CHAPTER 11

Recommendation Engines

Now a days, everyone experiences the increasing impact of products and applications that utilize machine learning. Out of all the applications of machine learning, recommendation engine techniques underwrite some of the most prolific utilities in everyday experience. Recommendation engines are a class of machine learning techniques that predict *a user* preference for an *item*. A user is a customer or a prospective customer of an item or a product. In essence, the purpose of an enterprise is to connect a customer to a product that the customer wants. Recommendation engines influence this important connection between the customer and the product. Hence, recommendation engines play a cardinal role in how a relevant product is introduced to a customer. Whether it is a snippet of news feed on a social media homepage, what to watch next in a streaming service, suggested article in an online news portal, a new friend recommendation, or a suggested product to buy from an e-commerce site, the influence of automated recommendation engines are inescapable in today's digital economy.

WHY DO WE NEED RECOMMENDATION ENGINES?

A brick and mortar store has a limitation of space which dictates the number of unique products it can hold. An enterprise with a few dozen products, by design, has a limited number of choices for its customers. In either case, it is relatively easy to predict what available products the customers want. It is common to see retailers recommending their top selling products to their customers. To capitalize on what readers might commonly buy, bookstores often display the top selling books, new releases, or store recommendations at the front of the store. Newspapers run top stories in politics, business, sports, and entertainment on the front page, anticipating they would be the most likely read articles. These tactics became the recommendations or product predictions for customer consumption as a whole. In this approach, there is one common recommendation list for *all* the customers. It served well for the customers because most of them purchased top selling products (and,

Data Science. DOI: https://doi.org/10.1016/B978-0-12-814761-0.00011-3

hence, they are in best sellers!) and served the enterprise well because they made it easy for customers to purchase the top sellers.

The legacy recommendation tactics that served the brick and mortar settings do not take into account an *individual* customer preference. After all, not all the customers are the same. Some may prefer business books, but not mystery novels, some may like to read sports news and may not really care about current affairs or politics. At a deeper level, none of the users are the same and everyone has their own unique interests and preferences of the products they want to consume. It would be ideal to have a unique recommendation list for every user, taking into consideration their unique interests and wants. The customers and the enterprises alike want a personalized storefront to maximize the positive experience of customer-product interactions. Both the editor-curated and best-selling lists are good global approximations of what customers would purchase. However, both the approaches do not fit the preferences of an individual user, unless the user is the prototypical average user.

The advent of online businesses greatly expanded the access of the product selection an enterprise can offer. In the span of a few years, customers could have instant access to half a million movies on a streaming service, millions of products on an e-commerce platform, millions of news articles and user generated content. Shelf space is not a constraint anymore and there is an abundance of unique products at the customer's disposal. However, not all products are consumed at the same rate. There are blockbuster movies and there is a long tail of obscure movies that are consumed less frequently. The long tail of products tend to serve customers with unique interests. A customer who is interested in thrillers might have watched the movie *The Girl with the Dragon Tattoo*, a Swedish-American psychological thriller from 2011. Given that the customer watched *The Girl with the Dragon Tattoo*, the customer might be extremely interested in an obscure subgenre of *Scandinavian Crime Fiction* and discovers titles like *Headhunters*, *The Killing*, *The Bridge*, etc. This pattern of consumption is made possible when there is an abundant selection of products that can be consumed instantly. It is also a scenario where the one size fits all approach of the best seller lists and common editorial staff picks are insufficient. The more customer-product connections an enterprise makes, the more it can keep the customer engaged. Moreover, the long tail products tend to be less expensive than licensing the best seller products. In those cases, it will be more profitable for the business to keep the customer engaged in the long tail products.

Search engines offer a good solution to discover items in the long tail only if the user knows what to look for. The information retrieval system works if the customer types in a specific movie title, like "Headhunters" or in some

cases, if the information retrieval system is capable, users can search for "Scandinavian crime fiction" and get a list of movies. But, what if there is no such thing as a Scandinavian crime fiction subgenre or if the user is unaware of such a genre? There is nothing recommended when the credits roll.

Most customers are best served by their personalized storefronts where they can discover the most relevant products for them, out of the millions of available products. If an enterprise can predict what the user is going to consume next, it will be serving both the customers and itself. Recommendation engines make the personalized experience happen.

APPLICATIONS OF RECOMMENDATION ENGINES

Recommendation engines have become an indispensable part of online, and in some cases offline, experiences. It is difficult to escape the reach of the applications that utilize recommendation engines every day. These models are behind some of the familiar phrases in digital life: friends you may know, who to follow, recommended connections, news feeds, tweet feeds, customers who bought this item also bought, featured products, suggested dates, you may also like, related stories, recommended movies, what to watch next, etc. Recommendation engines power these online experiences (Underwood, 2017). Some of the applications of the recommenders are:

Product recommendations: An e-commerce website or any organization that has thousands of products, use product recommendation engines. It is one of the earliest applications of recommenders in the digital realm (Schafer, Konstan, & Riedl, 2001). The recommended listings, which are predicted by the company as the most relevant products for a specific customer, are shown in the homepage, product pages, shopping cart, checkout, and even in the order confirmation page. The recommendation list is based on past purchases or past clickstream data of the customer. E-commerce companies often complement the list with the best seller lists, new products, and curated featured products. Once the customer is in a product page, assuming the customer is interested in the product, a new modified recommendation list is shown as "related products" or as "other customer also bought this," lists. Similar features can be extended to offline enterprises, to an extent. Customizable marketing offers, customer relevant coupon codes, and dynamic call center scripts are some of the ways recommendation engines are used in offline experiences.

Content recommendation: A personalized news feed is one way to increase the stickiness of content sites. Whether it is a desktop, mobile, or any other device, users have limited time to consume the content. Content

companies are motivated to keep users on the site by feeding them the most relevant news stories, a friend's social update, or a tweet. All these recommendations are based on the past behavior of the user and the behavior of other users on the site (Thorson, 2008).

Entertainment recommendation: Binge watching as a phenomenon was unknown just a few years ago. Digital consumption of entertainment products has become the predominant way the media is consumed in many segments. Music streaming providers have a vast collection of tracks and cater to a user's unique taste in music. They can recommend a new artist's track to a listener because it shares the same traits as the track *For What it is Worth* by Buffalo Springfield that has already been *liked* by the listener.

In 2006, when Netflix Inc. announced Netflix Prize, a public competition to award the best recommendation engine model that could beat the company's own recommender—CineMatch, research into recommendation engines came into the spotlight. For its part Netflix released about half a million anonymized user ratings of 17,000 movies. Movies are rated with a 1−5-star scale by the users, presumably after watching the movie. Even though, recommendation engines existed before this, new research began into this problem of predicting user-product interactions (Koren, Bell, & Volinsky, 2009).

11.1 RECOMMENDATION ENGINE CONCEPTS

Whatever the application may be, a recommendation engine should predict a customer's interest in a particular product and provide highly personalized recommendations to the customer. To do this, the recommendation engine needs some information about the customer, the product and the customer-product preference. The input information for the recommendation engine takes the form of the past inclinations of the customer's preference of the product, for example, the movie ratings by a customer of the content streaming service. The customer discloses this rating information with an implicit understanding that the company will keep offering them highly relevant titles and with the satisfaction of their voice being heard. The company uses this data point and the data points collected from other users to feed into the recommendation engine. They invest in recommendation engines in an effort to keep offering customers highly personalized content, keep customers engaged, and maintain the trust of the customers for giving them information about the products they like.

The general model of a recommendation engine is shown in Fig. 11.1. Throughout this chapter, the customer will be referred to as the *user* and the

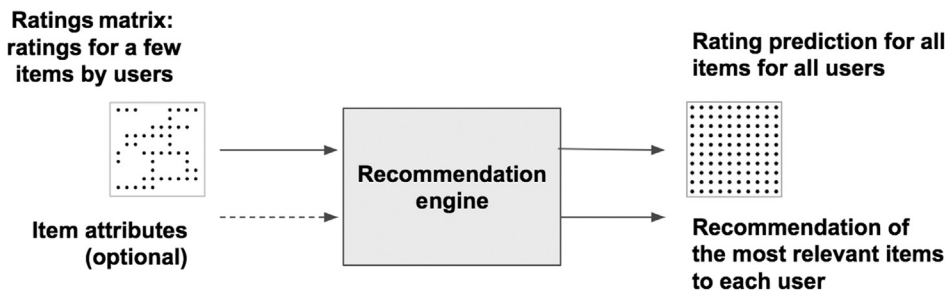

FIGURE 11.1
The general model of a recommendation engine.

Table 11.1 Known Ratings Matrix

	The Godfather	2001: A Space Odyssey	The Hunt for Red October	Fargo	The Imitation Game	...
Josephine	5	4	1		1	
Olivia			2	2	4	
Amelia	5	1	4	4	1	
Zoe	2	2	5	1	1	
Alanna	5	5		1	1	
Kim		4	1	2	5	
...						

product as an *item*. An item can be a product, movie, track, album, artist, topic, news article, etc. The user-item preference is quantified in the form of *ratings*. The ratings can be of any ordinal scale: 1−5-star ratings, scores, like or dislike flags, etc.

The most common input format for a recommendation engine is the *Utility matrix* or *Ratings matrix*. It is a simple two-dimensional crosstab with items in the columns, users in the rows and ratings in the cells. A few cells in the ratings matrix have rating values which indicate the user's preference for an item. Table 11.1 shows the ratings matrix with the user Josephine rating the movie *The Godfather* as 5 stars, on a 1−5-star scale. Alternatively, the cells in the ratings matrix may indicate a Boolean flag, whether or not a user has purchased, viewed, or searched for an item. In the ratings matrix with rating data, the blank cells may represent a few possibilities. Most likely the reason is that the user has not used or experienced the item yet. Other reasons can be that the user has not bothered to give a rating after using the item. In either case, the lack of rating cannot be assumed to indicate that the user disliked the item.

It is lucky to see so many ratings for the items by the users in the ratings matrix shown in Table 11.1. In reality, however, most of the cells would be empty in a ratings matrix. Say there is an "Etsy" style e-commerce company with 500,000 items and a couple of million users. Each user can give a 1−5-star rating for a product they are familiar with. If all the users rate all the items, that would be 10^{12} possible ratings! How many distinct products would a typical user buy? Perhaps 100. Out of which, how many items would a typical user rate? Perhaps 10. The remaining of 499,990 items for the user will have a blank cell in the ratings matrix. A sparse matrix provides both a challenge and an opportunity for recommendation engines. The model would have to sift through the user's potential preference in the other 499,990 items and provide a list of a dozen or so items the user would be most interested in. *The objective of the recommendation engine is to fill the blank cells of the ratings matrix with predicted ratings for each user and the item, which the user presumably hasn't used or experienced yet.*

The association analysis technique discussed in Chapter 6, Association Analysis is similar in approach to a recommendation engine. After all, both the techniques recommend items to a user and both employ machine learning algorithms. However, the results of the association analysis technique are not personalized to a specific user. If the association analysis model recommends beer the moment diapers are placed in the user's shopping basket, this same recommendation would be served to *all* users, whether or not a user liked beer or ever even bought beer in the past. It is a global rule-based recommendation for all users. Moreover, in the association analysis model, the session window of consideration is restricted to the current visit to the supermarket. The purchase history of the individual user in previous visits is not taken into account. In the case of recommendation engines, past ratings or purchase history of a specific user is central to building future user-item recommendations. The past history data is not readily accessible when a user enters a supermarket or is visiting a website as an anonymous user. In these cases, the global recommendation by association analysis provides a good approximation of user-item preferences.

Building up the Ratings Matrix

The ratings matrix shown in Table 11.1 can be represented with a ratings function f.

$$\textit{Ratings functionf}: U \times I \rightarrow R \tag{11.1}$$

where U is the set of users, I is the item set and R is the ratings matrix. The initial *known* ratings in matrix R is sparse because the user-item interaction is miniscule compared to all the possible item-user combinations. The objective

of the recommender is to fill the blank cells with predicted *unknown* ratings. The process of building a completed ratings matrix consists of three steps.

1. Assemble known ratings
2. Predict unknown ratings
3. Prediction evaluation

Step 1: Assemble Known Ratings

Collecting and assembling known ratings in the form of a ratings matrix is the most time-consuming step, riddled with data capture challenges. The ratings can be explicit, where the user directly provides ratings about an item. The user can be offered a list of previously consumed items or be provided with choices to select their favorite items out of a pre-selected list. The choice list is helpful to seed some ratings when a user is new to the service. The ratings could also be derived from the wish-lists where the user has expressed interest in an item. The challenge in the explicit rating is that it is not scalable. A typical user is not good at voluntarily providing a rating. How often does one leave a rating for a product that has been purchased or seen? The typical response rate is very low, in the range of 1%−10% (Weise, 2017). This limits the amount of data that can be used to build a reliable recommendation engine.

The implicit collection of ratings provides an alternative to, or complements, explicit data collection. Implicit data collection is inferred preference of an item from the past user actions, for example, searching for an item, previous purchases (not ratings), watching a movie, product views, etc. While one can get an abundance of data about user-item interactions using implicit collection, the challenge is that it is very difficult to interpret low or high ratings from implicit action. The user might have seen the movie and completely detested the movie. It is incorrect to assume that a user liked the movie just because the user watched it. A hybrid combination of both explicit and implicit rating is used in large scale production deployments to overcome the limitations of individual methods (Hu, Koren, & Volinsky, 2008).

Regardless of either explicit or implicit data collection, there is no available known rating for a new user or a new item. Hence, providing relevant recommendations is difficult when a new user or new item is introduced into the system. This is called the *cold start* problem. The recommenders cannot conclude any inferences from users or items when the system has not obtained sufficient data about them. Cold start is a major issue to recommenders because new items and users are continuously added to the system. Some of the algorithms used to build a recommendation engine discussed in this chapter are highly susceptible to dealing with the cold start problem, while others are robust.

Step 2: Rating Prediction

Predicting ratings for prospective user-item interaction is the key objective for a recommendation engine. In some cases, the objective is to show 1−5-star ratings for all the movies in the catalog indicating how much a user will prefer a movie. In most cases, however, it is not essential to predict ratings for *all* the user-item interactions. The objective is just to show only the *top* recommended items from the massive catalog of movies. Hence, it might be adequate to show items with top predicted ratings instead of all the items with the predicted ratings.

Step 3: Evaluation

Any machine learning-based prediction model needs to go through an evaluation step to check if the prediction is generalized and the model is not overfitting the training dataset. Some of the known rating data can be reserved as a test dataset to evaluate the training prediction. The performance of the recommendation engine is often measured as RMSE (root mean square error) or MAE (mean absolute error) of ratings, which measures the delta between the actual ratings and the predicted ratings. RMSE metric is penalized for larger errors, while MAE has the advantage of easy interpretation. The accuracy of the deployed recommendation engine has to be continuously measured whenever a new known rating is captured in the system. The new actual rating is compared against the previous prediction of the recommender.

The Balance

A good recommendation engine has reasonable accuracy of predicted ratings, balanced with a few other secondary objectives. Sometimes a user might not have discovered a particular genre and could benefit from serendipitous discovery of an item appearing on the recommendation list. A new popular movie might not have a user's favorite cast or director but might be worth recommending because the movie is popular. The recommendation engine accentuates a phenomenon called the *Filter bubble*, where users are cocooned with a singular viewpoint (Bozdag, 2013). It is not uncommon to find recommended articles leaning towards one political affiliation or all the recommended movies from one genre, because the user happened to sample a particular news article or watched a movie from a specific genre. The same user might like a different genre, say classic drama, had the user been provided with an opportunity to watch it. Providing a mix of diverse selection and enabling serendipitous discovery might complement the recommendation engine's predictions on user-item preferences. Companies deal with the

filter bubble problem by creating hybrid lists like new, noteworthy, or popular items along with the listing from recommendation engines.

11.1.1 Types of Recommendation Engines

The machine learning methods for building a recommendation engine can be classified into a taxonomy shown in Fig. 11.2. There is a wide selection of algorithms for building a recommendation engine, each with its own strengths and limitations (Bobadilla, Ortega, Hernando, & Gutiérrez, 2013). Association analysis is included as a type of recommendation engine method in this taxonomy, but it is distinct because of the absence of user personalization and the window of data it considers for a recommendation (Cakir & Aras, 2012). Recommendation engine methods are broadly classified into *Collaborative Filtering* and *Content-based Filtering*.

Collaborative filtering: Collaborative filtering is a class of recommenders that leverage only the past user-item interactions in the form of a ratings matrix. It operates under the assumption that similar users will have similar likes. It uses rating information from all *other* users to provide predictions for a user-item interaction and, thereby, whittles down the item choices for the users, from the complete item set. Hence, the name collaborative filtering. The collaborative filtering techniques can be further classified into:

1. ***Neighborhood methods***: Neighborhood methods predict the user-item preferences by first finding a cohort of users or items which are similar to the user or the item whose ratings are to be predicted. In the case of

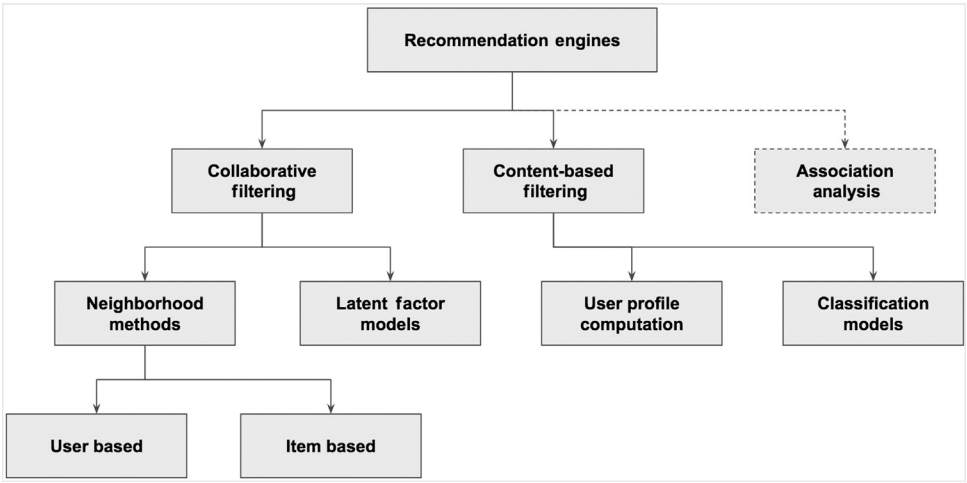

FIGURE 11.2
Taxonomy of recommendation engines.

the *user-based* neighborhood method, the algorithm is designed to find a cohort of users who have a similar rating pattern to the user in question and who have rated the item in question. The predicted rating is deduced from those cohort of similar users. The case is the same with the *item-based* neighborhood method where a cohort of similar items is found which have been given similar ratings when rated by the same user. The rating of similar items approximates the rating of the item in question for a user. Both these techniques leverage the similarity scores discussed in the k-Nearest Neighbor section in Chapter 4, Classification.

2. ***Latent factor models:*** Actively managed mutual fund managers look for suitable companies to invest in their pursuit to outperform the market averages. Out of hundreds of thousands of publicly traded companies, the fund managers select dozens of companies using certain criteria or factors. The factors can be management stability, market positioning, strength of the balance sheet, cash flow metrics, or market share of the products. They even explain the preference of the dozens of companies where they have taken positions using the factors. Factors are the same generalized dimensions that the company is ranked against (Company *Acme Inc*. has products with high market share) and the fund manager's preference for the factor (The fund *Cash cow* looks for companies with high market share). The latent factor models tend to explain the ratings by representing both the users and the items on a set of common factors. This set of factors is inferred from the known user-item ratings matrix. The factors can take the form of interpretable dimensions like "science-fiction movies with aliens" or uninterpretable dimensions. It maps both the users and items against the same set of factors. For example, if a science-fiction movie has a plot involving aliens and if a user has a preference for movies with aliens, a recommendation is made. Both the movie and the user are mapped with a degree of association with the factor—aliens.

Content-based filtering: In addition to the ratings matrix used by collaborative filtering methods, content-based filtering leverages *attribute* data about items. For example, it may use all the available information about the contents of a movie (item): the cast, the producer, the director, released year, genre, description, etc., in addition to the user-item ratings matrix. Content-based filtering uses the information about an item in the format of an *item profile.* After the item profile is sourced, the content-based recommenders predict the user-item preference with two different approaches:

1. ***User profile method:*** Once an item profile is developed, a user profile can be developed to show the affinity of a user to the attribute used in the item profile. For example, if a user has given high ratings for the

movies *Fargo*, *The Big Lebowski*, and *No Country for Old Men*, the user profile can be built to show the preference for the *Coen brothers'* (filmmakers, Joel Coen and Ethan Coen) work, who directed all the aforementioned movies. This technique is leveraged by user profile methods. The predicted rating for a user-item is computed by the similarity of the user and the item profiles. Since the movie *True Grit* is associated with the *Coen brothers* in the item profile and the user has shown an association with the *Coen brothers* in the user profile, a user-item recommendation can be made.

2. ***Supervised learning (Classification or Regression) models:*** Consider the case of a user who has shown explicit preference for a few movies by providing ratings. The item profile of the movies has a set of attributes, for example, the cast, the director, genre, key words in the plot description, etc., for all the items. If these two datasets were joined viz., the ratings matrix for *one user* and the item profile using the common link—*the item*, then movie ratings (response or label) and item attributes (input or predictor) are obtained for the user. The joined dataset is similar to the training data used to create classification or regression models. With the item profile and ratings matrix, it is possible to develop a classification or a regression model, say a decision tree, for *each* user and predict the rating for an unseen item using its attributes. By the supervised learning method, a personalized classification or regression model is built for every single user in the system. The model scores the items based on the user's affinity towards specific item attributes.

11.2 COLLABORATIVE FILTERING

Collaborative Filtering is based on a simple idea—a user will prefer an item if it is recommended by their like-minded friends. Suppose there is a new restaurant in town. If a friend, who happens to have the same interests as the user, raves about it—the user might like it as well. The idea makes intuitive sense. Collaborative filtering leverages this insight into an algorithm that predicts a user's preference of an item by finding a cohort of users who happen to have the same preference as the user. The preference or predicted rating for an item can be deduced by the rating given by a cohort of similar users.

The distinguishing feature of the collaborative filtering method is that the algorithm considers only the ratings matrix that is, the past user-item interactions. Hence, the collaborative filtering method is item domain independent. The same algorithm can be applied to predicting ratings for movies, music, books, and gardening tools. In fact, the algorithm does not have any knowledge about the items except the ratings given by the users.

Some of the real-world applications of collaborative filtering include Google News recommendations (Das, Datar, Garg, & Rajaram, 2007), music recommendations by Last.fm, and recommended Twitter follows (Gupta et al., 2013). The general framework for collaborative filtering is shown in Fig. 11.3. There are two distinct approaches in processing the ratings matrix and extracting the rating prediction. The neighborhood method finds a cohort of similar users or items. The latent factor method explains the ratings matrix through a set of dimensions called latent factors. *Matrix factorization* is the common realization of the latent factor method, where both users and items are mapped to a common set of self-extracted factors.

11.2.1 Neighborhood-Based Methods

Neighborhood methods calculate how similar the users (or the items) are in the known ratings matrix. The similarity scores or the measure of proximity, discussed in the k-Nearest Neighbor section in Chapter 4, Classification techniques, are used in neighborhood-based methods to identify similar users and items. Commonly used similarity measures are: Jaccard similarity, Cosine similarity, and Pearson correlation coefficient.

The general approach for neighborhood-based methods consists of two steps to find the predicted rating for a user-item:

1. Find the cohort of other similar users (or the items) who have rated the item (or the user) in question.
2. Deduce the rating from the ratings of similar users (or the items).

The two methods for the neighborhood-based systems, that is, user-based and item-based, are extremely similar. The former starts to identify similar users and the latter starts by identifying similarly rated items by the same

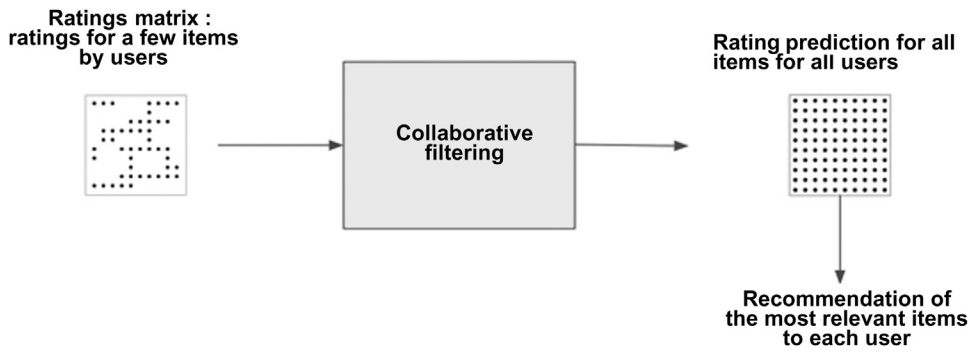

FIGURE 11.3
Collaborative filtering recommender.

users. For the ratings matrix shown in Table 11.1, both the methods share the same technique, where one starts with finding similar rows and the other finds similar columns.

User-Based Collaborative Filtering

The user-based collaborative filtering method operates on the assumption that similar users have similar likes. The two-step process of identifying new unseen user-item preferences consists of filtering similar users and deducing the ratings from similar users. The approach for user-based collaborative filtering is quite similar to the k-NN classification algorithm.

1. For every user x, find a set of N other users who are similar to the user x and who have rated the item i.
2. Approximate the rating of the user x for the item i, by aggregating (averaging) the rating of N similar users.

Consider the known ratings matrix shown in Table 11.2. Say the task is to find whether the user Olivia will like the movie *2001: A Space Odyssey*. The first step will be to find N users similar to the user Olivia who have already rated the movie *2001: A Space Odyssey*. The second step will be to deduce a rating for the user Olivia from the cohort of similar users.

Step 1: Identifying Similar Users

The users are similar if their rating *vectors* are close according to a distance measure. Consider the rating matrix shown in Table 11.2 as a set of rating vectors. The rating for the user Amelia is represented as $r_{amelia} = \{5,1,4,4,1\}$. The similarity between the two users is the similarity between the rating vectors. A quantifying metric is needed in order to measure the similarity between the user's vectors. Jaccard similarity, Cosine similarity, and Pearson correlation coefficient are some of the commonly used distance and

Table 11.2 Known Ratings Matrix and a User-item Rating Prediction

	The Godfather	2001: A Space Odyssey	The Hunt for Red October	Fargo	The Imitation Game
Josephine	5	4	1		1
Olivia		**?**	2	2	4
Amelia	5	1	4	4	1
Zoe	2	2	5	1	1
Alanna	5	5		1	1
Kim		4	1	2	5

similarity metrics. The cosine similarity measure between two nonzero user vectors for the user Olivia and the user Amelia is given by the Eq. (11.2)

$$\text{Cosine similarity } (|x \,.\, y|) = \frac{x\,y}{||x||\;||y||}$$

$$\text{Cosine similarity } (r_{\text{olivia}},\ r_{\text{amelia}}) = \frac{0\times5+0\times1+2\times4+2\times4+4\times1}{\sqrt{2^2+2^2+4^2}\times\sqrt{5^2+1^2+4^2+4^2+1^2}} = 0.53 \tag{11.2}$$

Note that the cosine similarity measure equates the lack of ratings as zero value ratings, which can also be considered as a low rating. This assumption works fine for the applications for where the user has purchased the item or not. In the movie recommendation case, this assumption can yield the wrong results because the lack of rating does not mean that the user dislikes the movie. Hence, the similarity measure needs to be enhanced to take into consideration the lack of rating being different from a low rating for an item. Moreover, biases in the ratings should also be dealt with. Some users are more generous in giving ratings than others who are more critical. The user's bias in giving ratings skews the similarity score between users.

Centered cosine similarity measure addresses the problem by normalizing the ratings across all the users. To achieve this, all the ratings for a user is subtracted from the average rating of the user. Thus, a negative number means below average rating and a positive number means above average ratings given by the same user. The normalized version of the ratings matrix is shown in Table 11.3. Each value of the ratings matrix is normalized with the average rating of the user.

Centered cosine similarity or *Pearson correlation* between the ratings for the users Olivia and Amelia is calculated by:

$$\text{Centered cosine similarity } (r_{\text{olivia}}, r_{\text{amelia}}) = \frac{0\times2.0+0\times-2.0+-0.7\times1.0+0.7\times1.0+1.3\times-2.0}{\sqrt{(-0.7)^2+(0.7)^2+(1.3)^2}*\sqrt{(2.0)^2+(-2.0)^2+(1.0)^2+(1.0)^2+(-2.0)^2}} = -0.65$$

Table 11.3 Normalized Ratings Matrix

	The Godfather	2001: A Space Odyssey	The Hunt for Red October	Fargo	The Imitation Game
Josephine	2.3	1.3	−1.8		−1.8
Olivia			−0.7	−0.7	1.3
Amelia	2.0	−2.0	1.0	1.0	−2.0
Zoe	−0.2	−0.2	2.8	−1.2	−1.2
Alanna	2.0	2.0		−2.0	−2.0
Kim		1.0	−2.0	−1.0	2.0

Table 11.4 User-to-User Similarity Matrix

	Josephine	Olivia	Amelia	Zoe	Alanna	Kim
Josephine	1.00	− 0.20	0.28	− 0.30	0.74	0.11
Olivia		1.00	− 0.65	− 0.50	− 0.20	0.90
Amelia			1.00	0.33	0.13	− 0.74
Zoe				1.00	0.30	− 0.67
Alanna					1.00	0.00
Kim						1.00

The similarity score can be pre-computed between all the possible pairs of users and the results can be kept ready in a user-to-user matrix shown in sample Table 11.4 for ease of calculation in the further steps. At this point the neighborhood or cohort size, k, has to be declared, similar to k in the k-NN algorithm. Assume k is 3 for this example. The goal of this step is to find three users similar to the user Olivia who have also rated the movie *2001: A Space Odyssey*. From the table, the top three users can be found similar to the user Olivia, who are Kim (0.90), Alanna (− 0.20), and Josephine (− 0.20).

Step 2: Deducing Rating From Neighborhood Users

$$\text{Rating } R = \sum_{\text{neighborhood}} (\text{similarity score} \times \text{rating}) \tag{11.3}$$

Once the cohort of users are found, deducing the predicted rating is straight forward. The predicted rating for Olivia for the movie *2001: A Space Odyssey* is the average of the ratings given by Kim, Alanna, and Josephine, for the same movie. One can use weighted average based on the similarity score for more accuracy and the general formula for the predicted ratings is given by Eq. (11.4).

$$r_{xi} = \frac{\sum_{u \in N} s_{xu} r_{ui}}{\sum_{u \in N} s_{xu}} \tag{11.4}$$

where N is a set of k users similar to user x and have rated the item i, r is the predicted rating for user x and item i. s_{xu} is the similarity score of user x and user u. Using Eq. (11.4) and the similarity matrix, the predicted rating for Movie 2: *2001: A Space Odyssey* can be computed for Olivia as:

$$r_{xi} = \frac{(0.9 \times 1) + (-0.2 \times 2) + (-0.2 \times 1.25)}{(0.9 - 0.2 - 0.2)} = 0.05$$

The predicted rating is 0.05 above the average score for Olivia, which is 2.7. The *final* predicted rating for the movie is therefore 2.75 stars. If k is 1, the rating would have been 3.7 stars.

The user-based neighborhood technique provides an intuitive way to fill the ratings matrix. The steps for finding similar users and rating deductions is repeated for every blank ratings cell in the ratings matrix. However, the collaborative filtering process can be time consuming. One way to speed-up the process is to pre-compute the user-to-user similarity matrix shown in Table 11.4 once for all the users and reuse the results for the first step in identifying similar users. The user-to-user similarity matrix should be updated for any new user. However, a new user similarity with other users can be computed only when their preference information is known—*cold start problem*!

The cold start problem is the most significant limitation of the collaborative filtering technique (Isinkaye, Folajimi, & Ojokoh, 2015). A new user to the system would not have item preference ratings. The system could ask the new user to declare the preferences by providing information on their most preferred items. Regardless, the user will have a minimal number of item ratings to start with. For the new users, the cohort neighborhoods will be small and, thus, matching the user with other cohort users will be limited, which in turn leads to less recommendations. Similarly, a new item will not be part of any previous user-item interactions and will lead to less recommendations for the new item to the users. In some cases, because of the fewer recommendations, the new items stay less popular. For existing users or items, the neighborhood method needs more data for the algorithm to be more effective, which can be a limiting factor for the applications that rely exclusively on the explicit data collection method.

The cold start problem can be mitigated by a couple of strategies. First, the system can ask all the new users to select or enter their preferred items once they have signed up. This can be selecting a few movies from a curated list of movies or tagging them from the movie catalog. The new user onboarding process incorporates this step, so the recommendation engines have some seed information about the new users. Second, the system can rely heavily on implicit data collection through search or clickstream activity until a solid item preference profile can be build.

Popular items are preferred by a large number of users. Collaborative filtering tends to recommend popular items to users because the cohort selection might be skewed towards higher ratings for popular items. Recommending the popular items is not necessarily a bad decision. However, one of the objectives of recommendation engine is to discover the personalized idiosyncratic and unique items of the user. Moreover, the nonpersonalized popular items are usually shown by *best seller* or *trending now* lists, which have the same content for all the users.

Item-Based Collaborative Filtering

The item-based neighborhood method operates on the assumption that users prefer items that are similar to previously preferred items. In this context, items that are similar tend to be rated similarly by the same users. If a user liked the dark comedy crime movie *Fargo*, then the user might like the crime thriller movie *No Country for Old Men*, provided that both movies are rated similarly by the same users. Both the movies are created by the same writers-directors and it is presumed that if a user has given high ratings for *Fargo*, the same user might have given high ratings for *No Country for Old Men* as well. If the rating patterns of two items are similar, then the items are in the neighborhood and their item vectors are close to each other. Hence, the item's neighbor is a set of other items that tend to get similar ratings when rated by the same user. The two-step process of identifying new user-item preference using item-based collaborative filtering include identifying similar items and deducing a rating from similar items.

1. For every item *i*, find a set of *N* other items which have similar ratings when rated by the same user.
2. Approximate the rating for the item *i* by aggregating (averaging) the rating of *N* similar items rated by the user.

The ratings matrix shown in Table 11.2 can be used to compute the predicted rating for the same unseen cell—for the user Olivia for the movie *2001: A Space Odyssey-using the item-based method*. To realize the item-based neighborhood method, the rating matrix shown in Table 11.2 has to be transposed (swapping rows and columns) and continued with the same steps as the user-based (or row based) neighborhood method. Table 11.5 shows the transposed version of the original ratings matrix.

The centered cosine or Pearson correlation coefficient metric is used to calculate the similarity between movies based on the ratings pattern. Since the objective is to find the rating for *2001: A Space Odyssey*, the similarity score would need to be found for all the movies with *2001: A Space Odyssey*.

Table 11.5 Transposed Ratings Matrix

	Josephine	Olivia	Amelia	Zoe	Alanna	Kim
The Godfather	5		5	2	5	
2001: A Space Odyssey	4	**?**	1	2	5	4
The Hunt for Red October	1	2	4	5		1
Fargo		2	4	1	1	2
The Imitation Game	1	4	1	1	1	5

Table 11.6 Normalized Ratings and Similarity With a Movie

	Josephine	Olivia	Amelia	Zoe	Alanna	Kim	Similarity with the Movie 2001: A Space Odyssey
The Godfather	2.3		2.0	−0.2	2.0		**−0.10**
2001: A Space Odyssey	1.3		−2.0	−0.2	2.0	1.0	**1.00**
The Hunt for Red October	−1.8	−0.7	1.0	2.8		−2.0	**−0.36**
Fargo		−0.7	1.0	−1.2	−2.0	−1.0	**0.24**
The Imitation Game	−1.8	1.3	−2.0	−1.2	−2.0	2.0	**−0.43**

Table 11.6 shows the centered rating values along with the similarity score of all the movies with *2001: A Space Odyssey*. The similarity score is calculated using Eq. (11.2) on the centered rating values. Since the centered ratings can be negative, the similarity score can be positive or negative. Depending on the number of neighbors specified, the top k neighbors to the movie *2001: A Space Odyssey* can now be narrowed down using the magnitude of the similarity score. Assume k is 2.

From Table 11.6 the nearest two movies to *2001: A space Odyssey* can be concluded, rated by Olivia, are *Fargo* and *The Hunt for Red October*. The predicted centered rating for the *2001: A space Odyssey* for Olivia, using Eq. (11.4) is:

$$r_{xi} = \frac{(0.24 \times -0.7) + (-0.36 \times -0.7)}{(0.24 - 0.36)} = -0.67$$

The normalized rating for Olivia and *2001: A space Odyssey* is −0.67 and the real ratings for *2001: A space Odyssey* by Olivia is 2. Note that the predicted ratings for the same user-item pair is different when user-based collaborative filtering is used.

User-Based or Item-Based Collaborative Filtering?

The neighborhood technique for predicting a rating for a user-item combination, either with user-based or item-based, is very similar. After all, if the ratings matrix is transposed at the beginning, the item-based approach is exactly the same as the user-based approach. However, the predicted rating is *different* when these two approaches are used on the *same* ratings matrix.

Conceptually, finding similar items is relatively easier than finding similar users. Items tend to get aligned with specific genres or types of the items. A movie can belong to either Classics or Science fiction genres or, less likely, in

both the genres. However, a user may like both Classics and Science fiction genres. It is common for the users to have interests in multiple genres and develop unique taste profiles. It is easier to find similar courtroom drama movies. It is difficult to find similar users when a user has preference to courtroom drama, science fiction, movies directed by Francis Ford Coppola and Pixar movies. If all the users prefer one genre, it is easy to isolate the preferences and, thereby, have no confounding of ratings. In practice, that doesn't work, and it is difficult to extract the user preference from the ratings matrix when users have multiple overlapping preference profiles. Hence, finding similar items is more effective than finding similar users and yields better performance in most applications.

In some business applications, the objective of the recommendation engine is to provide a list of top recommended items for each user. To accomplish this in the user-based approach, for every user, one can pre-compute the step of finding similar users and then aggregate the ratings for items. This vastly narrows the search space. However, in the item-based approach, all the item-to-item similarity combinations need to be pre-computed before getting into user level recommendation. The user-based approach has a leg up when it comes to computational time because most applications are concerned with providing recommendations at the user level.

Neighborhood based Collaborative Filtering - How to Implement

The neighbor-based method involves relatively simple calculations, albeit time consuming. The entire step-by-step process of finding similar users (or similarly rated items) and deducing a rating can be implemented in a programming tool or by using RapidMiner. If using the latter, ratings data should be prepared in the form of a ratings matrix to find similar users or items. Similar users can be found using the operator *Data to Similarity Data* operator with Cosine Similarity as the parameter. Alternatively, the prebuilt operators for Recommendation Engines using the Recommenders extension (Mihelčić, Antulov-Fantulin, Bošnjak, & Šmuc, 2012) can be leveraged. The operators that come with the extension expand the available library and abstract the low level tasks to high-level tasks geared toward Recommendation Engine application.

Dataset

The dataset used in this chapter is sourced from MovieLens database. It is a sample of real-world anonymized data from GroupLens Inc. The ratings data can be downloaded from the GroupLens website[1] which provides multiple data size options. The following implementation uses 100,000 ratings given

[1] http://grouplens.org/datasets/movielens

Table 11.7 MovieLens Datasets—Ratings and Movie Attributes

User ID	Movie ID	Rating	Timestamp
1	31	2.5	1260759144
1	1029	3	1260759179
1	1061	3	1260759182
1	1129	2	1260759185
1	1172	4	1260759205
1	1263	2	1260759151
1	1287	2	1260759187
1	1293	2	1260759148
1	1339	3.5	1260759125
1	1343	2	1260759131

Movie ID	Title	Genres
1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
2	Jumanji (1995)	Adventure|Children|Fantasy
3	Grumpier Old Men (1995)	Comedy|Romance
4	Waiting to Exhale (1995)	Comedy|Drama|Romance
5	Father of the Bride Part II (1995)	Comedy
6	Heat (1995)	Action|Crime|Thriller
7	Sabrina (1995)	Comedy|Romance
8	Tom and Huck (1995)	Adventure|Children
9	Sudden Death (1995)	Action
10	GoldenEye (1995)	Action|Adventure|Thriller

by 1000 users for 1700 titles. There are two datasets in each of the downloaded packages. The first dataset (ratings dataset) contains ratings, user IDs, and movie ID attributes. The second dataset (items dataset) contains limited metadata about each movie—movie ID, movie title, and concatenated genre attributes. The attributes of the movies can be further expanded from other third-party databases like IMDB[2] using the movie name. Table 11.7 shows the MovieLens input data.

Implementation Steps

The outline of the full RapidMiner process is shown in Fig. 11.4. The high-level process for a recommendation engine is no different from the other predictive analytics process. It contains data preparation, modeling, applying the model to a test set, and performance evaluation steps to predict the user-item ratings.

[2] http://imdb.com

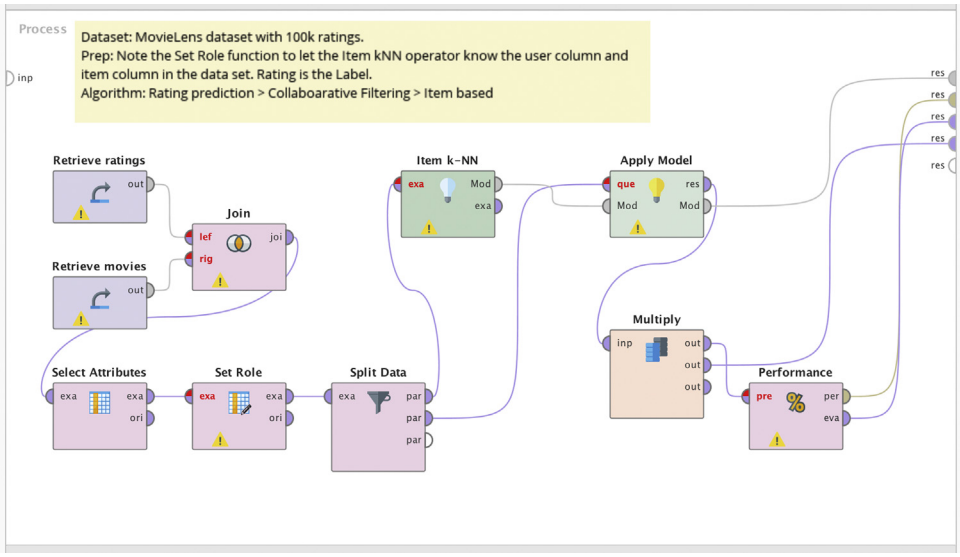

FIGURE 11.4
Item *k*-NN recommender process.

1. **Data Preparation:** The central operator in the neighborhood-based recommender process is the modeling operator called *User—KNN* or *Item—KNN*. The former implements a user-based and the latter implements an item-based neighborhood recommender. As a collaborative filtering method, the only input needed for the modeling operator is the ratings matrix. The ratings matrix is in the form of User identification, Item identification, and the label (rating). Any other attribute from the dataset is not used in the core modeling operator. Whatever the format of the input data, it needs to be transformed to a dataset with user identification, item identification, and numerical rating label.
   All the information needed for the modeling is available in the MovieLens rating dataset. However, more context can be added to the movies by joining the ratings dataset with the item dataset. Attributes not needed for the process are discarded using the *Select attribute* operator. The *Set role* operator is used to declare which attribute is "user identification," "item identification," and the label. The "user identification" and "item identification" are custom roles used by Recommender operators. Mapping the attribute user ID to "user identification" role and item ID to "item identification" role is

specified in the parameter settings for the *Set role* operator. The ratings data is randomly split between training (95%) and test dataset (5%) using the *Split data* operator

2. **Modeling:** The modeling operator used in this recommender process *item K-NN* can be found under Recommenders > Item rating prediction > Collaborative Filtering Rating Prediction. There are a few parameters to configure based on the application.
   a. *k*: Nearest neighbor cohort size. This is the same "*k*" as in *k*-NN classification. The cohort size is set as 10 for this process.
   b. Min and Max ratings: The ratings range in the training dataset. The movie ratings dataset has ratings from 0 to 5.
   c. Similarity measure: Pearson or Cosine similarity measure is the commonly used similarity measure for recommenders. Cosine similarity measure is used for this recommender process.
3. **Apply Model to test set:** The *Apply Model (Rating Prediction)* operator under Recommenders > Recommender Performance > Model Application is used to apply the training model to the test dataset. A portion of the original dataset that is reserved for testing is used as an input for the Apply Model operator which calculates the predicted ratings for each user-item combination in the test dataset. The Apply Model is also used in the final deployment where one can supply user-item combinations for which the rating is to be predicted.
4. **Performance evaluation:** The output of the predicted test dataset from the *Apply Model* operator is connected to the *Performance (Rating Prediction)* operator under the Recommender extension folder to evaluate if the predicted rating is close to the actual rating provided by the users. As in the *Apply Model* operator, the ratings range is declared in the parameters for *Performance (Rating Prediction)* operator.

The whole process shown in Fig. 11.4 can be saved and executed. The Results window shows the predicted values for the test dataset. In addition to the attributes in the original test dataset, a new prediction column is appended to the dataset. A sample of the first ten records are shown in Fig. 11.5, where recommender model prediction and the actual rating given by the users can be observed.

The performance vector tab in the Results window shows the output of the Performance evaluation operator. RMSE of the rating prediction is 0.873 stars for the neighborhood-based recommender using the MovieLens dataset. The MAE is 0.665 stars. On an average, the predicted star rating will be off from the actual prediction by 0.665 stars. Not bad for using a simple recommender model!

Row No.	rating	userId	movieId	title	genres	prediction
1	2.500	1	2455	Fly, The (1986)	Drama|Horror|Sci-Fi|Thriller	2.292
2	3	2	339	While You Were Sleeping (1995)	Comedy|Romance	3.382
3	3	2	410	Addams Family Values (1993)	Children|Comedy|Fantasy	3.447
4	3	2	539	Sleepless in Seattle (1993)	Comedy|Drama|Romance	3.232
5	3	2	550	Threesome (1994)	Comedy|Romance	3.399
6	3	2	588	Aladdin (1992)	Adventure|Animation|Childr...	3.954
7	4	2	661	James and the Giant Peach (1996)	Adventure|Animation|Childr...	3.418
8	4	2	720	Wallace & Gromit: The Best of Aardman ...	Adventure|Animation|Comedy	3.777
9	3	4	431	Carlito's Way (1993)	Crime|Drama	4.608
10	5	4	1194	Cheech and Chong's Up in Smoke (1978)	Comedy	4.297

FIGURE 11.5
Predicted ratings for Item *k*-NN recommender.

Conclusion

Neighborhood-based recommendation systems use just the ratings matrix as in input for training the model. It is content agnostic or domain independent and does not have to learn about the details of the items a priori. Moreover, the same model can be used to recommend different types of items. An ecommerce platform can recommend books, movies, and other products with the same recommendation engine. *The model using the neighborhood method does not care about what the items mean, it just focuses on the user-item interactions using past ratings*. As with all the collaborative filtering techniques, the neighborhood method suffers from the cold start problem and popular item bias.

One limitation of neighborhood techniques is the processing complexity and scalability. It is computationally intensive to find similar users or similar items for every user or an item, when there are millions of users and items. Since it is practically impossible to carry out these calculations at runtime, the neighborhood calculation for finding neighborhoods for each user or item can be pre-computed. If R is the ratings matrix, the processing complexity of the neighborhood-based approach is given by O(|R|). The pre-computation processing complexity increases to O(n.|R|) where, n is the number of items (Lee, Sun, & Lebanon, 2012). As the number of items or users increase in the system, so to do the computations required for collaborative filtering approaches. In the current form, collaborative filtering approaches do not scale well with the increase in users and items in a system.

The items like a pack of Oatmeal and a pack of Instant Oatmeal may look quite similar, but they are different items. Synonymy is the tendency of particularly similar items with different distinctions. Collaborative filtering techniques will have a hard time equating these two items because they do not have any

information on the content of the items. Ideally, two users having preferences on two similar items should be treated the same. Collaborative filtering techniques have difficulty managing the synonymy of the items.

GLOBAL BASELINE

The recommendation engine methods discussed in this chapter so far have underpinnings in machine learning. Alternatively, there is a simple straightforward (naive) method to calculate the predicted rating for a user-item interaction - Global baseline, as shown in Eq. (11.5).

$$r_{ui} = \mu + b_u + b_i \tag{11.5}$$

where μ is the global average of all ratings, b_u is the user bias. On average, the rating given by a user u is $\mu + b_u$, b_i is the item bias. On average, the rating for the item i by all the users is $\mu + b_i$. b_u and b_i is the delta from the global average of all the ratings, given by the user u and for the item i. For example, consider the average star rating for all the movies is 2.5. The user Amelia tends to be generous in her rating and on average she has given 3.5 stars for all the movies rated by her. The user bias is +1.0 stars. On an average the movie *Interstellar* received 2.0 stars from all the users. The item bias is −0.5. The predicted star rating, using Global baseline, by the user Amelia for the movie *Interstellar* is 3.0

$$r_{ui} = 2.5 + 1.0 - 0.5 = 3.0$$

The global average serves as a baseline to measure the performance of more complex machine learning-based recommenders discussed in this chapter.

11.2.2 Matrix Factorization

The ratings matrix discussed so far in this chapter has information on the user, item, and the strength of user-item interaction. For every user, the ratings information is at the level of individual item, that is, every individual movie, track, product. The number of items usually ranges from thousands to millions of unique values. If one were to ask someone what movies or books they would prefer to watch or read, the answer is usually in the form of some generalized dimension. For example, expressing the preference for science fiction movies, movies directed by Christopher Nolan, movies with strong female leads, crime novels or 1920s literature. The subsequent follow-up might be providing specific item examples which belong to those groupings, for example titles like, *Interstellar*, *Iron Lady*, *The Great Gatsby*, etc. The generalized categories can be a predefined genre, works by a creator, or sometimes it might be a yet-to-be named or vague categorization. Perhaps the user just likes a set of items with no specific generalized category.

The *Latent Factor* model, like the neighborhood methods, uses just the ratings matrix as the only input. It tries to generalize and explain the ratings matrix with a set of *latent factors* or generalized dimensions. The factors, which usually range from a dozen to a few hundred, are automatically inferred from the ratings matrix, as long as the number of factors is specified. There is no separate input to the model that provides pre-classified genres or factors. In fact, there are no interpretable names for the latent factors. The inferred

factors might resemble genre like classification (science fiction or family movies) and in some cases they are just uninterpretable groupings of items. It will be interesting to research and generalize why a group of items are rated highly against a factor. Once the factors are discovered, the model associates an item's membership towards a factor and a user's inclination towards the *same* factor. If the user and item are near to each other when plotted against a factor or a set of factors, then there is a strong user-item preference.

Fig. 11.6 shows the users (circles) and items (squares) mapped against two illustrative factors: production scale and comedic content. The items or movies are plotted on the chart based on how movies express themselves against the latent factors. The users are plotted based on their preferences against the same latent factors. From the chart, it can be concluded that the user Amelia prefers the movie *No country for Old men* because both the user and the movie are close to one another when expressed against the latent factors. Similarly, Alanna prefers the movie *Headhunters* instead of *Titanic*. The user-item preference is calculated by the dot product of the user vector and the item vector expressed latent factors. The similarity between the user and the item vectors dictates the preference of the user to the item (Koren et al., 2009).

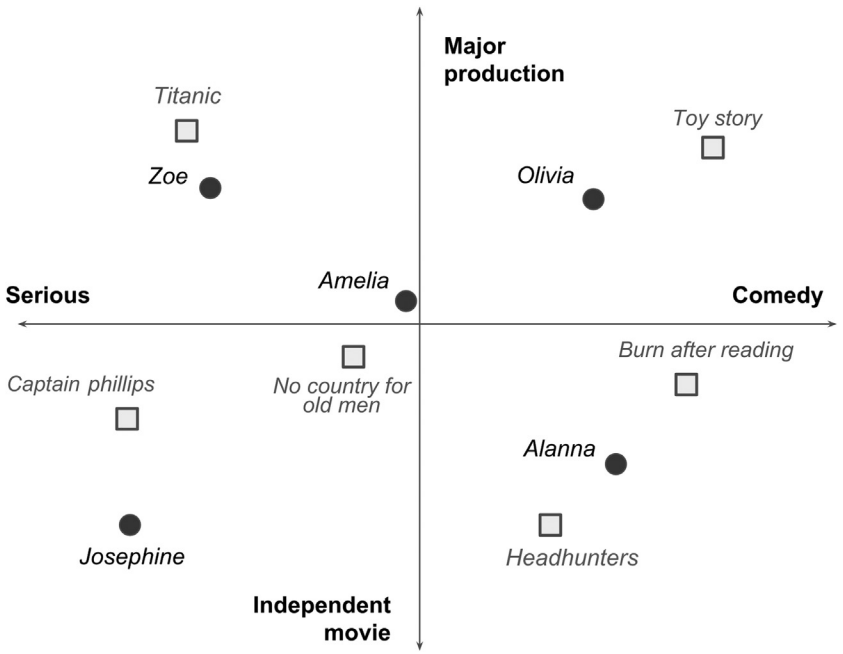

FIGURE 11.6
Items and users in latent factor space.

Matrix Factorization is a technique to discover the latent factors from the ratings matrix and to map the items and the users against those factors. Consider a ratings matrix R with ratings by n users for m items. The ratings matrix R will have $n \times m$ rows and columns. The matrix R can be decomposed into two thin matrices P and Q. P will have $n \times f$ dimensions and Q will have $m \times f$ dimensions where f is the number of latent factors. In the example used in Fig. 11.6, there are two latent factors. The matrix R can be decomposed in such a way that the dot product of the matrix P and transposed Q will yield a matrix with $n \times m$ dimensions that closely approximates the original ratings matrix R (Fig. 11.7).

$$R \approx P \cdot Q^T \tag{11.6}$$

The decomposition of the ratings matrix into the user matrix and the item matrix makes intuitive sense. The rating of a user-item combination, say Olivia to *Toy Story*, can be explained by Olivia's preference to comedy movies and whether the movie *Toy Story* is acclaimed highly in a comedic content scale or not. This generalization approach greatly reduces the size of the matrices. The massive $m \times n$ dimensional ratings matrix with 2 million users and a half a million items can be decomposed to two thin matrices: P with 2 million users against 100 factors and Q with half a million items against 100 factors. The dot product of P and Q will closely yield the original ratings matrix, with 2 million users and a half a million items, but with more information! *All* the cells in the rating matrix will be filled with the predicted ratings from the dot product, including the sparse known ratings and the vast unknown ratings, using the matrix dot product of P and Q.

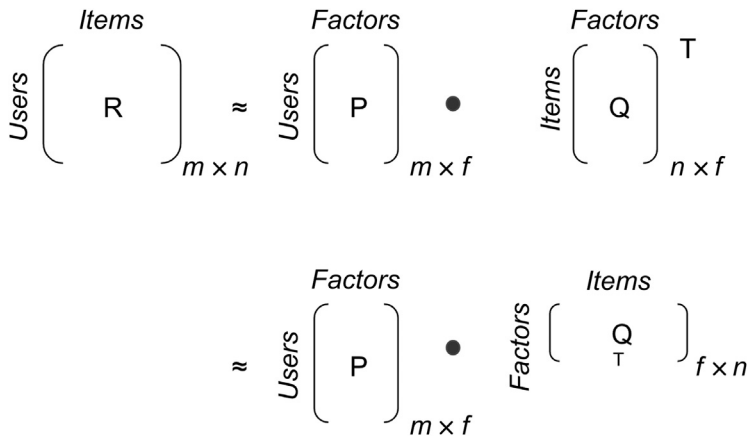

FIGURE 11.7
Decomposition of ratings matrix into latent factor matrices.

The approach is similar to expressing a number, 61, as a product of two numbers. As a prime number, it is not possible to decompose 61 as a product of two numbers, but the original number can be closely approximated with 6×10. The error in this decomposition is 1. Similarly, each rating r can be expressed by the user u for the item i as:

$$\widehat{r_{ui}} = p_u.q_i^T \tag{11.7}$$

where p_u is the user vector with f dimensions and q_i is the item vector with f dimensions. For each item i, the vector q_i measures the extent to which the item i possesses the latent factors. For each user u, the vector p_u measures the preference of the user for the items that have a high possession of those factors. The dot product of the vectors p_u and $q^T{}_i$ gives the approximation of the user's preference for the item.

The objective of the matrix factorization method is to learn the vectors P and Q from the ratings matrix R, where P expresses an item's rating in terms of the factors f and Q expresses the interest of users to the factors. P and Q should be learned in such a way that one can minimize the delta between known ratings and predicted ratings. Assume K is the set of known, non-blank cells in the ratings matrix R. The objective is to minimize the prediction error or the loss function, in Eq. (11.8)

$$\begin{aligned} &\min \sum_{(u,i)\in K} (r_{ui} - \widehat{r_{ui}})^2 \\ &\min \sum_{(u,i)\in K} (r_{ui} - p_u q_i^T)^2 \end{aligned} \tag{11.8}$$

where K is the set of (u,i) pairs of known ratings.

Similar to all machine learning algorithms for predictive tasks, overfitting is a problem for recommenders using the matrix factorization method. The key objective is to predict the ratings of items which are not rated, more than accurately predicting the ratings of the known items. To reduce the impact of overfitting, one can introduce *regularization* models (as introduced in Chapter 5: Regression Methods). Regularization is a technique that penalizes learning more complex and flexible models to avoid the risk of overfitting. Regularization avoids overfitting by minimizing the regularized square error shown in Eq. (11.9). It regularizes or forces the coefficient estimates toward zero. Hence, the magnitude of the learned parameters (coefficients) is penalized through regularization. The extent of regularization is controlled through the tuning parameter λ, which ranges from 0 (no effect) to ∞ (maximum effect)

$$\min \sum_{(u,i)\in K} \left(r_{ui} - p_u q_i^T\right)^2 + \lambda\left(||p_u||^2 + ||q_i||^2\right) \tag{11.9}$$

Much of the observed variance in the rating matrix can be explained by the user-item interactions. However, in real-world applications, there are significant biases that affect the user-item preference. After all, some users are more critical in providing ratings than others. Some blockbuster movies gather generous ratings because they are... blockbusters. To factor in the effects of bias to overall ratings the Global Baseline model can be used given by Eq. (11.5), as a proxy for overall bias in the system.

$$b_{ui} = \mu + b_u + b_i \tag{11.10}$$

The effects of the bias can be used in the predicted ratings, in addition to the user-item interactions as calculated by the matrix factorization method. Eq. (11.11) shows the predicted rating with bias taken into consideration.

$$\widehat{r_{ui}} = \mu + b_u + b_i + p_u.q_i^T \tag{11.11}$$

The overall objective function for the matrix factorization methods is shown in Eq. (11.12). The parameters for the learning algorithm are: regularization λ, user bias b_u, item bias b_i, and the global average rating μ

$$\min \sum_{(u,i)\in K} (r_{ui} - (\mu + b_u + b_i + p_u.q_i^T)^2 + \lambda(||p_u||^2 + ||q_i||^2 + {b_u}^2 + {b_i}^2) \tag{11.12}$$

The algorithm commonly used to learn the factor vectors p_u and q_i by minimizing the loss function is *Stochastic Gradient Descent* (SGD). SGD is an iterative optimization method based on gradient descent touched upon in chapter 5 and also in chapter 10, to minimize the objective function like the one in Eq. (11.9). For the given factor dimension f, the SGD algorithm initializes the vectors P and Q and calculates the error rate, which is the delta between the real and predicted ratings. The algorithm slowly changes the value of P and Q to minimize the error and halts when there is no meaningful change in the error rate (Gemulla, Nijkamp, Haas, & Sismanis, 2011).

Matrix Factorization - How to Implement

Implementing a matrix factorization-based recommender from scratch is an effort intensive process. The Recommenders extension (Mihelčić et al., 2012) offers prebuilt operators to implement Biased Matrix Factorization recommendation engines. As with the neighborhood-based recommender implementation, the MovieLens dataset from GroupLens[3] with 100,000 ratings given by 1000 users for 1700 titles is used for building the recommender using matrix factorization.

[3] http://grouplens.org/datasets/movielens

Implementation Steps

The outline of the RapidMiner process for the matrix factorization-based recommender is shown in Fig. 11.8. The process is similar to the process used in neighborhood-based methods, with the only change to the modeling operator—Biased Matrix Factorization (BMF)

1. **Data Preparation:** The central operator in this is process is the modeling operator called BMF. As a collaborative filtering method, the only input needed for the modeling operator is ratings matrix. The *Set role* operator is used to declare the attributes for "user identification", "item identification," and the label. The data is randomly split between training (95%) and test dataset (5%).
2. **Modeling:** The modeling operator is BMF—found under Recommenders > Item rating prediction > Collaborative Filtering Rating Prediction. These parameters can be configured in the modeling operator to suit the application:
   a. **Min and Max ratings:** The ratings range in the training dataset. The movie ratings dataset has ratings from 0 to 5.

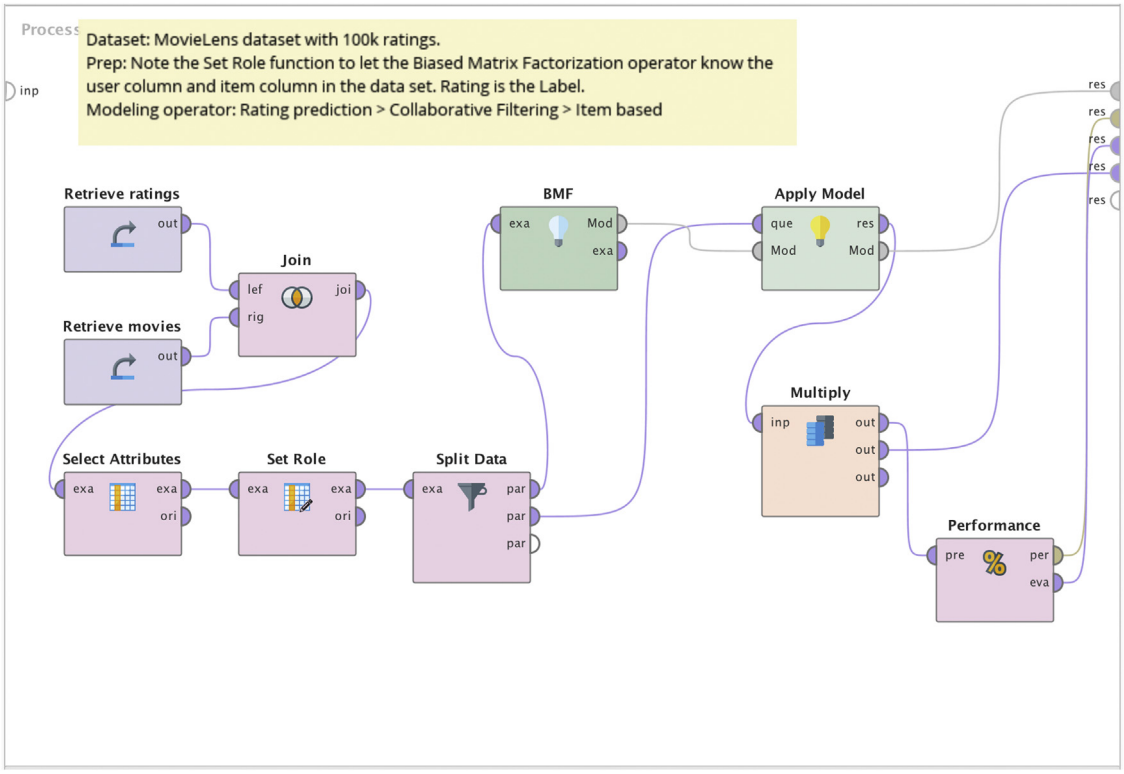

FIGURE 11.8
Matrix factorization recommender process.

b. **Num Factors (f)**: The number of latent factors inferred from the ratings dataset. Specifying the number of latent factors is critical in matrix factorization modeling. However, similar to k-NN and clustering techniques, specifying an optimal number of factors is tricky. Hence, the parameter optimization technique discussed in Chapter 15 Getting started with RapidMiner may need to be employed to find the best number of latent factors for the dataset to obtain optimal performance. In this process, the number of latent factors is set as 12.
c. **Bias:** Bias regularization parameter. The default value is set as 0.0001.
d. **Learn Rate (α):** Learning rate in the Stochastic Gradient Descent algorithm. SGD is used to optimize the parameters to minimize the function in Eq. (11.12). The value is set as 0.005 for this process.
e. **Regularization (λ):** Regularization tuning parameter. The default value is set as 0.015.

3. **Apply Model to test set:** The *Apply Model (Rating Prediction)* operator is used to apply the training model to the test dataset. The model and the test dataset serve as input to the apply model operator and the output is the predicted rating for all the test records.
4. **Performance evaluation:** The output of the predicted test dataset from the *Apply Model* operator is connected to the *Performance (Rating Prediction)* operator to evaluate if the predicted rating is close to the actual rating provided by the users.

The RapidMiner process shown in Fig. 11.8 can be saved and executed. The result window shows the predicted dataset, the recommender model and the performance vector. In addition to the attributes in the test dataset, a new

Row No.	rating	userId	movieId	title	genres	prediction
1	2.500	1	2455	Fly, The (1986)	Drama|Horror|Sci-Fi|Thriller	2.854
2	3	2	339	While You Were Sleeping (1995)	Comedy|Romance	3.078
3	3	2	410	Addams Family Values (1993)	Children|Comedy|Fantasy	2.813
4	3	2	539	Sleepless in Seattle (1993)	Comedy|Drama|Romance	3.059
5	3	2	550	Threesome (1994)	Comedy|Romance	2.838
6	3	2	588	Aladdin (1992)	Adventure|Animation|Childre...	3.505
7	4	2	661	James and the Giant Peach (19...	Adventure|Animation|Childre...	3.158
8	4	2	720	Wallace & Gromit: The Best of A...	Adventure|Animation|Comedy	3.733
9	3	4	431	Carlito's Way (1993)	Crime|Drama	4.411
10	5	4	1194	Cheech and Chong's Up in Smok...	Comedy	4.444

FIGURE 11.9
Predicted ratings using matrix factorization recommender.

rating prediction column is appended to the dataset. A sample of ten test records are shown in Fig. 11.9 with the original unseen ratings and the predicted ratings.

The results window also contains the performance vector which shows the RMSE and MAE for the ratings prediction for the test dataset. The RMSE for the BMF model is 0.953 stars and MAE is 0.730 stars.

11.3 CONTENT-BASED FILTERING

The collaborative filtering method uses the past user-item interaction data as the only input for building the recommenders. 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. This assumption makes intuitive sense. If a user liked the movies *Frago*, *No Country for Old Men*, and *Burn After Reading*, then the user would most likely prefer *The Big Lebowski*, which are all directed by the same people—the Coen brothers. Content-based recommendation engines keep suggesting an item to a user similar to the items rated highly by the same user. The user will most likely get recommendations about movies with the same cast or director as the user preferred in the past.

The distinguishing feature of the content-based recommendation engine is that it sources attributes of an item, also known as building the item profile. The attribute data about the movies are readily available in public databases like IMDB,[4] where the cast, directors, genre, description, and the year of the title can be sourced. The item attributes can be derived from structured catalogs, tags, or unstructured data from the item description and images.

The general framework of a content-based recommendation engine is shown in Fig. 11.10. 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.

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 catalog 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.

[4] https://www.imdb.com/

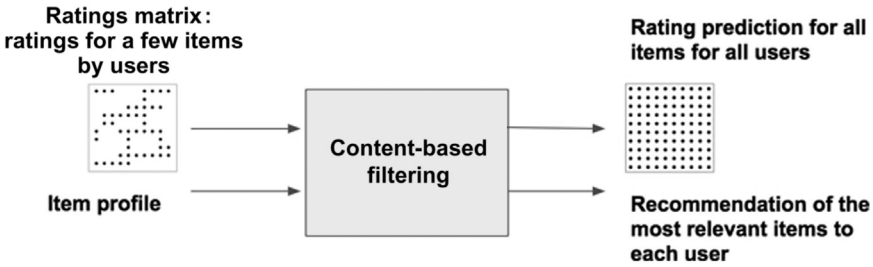

FIGURE 11.10
Model for content-based recommendation engine.

The user profile approach computes the preference of users to the item attributes from the ratings matrix. The proximity of the users and the items against the item attribute space indicates the preference of the user to the items. Whereas, the supervised learning approach treats the user preference of attributes as a *user level* classification or regression problem with the ratings matrix serving as the label (target) and item attributes as the predictors (feature). If one uses a decision tree as the supervised learning technique, then *each* user will have a personalized decision tree. The nodes in the decision tree will be checking an item attribute to predict whether the user will prefer the item or not.

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, subgenre, 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. Similar to the document vectors discussed in Chapter 9, Text Mining, the number of columns or attributes will be large and the matrix will be sparse. Table 11.8 shows a sample item profile or item feature matrix for the movies with the attributes as cast, directors, genre, etc.

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. Chapter 9, Text Mining discusses relevant tools like term frequency-inverse document frequency (TF-IDF) to

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

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. At the end item profile creation process, there will be a set of attributes that best describe the characteristics of each item. The matrix contains information on whether each item possess those attributes or not. Once the item profile is assembled, the recommendations can be extracted using either a user profile computation approach or a supervised learning approach.

11.3.1 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. Proximity measures like centered cosine metric discussed in the neighborhood based methods is used to measure the preference between a user and an item. The proximity measure between user and item is used to provide recommendations of the item to the user. This approach is similar to matrix factorization as both the methods express the ratings in terms of a user's preference to a set of features and items associated to the same features. Content-based recommenders using user profile computation differs from matrix factorization with regards to how the features are derived. Content-based recommenders start with a known feature set for the items and matrix factorization infers a set of specified features from the ratings matrix.

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. 11.11 shows the visual representation of the matrix operation.

Consider the two matrices shown in Tables 11.9 and 11.10. The matrix *R* is the ratings matrix with six users and five movies. This is a Boolean matrix with 1 indicating that the user likes the movie and the blank indicating no explicit preference for the movie. The matrix *I* is the item profile with *f* columns, starting with the cast, . . ., director, movie genre. In practice, *f* will span thousands of columns, which is a superset of all the cast members, directors, and genres of all the movies in the catalog.

The user profile *U* can be derived in such a way than the value-shown user profile is percent of the time that the feature appears in the movies liked by the user. For example, Olivia likes the movies *Fargo*, *Queen*, and *Eye in the Sky*. Two-thirds of all the movies liked by Olivia have Helen Mirren in the cast (*Queen* and *Eye in the Sky*). All the movies (*Forrest Gump* and *Sleepless in Seattle*) liked by Josephine have Tom Hanks in the cast. The generic formula for the cells in the user profile U is the number of times the feature appears

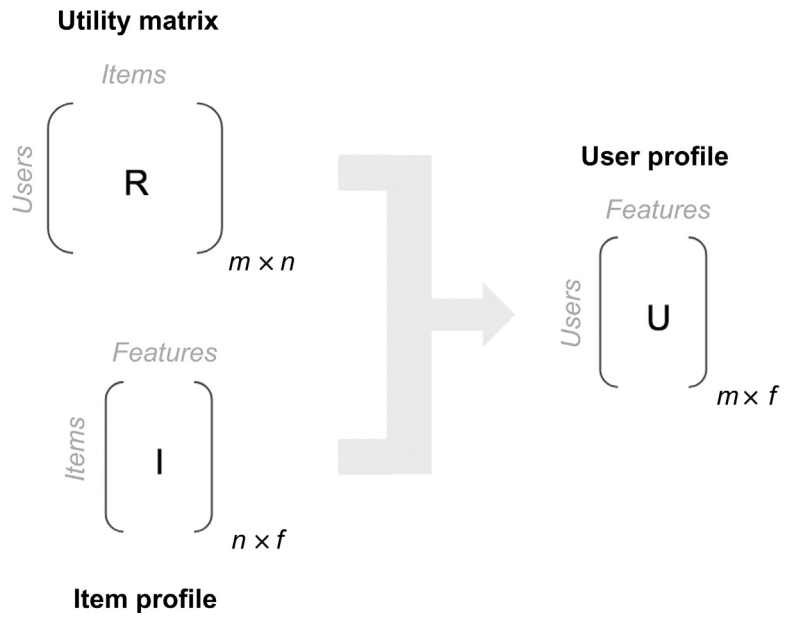

FIGURE 11.11
User profile from utility matrix and item profile.

Table 11.9 Ratings Matrix R

	Fargo	Forrest Gump	Queen	Sleepless in Seattle	Eye in the Sky
Josephine		1		1	
Olivia	1		1		1
Amelia		1			
Zoe	1				
Alanna					
Kim		1			

Table 11.10 Item Profile I

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

in the movies liked by the user divided by the number of movies liked by the user. Table 11.11 shows the user profile U.

The user feature vector for Amelia is $U = \{1,0,\ldots,0,0,\ldots,0,0\}$ and the item feature vector for *Fargo* is $I = \{0,0,\ldots,1,0,\ldots,0,0\}$ and *Forrest Gump* is $I = \{1,0,\ldots,0,0,\ldots,0,0\}$. Out of these two item vectors, *Forrest Gump* is closer (in fact, perfect match in this example) to Amelia's user vector and, hence, it gets recommended.

As more becomes known about the user's preferences (when the user "likes" a title), the user profile is updated. Consequently, the weights of the user-feature matrix are updated and the latest user profile is used to compare against all the items in the item profile. Note that in this method, unlike the collaborative filtering approach, no information from *other users* is needed to make recommendations. This feature makes the content-based recommendation system a good fit to address the cold start problem, especially when a new item is added to the system. When a new movie title is added to the system, the item attributes are already known a priori. Therefore, the new items can be instantly recommended to the relevant users. However, the content-based method needs more consistent information about each item to make

Table 11.11 User Profile *U*

	Tom Hanks	Helen Mirren	...	Joel Coen	Kathryn Bigelow	...	Romantic	Action
Josephine	1						1/2	
Olivia		2/3						1/3
Amelia	1							
Zoe				1				
Alanna								
Kim	1							

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.

Content-Based Filtering - How to Implement

In addition to the standard ratings matrix, a content-based recommender needs the item profile. Sourcing item attribute dataset is one more additional data pre-processing step to be built in the data science tool for creating a content-based recommendation engine. In RapidMiner, implementing the content-based recommenders can be accomplished using the Recommender extension operators.

Dataset

The same MovieLens[5] ratings matrix dataset used earlier in collaborative filtering is used to implement content-based recommenders. There are two datasets provided by MovieLens. The first datafile contains a ratings matrix, with user ID, movie ID, and ratings. The ratings matrix has 100,000 ratings given by 1000 users for 1700 titles. The movie datafile contains limited metadata about the movie ID: title and concatenated genres. This second dataset will serve as the item profile to build the content-based filtering.

Implementation steps

The outline of the RapidMiner process for the recommender is shown in Fig. 11.12. The high-level process for building a content-based recommendation engine consists of preparing the ratings matrix, preparing the item profile, recommender model building, applying the model to known test ratings, and the performance evaluation.

1. **Data preparation:** The modeling operator for the content-based recommender process is item attribute *k*-NN which can be found under

[5] GroupLens.org

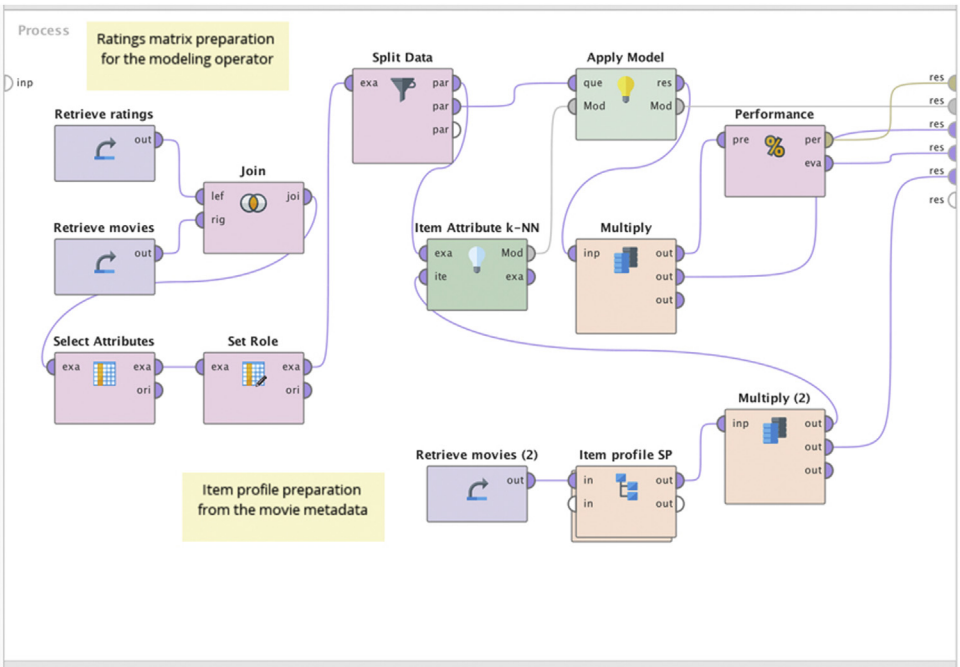

FIGURE 11.12
Recommender process using content-based filtering.

Recommenders > Item rating prediction > Attribute-based Rating Prediction. The inputs for the modeling operator are (1) the ratings matrix and (2) the item profile. The ratings matrix is in the form of User identification (user ID), Item identification (movie ID), and the label (Rating), similar to the process used in collaborative filtering. *Set role* operator is used to declare which attribute is the "user identification", "item identification," or the label.

2. **Item profile preparation:** Content-based recommenders will need the item profile for building the recommender model. The second dataset from MovieLens is in the form of Movie ID, Title, and Genres, as shown in Fig. 11.14A. The column Genres is a concatenated list of genres that a title will belong to with each genre separated by | character. For example, the movie *Toy Story* belongs to Adventure, Animation, Children, Comedy, and Fantasy genres. However, the format of the item profile that is needed by the *Item attribute k-NN* modeling operator is shown in Fig. 11.14C where each record has one distinct movie-genre combination. The movie *Toy Story* (Movie ID = 1) is pivoted into five rows, one for each genre (Genre = 4,5,6,7,11). The item profile dataset represented in Fig. 11.14A would need to be transformed into the item profile shown in Fig. 11.15C. The following section

highlights the fact that the most time-consuming part of a data science process is gathering and processing data. The subprocess of transforming the raw item profile into the one needed for the modeling operator is shown in Fig. 11.13A. The key steps in this data transformation task are:

a. ***Text mining***: The Genre attributes contain values like Adventure|Animation|Children|Comedy|Fantasy. The *text to attributes using text mining* operator transforms the concatenated values into independent attributes shown in Fig. 11.14B. This process is discussed in the Chapter 9, Text Mining. The key operator in the text mining subprocess is *Tokenize*—to convert the genre words to

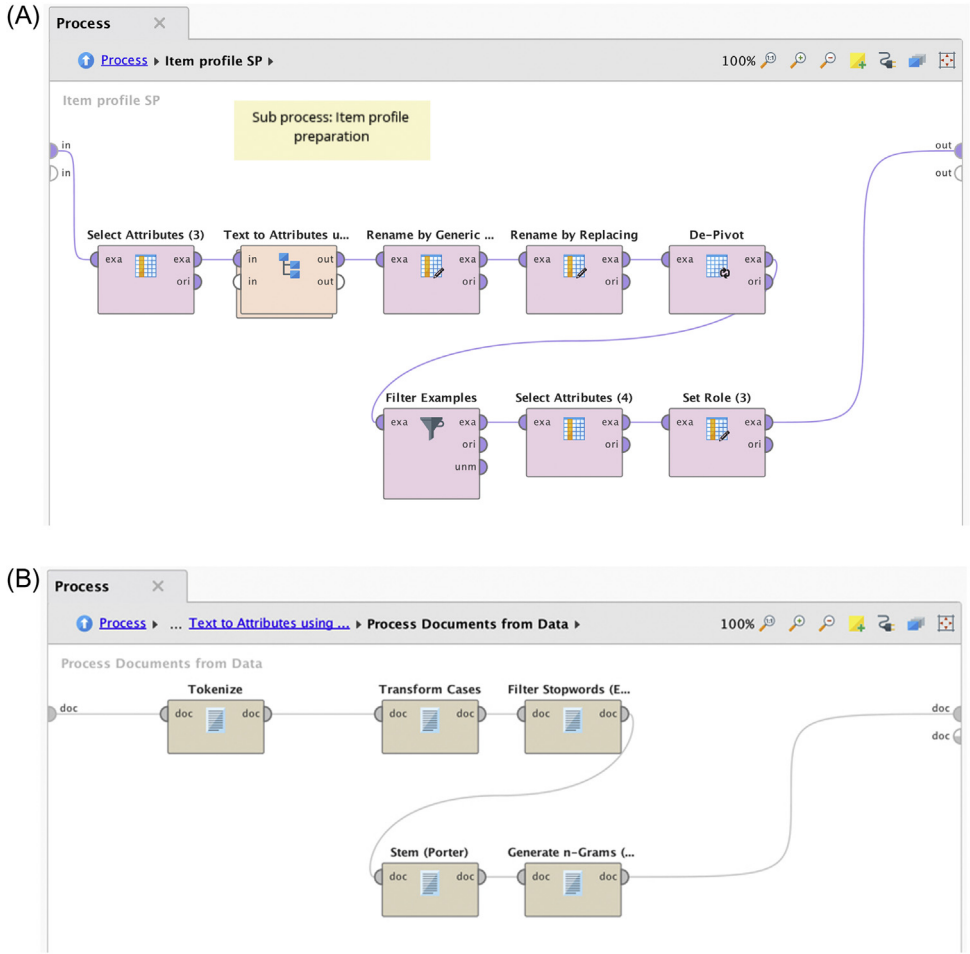

FIGURE 11.13

(A) Subprocess to create an Item profile. (B) Text mining subprocess to convert text to attributes.

(A)

Row No.	movieId	title	genres
1	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
2	2	Jumanji (1995)	Adventure|Children|Fantasy
3	3	Grumpier Old Men (1995)	Comedy|Romance
4	4	Waiting to Exhale (1995)	Comedy|Drama|Romance
5	5	Father of the Bride Part II (19...	Comedy
6	6	Heat (1995)	Action|Crime|Thriller
7	7	Sabrina (1995)	Comedy|Romance
8	8	Tom and Huck (1995)	Adventure|Children
9	9	Sudden Death (1995)	Action
10	10	GoldenEye (1995)	Action|Adventure|Thriller

(B)

movieId	(no genres ...	?	action	adventur	anim	children	comedi	crime	documentari
1	0	0	0	1	1	1	1	0	0
2	0	0	0	1	0	1	0	0	0
3	0	0	0	0	0	0	1	0	0
4	0	0	0	0	0	0	1	0	0
5	0	0	0	0	0	0	1	0	0
6	0	0	1	0	0	0	0	1	0
7	0	0	0	0	0	0	1	0	0
8	0	0	0	1	0	1	0	0	0
9	0	0	1	0	0	0	0	0	0
10	0	0	1	1	0	0	0	0	0

(C)

Row No.	movieId	Genre
1	1	4
2	1	5
3	1	6
4	1	7
5	1	11
6	2	4
7	2	6
8	2	11
9	3	7
10	3	17

FIGURE 11.14

(A) Item profile with concatenated genre. (B) Item profile with attributes. (C) Item profile with attribute information in rows.

attributes. The parameter to split the words is the character (|) which is specified in the *tokenize* operator.

b. ***Attribute naming for De-Pivot:*** Before the columns can be de-pivoted to rows, the columns should be renamed so the *de-Pivot* operator can work this dataset. The columns are renamed to generic attr1, attr2, . . ., etc.

c. ***De-Pivot*:** The *de-pivot* operator converts the columns information to rows. The Genre in the column for each movie is now a distinct row. This operator expands the row count of the table from count (movie ID) to count [movie ID × distinct (genre)].

d. ***Filtering and set role*:** The output of de-pivot has both the negative and positive examples. Only the positive examples are needed, and the negative examples can be filtered out. The *Set role* operator is used to declare Movie ID as "item identification" and Genre as "attribute identification". The dataset is now suitable to be used for the modeling operator. The attribute columns for the items are folded into one column (Genre) which indicates all the attributes the item possesses.

3. **Recommender Modeling**: The modeling operator *item attribute K-NN* receives both the ratings matrix and the item profile as the inputs. There are a few parameters to configure based on the application and the input data

   a. **k:** Nearest neighbor cohort size. The predicted rating is based on the distance between the user vector and the item vector. *k* is set as 10 in this process.

   b. **Min and Max ratings:** Ratings range in the dataset. It is set as 0 and 5 for minimum and maximum ratings.

4. **Apply Model to test set:** The *Apply Model (Rating Prediction)* operator is used to apply the training model to the test dataset. A portion of the original dataset that was reserved for testing is used as an input for the Apply Model operator which calculates the predicted ratings for each user-item combination in the test dataset.

5. **Performance evaluation:** The output of the predicted test dataset from the *Apply Model* operator is connected to the *Performance (Rating Prediction)* operator to evaluate if the predicted rating is close to the actual rating provided by the users.

The RapidMiner process shown in Fig. 11.12 can be saved and executed. The result window shows the original dataset with a new prediction column appended to the dataset. A sample of ten records are shown in Fig. 11.15. The result tab also has the performance vector.

The performance vector shown as the result of the performance evaluation operator reports the RMSE and MAE of rating prediction. RMSE of the rating

Row No.	rating	userId	movieId	title	genres	prediction
1	2.500	1	2455	Fly, The (1986)	Drama|Horror|Sci...	2.367
2	3	2	339	While You Were Sleeping (1995)	Comedy|Romance	3.174
3	3	2	410	Addams Family Values (1993)	Children|Comedy...	3.003
4	3	2	539	Sleepless in Seattle (1993)	Comedy|Drama|R...	3.359
5	3	2	550	Threesome (1994)	Comedy|Romance	2.999
6	3	2	588	Aladdin (1992)	Adventure|Anima...	3.429
7	4	2	661	James and the Giant Peach (1996)	Adventure|Anima...	3.737
8	4	2	720	Wallace & Gromit: The Best of A...	Adventure|Anima...	4.053
9	3	4	431	Carlito's Way (1993)	Crime|Drama	4.746
10	5	4	1194	Cheech and Chong's Up in Smok...	Comedy	4.186

FIGURE 11.15
Predicted rating using content-based filtering.

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

prediction is 0.915 stars for the neighborhood-based recommender using the MovieLens dataset. The MAE is 0.703 stars.

11.3.2 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. Consider the user-item ratings matrix shown in Table 11.8 and the item profile matrix shown in Table 11.10. 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 11.12. This matrix is strikingly similar to the training data used in the supervised models discussed in Chapter 4, Classification and Chapter 5, Regression Methods. With the exception of the first column (Movie

title) which can be ignored as ID, all the attributes in the item profile table are features or independent variables in the training set. The last column, newly introduced indicator for one user, is the class label or the dependent variable. A classification model can be built to generalize the relationship between the attributes to the user preference and the resultant model can be used to predict the preference for any unseen new items for a specific user.

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. Classification models predict user preference of the item attributes. The supervised learning model-based approach treats recommendation tasks as a user-specific classification or regression problem and learns a classifier for the user's likes and dislikes based on the product features.

In the supervised learning model—based approach, each user has a personalized decision tree. 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. 11.16

The decision tree shown in Fig. 11.16 is a classification tree for the user Oliva using the item profile shown in Table 11.12. For another user, the tree would be different. For the ratings prediction problem, one could use regression models to predict the numerical ratings. Given that a user provides explicit ratings only for a few items, the challenge is inferring a pattern from a few positive ratings for items and from thousands of attributes of those items. Complementing the explicit ratings with the implicit outcomes (watching a movie, product views, items in the cart) will be helpful to boost the recommendations.

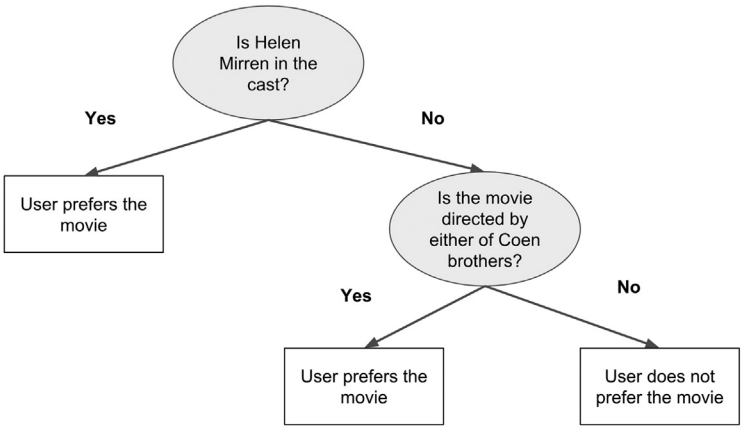

FIGURE 11.16
Personalized decision tree for one user in the system.

Supervised Learning Models - How to Implement

The supervised learning (classification) model approach for a content-based recommendation engine builds a classification model for each user. Hence, the model building is shown for one user and the process can be repeated in the loop for each user in the ratings matrix. The implementation step closely resembles the modeling process discussed in Chapter 4, Classification. The rule induction modeling technique is used in this recommender implementation. It can be replaced with various other classification or regression modeling techniques to suit the application and the data.

Dataset

The datasets used for the process is from MovieLens database with two datasets. The first dataset (ratings dataset) contains ratings, user ID and movie ID attributes. The second dataset (item dataset) contains limited metadata about each movie—movie ID, movie title, and concatenated genre attributes. To create the supervised learning model, these two datasets have to be merged to one dataset that has labels and attributes for one user.

Implementation steps

The outline of the complete RapidMiner process is shown in Fig. 11.17. The high-level process for a model-based approach to content-based recommendation engine contains: data preparation for classification modeling, model building, apply the model to test set and performance evaluation.

1. **Data Preparation:** The movie dataset structure is shown in Fig. 11.18A. Each movie has a title and the concatenated genres separated by | symbol. The first step is to convert this data structure to the structure

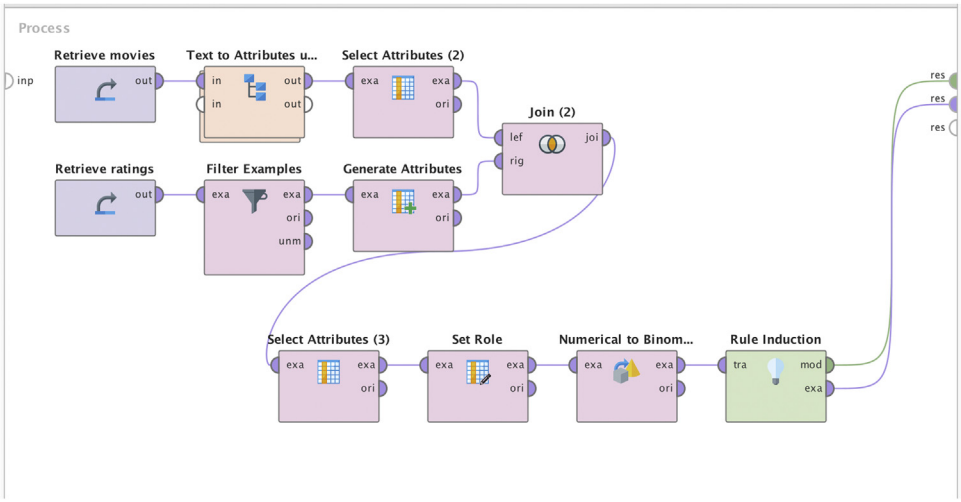

FIGURE 11.17
Classification process for one user in the system.

shown in Fig. 11.18B. The structure shown in Fig. 11.18B is conducive for classification modeling where each genre is shown as a separate attribute, with the value as 0 or 1 based on whether the movie is of that genre. For example, the movie *Toy story (movie id = 1)* is listed in

(A)

Row No.	movieId	title	genres
1	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
2	2	Jumanji (1995)	Adventure|Children|Fantasy
3	3	Grumpier Old Men (1995)	Comedy|Romance
4	4	Waiting to Exhale (1995)	Comedy|Drama|Romance
5	5	Father of the Bride Part II (19...	Comedy
6	6	Heat (1995)	Action|Crime|Thriller
7	7	Sabrina (1995)	Comedy|Romance
8	8	Tom and Huck (1995)	Adventure|Children
9	9	Sudden Death (1995)	Action
10	10	GoldenEye (1995)	Action|Adventure|Thriller

(B)

movieId	(no genres ...	?	action	adventur	anim	children	comedi	crime	documentari
1	0	0	0	1	1	1	1	0	0
2	0	0	0	1	0	1	0	0	0
3	0	0	0	0	0	0	1	0	0
4	0	0	0	0	0	0	1	0	0
5	0	0	0	0	0	0	1	0	0
6	0	0	1	0	0	0	0	1	0
7	0	0	0	0	0	0	1	0	0
8	0	0	0	1	0	1	0	0	0
9	0	0	1	0	0	0	0	0	0
10	0	0	1	1	0	0	0	0	0

(C)

Row No.	userId	rating_b	(no genres ...	?	action	adventur	anim	children
1	549	false	0	0	0	0	0	0
2	549	true	0	0	0	0	0	0
3	549	false	0	0	0	0	0	0
4	549	true	0	0	0	0	0	0
5	549	true	0	0	0	0	0	0
6	549	false	0	0	0	0	0	0
7	549	true	0	0	0	0	0	0
8	549	true	0	0	0	0	0	0
9	549	true	0	0	0	0	0	0
10	549	true	0	0	0	0	0	0

FIGURE 11.18

(A) Item profile with concatenated genre. (B) Item profile with genre as attributes. (C) Item profile with class label for the user 549.

following categories: Adventure, Comedy, Animation, Children, and Fantasy.

Text mining data processing is used to convert the data structure from Fig. 11.18A to B. In the text mining subprocess, the genre attribute is converted to text. Each word is tokenized from the concatenated genre text using the delimiter |. Each token is then case converted and stemmed to provide word vector for each movie. The text mining subprocess is shown in Fig. 11.19.

The movie-genre vector shown in Fig. 11.18B, can now be merged with the preference information of a user. The ratings matrix is filtered for one user, user 549, using the *Filter example* operator. To simplify the rating scale, the 0−5 scale is converted to boolean true (3 or more) or false (below 3) using the formula in *Generate Attributes*. A numeric rating label is preferable in production models because it contains more information than the boolean true/false. The five-point rating is converted from numeric to categorical just to illustrate a simple classification model for this implementation. The rating information for the user can be merged to the movie profile using the inner *Join* operator. In this process, the result set will just contain the titles rated by the user. The resultant dataset is shown in Fig. 11.18C, where rating_b is the preference class label and the rest of the columns are predictors.

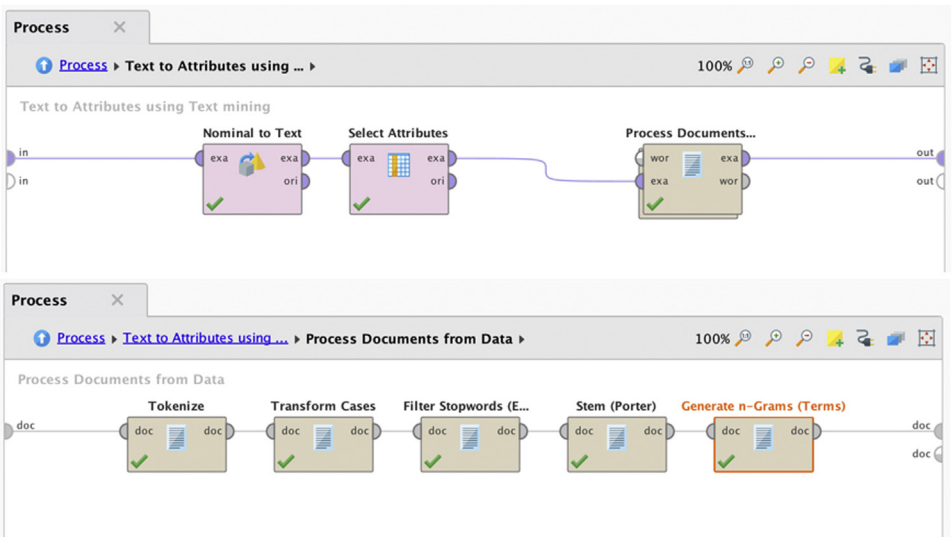

FIGURE 11.19

Subprocess to convert the item profile to classification training set.

2. **Modeling:** A rule induction is used for classification modeling. The rule induction model and parameters are discussed in Chapter 4, Classification. The parameters specified in this process are: criterion = information gain and sample ration = 0.9. A decision tree or logistic regression can be used as an alternative to the rule induction.

The whole process shown in Fig. 11.17 can be saved and executed. The result window shows the model from rule induction (Fig. 11.20). Ideally, this model should have been tested with a test dataset and performance evaluated. Based on the rule induction model, user 549 has a preference for the Crime genre. This model can be run against all other movies in the catalog which are not rated and the predicted rating can be used to recommend titles to the user 549.

The model built using the RapidMiner process is customized for one user, user 549. This entire process can be looped for each user and the model can be stored in a separate dataset. The looping process is discussed in Chapter 15 Getting started with RapidMiner. Each user in the system will have a personalized model. The effectiveness of the model depends on each user's past rating and is independent of ratings from other users.

Content-based recommendation engines are better at explaining why the system is making the recommendation because it has generalized the features the user is interested in. For example, the system might recommend *Apollo 13*, *Saving private Ryan*, and *Captain Phillips* because the user is interested in movies that have Tom Hanks in the cast. Unlike the collaborative filtering method, data from other users is not needed for the content-based systems, however, additional data from the items are essential. This feature is significant because when a new user is introduced into the system, the recommendation engine does not suffer from the cold start problem. When a new item is introduced, say a new movie, the attributes of the movie are already known. Hence, the addition of a new item or a user is quite seamless for the recommenders.

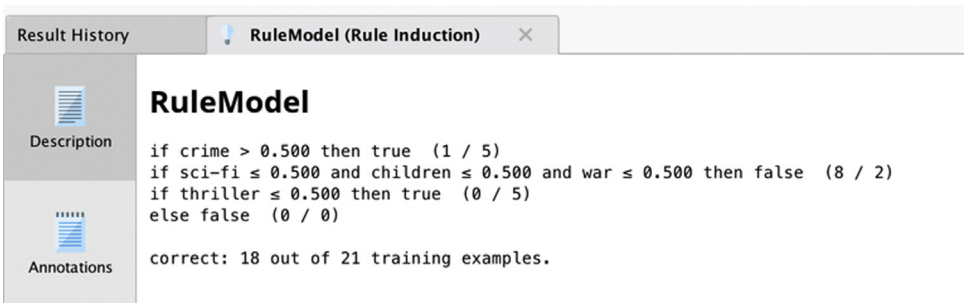

FIGURE 11.20
Rule induction model for one user in the system.

The key datasets involved in the recommendation, that is, the item profile, user profile or classification models, and the recommendation list, can be pre-computed. Since the main objective in many cases is finding the top recommended items instead of filling the complete ratings matrix, decision trees can focus only on the attributes relevant to the user.

Content-based recommenders tend to address unique preferences for the user. For example: users interested in Scandinavian crime thrillers. There may not be enough users who like this subgenre for collaborative filtering to be effective because a sizable cohort of users is needed to prefer these unique genres and other items in the catalog.

Even though rating information from other users are not necessary, an exhaustive item profile is essential for obtaining the relevant recommendations from content-based systems. The features of the item are hard to get to begin with and some attributes like genre taxonomy are difficult to master. Blockbuster items, are by definition, watched by a wider audience, beyond the fanatics of a particular genre. For example, one doesn't have to be a fan of science fiction to watch *Avatar*. Just because a user watched *Avatar* it doesn't mean that the recommender can be inundated with other science fiction movies. Special handling is required so the system doesn't conflate the success of blockbusters with the user preference for specific attributes in the blockbusters.

Content-based recommenders are content specific. For example, it makes logical sense to recommend Moroccan cooking utensils if the user has shown an interest in books focused on North African cuisine, if the ecommerce platform offers both the categories. Content-based systems will find this task to be difficult because the knowledge gained in a books category is hard to translate to a kitchen utensils category. Collaborative filtering is content agnostic.

11.4 HYBRID RECOMMENDERS

Each recommendation engine discussed in this chapter has a central idea and solves the problem of predicting the unseen user-item rating through a unique approach. 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. The diversity of the base models can be imparted by selecting different

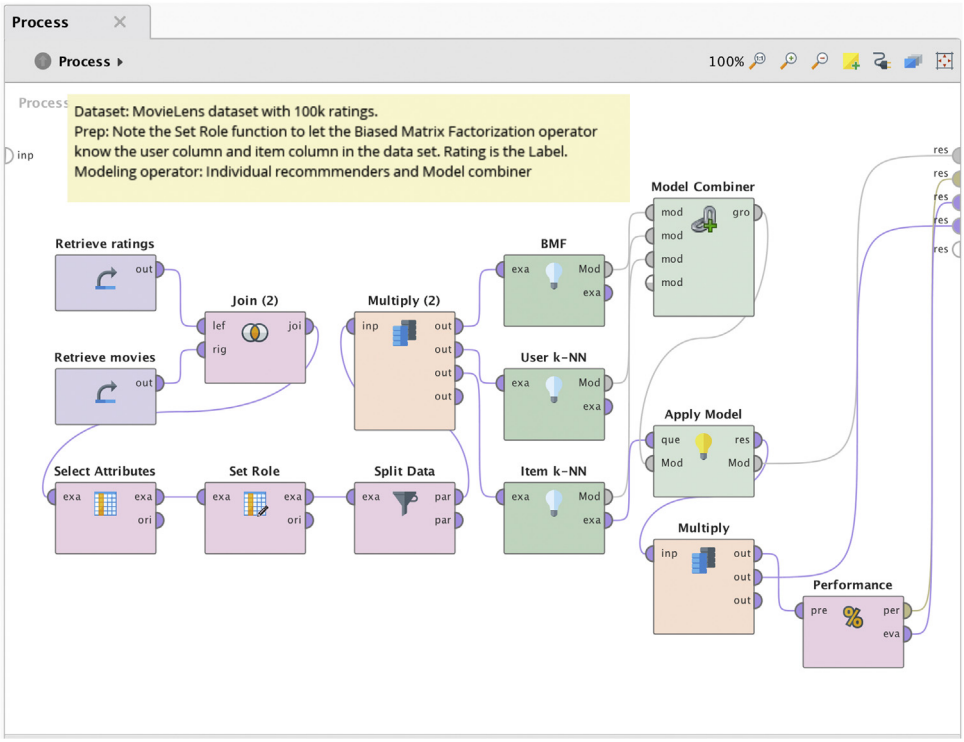

FIGURE 11.21
Hybrid recommenders using model combiner.

modeling techniques like neighborhood BMF, content-based and supervised model-based recommenders.

Fig. 11.21 shows the RapidMiner process of Hybrid recommender with base recommenders such as BMF, User K-NN and Item K-NN. The *Model Combiner* operator is the ensemble operator which combines all three models into one meta operator. It weighs the base models equally. All the base recommenders operate on the same training dataset. The performance of this hybrid recommender process for the MovieLens dataset is RMSE of 0.781 and MAE of 0.589. In this case, the performance of the Hybrid recommender is better than the results achieved with neighborhood or matrix factorization alone.

11.5 CONCLUSION

How does one know if the recommendations are relevant for the users? One would hope to see users taking up the offer of using the recommended items

(watching, purchasing, reading,...) more than users acting on a random selection of items. An A/B experimental testing of "with" and "without" recommendation engines can help determine the overall impact of the recommendation engine in the system.

The data used by recommendation engines is considered sensitive under any standards because it is at the user level and indicates a user's preferences. A great deal of information about a person can be deduced when one knows the books they read, movies they watch, or the products they buy, let alone the combination of all three! (Siegel, 2016) Users implicitly entrust enterprises to safeguard their private information by implicitly or explicitly agreeing to the terms and conditions. Users prefer personalized experiences and they are willing to give some information about themselves by explicit rating information and implicit views or search information. How is the need for personalization balanced with the privacy of users? Transparency can be the enabler that strikes a balance between privacy and the benefits of personalization. Users have the right to know what inputs about them are used for recommenders: list of previously liked titles, searches, or any other clickstream data used as an input for recommenders. Users can inspect or remove their preference for the titles. The explainability of recommendations increases the trust that users have in the system. For example: one might like a list of titles because they have watched *Fargo*. In some jurisdictions for certain applications, users have the legal right to know why a recommendation or disposition is made about a user.

Personalization can lead to filter bubble, which is a state of intellectual isolation (Bozdag, 2013). Hyper personalized recommendation engines keep feeding items that a user is interested in and attenuate the usage of more diverse set of items. Personalization in news feeds causes reinforcement of one's political leaning and results in an echo chamber of viewpoints. Personalization, in a way, acts as a censor to filter out viewpoints that the user doesn't explicitly prefer. The algorithms that control personalization, especially in news sites and social media, play a significant role in shaping the political opinions of people. Enterprises have responded to the bias of filter bubbles by introducing the equivalent of best sellers: trending now, top news, etc.,—where a selection of un-personalized item selections are provided to the users, in addition to the personalized recommendations.

Summary of the Types of Recommendation Engines

See Table 11.13 for a summary of different 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.		

References

Bobadilla, J., Ortega, F., Hernando, A., & Gutiérrez, A. (2013). Recommender systems survey. *Knowledge-Based Systems*, *46*, 109–132. Available from https://doi.org/10.1016/J.KNOSYS.2013.03.012.

Bozdag, E. (2013). Bias in algorithmic filtering and personalization. *Ethics and Information Technology*, *15*(3), 209–227. Available from https://doi.org/10.1007/s10676-013-9321-6.

Cakir, O., & Aras, M. E. (2012). A Recommendation engine by using association rules. *Procedia–Social and Behavioral Sciences*, *62*, 452–456. Available from https://doi.org/10.1016/j.sbspro.2012.09.074.

Das, A.S., Datar, M., Garg, A., & Rajaram, S. (2007). Google news personalization. In: *Proceedings of the 16th international conference on World Wide Web–WWW '07* (p. 271). New York: ACM Press. <https://doi.org/10.1145/1242572.1242610>.

Gemulla, R., Nijkamp, E., Haas, P.J., & Sismanis, Y. (2011). Large-scale matrix factorization with distributed stochastic gradient descent. In: *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining–KDD '11* (p. 69). New York: ACM Press. <https://doi.org/10.1145/2020408.2020426>.

Gupta, P., Goel, A., Lin, J., Sharma, A., Wang, D., & Zadeh, R. (2013). WTF: The who to follow service at twitter. In: *Proceedings of the 22nd international conference on World Wide Web - WWW '13* (pp. 505–514). New York: ACM Press. <https://doi.org/10.1145/2488388.2488433>.

Hu, Y., Koren, Y., & Volinsky, C. (2008). Collaborative filtering for implicit feedback datasets. In *2008 eighth IEEE international conference on data mining* (pp. 263–272). IEEE. <https://doi.org/10.1109/ICDM.2008.22>.

Isinkaye, F. O., Folajimi, Y. O., & Ojokoh, B. A. (2015). Recommendation systems: Principles, methods and evaluation. *Egyptian Informatics Journal*, *16*(3), 261–273. Available from https://doi.org/10.1016/J.EIJ.2015.06.005.

Koren, Y., Bell, R., & Volinsky, C. (2009). Matrix factorization techniques for recommender systems. *Computer*, *42*(8), 30–37. Available from https://doi.org/10.1109/MC.2009.263.

Lee, J., Sun, M., & Lebanon, G. (2012). *A comparative study of collaborative filtering algorithms*, pp. 1–27. Retrieved from <http://arxiv.org/abs/1205.3193>.

Mihelčić, M., Antulov-Fantulin, N., Bošnjak, M., & Šmuc, T. (2012). Extending RapidMiner with recommender systems algorithms. In: *RapidMiner community meeting and conference*.

Schafer, J. Ben, Konstan, J. A., & Riedl, J. (2001). E-commerce recommendation applications. *Data Mining and Knowledge Discovery*, *5*(1/2), 115–153. Available from https://doi.org/10.1023/A:1009804230409.

Siegel, Eric (2016). *Predictive analytics: The power to predict who will click, buy, lie, or die*. John Wiley & Sons Incorporated.

Thorson, E. (2008). Changing patterns of news consumption and participation. *Information, Communication & Society*, *11*(4), 473–489. Available from https://doi.org/10.1080/13691180801999027.

Underwood, C. (2017). *Use cases of recommendation systems in business–current applications and methods*. Retrieved on June 17, 2018, From <https://www.techemergence.com/use-cases-recommendation-systems/>.

Weise, E. (2017). That review you wrote on Amazon? Priceless. Retrieved on October 29, 2018, From < https://www.usatoday.com/story/tech/news/2017/03/20/review-you-wrote-amazon-priceless/99332602/ > .