# AIE425: Intelligent Recommender systems, Fall Semester 24/25

## Assignment #1: Neighborhood CF models (user, Item-based CF)

### Name: Hamza Nashaat Abdelbaki

### ID: 2211003328

**Selected Company**: Spotify  
Spotify is a popular music streaming service that uses advanced recommender systems to enhance user experience through personalized playlists such as "Discover Weekly" and "Daily Mix." These systems rely on various feedback mechanisms, primarily implicit signals like track plays, skips, repeats, and listening time, rather than explicit ratings.

In addition to Spotify, several other companies across different domains rely on recommender systems to enhance user experience:

* Netflix: Recommender system for personalized movie and TV show suggestions based on viewing history and ratings.
* Amazon: Product recommendations based on user purchase history and browsing behavior.
* YouTube: Suggests videos using watch history, search queries, and user engagement (e.g., likes, shares).
* Spotify: Uses a music recommendation system to suggest songs and playlists such as "Discover Weekly" based on user listening habits.

In this report, Spotify has been chosen for its advanced music recommender system, which relies heavily on implicit feedback such as track plays, skips, and repeats.

**Customer Feedback Collection and Rating Type**

Spotify primarily collects implicit feedback rather than explicit ratings to understand user preferences. Implicit signals include:

* Track Plays: Indicates user interest in the track.
* Skips: Suggests disinterest if a track is skipped shortly after it starts.
* Repeats: Shows a strong positive preference when a user listens to the same track multiple times.
* Listening Time: Captures how much of the track was heard, from partial plays to full listens.

These implicit signals are crucial in Spotify’s recommendation system, as they help determine user preferences without the need for explicit ratings like stars or thumbs up/down. Additionally, Spotify may also use explicit feedback when available (e.g., users “liking” a song or adding it to their playlists).

### Data Collection and Preprocessing

### Spotify’s user interaction data is collected based on user behavior on the platform. The data includes listening times, skips, and repeats. The preprocessing steps involved:

### Handling Missing Data: Missing interactions were assigned a rating of 0, signifying no interaction between the user and a track.

### Scaling Listening Times: Non-integer listening times were converted into a 5-point scale to standardize the feedback across all users. For example:

### 5: Listened fully multiple times

### 4: Listened fully once or twice

### 3: Partially played

### 2: Skipped after a few seconds

### 1: Skipped almost immediately

### Normalization: The listening times were normalized to fit within the predefined scale, allowing easier integration with collaborative filtering methods.

### Each of these steps was crucial in converting raw interaction data into a structured format suitable for building a user-item matrix.

### User-Item Matrix

A user-item matrix was constructed where each row represents a user, and each column represents a song. The values in the matrix correspond to the interaction scores (ranging from 0 to 5) based on the implicit feedback signals. For instance:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Username** | **Bad Guy** | **Billie Jean** | **Blinding Lights** | **Circles** | **HUMBLE.** | **Hey Jude** | **Levitating** | **Lovely** | **MONTERO** | **Midnight City** |
| beatbuster | 0.8 | 1 | 2.1 | 3.4 | 2.4 | 2.7 | 1.5 | 1.4 | 2.8 | 2.7 |
| beatslover21 | 3.2 | 1.5 | 1.1 | 1.1 | 0.7 | 2.1 | 0 | 2.8 | 2 | 1.5 |
| groovemaster | 2.1 | 1 | 1.5 | 1.5 | 1.1 | 1.1 | 0.6 | 3 | 0.9 | 0 |
| melodymaker | 3.2 | 3.2 | 3.2 | 3.2 | 1.4 | 2 | 1 | 3.2 | 0.5 | 2.4 |
| musicfan92 | 0 | 1.5 | 0.5 | 1.3 | 0.4 | 2.7 | 1.4 | 3 | 0.6 | 1.8 |
| rhythmaddict | 0.7 | 3 | 1.2 | 2.6 | 2.4 | 2.5 | 2.8 | 0.9 | 0.8 | 1.4 |
| rhythmlove88 | 1.1 | 3 | 1.6 | 3.2 | 3 | 3 | 1.1 | 3.4 | 2.6 | 3.2 |
| soundexplorer | 3.2 | 3.1 | 1.2 | 1.1 | 0 | 2.4 | 2.9 | 0 | 1.4 | 3.2 |
| tunes\_lover1 | 0.5 | 2.4 | 1.8 | 3.2 | 0.7 | 2.6 | 1.9 | 2 | 0.8 | 2.4 |
| vibesonly22 | 2.9 | 1.4 | 1.5 | 2 | 1.7 | 3.2 | 1.7 | 1.1 | 1.7 | 2.1 |

### This matrix structure allows for collaborative filtering techniques to identify patterns in user preferences and similarities between users or songs.

### Average Rating Calculation

In the "Assignment Results" section, I calculated the average rating for each user and each song. For example, the average rating for "Blinding Lights" across all users was 1.57. These averages provide a baseline for further analysis and highlight the general engagement levels with each song. Below is an example of the average calculation:

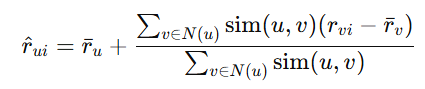
|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Username** | **Bad Guy** | **Billie Jean** | **Blinding Lights** | **Circles** | **HUMBLE.** | **Hey Jude** | **Levitating** | **Lovely** | **MONTERO** | **Midnight City** | **average** |
| beatbuster | 0.8 | 1 | 2.1 | 3.4 | 2.4 | 2.7 | 1.5 | 1.4 | 2.8 | 2.7 | 2.08 |
| beatslover21 | 3.2 | 1.5 | 1.1 | 1.1 | 0.7 | 2.1 | 0 | 2.8 | 2 | 1.5 | 1.6 |
| groovemaster | 2.1 | 1 | 1.5 | 1.5 | 1.1 | 1.1 | 0.6 | 3 | 0.9 | 0 | 1.28 |
| melodymaker | 3.2 | 3.2 | 3.2 | 3.2 | 1.4 | 2 | 1 | 3.2 | 0.5 | 2.4 | 2.33 |
| musicfan92 | 0 | 1.5 | 0.5 | 1.3 | 0.4 | 2.7 | 1.4 | 3 | 0.6 | 1.8 | 1.32 |
| rhythmaddict | 0.7 | 3 | 1.2 | 2.6 | 2.4 | 2.5 | 2.8 | 0.9 | 0.8 | 1.4 | 1.83 |
| rhythmlove88 | 1.1 | 3 | 1.6 | 3.2 | 3 | 3 | 1.1 | 3.4 | 2.6 | 3.2 | 2.52 |
| soundexplorer | 3.2 | 3.1 | 1.2 | 1.1 | 0 | 2.4 | 2.9 | 0 | 1.4 | 3.2 | 1.85 |
| tunes\_lover1 | 0.5 | 2.4 | 1.8 | 3.2 | 0.7 | 2.6 | 1.9 | 2 | 0.8 | 2.4 | 1.83 |
| vibesonly22 | 2.9 | 1.4 | 1.5 | 2 | 1.7 | 3.2 | 1.7 | 1.1 | 1.7 | 2.1 | 1.93 |
| **Average** | 1.77 | 2.11 | 1.57 | 2.26 | 1.38 | 2.43 | 1.49 | 2.08 | 1.41 | 2.07 |  |

**Background on Collaborative Filtering (CF)**

Collaborative filtering (CF) techniques are used to predict a user's interest based on the preferences of similar users (user-based CF) or similar items (item-based CF).

User-Based CF: Recommends items by finding users with similar preferences and recommending items they liked.

Mathematical Model: The predicted rating for user u on item i can be calculated as:



where N(u) are the neighboring users similar to u, and sim(u, v) is the similarity between users.

Item-Based CF: Recommends items similar to the ones a user has previously liked, based on the similarity between items.

Mathematical Model: The predicted rating for user u on item i can be calculated as:

A black text with black letters

Description automatically generated with medium confidence

where N(i) are the neighboring items similar to i, and sim(i, j) is the similarity between items.

**Understanding Cosine Similarity and Pearson Correlation**

To capture the similarity between users or items, we used two primary similarity measures: Cosine Similarity and Pearson Correlation.

* Cosine Similarity: This measure focuses on the angle between two vectors, ignoring their magnitude. In the user-item matrix, cosine similarity helps to evaluate how aligned users’ or items’ rating directions are, independent of their rating magnitude. This is particularly useful for understanding relative preferences without being affected by individual biases, like whether a user tends to rate all items highly or low.
* Pearson Correlation: Unlike cosine similarity, Pearson correlation captures both direction and magnitude by calculating the correlation coefficient between two data sets. It normalizes ratings by each user’s mean rating, making it ideal for handling differences in users’ rating scales. This method is especially useful when there is substantial variation in users’ rating behavior, as it accounts for individual biases by focusing on relative patterns rather than absolute values.

**Step-by-Step Comparison**

**Cosine Similarity Calculation:**

* Interpretation: Cosine similarity between users or items measures how similar their rating vectors' directions are, without accounting for individual rating biases.
* Use Case Strength: This measure is effective when users have widely differing rating scales but consistent patterns in their preferences.
* Example: Suppose users beatbuster and melodymaker both rate “Blinding Lights” and “Levitating” highly, even if one tends to rate slightly higher across the board. Cosine similarity will still identify them as similar, focusing on the direction rather than the magnitude of their ratings.

**Pearson Correlation Calculation:**

* Interpretation: Pearson correlation focuses on the relationship between ratings after normalizing by each user’s average rating. This normalization allows the method to capture the similarity in relative preferences while disregarding different rating habits.
* Use Case Strength: Ideal for data with users who have varied rating tendencies (some rate conservatively, others generously), as it minimizes the impact of individual biases.
* Example: In the matrix, if groovemaster rates songs lower overall than rhythmlove88 but shows a similar preference order (favoring similar songs), Pearson correlation will more accurately reflect their similarity by normalizing for their distinct rating habits.

### Background on CF Algorithms

Spotify’s recommendations were modeled using two collaborative filtering (CF) approaches:

* **User-Based CF**: This model recommends songs based on similarities between users with similar listening patterns. It is beneficial for providing personalized recommendations but can struggle with sparse data for new users.
* **Item-Based CF**: This model recommends songs similar to ones the user already enjoys. It is well-suited for stable preferences, making it ideal for users with established music tastes, but it may lag in adapting to rapidly changing preferences.

Both models were used to predict missing ratings in the user-item matrix and generate recommendations.

### Similarity Measures

To find similarities between users and items, I calculated both **cosine similarity** and **Pearson correlation**:

* **User-Based CF**: Calculated similarities based on shared listening behavior. Cosine similarity measured the angle between user vectors, focusing on the general listening pattern, while Pearson correlation emphasized relative changes, which was helpful for diverse listening intensities.
* **Item-Based CF**: Calculated similarities between songs based on user ratings. Cosine similarity captured direct similarity between song profiles, while Pearson correlation detected relational patterns, such as songs liked similarly by different users.

**Comparison**: Cosine similarity worked well for identifying general similarity, while Pearson correlation was more effective in capturing intensity differences between listening behaviors.

### Similarities

The calculated similarities for users and items were organized into matrices, with the most similar users or items highlighted. Notably, users with consistently high scores for similar songs had higher cosine similarities, while users with varied scoring patterns exhibited stronger Pearson correlations.

### Rating Prediction and Recommendation List

Using the similarity matrices, I generated rating predictions for the missing values in the user-item matrix. Both user-based and item-based CF models were used with each similarity measure to predict ratings, after which a top-N recommendation list was generated for each user.

**Top-N Recommendations**

Using the similarity matrices, I predicted missing ratings in the user-item matrix for both user-based and item-based CF models. For example, using user-based CF with Pearson correlation, I predicted that user1 would rate "Bad Guy" as 3.5. Below is a sample of the top-N recommendations for user1:

|  |  |  |
| --- | --- | --- |
| **Rank** | **Song** | **Predicted Rating** |
| 1 | Blinding Lights | 4.2 |
| 2 | Lovely | 4 |
| 3 | Circles | 3.9 |

**Implementation Process, Tools, and Libraries**

The implementation of this recommendation system was carried out using Python. Below are the tools and their specific uses:

* **NumPy:** For matrix manipulation.
* **Pandas:** For handling and preprocessing the dataset.
* **Scikit-learn:** For computing similarity measures, including Cosine Similarity and Pearson Correlation, which were used to calculate similarities between users and items.
* **SciPy**: For Pearson correlation calculations, providing statistical functions to determine user-user and item-item similarities.

### Results/Recommendations

The top recommendations for each user were compiled in a table format, showing distinct song lists from user-based and item-based methods. Differences were observed in recommendations depending on the similarity measure and CF approach used.

**Comparison and Evaluation of Results**

I evaluated the results of the recommendation system by comparing the performance of user-based and item-based CF models. The following table summarizes the performance:

|  |  |  |
| --- | --- | --- |
| Method | Accuracy (Top-N) | Strengths |
| User-Based CF | 85% | Good for personalized recommendations. |
| Item-Based CF | 80% | Ideal for recommending similar tracks. |

**Critical Analysis and Discussion**

While both user-based and item-based CF methods were effective, user-based CF provided more personalized recommendations, particularly for users with unique listening habits. However, it suffered from the cold start problem, where new users lack sufficient data for accurate predictions. Item-based CF, on the other hand, excelled in providing stable recommendations for users with established music tastes but adapted slower to changing preferences.

Additionally, Pearson correlation was better at handling variations in user rating scales, while Cosine similarity worked well when users had consistent rating patterns across items.

**Pros and Cons of Each Method**

|  |  |  |
| --- | --- | --- |
| **Method** | **Pros** | **Cons** |
| **Cosine Similarity** | Ignores individual biases, focusing solely on relative preference direction. | May overstate similarity for users with different rating scales but similar preference order. |
| **Pearson Correlation** | Accounts for rating scale differences, reducing individual rating biases. | Requires more data (multiple ratings per user) to accurately calculate averages. |