In reviewing these rules, we see that the information can be compressed. First, rule #1, which appears from the confidence level to be a very promising rule, is probably meaningless. It says: "If Italian cooking books have been purchased, then cookbooks are purchased." It seems likely that Italian cooking books are simply a subset of cookbooks. Rules 14,15, and 16 involve the same trio of books, with different antecedents and consequents. The same is true of rules 17 and 18 as well as 19 and 20. (Pairs and groups like this are easy to track down by looking for rows that share the same support.) This does not mean that the rules are not useful. On the contrary, it can reduce the number of itemsets to be considered for possible action from a business perspective.

14.2 Collaborative Filtering[3]

Recommendation systems are a critically important part of websites that offer a large variety of products or services. Examples include Amazon.com, which offers millions of different products; Netflix has thousands of movies for rental; Google searches over huge numbers of webpages; Internet radio websites such as Spotify and Pandora include a large variety of music albums by various artists; travel websites offer many destinations and hotels; social network websites have many groups. The recommender engine provides personalized recommendations to a user based on the user's information as well as on similar users' information. Information means behaviors indicative of preference, such as purchase, ratings, and clicking.

The value that recommendation systems provide to users helps online companies convert browsers into buyers, increase cross-selling, and build loyalty.

Collaborative filtering is a popular technique used by such recommendation systems. The term *collaborative filtering* is based on the notions of identifying relevant items for a specific user from the very large set of items ("filtering") by considering preferences of many users ("collaboration").

The Fortune.com article "Amazon's Recommendation Secret" (June 30, 2012) describes the company's use of collaborative filtering not only for providing personalized product recommendations, but also for customizing the entire website interface for each user:

> At root, the retail giant's recommendation system is based on a number of simple elements: what a user has bought in the past, which items they have in their virtual shopping cart, items they've rated and liked, and what other customers have viewed and purchased. Amazon calls this homegrown math "item-to-item collaborative filtering," and it's used this algorithm to heavily customize the browsing experience for returning customers.

Data Type and Format

Collaborative filtering requires availability of all item–user information. Specifically, for each item–user combination, we should have some measure of the user's preference for that item. Preference can be a numerical rating or a binary behavior such as a purchase, a 'like', or a click.

For n users (u_1, u_2, …, u_n) and p items (i_1, i_2, …i_p), we can think of the data as an $n \times p$ matrix of n rows (users) by p columns (items). Each cell includes the rating or the binary event corresponding to the user's preference of the item (see schematic in Table 14.9). Typically not every user purchases or rates every item, and therefore a purchase matrix will have many zeros (it is sparse), and a rating matrix will have many missing values. Such missing values sometimes convey "uninterested" (as opposed to non-missing values that convey interest).

Table 14.9 Schematic of matrix format with ratings data

	Item ID			
User ID	I_1	I_2	$\cdots$	I_p
U_1	$r_{1,1}$	$r_{1,2}$	$\cdots$	$r_{1,p}$
U_2	$r_{2,1}$	$r_{2,2}$	$\cdots$	$r_{2,p}$
$\vdots$				
U_n	$r_{n,1}$	$r_{n,2}$	$\cdots$	$r_{n,p}$

When both n and p are large, it is not practical to store the preferences data ($r_{u,i}$) in an $n \times p$ table. Instead, the data can be stored in many rows of triplets of the form (U_u, I_i, $r_{u,i}$), where each triplet contains the user ID, the item ID, and the preference information.

Example 3: Netflix Prize Contest

We have been considering both association rules and collaborative filtering as unsupervised techniques, but it is possible to judge how well they do by looking at holdout data to see what users purchase and how they rate items. The famous Netflix contest, mentioned in Chapter 13, did just this and provides a useful example to illustrate collaborative filtering, though the extension into training and validation is beyond the scope of this book.

In 2006, Netflix, the largest movie rental service in North America, announced a one million USD contest (www.netflixprize.com) for the purpose of improving its recommendation system called *Cinematch*. Participants were provided with a number of datasets, one for each movie. Each dataset included all the customer ratings for that movie (and the timestamp). We can think of one large combined dataset of the form [customer ID, movie ID, rating, date] where each record includes the rating given by a certain customer to a certain movie on a certain date. Ratings were on a 1–5 star scale. Contestants were asked to develop a recommendation algorithm that would improve over the existing Netflix system. Table 14.10 shows a small sample from the contest data, organized in matrix format. Rows indicate customers and columns are different movies.

Table 14.10 Sample of records from the Netflix Prize contest, for a subset of 10 customers and 9 movies

	Movie ID								
Customer ID	1	5	8	17	18	28	30	44	48
30878	4	1			3	3	4	5	
124105	4								
822109	5								
823519	3		1	4		4	5		
885013	4	5							
893988	3						4	4	
1248029	3					2	4		3
1503895	4								
1842128	4						3		
2238063	3								

It is interesting to note that the winning team was able to improve their system by considering not just the ratings for a movie, but *whether* a movie was rated by a particular customer or not. In other words, the information on which movies a customer decided to rate turned out to be critically informative of customers' preferences, more than simply considering the 1–5 rating information[4]:

> Collaborative filtering methods address the sparse set of rating values. However, much accuracy is obtained by also looking at other features of the data. First is the information on which movies each user chose to rate, regardless of specific rating value ("the binary view"). This played a decisive role in our 2007 solution, and reflects the fact that the movies to be rated are selected deliberately by the user, and are not a random sample.

This is an example where converting the rating information into a binary matrix of rated/unrated proved to be useful.

User-Based Collaborative Filtering: "People Like You"

One approach to generating personalized recommendations for a user using collaborative filtering is based on finding users with similar preferences, and recommending items that they liked but the user hasn't purchased. The algorithm has two steps:

1. Find users who are most similar to the user of interest (neighbors). This is done by comparing the preference of our user to the preferences of other users.
2. Considering only the items that the user has *not* yet purchased, recommend the ones that are most preferred by the user's neighbors.

This is the approach behind Amazon's "Customers Who Bought This Item Also Bought..." (see Figure 14.1). It is also used in a Google search for generating the "Similar pages" link shown near each search result.

Step 1 requires choosing a distance (or proximity) metric to measure the distance between our user and the other users. Once the distances are computed, we can use a threshold on the distance or on the number of required neighbors to determine the nearest neighbors to be used in Step 2. This approach is called "user-based top-*N* recommendation."

A nearest-neighbors approach measures the distance of our user to each of the other users in the database, similar to the *k*-nearest-neighbors algorithm (see Chapter 7). The Euclidean distance measure we discussed in that chapter does not perform as well for collaborative filtering as some other measures. A popular proximity measure between two users is the Pearson correlation between their ratings. We denote the ratings of items $I_1, \ldots, I_p$ by user U_1 as $r_{1,1}, r_{1,2}, \ldots, r_{1,p}$ and their average by $\overline{r}_1$. Similarly, the ratings by user U_2 are $r_{2,1}, r_{2,2}, \ldots, r_{2,p}$, with average $\overline{r}_2$. The correlation proximity between the two users is defined by

$$\text{Corr}(U_1, U_2) = \frac{\sum (r_{1,i} - \overline{r}_1)(r_{2,i} - \overline{r}_2)}{\sqrt{\sum (r_{1,i} - \overline{r}_1)^2}\sqrt{\sum (r_{2,i} - \overline{r}_2)^2}}, \tag{14.1}$$

where the summations are only over the items co-rated by both users.

To illustrate this, let us compute the correlation between customer 30878 and customer 823519 in the small Netflix sample in Table 14.10. We'll assume that the data shown in the table is the entire information. First, we compute the average rating by each of these users:

$$\overline{r}_{30878} = (4 + 1 + 3 + 3 + 4 + 5)/6 = 3.333$$
$$\overline{r}_{823519} = (3 + 1 + 4 + 4 + 5)/5 = 3.4$$

Note that the average is computed over a different number of movies for each of these customers, because they each rated a different set of movies. The average for a customer is computed over *all* the movies that a customer rated. The calculations for the correlation involve the departures from the average, but *only for the items that they co-rated*. In this case the co-rated movie IDs are 1, 28, and 30:

$$\text{Corr}(U_{30878}, U_{823519}) =$$
$$\frac{(4 - 3.333)(3 - 3.4) + (3 - 3.333)(4 - 3.4) + (4 - 3.333)(5 - 3.4)}{\sqrt{(4 - 3.333)^2 + (3 - 3.333)^2 + (4 - 3.333)^2}\sqrt{(3 - 3.4)^2 + (4 - 3.4)^2 + (5 - 3.4)^2}}$$
$$= 0.6/1.75 = 0.34$$

The same approach can be used when the data are in the form of a binary matrix (e.g., purchased or didn't purchase.)

Another popular measure is a variant of the Pearson correlation called *cosine similarity*. It differs from the correlation formula by not subtracting the means. Subtracting the mean in the correlation formula adjusts for users' different overall approaches to rating—for example, a customer who always rates highly vs. one who tends to give low ratings.[5]

For example, the cosine similarity between the two Netflix customers is:

$$\text{Cos Sim}(U_{30878}, U_{823519}) = \frac{4 \times 3 + 3 \times 4 + 4 \times 5}{\sqrt{4^2 + 3^2 + 4^2}\sqrt{3^2 + 4^2 + 5^2}}$$
$$= 44/45.277 = 0.972$$

Note that when the data are in the form of a binary matrix, say, for purchase or no-purchase, the cosine similarity must be calculated over all items that either user has purchased; it cannot be limited to just the items that were co-purchased.

> Collaborative filtering suffers from what is called a *cold start*: it cannot be used as is to create recommendations for new users or new items. For a user who rated a single item, the correlation coefficient between this and other users (in user-generated collaborative filtering) will have a denominator of zero and the cosine proximity will be 1 regardless of the rating. In a similar vein, users with just one item, and items with just one user, do not qualify as candidates for nearby neighbors.

For a user of interest, we compute his/her similarity to each of the users in our database using a correlation, cosine similarity, or another measure. Then, in Step 2, we look only at the *k* nearest users, and among all the other items that similarity, or another measure. Then, in Step 2, we look only at the k nearest users, and among all the other items that they rated/purchased, we choose the best one and recommend it to our user. What is the best one? For binary purchase data, it is the item most purchased. For rating data, it could be the highest rated, most rated, or a weighting of the two.

The nearest-neighbors approach can be computationally expensive when we have a large database of users. One solution is to use clustering methods (see Chapter 15) to group users into homogeneous clusters in terms of their preferences, and then to measure the distance of our user to each of the clusters. This approach places the computational load on the clustering step that can take place earlier and offline; it is then cheaper (and faster) to compare our user to each of the clusters in real time. The price of clustering is less accurate recommendations, because not all the members of the closest cluster are the most similar to our user.

Item-Based Collaborative Filtering

When the number of users is much larger than the number of items, it is computationally cheaper (and faster) to find similar items rather than similar users. Specifically, when a user expresses interest in a particular item, the item-based collaborative filtering algorithm has two steps:

1. Find the items that were co-rated, or co-purchased, (by any user) with the item of interest.
2. Recommend the most popular or correlated item(s) among the similar items.

Similarity is now computed between items, instead of users. For example, in our small Netflix sample (Table 14.10), the correlation between movie 1 (with average $\bar{r}_1 = 3.7$) and movie 5 (with average $\bar{r}_5 = 3$) is:

$$\text{Corr}(I_1, I_5) = \frac{(4-3.7)(1-3)+(4-3.7)(5-3)}{\sqrt{(4-3.7)^2+(4-3.7)^2}\sqrt{(1-3)^2+(5-3)^2}} = 0$$

The zero correlation is due to the two opposite ratings of movie 5 by the users who also rated 1. One user rated it 5 stars and the other gave it a 1 star.

In like fashion, we can compute similarity between all the movies. This can be done offline. In real time, for a user who rates a certain movie highly, we can look up the movie correlation table and recommend the movie with the highest positive correlation to the user's newly rated movie.

According to an industry report[6] by researchers who developed the Amazon item-to-item recommendation system,

> "[The item-based] algorithm produces recommendations in real time, scales to massive data sets, and generates high-quality recommendations."

The disadvantage of item-based recommendations is that there is less diversity between items (compared to users' taste), and therefore, the recommendations are often obvious.

Table 14.11 shows R code for collaborative filtering on a simulated data similar to the Netflix data using the recommenderlab package.

Table 14.11 Collaborative Filtering in R

code for collaborative filtering

```
# simulate matrix with 1000 users and 100 movies
m <- matrix(nrow = 1000, ncol = 100)
# simulated ratings (1% of the data)
m[sample.int(100*1000, 1000)] <- ceiling(runif(1000, 0, 5))
## convert into a realRatingMatrix
r <- as(m, "realRatingMatrix")

library(recommenderlab)
# user-based collaborative filtering
UB.Rec <- Recommender(r, "UBCF")
pred <- predict(UB.Rec, r, type="ratings")
as(pred, "matrix")
```

```
# item-based collaborative filtering
IB.Rec <- Recommender(r, "IBCF")
pred <- predict(IB.Rec, r, type="ratings")
as(pred, "matrix")
```

Advantages and Weaknesses of Collaborative Filtering

Collaborative filtering relies on the availability of subjective information regarding users' preferences. It provides useful recommendations, even for "long tail" items, if our database contains sufficient similar users (not necessarily many, but at least a few per user), so that each user can find other users with similar tastes. Similarly, the data should include sufficient per-item ratings or purchases. One limitation of collaborative filtering is therefore that it cannot generate recommendations for new users, nor for new items. There are various approaches for tackling this challenge.

User-based collaborative filtering looks for similarity in terms of highly-rated or preferred items. However, it is blind to data on low-rated or unwanted items. We can therefore not expect to use it as is for detecting unwanted items.

User-based collaborative filtering helps leverage similarities between people's tastes for providing personalized recommendations. However, when the number of users becomes very large, collaborative filtering becomes computationally difficult. Solutions include item-based algorithms, clustering of users, and dimension reduction. The most popular dimension reduction method used in such cases is singular value decomposition (SVD), a computationally superior form of principal components analysis (see Chapter 4).

Although the term "prediction" is often used to describe the output of collaborative filtering, this method is unsupervised by nature. It can be used to generate predicted ratings or purchase indication for a user, but usually we do not have the true outcome value in practice. One important way to improve recommendations generated by collaborative filtering is by getting user feedback. Once a recommendation is generated, the user can indicate whether the recommendation was adequate or not. For this reason, many recommender systems entice users to provide feedback on their recommendations.

Collaborative Filtering vs. Association Rules

While collaborative filtering and association rules are both unsupervised methods used for generating recommendations, they differ in several ways:

Frequent itemsets vs. personalized recommendations Association rules look for frequent item combinations and will provide recommendations only for those items. In contrast, collaborative filtering provides personalized recommendations for every item, thereby catering to users with unusual taste. In this sense, collaborative filtering is useful for capturing the "long tail" of user preferences, while association rules look for the "head." This difference has implications for the data needed: association rules require data on a very large number of "baskets" (transactions) in order to find a sufficient number of baskets that contain certain combinations of items. In contrast, collaborative filtering does not require many "baskets," but does require data on as many items as possible for many users. Also, association rules operate at the basket level (our database can include multiple transactions for each user), while collaborative filtering operates at the user level.

Because association rules produce generic, impersonal rules (association-based recommendations such as Amazon's "Frequently Bought Together" display the same recommendations to all users searching for a specific item), they can be used for setting common strategies such as product placement in a store or sequencing of diagnostic tests in hospitals. In contrast, collaborative filtering generates user-specific recommendations (e.g., Amazon's "Customers Who Bought This Item Also Bought...") and is therefore a tool designed for personalization.

Transactional data vs. user data Association rules provide recommendations of items based on their co-purchase with other items in *many transactions/baskets*. In contrast, collaborative filtering provides recommendations of items based on their co-purchase or co-rating by even a small number of other *users*. Considering distinct baskets is useful when the same items are purchased over and over again (e.g., in grocery shopping). Considering distinct users is useful when each item is typically purchased/rated once (e.g., purchases of books, music, and movies).

Binary data and ratings data Association rules treat items as binary data (1 = purchase, 0 = nonpurchase), whereas collaborative filtering can operate on either binary data or on numerical ratings.

Two or more items In association rules, the antecedent and consequent can each include one or more items (e.g., IF milk THEN cookies and cornflakes). Hence, a recommendation might be a bundle of the item of interest with

IF milk THEN cookies and cornflakes”). Hence, a recommendation might be a bundle of the item of interest with multiple items (“buy milk, cookies, and cornflakes and receive 10% discount”). In contrast, in collaborative filtering, similarity is measured between *pairs* of items or pairs of users. A recommendation will therefore be either for a single item (the most popular item purchased by people like you, which you haven’t purchased), or for multiple single items which do not necessarily relate to each other (the top two most popular items purchased by people like you, which you haven’t purchased).

These distinctions are sharper for purchases and recommendations of non-popular items, especially when comparing association rules to user-based collaborative filtering. When considering what to recommend to a user who purchased a popular item, then association rules and item-based collaborative filtering might yield the same recommendation for a single item. But a user-based recommendation will likely differ. Consider a customer who purchases milk every week as well as gluten-free products (which are rarely purchased by other customers). Suppose that using association rules on the transaction database we identify the rule “IF milk THEN cookies.” Then, the next time our customer purchases milk, s/he will receive a recommendation (e.g., a coupon) to purchase cookies, whether or not s/he purchased cookies, and irrespective of his/her gluten-free item purchases. In item-based collaborative filtering, we would look at all items co-purchased with milk across all users and recommend the most popular item among them (which was not purchased by our customer). This might also lead to a recommendation of cookies, because this item was not purchased by our customer.[7] Now consider user-based collaborative filtering. User-based collaborative filtering searches for similar customers—those who purchased the same set of items—and then recommend the item most commonly purchased by these neighbors, which *was not purchased by our customer*. The user-based recommendation is therefore unlikely to recommend cookies and more likely to recommend popular gluten-free items that the customer has not purchased.