Item-Based Collaborative Filtering for Music Recommendation

A Co-occurrence Matrix Approach

Your Name

June 29, 2025

Overview

Music Recommendation Challenge

Objective

Build a personalized music recommendation system using item-based collaborative filtering

Dataset Structure:

- **user_id**: Unique identifier for each user
- **song_id**: Unique identifier for each track
- **listen_count**: Number of times user played the song

Data Characteristics:

- No ratings, only interaction counts
- Real-world music streaming data

Key Statistics:

- Unique users: Large scale
- Unique songs: Thousands
- Train/Test split: 80/20
- Data format: User-Item interaction matrix

Data Quality:

- No missing values
- No duplicate entries

Dataset Details & Preprocessing

Core Data Fields

user_id User identifier - represents individual music listeners

song_id Track identifier - represents individual songs in the catalog

listen_count Frequency of interaction - how many times user played the song

Why These Fields Matter:

- **user_id**: Enables personalization
- **song_id**: Items to recommend
- **listen_count**: Measures preference strength

Data Insights:

- Co-occurrence patterns reveal similarity

Preprocessing Steps:

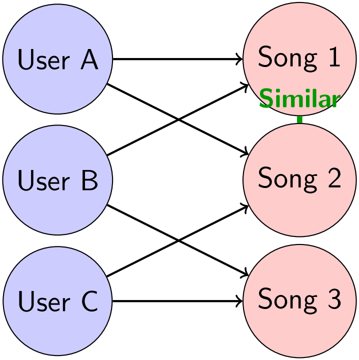
1. Load user-track interaction data
2. Filter users with 30 unique songs
3. Split into train/test (80/20)
4. Create user-item interaction matrix

Result: Clean, dense dataset focusing on engaged users with sufficient listening history

Item Similarity Principle

Core Assumption

"Songs listened to by similar users are likely to be similar to each other"



Algorithm Workflow

Algorithm 1 Item Similarity Recommender

Input: User ID, Training Data
Get user's listening history: $S_u = \{s_1, s_2, ..., s_k\}$
Get all songs in system: $S_{all} = \{s_1, s_2, ..., s_n\}$
For each song pair (s_i, s_j):
    Compute Jaccard similarity using co-occurrence
Build co-occurrence matrix $M_{k \times n}$
Compute weighted similarity scores
Rank songs by similarity scores
Filter out already-listened songs
Return: Top-10 recommendations

Co-occurrence Matrix Construction

Jaccard Similarity Formula

$$J(s_i, s_j) = \frac{|U_{s_i} \cap U_{s_j}|}{|U_{s_i} \cup U_{s_j}|} \tag{1}$$

where U_{s_i} = set of users who listened to song s_i

Matrix Dimensions: $k \times n$

- k = user's songs, n = total songs
- Each cell = Jaccard similarity

Quick Example: Song A: {User1, User2, User3}, Song B: {User2, User3, User4}
Jaccard $= \frac{2}{4} = 0.5$ (50% similar)
Matrix: $k \times n$ (k = user's songs, n = total)

Weighted Score Calculation

$$Score(s_j) = \frac{1}{k}\sum_{i=1}^{k} J(s_i, s_j) \tag{2}$$

Example: User {Rock, Pop, Jazz}, Blues similarities {0.6, 0.2, 0.8}
Score $= \frac{0.6+0.2+0.8}{3} = 0.53$

Key Implementation Details

Data Preprocessing

- Filter users with 30 unique songs (active users)
- 80/20 train-test split

Similarity Computation

```
# Jaccard Index calculation
users_intersection = users_i.intersection(users_j)
users_union = users_i.union(users_j)
similarity = len(users_intersection) / len(users_union)
```

Recommendation Generation

- Compute average similarity across all user's songs
- Sort by similarity score (descending)
- Filter out previously listened songs
- Return top-10 recommendations

Evaluation Methodology

Metrics Used

- **Precision@K**: $\frac{\text{Relevant items in top-K}}{\text{K}}$
- **Recall@K**: $\frac{\text{Relevant items in top-K}}{\text{Total relevant items}}$
- Evaluated for K = 1, 2, ..., 10

Experimental Setup

- User sample: 5% of test users
- Test data: 1000 records subset
- Recommendations: Top-10 per user
- Ground truth: User's actual listening in test set

Precision-Recall Curve: Shows trade-off between precision and recall across different K values

Results: Impact of User Filtering

Before Filtering (All Users):

- **Precision@10**: Up to 20%
- **Recall@10**: Up to 12.5%
- Curve position: Center to upper-left
- Includes casual users (1-5 songs)

After Filtering (30 songs):

- **Precision@10**: 7.5-10%
- **Recall@10**: 2.5%
- Curve position: Lower-left corner
- Only active, engaged users

Performance Comparison:

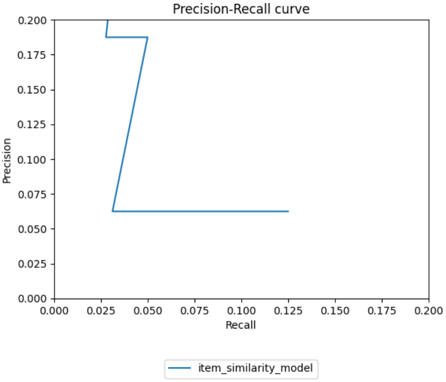
Metric	Before	After
Max Precision	20%	10%
Max Recall	12.5%	2.5%
User Quality	Mixed	High
Realism	Low	High

Why the Drop?

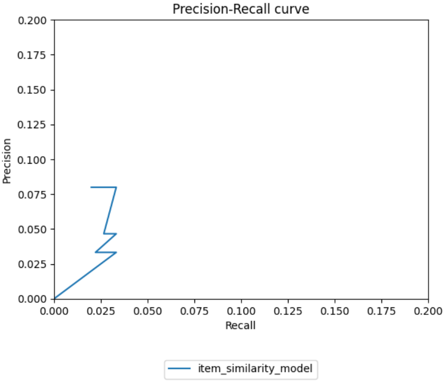
- Casual users = easy predictions
- Active users = complex preferences
- Filtered results more realistic

Precision-Recall Curves Comparison

Before Filtering



After Filtering (30 songs)



Key Insight

Performance drop is expected and desirable - it indicates proper evaluation methodology focusing on meaningful user segments

Why User Filtering Improves Evaluation Quality

The "Easy Prediction" Problem

Casual Users (1-5 songs): Small test sets $\rightarrow$ Artificially high precision/recall

Mathematical Example:

Casual User:

$$\text{Test songs} = 1$$
$$\text{Correct predictions} = 1$$
$$\text{Precision} = \frac{1}{1} = 100\%$$

Active User:

$$\text{Test songs} = 8$$
$$\text{Correct predictions} = 1$$
$$\text{Precision} = \frac{1}{8} = 12.5\%$$

Business Relevance:

- **Active users**: Generate revenue, stay engaged
- **Casual users**: May not return, limited value
- **Filtering focus**: Real users
- **Realistic metrics**: True system performance

Industry Standard: Music platforms evaluate on users with sufficient listening history (20-50 songs)

Strengths & Weaknesses

Advantages:

- **Personalized**: Based on actual user behavior
- **Explainable**: "Because you liked X, try Y"
- **No content needed**: Only interaction data
- **Domain agnostic**: Works across different domains
- **Quality recommendations**: Similar items tend to be relevant

Limitations:

- **Cold start**: Can't recommend to new users
- **Sparsity**: Matrix mostly zeros
- **Scalability**: Expensive computation for large datasets
- **Echo chamber**: May lack diversity
- **Popular bias**: Tends to recommend popular items

Best Use Cases

Music streaming, e-commerce, movie recommendations where user-item interactions are abundant

Summary & Future Work

Key Achievements

- Successfully implemented item-based collaborative filtering
- **Demonstrated importance of user filtering** for realistic evaluation
- Achieved meaningful precision-recall performance (7-10% precision for active users)
- **Revealed evaluation bias** in unfiltered data (artificially high metrics)
- Created scalable, interpretable recommendation system

Critical Insights

- **User filtering is essential**: Prevents inflated metrics from casual users
- **Performance drop is good**: Indicates realistic, challenging evaluation
- **Focus on engaged users**: More business-relevant metrics

Thank you for your attention!