collaborative filtering from a quadratic-complexity process into a scalable, structured system capable of handling sparse, biased, and long-tail datasets effectively.