Recommendation Engines

Part-2

1. CONTENT-BASED FILTERING

Content or attribute-based recommenders use the explicit properties of an item (attributes) in addition to the past user-item interaction data as inputs for the recommenders. They operate under the assumption that the items with similar properties have similar ratings at the user level.

The general framework of a content-based recommendation engine is shown in Fig. The model consumes ratings matrix and item profiles. The output of the model either fills the entire ratings matrix or provides just the top item recommendation for each user.

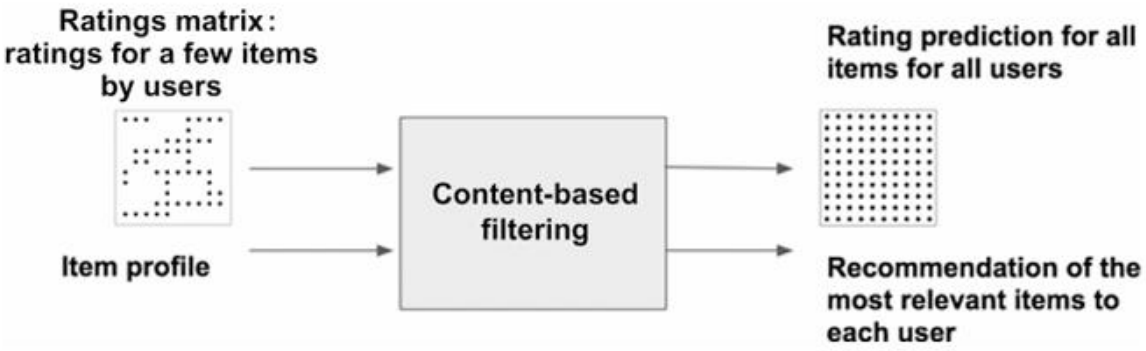

FIGURE Model for content-based recommendation engine.

Predicting ratings using a content-based recommendation method involves two steps.

The first step is to build a good item profile. Each item in the cata log can be represented as a vector of its profile attributes.

The second step is to extract the recommendations from the item profile and ratings matrix. There are two distinct methods used for extracting recommendations: a user profile-based approach and a supervised learning-based approach.

2.Building an Item Profile

An item profile is a set features or discrete characteristics about an item in the form of a matrix. Features, also called attributes, provide a description of an item. Each item can be considered as a vector against the set of attributes.

In case of the books, the attributes may be the publisher, author, genre, sub genre, etc.

In the case of movies, the attributes may be individual cast members, year, genre, director, producer, etc.

A matrix can be built with columns as the universe of attributes for all the items where each row is a distinct item. The cells can be Boolean flags indicating if the item is associated with the attribute or not.

Table shows a sample item profile or item feature matrix for the movies with the attributes as cast, directors, genre, etc.

Table 11.8 Item Profile

Movie	Tom Hanks	Helen Miren	...	Joel Coen	Kathryn Bigelow	...	Romantic	Action
Fargo				1				
Forrest Gump	1							
Queen		1						
Sleepless in Seattle	1						1	
Eye in the Sky		1						1

The item profile can be sourced from the providers of the item (e.g., product sellers in an e-commerce platform) or from third-party metadata providers (IMDB has metadata on a vast selection of movies). The item description provides a wealth of features about the products.

Text Mining relevant tools like term frequency-inverse document frequency (TF-IDF) to extract features from documents such as item description. If the items are news articles in an online news portal, text mining is used to extract features from news articles. In this case, the item profile contains important words in the columns and the cells of the matrix indicate whether the words appear in the document of the individual items.

Once the item profile is assembled, the recommendations can be extracted using either a user profile computation approach or a super vised learning approach.

3 User Profile Computation

The user profile approach computes the user-item preference by building a user feature matrix in addition to the item feature matrix. The user feature matrix or the user profile maps the user preference to the same features used in the item feature matrix, thereby, measuring the strength of preference of the user to the features. Just like item profile vector, a user profile can be represented in a vector form in the feature space. The proximity of the user and the item vector indicates the strength of preference of the items to the users.

The user profile is built from the combination of the item profile and the known ratings matrix. Suppose R is the ratings matrix with m users and n items. I is the item profile matrix with n items and f features or attributes. The extracted user profile will be the matrix U with m users and exactly the same f features from the item profile. Fig. shows the visual representation of the matrix operation.

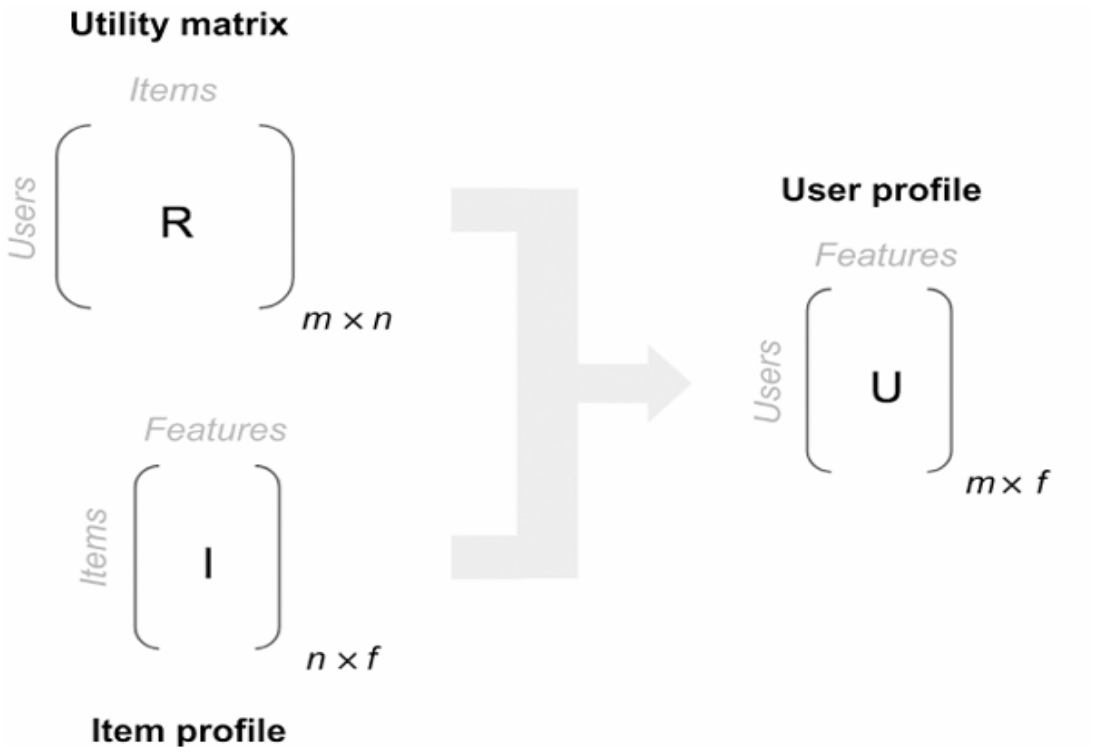

FIGURE User profile from utility matrix and item profile

However, the content based method needs more consistent information about each item to make meaningful recommendations to the users. The content-based recommenders do not entirely address the cold start problem for new users. Some information is still needed on the new user's item preferences to make a recommendation for new users.

How It Works

1. **Feature Extraction**
   - Each item (e.g., movies, books, or products) is described using features.
   - Example: A movie might have features like genre, director, actors, and keywords.
2. **User Profile Creation**
   - The system builds a user profile based on past interactions.
   - Example: If a user watches many action movies, their profile will have a preference for action-related features.
3. **Similarity Calculation**
   - When recommending new items, the system compares their features with the user's preferences.
   - This can be done using techniques like:
     - **Cosine Similarity** (measures similarity between vectors)
     - **TF-IDF (Term Frequency-Inverse Document Frequency)** (common in text-based recommendations)
     - **Neural Networks or Deep Learning** (for advanced systems)
4. **Recommendation Generation**
   - The system ranks items based on similarity scores and suggests the most relevant ones.

Let's take a numerical example using **Tollywood (Telugu) movies** and a **Content-Based Filtering** approach based on genres.

Step 1: Movie Dataset (Feature Matrix)

We consider five Tollywood movies with three genre features: **Action, Comedy, and Drama** (values represent the presence of the genre).

Movie	Action	Comedy	Drama
RRR	1.0	0.2	0.3
Pushpa	0.9	0.1	0.4
Jathi Ratnalu	0.1	1.0	0.5
Arjun Reddy	0.2	0.1	1.0
Baahubali	1.0	0.3	0.2

Step 2: User Profile

Let's say a user watched **RRR** and **Pushpa**. We take the average of their feature vectors:

$$\text{User Profile} = \frac{(1.0, 0.2, 0.3) + (0.9, 0.1, 0.4)}{2} = (0.95, 0.15, 0.35)$$

User Profile = (0.95, 0.15, 0.35)

Step 3: Compute Similarity (Cosine Similarity)

We compute **cosine similarity** between the user profile and other movies to find recommendations.

$$\text{Cosine Similarity} = \frac{A \cdot B}{||A|| \times ||B||}$$

Similarity Calculation

For **Jathi Ratnalu** (0.1, 1.0, 0.5):

$$\cos(\theta) = \frac{(0.95 \times 0.1) + (0.15 \times 1.0) + (0.35 \times 0.5)}{\sqrt{(0.95^2 + 0.15^2 + 0.35^2)} \times \sqrt{(0.1^2 + 1.0^2 + 0.5^2)}}$$

= 0.3656

Similarly, we compute for **Arjun Reddy** and **Baahubali**.

- Baahubali similarity = **0.9789**
- Arjun **Reddy** similarity = **0.5292**

Step 4: Ranking & Recommendations

Sorting by similarity:

1. **Baahubali** (0.9789) → Highest similarity
2. **Arjun Reddy** (0.5292)
3. **Jathi Ratnalu** (0.3656)

Thus, the system **recommends "Baahubali" as the top choice** for the user.

Advantages

✓ Works well even with few users (no cold-start issue for items).
✓ Personalized recommendations based on user preferences.
✓ No need for data from other users.

Disadvantages

✗ Struggles with new users (cold-start problem).
✗ May lead to a "filter bubble" (limited diversity in recommendations).
✗ Requires good feature engineering.

SUPERVISED LEARNING MODELS

A supervised learning model-based recommender approaches the problem of user-item preference prediction at the individual user level. If a user has expressed interest in a few items and if those items have features, then the interest of the users to those features can be inferred.

In the supervised learning model based approach, each user has a personalized decision tree.

The item profile matrix can be customized just for one user, say Olivia, by introducing a new column in the item profile to indicate whether Olivia likes the movie. This yields the item profile matrix for one user (Olivia) shown in Table.

Table 11.12 Item Profile With Class Label for One User

Movie	Tom Hanks	Helen Mirren	...	Joel Coen	Kathryn Bigelow	...	Romantic	Action	Class label for Olivia
Fargo				1					1
Forrest Gump	1								0
Queen		1							1
Sleepless in Seattle	1						1		0
Eye in the Sky		1						1	1

The classification model, say a decision tree, can be built by learning the attribute preferences for Olivia and the model can be applied to the catalog for all the movies not seen by Oliva.

Suppose one has a straightforward preference for movies: they only like it if the movie has Helen Mirren in the cast or is directed by the Coen brothers. Their personalized decision tree would be like the one in Fig.

The decision tree shown in Fig. is a classification tree for the user Oliva using the item profile shown in Table. For another user, the tree would be different.

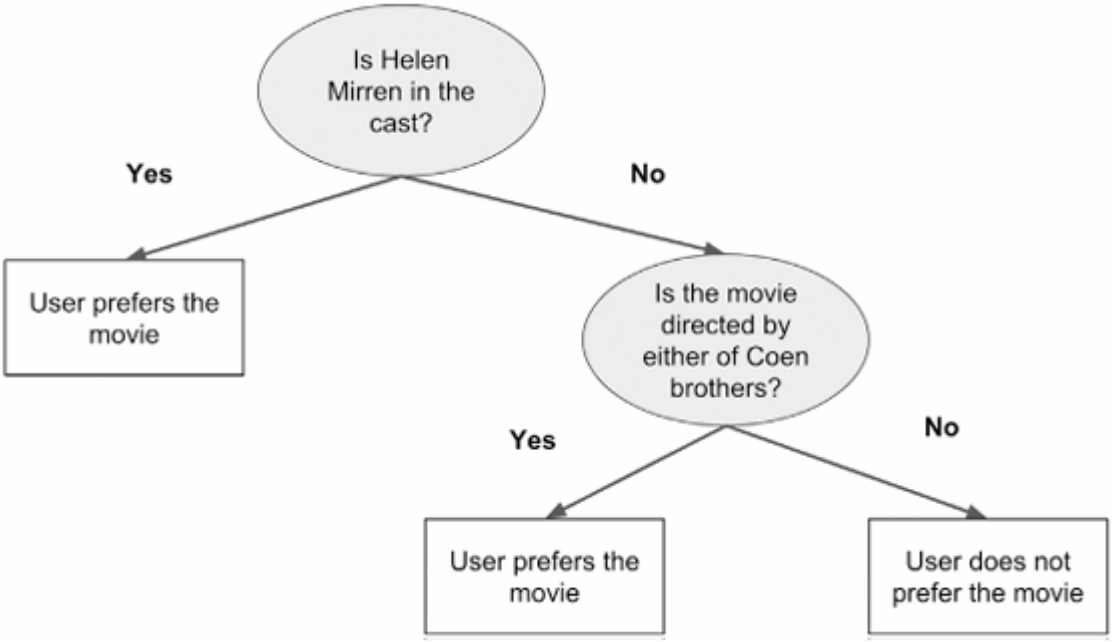

FIGURE Personalized decision tree for one user in the system.

HYBRID RECOMMENDERS

Each recommendation model has its own strengths and limitations, that is, each works better in specific data setting than others. Some recommenders are robust at handling the cold start problem, have model biases, and tend to overfit the training dataset.

As in the case of the ensemble classifiers, hybrid recommenders combine the model output of multiple base recommenders into one hybrid recommender. As long as the base models are independent, this approach limits the generalization error, improves the performance of the recommender, and overcomes the limitation of a single recommendation technique.

Comparison of the Types of Recommendation Engines

Table 11.13 Comparison of Recommendation Engines

Type	Inputs	Assumption	Approach	Advantages	Limitations	Use Cases
Collaborative filtering	Ratings matrix with user-item preferences	Similar users or items have similar likes	Derives ratings from like-minded users (or items) and the corresponding known user-item interactions	The only input needed is the ratings matrix. Domain agnostic. More accurate than content-based filtering in most applications	Cold start problem for new users and items. Sparse entries in the rating matrix leads to poor coverage and recommendations. Computation grows linearly with the number of items and users.	eCommerce, music, new connection recommendations from Amazon, Last.fm, Spotify, LinkedIn, and Twitter
User-based CF (neighborhood)	Ratings matrix	Similar users rate items similarly	Finds a cohort of users who have provided similar ratings. Derives the outcome rating from the cohort users	User-to-user similarity can be pre-computed		
Item-based CF (neighborhood)	Ratings matrix	Users prefer items that are similar to previously preferred items	Finds a cohort of items which have been given similar ratings by the same users. Derives rating from cohort items	More accurate than user-based CF		
Latent matrix factorization	Ratings matrix	User's preference of an item can be better explained by their preference of an item's characteristics	Decomposes the User-Item matrix into two matrices (P and Q) with latent factors. Fills the blank values in the ratings matrix by dot product of P and Q	Works in sparse matrix. More accurate than neighborhood-based collaborative filtering.	Cannot explain why the prediction is made	
Content-based filtering	User-item rating matrix and item profile	Recommends items similar to those the user liked in the past	Abstracts the features of the item and builds an item profile. Uses the item profile to evaluate the user preference for the attributes in the item profile	Addresses cold start problem for new items and new users. Can provide explanations on why the recommendation is made.	Requires item profile dataset in addition to the ratings matrix. Recommenders are domain specific. Popular items skew the results	Music recommendation from Pandora and CiteSeer's citation indexing
User profile based	User-item rating matrix and item profile	User's preference for an item can be expressed by their preference for an item attribute	Builds a user profile with the same attributes as the item profile. Computes the rating based on similarity of the user profile and the item profile	Provides descriptive recommendations.		
Supervised learning model based	User-item rating matrix and Item profile	Every time a user prefers an item, it is a vote of preference for the item's attributes	A personalized classification or regression model for every single user in the system. Learns a classifier based on user likes or dislikes of an item and its relationship with the item attributes	Every user has a separate model and could be independently customized.		