RS Module 2 Notes

Collaborative Filtering Based Recommendations

The main idea of collaborative recommendation approaches is to exploit information about the past behavior or the opinions of an existing user community for predicting which items the current user of the system will most probably like or be interested in. These types of systems are in widespread industrial use today, in particular as a tool in online retail sites to customize the content to the needs of a particular customer and to thereby promote additional items and increase sales.

Feature	User-Based Nearest Neighbor	Item-Based Nearest Neighbor
Core Idea	Recommends items based on similar users	Recommends items similar to those a user liked
Data Focus	User-user similarity	Item-item similarity
Cold-Start Problem	New users struggle to get recommendations	New items are hard to recommend
Diversity of Recommendations	Can provide more diverse recommendations	Less diverse, can lead to overspecialization
Explainability	Harder to explain (based on user behavior)	Easier to explain (based on item similarity)
Scalability	Scales poorly with many users	Scales better with many users, more scalable
Sparsity Problem	Suffers from sparsity in user-item interactions	Less affected but still vulnerable to sparsity
Popular Use Case	Social media platforms, e.g., Facebook	E-commerce platforms, e.g., Amazon
Computational Cost	Higher due to user-user similarity computation	Lower since item-item similarity is more stable

User-based Nearest Neighbor Recommendation

In **user-based collaborative filtering**, the system recommends items to a user based on the preferences of other similar users (neighbors). The key idea is that if two users have a similar taste or rated items similarly in the past, the items that one user liked are likely to be liked by the other.

Steps Involved:

1. **Calculate Similarity between Users**: Measure how similar two users are based on their past ratings. Common similarity measures include:
   - **Cosine Similarity**: Measures the cosine of the angle between two user rating vectors.
   - **Pearson Correlation**: Measures the linear correlation between the ratings of two users.

$$\text{Pearson Correlation}(A, B) = \frac{\sum_{i=1}^{n}(A_i - \bar{A})(B_i - \bar{B})}{\sqrt{\sum_{i=1}^{n}(A_i - \bar{A})^2}\sqrt{\sum_{i=1}^{n}(B_i - \bar{B})^2}}$$

   Interpretation
   A value close to 1 indicates a strong positive correlation (the users rate similarly).
   A value close to -1 indicates a strong negative correlation (the users rate inversely).
   A value of 0 indicates no linear relationship between the ratings.

   Example: If users A and B have rated many items similarly, they will have a high similarity score.
2. **Find Nearest Neighbors**: Identify users who have the highest similarity scores with the target user.
3. **Recommend Items**: Once the neighbors are found, the system recommends items that the similar users liked but the target user has not interacted with yet.
   Example: If User A liked Action and Drama movies and User B, who is similar to A, has not seen many action movies, the system will recommend Action movies to B.

Item-based Nearest Neighbor Recommendation

In **item-based collaborative filtering**, the focus is on finding relationships between items rather than users. The system recommends items that are similar to the items a user has liked in the past.

Steps Involved:

1. **Calculate Item Similarity**: Measure the similarity between items based on the users who have rated them. Common similarity measures include:
   - **Cosine Similarity**: Between item rating vectors.

$$\text{Cosine Similarity}(A, B) = \frac{\bar{A} \cdot \bar{B}}{\|A\|\|B\|}$$

   Interpretation:
   A value of 1 means the vectors (or users/items) are perfectly similar.

A value of 0 means no similarity.
A value of -1 means they are completely dissimilar.

- **Adjusted Cosine Similarity**: Adjusts for differences in user rating tendencies.

$$sim(a,b) = \frac{\sum_{u \in U}(r_{u,a} - \bar{r_u})(r_{u,b} - \bar{r_u})}{\sqrt{\sum_{u \in U}(r_{u,a} - \bar{r_u})^2}\sqrt{\sum_{u \in U}(r_{u,b} - \bar{r_u})^2}}$$

Example: If many users who liked movie A also liked movie B, these two movies will have a high similarity score.

2. **Find Nearest Neighbor Items**: For each item a user has rated highly, find other similar items (neighbors).
3. **Recommend Items**: Recommend the items that are most similar to the ones the user liked.
   Example: If a user has given high ratings to multiple action movies, the system will recommend other similar action movies.

Advantages:

- Item-based approaches are more stable than user-based methods because item similarity doesn't change as frequently as user behavior.

Model-based Collaborative Filtering

This approach uses machine learning models to make recommendations based on user-item interaction data. Instead of directly computing similarities, a model is trained to predict user preferences.

Common Techniques:

1. **Matrix Factorization (SVD - Singular Value Decomposition)**: The user-item interaction matrix is decomposed into two lower-dimensional matrices (one for users, one for items). These matrices capture latent factors that represent hidden features in user preferences and item characteristics.

$$R_{m \times n} \approx U_{m \times k} \times V^T_{n \times k}$$

   Where R is the original user-item rating matrix, and U and V are the matrices with latent features.
   Example: In a movie recommendation system, latent factors might capture concepts like genre or director preference, even if this information is not explicitly provided.
2. **K-Nearest Neighbor (KNN)**: This algorithm identifies the k most similar users or items based on the similarity metrics and then uses those for recommendations.
3. **Probabilistic Matrix Factorization (PMF)**: A probabilistic approach that assumes there is some underlying distribution governing user-item ratings.

Advantages of Model-Based Methods

1. They can capture complex patterns in data.
2. They work well for large-scale datasets and can better generalize than memory-based methods.

Pre-processing Based Approaches

Before applying collaborative filtering algorithms, pre-processing can enhance the data quality and algorithm performance.

1. **Data Normalization**: Adjust user ratings to account for biases. For instance, some users might consistently give high or low ratings. Normalizing the ratings can improve recommendation accuracy.
   Example: If a user gives high ratings to most items, their ratings could be normalized to reflect their relative preferences rather than absolute scores.
2. **Dimensionality Reduction**: Techniques like **Principal Component Analysis (PCA)** are used to reduce the number of features, helping with scalability and noise reduction.
3. **Handling Missing Values**: In a sparse matrix, where many ratings are missing, techniques such as imputation or matrix factorization can fill in the gaps.

Advantages and Drawbacks of Collaborative Filtering

Collaborative filtering has its own advantages and limitations over other filtering techniques, some of which are listed below:

Advantages of Collaborative Filtering:

1. **No Domain Knowledge Required**: Collaborative filtering does not rely on item features (like content-based filtering does). It purely uses user-item interaction data, such as ratings or clicks, meaning it can be applied across various domains without needing to understand the nature of the items (e.g., movies, books, products).
2. **Handles Complex User Behavior**: It captures the underlying preferences and behavior patterns of users by analyzing historical interactions. It can recommend items that a user might not have explicitly searched for but are likely to enjoy based on similar users or items.
3. **Discovering New Interests**: Collaborative filtering can help users discover new items by recommending items that similar users have liked. This can result in serendipitous recommendations, offering things outside of the user's normal preferences.
4. **Dynamic**: It adapts quickly to changing user preferences. As user behavior changes (e.g., new ratings or interactions), collaborative filtering models can be updated to reflect current tastes.
5. **Personalization**: The system generates highly personalized recommendations by learning from the specific behavior and preferences of each user, leading to better user

satisfaction.

Drawbacks of Collaborative Filtering:

1. **Cold-Start Problem**: Collaborative filtering requires sufficient data to make meaningful recommendations. When new users or items are introduced to the system (i.e., users with no interaction history or newly added items), the system lacks the data to generate recommendations, known as the **cold-start problem**.
2. **Sparsity**: In large datasets, user-item interactions tend to be sparse, meaning most users only interact with a small subset of items. This sparsity can make it difficult to find similar users or items, leading to poor recommendations.
3. **Scalability**: As the number of users and items grows, the computational complexity of finding nearest neighbors in user-based or item-based collaborative filtering can become a challenge. This requires significant computational resources and optimization for large-scale systems.
4. **Popularity Bias**: Collaborative filtering tends to recommend popular items that have been interacted with by many users. This can result in lesser-known items or niche products being underrepresented in recommendations, limiting diversity.
5. **Over-Specialization**: Recommendations might become too narrow, especially in **user-based filtering**. A user might keep receiving recommendations similar to what they've already interacted with, limiting exploration of new types of content.
6. **Data Privacy Concerns**: Collaborative filtering depends heavily on collecting and analyzing user behavior data. This raises privacy concerns, as sensitive user data like browsing history, clicks, and purchase history are used for generating recommendations.

Attacks on Collaborative Recommender Systems

Recommender systems can be vulnerable to different types of attacks, where malicious users attempt to manipulate the system’s outputs.

Types of Attacks:

Profile Injection Attacks (Shilling Attacks): Malicious users inject fake profiles with the aim of influencing recommendations. They can either:

1. **Push Attack**: Aim to promote certain items by giving them artificially high ratings.
2. **Nuke Attack**: Aim to demote certain items by giving them artificially low ratings.
   Example: In an e-commerce system, a seller might create fake user accounts to upvote their product, pushing it higher in the recommendations.

Types of Profile Injection Attacks:

1. **Random Attack:** In the random attack, all item ratings of the injected profile (except the target item rating, of course) are filled with random values drawn from a normal

distribution that is determined by the mean rating value and the standard deviation of all ratings in the database. The intuitive idea of this approach is that the generated profiles should contain “typical” ratings so they are considered as neighbors to many other real profiles.

2. **Average Attack:** A bit more sophisticated than the random attack is the average attack. In this method, the average rating per item is used to determine the rating values for the profile to be injected. Intuitively, the profiles that are generated based on this strategy should have more neighbors, as more details about the existing rating datasets are taken into account.
3. **Bandwagon Attack:** The bandwagon attack exploits additional, external knowledge about a rating database in a domain to increase the chances that the injected profiles have many neighbors. In nearly all domains in which recommenders are applied, there are “blockbusters” – very popular items that are liked by a larger number of users. The idea, therefore, is to inject profiles that – besides the high or low rating for the target items – contain only high rating values for very popular items.
4. **Segment Attack:** The rationale for segment attack is straightforwardly derived from the well-known marketing insight that promotional activities can be more effective when they are tailored to individual market segments. When designing an attack that aims to push item A, the problem is thus to identify a subset of the user community that is generally interested in items that are similar to A.
5. **Special Nuke Attack:** (Love/hate attack) In the corresponding attack profiles, the target item is given the minimum value, whereas some other randomly chosen items (those in the filler set) are given the highest possible rating value.

Other Types of Attacks:

Data Poisoning: This involves inserting misleading data into the training dataset to degrade the performance of the recommender system. Poisoned data skews the recommendations away from their true target.

Reverse Attacks: Malicious users might exploit the system to get biased recommendations for themselves by creating patterns in their own profile that the system might interpret as similar to high-rated items.

Defense Mechanisms

1. **Anomaly Detection**: Detecting and removing fake profiles or ratings that don't fit user behavior patterns.
2. **Robust Collaborative Filtering**: Developing algorithms that are resistant to attacks, such as those that use weighted averages or trust-based models to down-weight suspicious users.
3. **Trust-based Filtering**: Incorporating trust networks into recommendations, where users’ trust ratings of each other influence recommendations, helping to filter out malicious actors.

Summary

Collaborative filtering is one of the most widely used techniques in recommender systems. Whether it’s user-based or item-based, the system leverages the preferences of similar entities to make recommendations. Modern approaches often involve model-based techniques like matrix factorization, which captures latent patterns in the data. Additionally, addressing challenges like data sparsity, scalability, and vulnerabilities (such as shilling attacks) is critical to building robust and reliable recommender systems.