Collaborative Filtering Recommender Systems

J. Ben Schafer, Dan Frankowski, Jon Herlocker, and Shilad Sen

Department of Computer Science
University of Northern Iowa
Cedar Falls, IA 50614-0507
schafer@cs.uni.edu

Department of Computer Science
University of Minnesota
4-192 EE/CS Building
200 Union St. SE
Minneapolis, MN 55455
{dfrankow , ssen } @ cs.umn.edu

School of Electrical Engineering and Computer Science
Oregon State University
102 Dearborn Hall
Corvallis, OR 97331
herlock@eecs.oregonstate.edu

Abstract. One of the potent personalization technologies powering the adaptive web is collaborative filtering. Collaborative filtering (CF) is the process of filtering or evaluating items through the opinions of other people. CF technology brings together the opinions of large interconnected communities on the web, supporting filtering of substantial quantities of data. In this chapter we introduce the core concepts of collaborative filtering, its primary uses for users of the adaptive web, the theory and practice of CF algorithms, and design decisions regarding design of rating systems and acquisition of ratings. We also discuss how to evaluate CF systems, and the evolution of rich interaction interfaces. We close the chapter with discussions of the challenges of privacy particular to a CF recommendation service and important open research questions in the field.

1 Introduction

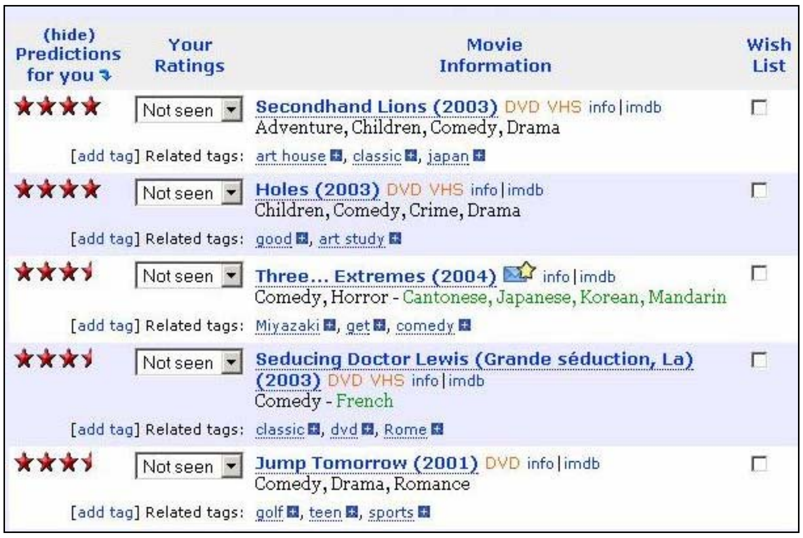
Collaborative Filtering is the process of filtering or evaluating items using the opinions of other people. While the term collaborative filtering (CF) has only been around for a little more than a decade, CF takes its roots from something humans have been doing for centuries - sharing opinions with others.

For years, people have stood over the back fence or in the office break room and discussed books they have read, restaurants they have tried, and movies they have seen – then used these discussions to form opinions. For example, when enough of Amy's colleagues say they liked the latest release from Hollywood, she might decide that she also should see it. Similarly, if many of them found it a disaster, she might decide to spend her money elsewhere. Better yet, Amy might observe that Matt recommends the types of films that she finds enjoyable, Paul has a history of recommending films that she despises, and Margaret just seems to recommend everything. Over time, she learns whose opinions she should listen to and how these opinions can be applied to help her determine the quality of an item.

Computers and the web allow us to advance beyond simple word-of-mouth. Instead of limiting ourselves to tens or hundreds of individuals the Internet allows us to consider the opinions of thousands. The speed of computers allows us to process these opinions in real time and determine not only what a much larger community thinks of an item, but also develop a truly personalized view of that item using the opinions most appropriate for a given user or group of users.

1.1 Core Concepts

While this chapter considers a variety of CF systems, we introduce the topic through MovieLens[1]. MovieLens is a collaborative filtering system for movies. A user of MovieLens rates movies using 1 to 5 stars, where 1 is “Awful” and 5 is “Must See”. MovieLens then uses the ratings of the community to recommend other movies that user might be interested in (**Figure 1**), predict what that user might rate a movie, or perform other tasks.



(hide) Predictions for you	Your Ratings	Movie Information	Wish List
★★★★	Not seen	Secondhand Lions (2003) DVD VHS info | imdb Adventure, Children, Comedy, Drama [add tag] Related tags: art house, classic, japan	☐
★★★★	Not seen	Holes (2003) DVD VHS info | imdb Children, Comedy, Crime, Drama [add tag] Related tags: good, art study	☐
★★★½	Not seen	Three... Extremes (2004) info | imdb Comedy, Horror - Cantonese, Japanese, Korean, Mandarin [add tag] Related tags: Miyazaki, get, comedy	☐
★★★½	Not seen	Seducing Doctor Lewis (Grande séduction, La) (2003) DVD VHS info | imdb Comedy - French [add tag] Related tags: classic, dvd, Rome	☐
★★★½	Not seen	Jump Tomorrow (2001) DVD info | imdb Comedy, Drama, Romance [add tag] Related tags: golf, teen, sports	☐

Figure 1: MovieLens uses collaborative filtering to predict that this user is likely to rate the movie “Holes” 4 out of 5 stars.

[1] http://www.movielens.org/

To be more formal, a *rating* consists of the association of two things – user and item. One way to visualize ratings is as a matrix (**Table 1**). Without loss of generality, a ratings matrix consists of a table where each row represents a user, each column represents a specific movie, and the number at the intersection of a row and a column represents the user's rating value. The absence of a rating score at this intersection indicates that that user has not yet rated the item.

Table 1: A MovieLens ratings matrix. Amy rated the movie Sideways a 5. Matt has not seen The Matrix.

	The Matrix	**Speed**	**Sideways**	**Brokeback Mountain**
Amy	1	2	5	
Matt		3	5	4
Paul	5	5	2	1
Cliff	5	5	5	5

The term *user* refers to any individual who provides ratings to a system. Most often, we use this term to refer to the people using a system to receive information (e.g., recommendations) although it also refers to those who provided the data (ratings) used in generating this information.

CF systems determine the quality of *items*. Items can consist of anything for which a human can provide a rating, such as art, books, CDs, journal articles, or vacation destinations.

Ratings in a collaborative filtering system can take on a variety of forms.

- Scalar ratings can consist of either numerical ratings, such as the 1-5 stars provided in MovieLens or ordinal ratings such as strongly agree, agree, neutral, disagree, strongly disagree.
- Binary ratings model choices between agree/disagree or good/bad.
- Unary ratings can indicate that a user has observed or purchased an item, or otherwise rated the item positively. The absence of a rating indicates that we have no information relating the user to the item (perhaps they purchased the item somewhere else).

Ratings may be gathered through explicit means, implicit means, or both. *Explicit ratings* are those where a user is asked to provide an opinion on an item. *Implicit ratings* are those inferred from a user's actions. For example, a user who visits a product page perhaps has some interest in that product while a user who subsequently purchases the product may have a much stronger interest in that product. The issues of design decisions and tradeoffs regarding collection of different types of ratings are discussed in Section 4.

1.2 The Beginning of Collaborative Filtering

As a formal area of research, collaborative filtering got its start as a means to handle the shifting nature of text repositories. As content bases grew from mostly "official" content, such as libraries and corporate document sets, to "informal" content such as discussion lists and e-mail archives, the challenge of finding quality items shifted as well. Pure content-based techniques were often inadequate at helping users find the documents they wanted. Keyword-based representations could do an adequate job of describing the topic of documents, but could do little to help users understand the nature or quality of those documents. Hence, a keyword search for “Chicago Rocks” might yield not only scholarly articles by the Chicago Rocks and Minerals Society but also the “shallower” posting to a music bulletin board regarding the 1970s rock band.

In the early 1990s there seemed to be two possible solutions to this new challenge:

1. wait for improvements in artificial intelligence that would allow better automated classification of documents, or
2. bring human judgment into the loop.

While the challenges of automated classification have yet to be overcome, human judgment has proved valuable and relatively easy to incorporate into semi-automated systems[2].

The Tapestry system, developed at Xerox PARC, took the first step in this direction by incorporating user actions and opinions into a message database and search system [17]. Tapestry stored the contents of messages, along with metadata about authors, readers, and responders. It also allowed any user to store annotations about messages, such as "useful survey" or "Phil should see this!" Tapestry users could form queries that combined basic textual information (e.g. contains the phrase "recommender systems") with semantic metadata queries (e.g. written by John OR replied to by Joe) and annotation queries (e.g. marked as "excellent" by Chris). This model has become known as pull-active collaborative filtering, because it is the responsibility of the user who desires recommendations to actively pull the recommendations out of the database.

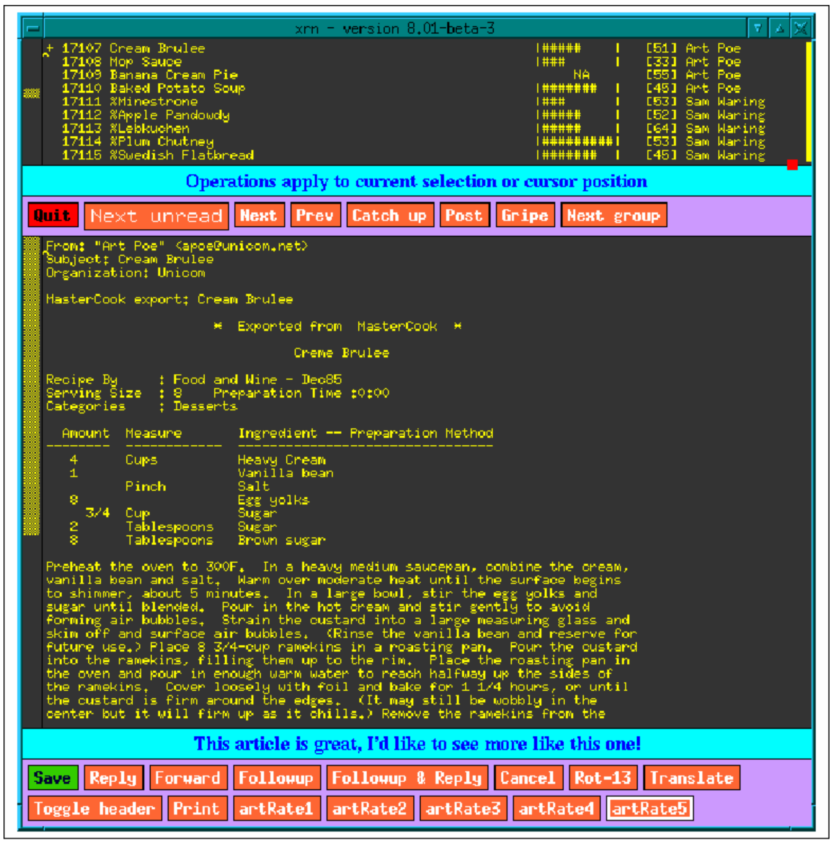
Soon after the emergence of Tapestry, other researchers began to recognize the potential for exploiting the human "information hubs" that seem to naturally occur within organizations. Maltz and Ehrlich [37] developed a push-active collaborative filtering recommender system that made it easy for a person reading a document to push that document on to others in the organization who should see it. This type of push-recommender role has become popular, with many people today serving as "joke hubs" who receive jokes from all over and forward them to those they believe would appreciate them (though often with far less discriminating thought than was envisioned).

A limitation of active collaborative filtering systems is that they require a community of people who know each other. Pull-active systems require that the user

[2] For a slightly more broad discussion on the differences between collaborative filtering and content filtering, see Section 2.4 of this chapter.

know whose opinions to trust; push-active systems require that the user know to whom particular content may be interesting. Automated collaborative filtering (ACF) systems relieve users of this burden by using a database of historical user opinions to automatically match each individual to others with similar opinions.

The early ACF systems included GroupLens [46,30] in the domain of Usenet newsgroup articles, Ringo [52] in the domain of music and musical artists, and Bellcore's Video Recommender [24] in the domain of movies. While a more formal discussion of recommendation algorithms follows in Section 3, each of these systems follow a process of gathering ratings from users, computing the correlations between pairs of users to identify a user's "neighbors" in taste space, and combining the ratings of those neighbors to make recommendations. GroupLens used a very explicit interface where ratings of Usenet newsgroup articles were entered manually by keystroke or button, and ratings were displayed numerically or graphically (**Figure 2**). Taking this a step further, both Ringo and Video Recommender were accessible through the web and email and provided simple features for community interaction.


xrn - version 8.01-beta-3

```
+ 17107 Cream Brulee                              |#####     |   [51] Art Poe
  17108 Mop Sauce                                 |###       |   [33] Art Poe
  17109 Banana Cream Pie                              NA         [55] Art Poe
  17110 Baked Potato Soup                         |#######   |   [45] Art Poe
  17111 %Minestrone                               |###       |   [53] Sam Waring
  17112 %Apple Pandowdy                           |#####     |   [52] Sam Waring
  17113 %Lebkuchen                                |#####     |   [64] Sam Waring
  17114 %Plum Chutney                             |##########|   [53] Sam Waring
  17115 %Swedish Flatbread                        |#######   |   [45] Sam Waring
```

Operations apply to current selection or cursor position
Quit
Next unread
Next
Prev
Catch up
Post
Gripe
Next group

```
From: "Art Poe" <apoe@unicom.net>
Subject: Cream Brulee
Organization: Unicom

MasterCook export: Cream Brulee

                      *  Exported from  MasterCook  *

                                 Creme Brulee

Recipe By     : Food and Wine - Dec85
Serving Size  : 8    Preparation Time :0:00
Categories    : Desserts

  Amount  Measure       Ingredient -- Preparation Method
--------  ------------  --------------------------------
   4      Cups          Heavy Cream
   1                    Vanilla bean
          Pinch         Salt
   8                    Egg yolks
     3/4  Cup           Sugar
   2      Tablespoons   Sugar
   8      Tablespoons   Brown sugar

Preheat the oven to 300F.  In a heavy medium saucepan, combine the cream,
vanilla bean and salt.  Warm over moderate heat until the surface begins
to shimmer, about 5 minutes.  In a large bowl, stir the egg yolks and
sugar until blended.  Pour in the hot cream and stir gently to avoid
forming air bubbles.  Strain the custard into a large measuring glass and
skim off and surface air bubbles.  (Rinse the vanilla bean and reserve for
future use.) Place 8 3/4-cup ramekins in a roasting pan.  Pour the custard
into the ramekins, filling them up to the rim.  Place the roasting pan in
the oven and pour in enough warm water to reach halfway up the sides of
the ramekins.  Cover loosely with foil and bake for 1 1/4 hours, or until
the custard is firm around the edges.  (It may still be wobbly in the
center but it will firm up as it chills.) Remove the ramekins from the
```

Figure 2: A modified Xrn news reader. The GroupLens project added article predictions (lines of ### on the top right) and article rating buttons (bottom).

1.3 Collaborative Filtering and the Adaptive Web

These early collaborative filtering systems were designed to explicitly provide users with information about items. That is, users visited a website for the purpose of receiving recommendations from the CF system. Later, websites began to use CF systems behind the scenes to adapt their content to users, such as choosing which news articles a website should be presenting prominently to a user.

Providers of information on the web must deal with limited user attention and limited screen space. Collaborative filtering can predict what information users are likely to want to see, enabling providers to select subsets of information to display in the limited screen space. By placing that information prominently, it enables the user to maximize their limited attention. In this way, collaborative filtering enables the web to adapt to each individual user's needs.

The remainder of this chapter will discuss collaborative filtering in more depth by considering:

- The tasks for which users might use a CF system, things a CF system is good at, and the kinds of domains for which CF is appropriate (Section 2)
- Algorithms that CF systems employ (Section 3)
- How types of ratings in a CF system affect design choices (Section 4)
- How to evaluate and compare recommenders (Section 5)
- Trends in the development of more interactive and explicitly social interfaces (Section 6)
- The challenges to privacy and trust within CF systems (Section 7)
- Open questions in the continuing development of CF systems (Section 8)

2 Uses For Collaborative Filtering

Thus far, we have only briefly introduced collaborative filtering systems. However, we may have still left readers asking the question "for what purposes is CF appropriate?" In this section we consider this question by exploring user tasks that CF supports, then the services that CF systems provide, and finally, contrasting CF with content filtering, a technique that supports many of the same tasks, but using different technology. Throughout, we explore both well-understood technologies, and thought-provoking proposals that are not as well understood.

2.1 User Tasks

Designers of adaptive websites should carefully identify the possible tasks users may wish to accomplish with their site as different tasks may require different design decisions. From a marketing perspective, this is the value added by the CF system. In this section, we consider user tasks for which collaborative filtering is useful.

Tasks for which people use collaborative filtering that have been studied include:

1. **Help me find new items I might like.** In a world of information overload, I cannot evaluate all things. Present a few for me to choose from. This has been applied most commonly to consumer items (music, books, movies), but may also be applied to research papers, web pages, or other ratable items.
2. **Advise me on a particular item.** I have a particular item in mind; does the community know whether it is good or bad?
3. **Help me find a user (or some users) I might like**. Sometimes, knowing who to focus on is as important as knowing what to focus on. This might help with forming discussion groups [34], matchmaking, or connecting users so that they can exchange recommendations socially.
4. **Help our group find something new that we might like.** CF can help groups of people find items that maximize value to group as a whole [41]. For example, a couple that wishes to see a movie together or a research group that wishes to read an appropriate paper.
5. **Domain-specific tasks**. For example, a research paper recommender [55] might also wish to support
    a) recommend papers that this paper should cite
    b) recommend papers that should cite this paper

Moreover, there are likely many tasks that are still undiscovered. Others are not yet well documented in the research literature, although they could be supported by the ratings data that collaborative filtering often has available. For example:

6. **Help me find an item, new or not.** For example, I might wish a "balanced diet" of restaurants, including ones I've gone to before; or, I might wish to go to a restaurant with a group of people, even if some have already been there; or, I might wish to purchase some groceries that are appropriate for my shopping cart, even if I've already bought them before.
7. **Domain-specific tasks**. For example, a recommender for a movie and a restaurant for 1) a first date versus 2) a guys' night out.

Note that "domain-specific tasks" are on both lists. Recommenders for some domain-specific tasks have been explored; many have not. To date, much research has focused on more abstract tasks (like "find new items") while not probing deeply into the underlying user goals (like "find a movie for a first date").

2.2 Collaborative Filtering System Functionality

There are also broad abstract families of tasks that CF systems support. It is no accident that this system functionality is related to the user tasks of the previous section. Ideally, the system would support all user tasks, although mapping a real application to the functionality of an actual CF system can be challenging. In any case, here are the broad families of common CF system functionality:

1. **Recommend items.** Show a list of items to a user, in order of how useful they might be. Often this is described as predicting what the user would rate the item,

then ranking the items by this predicted rating. However, some successful recommendation algorithms do not compute predicted rating values at all. For example, Amazon's recommendation algorithm aggregates items similar to a user's purchases and ratings without ever computing a predicted rating [33]. Instead of displaying a personalized predicted rating, their user interface displays the average customer rating. As a result, the recommendation list may appear out of order with respect to the displayed average rating value. In many applications, picking the top few items well is crucial; producing predicted values is secondary.

2. **Predict for a given item.** Given a particular item, calculate its predicted rating. Note that prediction can be more demanding than recommendation. To recommend items, a system only needs to be prepared to offer a few alternatives, but not all. Some algorithms take advantage of this to be more scalable by saving memory and computation time [33, 47]. To provide predictions for a particular item, a system must be prepared to say something about any requested item, even rarely rated ones. How does a system decide how a particular user would rate a requested item if very few users – let alone users similar to the particular user – have rated the item? Personalized predictions may be challenging, if not impossible.
3. **Constrained recommendations: Recommend from a set of items.** Given a particular set or a constraint that gives a set of items, recommend from within that set. For example:

   > "Consider the following scenario. Mary's 8-year-old nephew is visiting for the weekend, and she would like to take him to the movies. She would like a comedy or family movie rated no "higher" than PG-13. She would prefer that the movie contain no sex, violence or offensive language, last less than two hours and, if possible, show at a theater in her neighborhood. Finally, she would like to select a movie that she herself might enjoy." [50]

   Schafer et al. [50] propose a "meta-recommendation system" that generates recommendations from a blending of multiple recommendation sources. Users define preferences and requirements through a web form that restricts the set of potential candidate items. Recommendations are based on a ranking of how well the items within this set match the provided preferences. Adomavicius et al. [1] call this "flexibility," and propose a SQL-like language as a desired extension in a "next-generation" recommendation system. Such a system might accept queries such as "RECOMMEND Movie TO User BASED ON Rating FROM MovieRecommender WHERE Movie.Length < 120 AND Movie.Rating < 3 AND User.City = Movie.Location." Similar techniques are discussed in *Chapter AT7* [53].

2.3 Suitable domains for collaborative filtering

One might simply take a user application, implement it with a CF system, and hope it will work. However, CF is better known to be effective in domains with certain properties. It seems useful to acquaint ourselves with them, and consider whether the user application is a good fit. We group these properties below into data distribution, underlying meaning, and data persistence.

Note that with special consideration, CF can be successfully applied in domains that do not have some of the properties below. We simply list them to provoke thought and discussion about what domains are easy or hard with collaborative filtering.

Data distribution. These properties are about the numbers and shape of the data:

1. **There are many items.** If there are only a few items to choose from, the user can learn about them all without need for computer support.
2. **There are many ratings per item.** If there are only a few ratings per item, there may not be enough information to provide useful predictions or recommendations.
3. **There are more users rating than items to be recommended.** A corollary of the previous paragraph is that often you'll need more users than the number of items that you want to be able to capably recommend. More precisely, if there are few ratings per user, you'll need many users. Lots of systems are like this. For example, this makes web pages a challenging domain, especially if the system requires explicit ratings. Google[3], a popular search engine, claims to index 8 billion web pages at present, which is more than the number of people in the world, not to mention the number who have access to computers. As another example, with one million users, a CF system might be able to make recommendations for a hundred thousand items, but may only be able to make confident predictions for ten thousand or fewer, depending on the distribution of ratings across items. The ratings distribution is almost always very skewed: a few items get most of the ratings, a long tail of items that get few ratings. Items in this long tail will not be confidently predictable.
4. **Users rate multiple items.** If a user rates only a single item, this provides some information for summary statistics, but no information for relating the items to each other.

Underlying meaning. These properties are of the underlying meaning of the data:

1. **For each user of the community, there are other users with common needs or tastes.** CF works because people have needs or tastes in common. If a person has tastes so unique that they are not shared by anybody else, then CF cannot provide any value. More generally, CF works better when each user can find many other users who share their tastes in some fashion.
2. **Item evaluation requires personal taste.** In cases where there are objective criteria for goodness that can be automatically computed, those criteria may be better applied by means other than collaborative filtering, e.g., search algorithms. Collaborative filtering allows users with similar tastes to inform each other. CF adds substantial value when evaluation of items is largely subjective (e.g., music), or when those items have many different objective criteria that need to be subjectively weighed against each other (e.g., cars). Sometimes there are objective criteria that can help (e.g., only recommend books written in English), but if

[3] http://www.google.com/

recommendation can be performed using *only* objective criteria, then CF is not useful.

3. **Items are homogenous.** That is to say, by all objective consumption criteria they are similar, and they differ only in subjective criteria. Music albums are like this. Most are similarly priced, similar to buy, of a similar length. Books or research papers are also like this. Items sold at a department store are not like this: some are cheap, some very expensive. For example, if you buy a hammer, perhaps you should not be recommended a refrigerator.

Data persistence. These are properties of how long the data is relevant:

1. **Items persist.** Not only does a CF system need a single item to be rated by many people, but also requires that people share multiple rated items – that there is overlap in the items they rate. If I've rated item A and I want a prediction for item B, most CF algorithms require multiple users to have rated both A and B. If I've rated item A and I want recommendations, most CF algorithms require that multiple users have rated A and some other items. All of this means that if items are only important for a short time, these requirements are hard to meet. For example, news stories: many appear per day, and many probably are only interesting for a few days.
2. **Taste persists.** CF has been most successful in domains where users' tastes don't change rapidly: e.g., movies, books, and consumer electronics. If tastes change frequently or rapidly, then older ratings may be less useful. An example might be clothing, where someone's taste from five years ago may not be relevant.

The properties of the preceding sections represent simplifications of the world where CF is most easily applied. In fact, applying CF in domains where these properties do not hold can provide both interesting applications and interesting research areas. For example, one might try to apply CF to non-homogenous items by using constrained recommendations, or applying external constraints (called *business rules* in the business world). Likewise, in order to perform system tasks for non-persistent items, one might try to apply content filtering, which is explored in the next section.

2.4 Comparing collaborative filtering to content-based filtering

Collaborative filtering uses the assumption that people with similar tastes will rate things similarly. *Content-based filtering* uses the assumption that items with similar objective features will be rated similarly. For example, if you liked a web page with the words "tomato sauce," you'll like another web page with the words "tomato sauce." The challenge is to cleanly extract the features of items that are most predictive. One then builds a user profile of features from the items a user has rated, and then compares that user profile to item profiles of new items whose features are extracted [4]. For more information, refer to *Chapter AT5* [43].

Content-based filtering and collaborative filtering have long been viewed as complementary [1]. Content-based filtering can predict relevance for items without ratings (e.g., new items, high-turnover items like news articles, huge item spaces like web pages); collaborative filtering needs ratings for an item in order to predict for it. On the other hand, content-based filtering needs content to analyze, and content can be scarce in some domains (e.g., movies, music, restaurants, and books without text reviews available); collaborative filtering does not require content. A content filtering model can only be as complex as the content it has access to. For instance, if the system only has genre metadata for movies, the model can only incorporate this one extremely coarse dimension. Furthermore, if there is no easy way to automatically extract a feature, then content-based filtering cannot consider that feature. For example, while people find the quality of multimedia data (e.g., images, video, or audio) for web pages important, it is difficult to automatically extract this information [4]. Collaborative filtering allows evaluation of such features, because people are doing the evaluating.

Content-based filtering may over-specialize. Items are recommended that match the content features in the user's interest profile or query. Items that do not contain the exact features specified in the interest profile may not get recommended even if they are similar (e.g., due to synonymy in keyword terms). Researchers generally believe collaborative filtering leads to more unexpected or different items that are equally valuable. Some people call this property of recommendations *novelty* or *serendipity*[21]. (See 1.6.2 for a more complete discussion.) However, collaborative filtering has also been shown to over-specialize in some cases [57].

Content-based filtering (CBF) and collaborative filtering may be manually combined by the end-user specifying particular features, essentially constraining recommendations to have certain content features [50]. More often they are automatically combined, sometimes called a *hybrid* approach. There are many ways to combine them, and no consensus exists among researchers [5, 11, 12, 19, 44]. However, such systems generally use the content analysis to identify items that meet the immediate need of the user, and use CF to try and capture features like quality that are hard to automatically analyze. For a more detailed look at these techniques, refer to *Chapter AT6* [9].

3 Collaborative Filtering Algorithms: Theory and Practice

Over the past decade, collaborative filtering algorithms have evolved from research algorithms intuitively capturing users' preferences to algorithms that meet the performance demands of large commercial applications. In this section we explore some of the most widely known collaborative filtering algorithms. Although a good deal of theoretical literature describes CF algorithms, little information is available to assist practitioners in building CF systems. We highlight not only the theoretical definition of these algorithms but their practical challenges and, where applicable, suggest techniques to address these challenges.

Breese et al. [8] described CF algorithms as separable into two classes: *memory-based* algorithms that require all ratings, items, and users be stored in memory and *model-based* algorithms that periodically create a summary of ratings patterns offline. Pure memory-based models do not scale well for real-world application. Thus, almost all practical algorithms use some form of pre-computation to reduce run-time complexity. As a result, current practical algorithms are either pure model based algorithms or are a hybrid of some pre-computation combined with some ratings data in memory.

A more useful organization of collaborative filtering algorithms splits them into *non-probabilistic* algorithms and *probabilistic* algorithms. We consider algorithms to be probabilistic if they are based on an underlying probabilistic model. That is, they represent probability distributions when computing predicted ratings or ranked recommendation lists. In general, non-probabilistic models are widely used by practitioners. Probabilistic models have been gaining favor, however, particular in the machine learning community.

3.1 Non-probabilistic Algorithms

The most well-known CF algorithms are nearest neighbor algorithms. We introduce the two different classes of nearest neighbor CF algorithms: user-based nearest neighbor and item-based nearest neighbor. We also explore more briefly non-probabilistic algorithms that transform or cluster the ratings space to reduce the ratings space dimensionality. Other commonly cited algorithms not discussed here include graph-based algorithms [2], neural networks [7], and rule-mining algorithms [20].

User-based Nearest Neighbor Algorithms

Early algorithms generated predictions for users based on ratings from similar users. We call these similar users *neighbors*. If a user *n* is similar to a user *u,* we say that *n* is a *neighbor* of *u*. User-based algorithms generate a prediction for an item *i* by analyzing ratings for *i* from users in *u*'s neighborhood. Naively, we could average all neighbors' ratings for item *i*. Equation 1 gives the naïve user formulation, where r_{ni} is neighbor *n*'s rating for item *i*.

$$pred(u,i) = \frac{\sum_{n \subset neighbors(u)} r_{ni}}{number\ of\ neighbors} \tag{1}$$

However, we want to weight ratings from users who are similar to *u* more heavily. Thus, if *userSim(u,n)* is a measure of the similarity between a target user *u* and a neighbor *n*, a prediction can be given by equation 2.

$$pred(u,i) = \sum_{n \subset neighbors(u)} userSim(u,n) \cdot r_{ni} \tag{2}$$

Unfortunately, if the similarities of the neighbors do not add up to one, this prediction will be incorrectly scaled. Accordingly equation 3, normalizes the prediction by dividing by the sum of the neighbors' similarities.

$$pred(u,i) = \frac{\sum_{n \subset neighbors(u)} userSim(u,n) \cdot r_{ni}}{\sum_{n \subset neighbors(u)} userSim(u,n)} \quad (3)$$

Finally, users vary in their use of rating scales. To compensate for ratings scale variations, equation 4 *average adjusts* for users' mean ratings..

$$pred(u,i) = \bar{r}_u + \frac{\sum_{n \subset neighbors(u)} userSim(u,n) \cdot (r_{ni} - \bar{r}_n)}{\sum_{n \subset neighbors(u)} userSim(u,n)} \quad (4)$$

The GroupLens system for Usenet newsgroups, one of the first CF systems, defined *userSim(),* in equation 4 using the Pearson correlation [46]. The Pearson correlation coefficient is calculated by comparing ratings for all items rated by both the target user and the neighbor (e.g. *corated* items). Equation 5 gives the formula for Pearson correlation between user *u* and neighbor *n*, where $\mathrm{CR}_{\mathrm{u,n}}$. denotes the set of corated items between *u* and *n.*

$$userSim(u,n) = \frac{\sum_{i \subset \mathrm{CR}_{\mathrm{u,n}}} (r_{ui} - \bar{r}_u)(r_{ni} - \bar{r}_n)}{\sqrt{\sum_{i \subset \mathrm{CR}_{\mathrm{u,n}}} (r_{ui} - \bar{r}_u)^2} \sqrt{\sum_{i \subset \mathrm{CR}_{\mathrm{u,n}}} (r_{ni} - \bar{r}_n)^2}} \quad (5)$$

Pearson correlation ranges from 1.0 for users with perfect agreement to -1.0 for perfect disagreement users. Negative correlations are generally believed to not be valuable in increasing prediction accuracy [22].

Practical Challenges of User-based Algorithms

The user-based nearest neighbor algorithm captures how word-of-mouth recommendation sharing works and it can detect complex patterns given enough users; however it has practical challenges.

Ratings data is often sparse, and pairs of users with few coratings are prone to skewed correlations. For example, if users share only three corated items, it is not uncommon for the ratings to match almost exactly (a similarity score of 1). If such similarities are not adjusted, these skewed neighbors can dominate a user's neighborhood.

Another problem with Pearson correlation is that it fails to incorporate agreement about a movie in the population as a whole. For instance, two users agreement about a universally loved movie is much less important than agreement for a controversial movie. Pearson correlation does not capture this distinction. Some user-based algorithms account for global item agreement by including weights inversely proportional to an item's popularity when calculating user correlations [8].

The original user-based algorithm as implemented in GroupLens included all users in a CF system in a prediction neighborhood [50]. Later algorithms improved accuracy and efficiency by limiting the prediction calculation to a user's closest *k* neighbors [22].

Most importantly, calculating a user's perfect neighborhood is expensive - requiring comparison against all other users. Thus, in a naïve implementation, the time and memory requirements of user-based algorithms scale linearly with the number of users and ratings. Researches have tried many techniques to reduce processing time and memory consumption:

- *Subsampling* - In sampling, a subset of users is selected prior to prediction computation. Neighborhood computation time remains fixed, and schemes have been proposed to intelligently choose neighbors in order to achieve virtually identical accuracy.
- *Clustering* - Clustering algorithms have been used to quickly locate a user's neighbors [33]. In these schemes, a user is compared to groups of users, rather than individual users. Clusters of users similar to the target are quickly discovered, and nearest neighbors can be selected from the most similar clusters. Both k-means clustering [35], and hierarchical divisive [28] and agglomerative clustering [31] can segment users into clusters. One challenge in using clustering is that clustering schemes use distance functions, such as Pearson correlation to both form the clusters and measure distance from a cluster. However, due to missing data, distance functions generally do not obey the triangle equality and are not true mathematical metrics[4]. This can lead to unintuitive and unstable clustering.

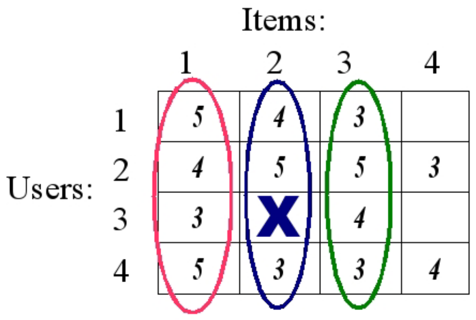
Item-based Nearest Neighbor Algorithms

Item-based nearest neighbor algorithms are the transpose of the user-based algorithms. While user-based algorithms generate predictions based on similarities between users, item-based algorithms generate predictions based on similarities between items [47]. The prediction for an item should be based on a user's ratings for similar items. Consider the ratings matrix shown in **Figure 3**.

Assume we are trying to predict a rating for user #3, item #2 (marked by the X). First, we observe that item #2's ratings are very similar to item #3's ratings, but not as similar to item #1's ratings. We now try to predict the rating "X" by building a weighted average of user #3's other ratings (3 for item #1 and 4 for item #3). Since

[4] A distance metric has four properties: it is non-negative, the identity distance is 0, it is reflexive, and the triangle equality holds. The triangle equality is generally most difficult requirement to meet.

item 2 is similar to item 3, we might guess that the rating for item #3 is more important. We conclude that a good guess is *0.25*3 + 0.75*4 = 3.75*.



Users:	Items: 1	2	3	4
1	5	4	3	
2	4	5	5	3
3	3	X	4	
4	5	3	3	4

Figure 3: An item-based nearest-neighbor algorithm generates predictions based on similarities between items. Observe that item two is fairly similar to item three and moderately similar to item one.

We have just outlined the item-based prediction algorithm, which we formalize in equation 6. A prediction for a user *u* and item *i* is composed of a weighted sum of the user *u*'s ratings for items most similar to *i*.

$$pred(u,i) = \frac{\sum_{j \in ratedItems(u)} itemSim(i,j) \cdot r_{ui}}{\sum_{j \in ratedItems(u)} itemSim(i,j)} \qquad \textbf{(6)}$$

Note that in equation 6, *itemSim()* is a measure of item similarity, not user similarity. Average correcting is not needed when generating the weighted sum because the component ratings are all from the same target user.

Several variations exist for calculating the similarity for a pair of items (*i*, *j*). Adjusted-cosine similarity, the most popular (and believed to be most accurate) similarity metric, is computed using all users who have rated both item *i* and *j*. Equation 7 gives the formula for adjusted-cosine similarity, where $RB_{i,j}$ denotes the set of users who have rated both item *i* and item *j*.

$$itemSim(i,j) = \frac{\sum_{u \subset RB_{i,j}} (r_{ui} - \bar{r}_u)(r_{uj} - \bar{r}_u)}{\sqrt{\sum_{u \subset RB_{i,j}} (r_{ui} - \bar{r}_u)^2} \sqrt{\sum_{u \subset RB_{i,j}} (r_{uj} - \bar{r}_u)^2}} \qquad \textbf{(7)}$$

The only difference from Pearson correlation is that average adjusting is performed with respect to the user, not the item. As in the user Pearson correlation, the correlation value ranges from –1.0 to 1.0.

There is evidence that item-based nearest neighbor algorithms are more accurate in predicting ratings than their user-based counterparts [47].

Practical Challenges in Item-based Algorithms

Theoretically, the size of the model could be as large as the square of the number of items. In practice, we can substantially reduce this size by only storing correlations for item pairs with more than *k* coratings. Sarwar et al. prune the model even further by only retaining the top *n* correlations for each item. Such modifications yield item-based algorithms that are relatively efficient in both memory usage and CPU performance. Note that pruning many of the correlations means that it may be more difficult to make a prediction for a given target item and user, since the items correlated with the user's ratings may not contain the target item.

As in the user algorithm, item pairs with few coratings can lead to skewed correlations and care must be exercised to not let skewed correlations dominate a prediction.

Non-probabilistic Dimensionality Reduction Algorithms

Large CF applications may support millions of users and items [33]. Other domains may have such a sparsity of ratings that there are few coratings. Several algorithms reduce domain complexity by mapping the item space to a smaller number of underlying "dimensions." Intuitively, these dimensions might represent the latent topics or tastes present in those items. The smaller "latent" dimensions reduce run-time performance needs and lead to larger numbers of co-rated dimensions. These techniques define a mapping between a user's ratings and their underlying tastes. An item's prediction can then be generated based on a user's underlying tastes. Mapping functions generally consist of simple vector operations, and predictions for an item can be calculated in constant time. Vector-based techniques for extracting underlying dimensions include support vector decomposition [48], principal component analysis [18], and factor analysis [10].

Practical Challenges in Dimensionality Reduction Algorithms

Mathematical dimensionality reduction techniques such as singular value decomposition [48] and principal component analysis [18] require an extremely expensive offline computation step to generate the latent dimensional space. Practical implementation of these techniques generally requires the use of heuristic methods for incrementally updating the latent dimensional space without having to entirely recompute – such as the folding-in technique for singular value decomposition. However, the primary challenge to utilizing such techniques is the mathematical complexity – which can lead to challenges debugging and maintaining software utilizing those techniques. While there is some evidence that these techniques can improve accuracy in predicting ratings [49], for the most part, the improvement has not been substantial enough to overcome the practical challenges of complexity.

Association Rule Mining

Association mining techniques build models based on commonly occurring patterns in the ratings matrix [20, 32]. For example, we may observe that users who rated item 1 highly often rate item 2 highly. A particular rule is represented by an input condition (e.g. item 1 rated highly) and a result condition (e.g. item 2 rated highly). The *support* of a rule represents the fraction of users who have rated both the input and result conditions, and the *accuracy* of a rule is the fraction of users with the input condition that exhibit the result condition.

In order to generate a predicted rating for a user *u* and item *i*, we first select the rules with a result condition of item *i* that only include items rated by user *u*. We then use a heuristic to translate the support, accuracy, and ratings for input conditions into a predicted rating.

For more information, refer to *Chapter MT3* [39].

Practical Challenges in Association Rule Mining

Naïve association rules can treat each rating value as independent. For example, a rating of 1 for a particular item is different than a rating of 2, even though both may be interpreted as the user indicating dissatisfaction with the item. This independence can dramatically increase the sparsity of an already sparse space. To overcome this, implementers generally place "similar" ratings into bins using one of several strategies:

- *High and low ratings bins* – Divide ratings into two bins; those above and those below a user's average rating.
- *High ratings* – Only consider ratings above a user's average when building rules.
- *All ratings* – Treat all ratings as identical when building rules.

A general drawback in association mining is that, since rating bins are treated discretely, we lose any notion of the numeric relationship among ratings. Although this relationship is theoretically meaningful, in practice it seems to have little impact.

Association rule mining in non-CF domains often looks for input patterns consisting of multiple items (e.g. if the user rated items 1 and 2 highly, they will rate item 3 highly). While these patterns may be useful, mining the patterns is too slow in CF domains due to the extremely high dimensionality.

3.2 Probabilistic Algorithms

Probabilistic CF algorithms explicitly represent probability distributions when computing predicted ratings or ranked recommendation lists. In general, probabilistic algorithms try to leverage well-understood formalisms of probability.

Most probabilistic CF algorithms calculate the probability that, given a user *u* and a rated item *i*, the user assigned the item a rating of *r*: $p(r|u,i)$. We calculate a predicted rating based on either the most probable rating value or the expected value of *r*. Equation 8 gives the formula for user *u*'s expected rating for item *i*.

$$E(r \mid u,i) = \sum_r r \cdot p(r \mid u,i) \qquad \textbf{(8)}$$

The most popular probabilistic framework involves Bayesian-network models that derive probabilistic dependencies among users or items. Some of the earliest probabilistic CF algorithms were proposed by Breese et al., who describe a method for deriving and applying Bayesian networks using decision trees to compactly represent probability tables [8]. A separate tree is constructed for every recommendable item. The branch chosen at a node in the tree is dependent on the user's rating (or lack of rating) for a particular item. Nodes in the tree store a probability vector for user's ratings of the predicted item. In theory, non-naïve Bayesian networks improve upon standard item-based algorithms by modeling dependencies between input items used to calculate a prediction. However for multi-valued ratings, there has been no published evidence of Bayesian networks consistently outperforming item-based nearest neighbor algorithms.

There has also been a good amount of work on developing probabilistic clustering/dimensionality reduction techniques. Probabilistic dimension reduction techniques introduce a hidden variable *p(z|u)* that represents the probability a user belongs to the hidden class *z*. Equation 9 gives the formula for calculating the probability of user *u* rating item *i* value *r*.

$$p(r \mid u,i) = \sum_z p(r \mid i,z)p(z \mid u) \tag{9}$$

The corresponding prediction is the expectation of the rating value (equation 10).

$$E(r \mid u,i) = \sum_r \left(r \cdot \sum_z p(r \mid z,i)p(z \mid u) \right) \tag{10}$$

Hoffman presents an expectation maximization (EM) algorithm for CF that estimates latent classes *z* with Gaussian probability distributions [26]. Clustering algorithms also have been used to estimate latent classes [56].

One advantage of probabilistic algorithms is that they can produce a probability distribution across possible rating values – information that captures the likelihood of each possible rating value. From this information, not only can you compute the most probable rating, you can also compute a likelihood of that rating being correct – thus capturing the algorithm's confidence. There has been a recent attempt to create a hybrid approach that utilizes the nearest neighbor algorithm, but represents ratings as discretized probability distributions rather than a point rating [36].

3.3 Over-arching Practical Concerns

Regardless of choice of algorithm, real-world CF systems need to address several problems that are generally not covered in research literature.

Adjust for few ratings

Items and users with few ratings can inappropriately bias CF results. Algorithms may take steps to adjust for users, items, and user and item pairs with few co-ratings (we'll generally call these *rarely-rated entities*). We will compare techniques for adjusting for rarely-rated entities, using a user-based algorithm as an example:

1. *Discard rarely-rated entities* – Algorithms often only incorporate data with greater than *k* ratings. In a user-based algorithm, for example, we would discard neighbors with fewer than k co-ratings with the target user. Although this is a simple and clean approach it can decrease the coverage of the CF system.
2. *Adjust calculations for rarely-rated entities*– This technique adjusts calculations for rarely-rated entities by pulling them closer to an expected mean. For instance, Pearson similarities for users with few co-ratings may be adjusted closer to 0. CF systems often make the adjustment amount inversely proportional to the number of ratings. Although adjustment can be effective, tuning adjustment parameters can be difficult and unstable.
3. *Incorporate a prior belief* – We can avoid skew by incorporating artificial data points that match an expected distribution. For example, we may believe that users ratings will generally match a probability distribution *p*. We can incorporate this prior belief into user correlation calculation by including *k* artificial co-rated items whose ratings are independently drawn from *p*.

Prediction vs. Recommendation

Prediction and Recommendation tasks place different requirements on a CF system. To recommend items, a system must be prepared to know about a subset of items, but perhaps not all. Some algorithms save memory and computation time by taking advantage of this [33, 47]. To provide predictions for a particular item, a system must store information about every item, even rarely rated ones. Algorithms that are required to present personalized predictions for many items often have larger memory requirements.

On the other hand, recommendation tasks require calculation of predictions for many (if not all) items. A single prediction request can therefore afford a more expensive prediction calculation than a recommendation request.

Confidence Metrics

CF systems can supply a confidence metric that indicates the support for a particular prediction. Applications may choose to not display predictions with confidence measures below a certain threshold.

Confidence measures can also be used when selecting items for recommendation. CF algorithms generally choose to recommend those items with highest predicted ratings. Some CF systems may choose to tradeoff items with high predictions and low confidence for items with less-high predictions and high confidence.

Confidence measures are specific to each CF algorithm. Probabilistic algorithms may be able to use their computed probability distributions to estimate confidence. User-based algorithms often use confidence measures that incorporate the agreement for an item in a user's neighborhood, and the number of corated items between neighbors and the user. Item-based algorithms may measure the number of ratings for correlated pairs of items contributing to a prediction.

4 Acquiring Ratings: Design Tradeoffs

Ratings data from users on items are what enable collaborative filtering. In this section we will discuss in more depth the different kinds of ratings data that can be used and key concepts and decisions involved with acquiring ratings for collaborative filtering systems.

4.1 Explicit versus Implicit Ratings: Tradeoff

Explicit ratings provided by users provide the most accurate description of a user's preference for an item with the least amount of data. However, because explicit ratings require additional work from the user, it can be challenging to collect ratings – particularly when creating a new CF service. On the other hand, implicit ratings – observations of user behavior from which preference can be inferred – can be collected with little or no cost to the user, but ratings inference may be imprecise. As an example, consider using "time spent reading information about a product" as an implicit rating for that product. Intuitively, if a user spends a lot of time reading about a product, we might conclude that they would be interested in purchasing that product. However, there are reasons that this inference could be inaccurate – the user may have taken a coffee break just after opening the product info page, or the user may have concluded that the product was inappropriate after spending the time to read about it. Thus if implicit ratings are used, there is more uncertainty in the computation. Other examples of implicit ratings are discussed in Oard and Kim [40].

The more ratings that you have, the more uncertainty in the ratings you can handle. Uncertainty in rating values, including implicit ones is handled by aggregating ratings – collecting multiple observations of variables that are predictive of a rating and combining them into a single estimated rating – either by voting [15] or averaging [46, 52]. Thus if you are able to collect large numbers of ratings, then the errors introduced by uncertainty of implicit ratings can be canceled out by aggregation. In such a situation, you may be able to build a very successful CF system without explicit ratings. Good examples from the music domain are AudioScrobbler[5] and MusicStrands[6], which track every single song you play. With music, after enough ratings (plays) have accumulated, these implicit ratings may represent user taste much better than small explicit ratings scales. A five point rating scale only allows you to group a user's rated items into five ranks – the CF system cannot distinguish difference in taste between items with the same rating value. When using the implicit play count, user may play individual songs thousands of times, and since each song is likely to be played a different number of times, a more complete ranking of items a

[5] AudioScrobbler is owned by Last.fm which can be found at http://www.last/fm/index.php
[6] http://www.musicstrands.com/

user likes can be created. If you cannot capture large numbers of implicit ratings, then you will most likely need some form of explicit rating.

4.2 The Challenge of Collecting Explicit Ratings

Explicit ratings require dedicated attention of the user. Early researchers believed that users would not invest the time rating items required for CF systems. From an economic perspective it would appear that if incremental recommendations are free, then everybody would wait for others to identify what was good and there would be insufficient ratings [3]. However, during the past decade, experience has demonstrated that collecting explicit ratings is not as challenging as previously thought.

The first reason is that – in order to succeed – a CF system doesn't need lots of ratings from all people. Instead you just need a relatively small number of "early adopters" who rate frequently and continuously. These early adopters provide sufficient information to generate recommendations for the remaining users of the system. The remaining users must each then just provide a limited number of ratings in order for the system to learn their preferences.

The second reason that collecting explicit ratings is easier than previously expected is that users appear to gain many benefits from rating other than higher quality recommendations. Although no conclusive studies have been done, researchers and practitioners have proposed that users gain the following rewards from rating:

- An increased feeling of having contributed to advancing a community
- Gratification from having one's opinion's voiced and valued
- An ability to use the CF system as an extension of their memory of what they like and dislike.

Maintainers of CF systems sometimes use incentives to encourage users to provide more explicit ratings. For example, sites may exchange user ratings for "site points." These site points can be exchanged for rewards (e.g. t-shirts and hats) or privileges (e.g. the right to view privileged content). Incentives can increase the number of ratings provided by users, but users who rate only when provided with incentives are often price sensitive, and will move to a more rewarding opportunity if the reward to rate drops or the reward to participate in another community is increased.

4.3 Rating Scales

Another significant design decision involves choosing the explicit rating scale. The finer grained the scale, the more information you will have regarding each user's preference. Finer grained scales require more complex user interfaces. The most common types of ratings are shown in **Table 2**.

At some point, increasing the precision of the rating scale further may fail to add value. If a very precise scale is selected, such as 1-100, you are unlikely to get a user to give the same rating for an item if you ask them at different points in time – thus you increase the uncertainty in the rating. Perhaps the most important consideration is the desires of the user population. Users may feel that they cannot fully describe their tastes with few possible rating values. In MovieLens, users were frustrated that they were not able to give ratings as precise as the systems predictions of their ratings – predicted ratings were to the closest half point while user ratings were integers [13].

Table 2: Most common explicit rating scales.

Rating Scale	**Description**
Unary	Good or “don’t know”
Binary	Good or Bad
Integer “Likert”-like	Integers: 1-5, 1-7, or 1-10

4.4 Cold Start Issues

The “cold-start” problem describes situations in which a recommender is unable to make meaningful recommendations due to an initial lack of ratings. This problem can significantly degrade CF performance. It can occur under three scenarios.

New User. When a user first registers with a CF service, they have no ratings on record. Thus no personalized predictions can be given. For example, a new user to MovieLens has no ratings in the system, so a neighborhood of similar users can not be calculated. This may be solved in several ways. For example, by a) having the user rate some initial items before they can use the service; b) displaying non-personalized recommendations (population averages) until the user has rated enough; c) asking the user to describe their taste in aggregate, e.g., “I like science fiction movies”; d) asking the user for demographic information, or e) using ratings of other users with similar demographics as recommendations.

New Item. When a new item is added to a CF system, it has no ratings, so it will not be recommended. For example, MovieLens is unable to recommend new Hollywood releases until someone has entered an initial rating. Unfortunately, in many domains, users are less likely to rate items that are not recommended to them. Generally this is not a show-stopper, because most good items can be discovered through means other than the CF system and will get eventually rated. Users also tend to be forgiving of systems that don’t recommend obscure items. However, in domains where there may be many “sleepers” – unrated items that are very good, several techniques can be used, including: a) recommending items through non-CF techniques – content analysis or metadata, and b) randomly selecting items with few or no ratings and asking users to rate those items.

New Community. The biggest cold-start problem is bootstrapping a new community. If a new service’s value is in its personalized CF recommendations, then without ratings it may not have sufficient differentiating value – thus not retain users long enough to build up ratings. The most common solution is to provide rating incentives to a small “bootstrap” subset of the community, before inviting the entire community to use the service. Other approaches are to maintain users’ interest through alternate services, initially generate recommendations using non-CF approaches, or to start with a set of ratings from another source outside the community.

5 Evaluation

Evaluation measures how well a collaborative filtering system is meeting its goals, either in absolute terms or in relation to alternative CF systems. Unfortunately, there is no well-accepted metric that can evaluate all-important criteria related to the performance of a CF system. The appropriate metric to choose may depend on the type of items being recommended, the user tasks supported by the CF system, and any external goals that the service providers may have (e.g., promotional or inventory depletion). An in-depth discussion of evaluation considerations of collaboration filtering systems can be found in Herlocker et al. [21]. In this section, we first discuss accuracy, which is generally considered the most important criteria to evaluate, and then discuss more briefly some of the other criteria that may be important to evaluate and their associated metrics.

5.1 Accuracy

The most prominent evaluation metrics in the research literature measure the *accuracy* of the system's predictions. Accuracy can either be measured as the magnitude of error between the predicted rating and the true rating, or the magnitude of error between the predicted ranking and the “true” ranking. *Predictive accuracy* is the ability of a collaborative filtering system to predict a user's rating for an item. The standard method for computing predictive accuracy is *mean absolute error (MAE)* – the average absolute difference between the predicted rating and the actual rating given by a user. The advantage of MAE is that it is simple, well understood, and traditional significance tests can be applied to it. Furthermore, MAE seems to intuitively capture the quality of a CF system – we want predictions to be as close as possible to the true ratings. However, MAE has proven to be an unreliable measure of a ranked recommendation list [36]. Users perceive errors at the top of a recommendation list as much more costly than similar errors at the bottom of lists. MAE does not differentiate between errors at the top and errors at the bottom of lists.

Rank accuracy metrics attempt to compute the utility of a recommendation list to a user. Common rank accuracy metrics include precision [36, 47] and half-life utility [8]. Precision is the percentage of items in a recommendation list that the user would rate as useful. In CF, it is often computed at varying lengths of recommendation list (1, 3, 5, etc). The half-life utility metric computes a value for a ranked list that is intended to capture percentage of the maximum utility achieved by the ranked list in question. The maximum utility is achieved if all of the items rated as useful appear above all the items rated as not useful. In the half-life utility metric, mistakes at the top of the ranked list are weighted exponentially greater than mistakes further down the list.

If the user interface of the collaborative filtering system primary provides ranked lists of “best-bet” recommendations, then the accuracy of the system should be evaluated with a rank accuracy metric. If the system displays predictions of ratings directly to the user, then it is important to evaluate the system with a predictive accuracy metric. In many cases, it may make sense to use both.

5.2 Beyond Accuracy

While many of the published evaluations of CF systems measure accuracy, researchers and practitioners have come to learn that accuracy is not the only criteria of interest, and in some cases, may not even be the most important. Several other evaluation criteria have been explored.

- *Novelty* is the ability of a CF system to recommend items that the user was not already aware of. While non-novel recommendations can still be valuable, for many applications novelty is one of the most valued characteristics of the CF system's recommendations. Even stronger than novelty is the idea of *serendipity*, where users are given recommendations for items that they would not have seen given their existing channels of discovery. To illustrate the distinction, consider a news article recommender. A traditional content-based personalization system may generate recommendations that are not novel, because if I say I like a particular news article, then it will recommend other news articles with similar text, including stories about the exact same news event. A system tuned for novelty will work hard to not recommend news stories to me of which I am already aware. A serendipitous system would recommend to me news articles about topics that I have never read about before. Researchers have studied how to adjust algorithms to promote serendipity and novelty [47], but measuring novelty is challenging because it requires live user studies where participants indicate if a recommendation was novel.
- *Coverage* is the percentage of the items known to the CF system for which the CF system can generate predictions. It is also possible to compute variants such as the percentage of items that have the potential of being recommended to users, as performance optimizations in recommendations may prevent certain items from ever being recommended [49].
- *Learning Rate* measures how quickly the CF system becomes an effective predictor of taste as data begins to arrive. Generally these are computed per-user, measuring the number of ratings that a user has to provide before they are getting high quality personalized predictions [51].
- *Confidence* describes a CF system's ability to evaluate the likely quality of its predictions. Most CF systems generate rankings based on the most probable predicted rating. A CF system that can accurately compute its confidence in a prediction has the ability to limit recommendations to high confidence ones, leading to a tradeoff of fewer false positives in return for decreased coverage and possibly decreased novelty. If confidence in predictions can be computed, it can be displayed to users to help them decide if the risk-return ratio is appropriate [23].
- *User satisfaction metrics*. The metrics described above are only a sample of possible evaluation metrics. In particular there are many more metrics that can be applied if researchers have the ability to present a system to users, and measure how users perceive the system. This can be accomplished either by surveying the users or measuring retention and use statistics. Good examples include Swearingen and Sinha [54] and Dahlen et al. [14].

- *Site performance metrics*. In addition to the more mathematical and often "offline" metrics described above, websites may choose to use fairly simple site analysis metrics when adding a recommender to a site or modifying the design of an existing recommender. Such metrics might include tracking an increase in items purchased or downloaded, an increase in overall user revenue, or an increase in overall user retention. While such trends are easy to track and measure, they may be difficult to correlate to specific changes to an active website.

In conclusion, it is best to select a suite of metrics that will evaluate the criteria that are most important for the successful operation of a particular CF system. For example, if you are using CF to generate a top-5 recommendations list for your web site, then you might compute precision at top-5, top-3, and top-1. Furthermore, if the goal of your web site recommendations is to introduce your users to new things, then you might also do some user studies where you shown recommendation lists to users and ask them to rate the novelty of those recommendations. Predictive accuracy metrics like MAE may not be so useful if you are not displaying predicted rating values to users.

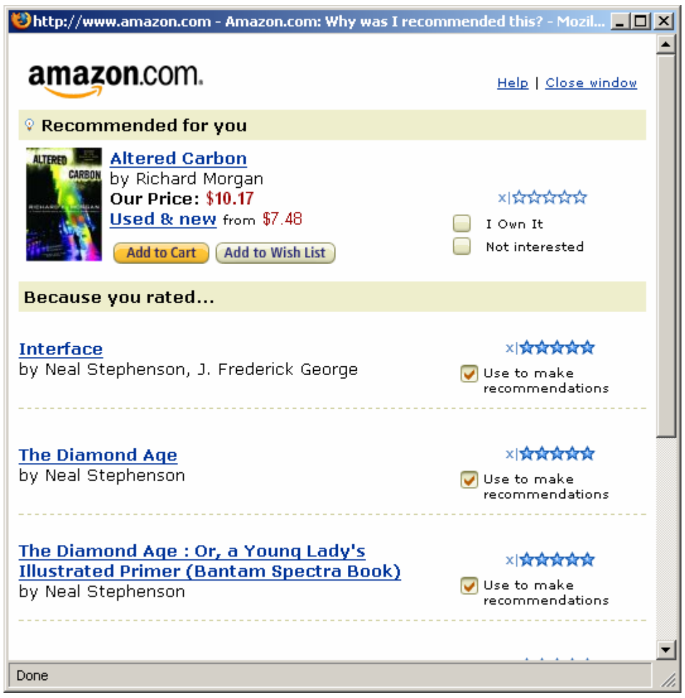
6 Rich Interfaces & Social Navigation

Early user interfaces for CF systems simply provided ranked list of recommendations, potentially with predicted ratings. The recommendation engine was a "black-box" – there was no transparency into how a prediction was computed [14, 17, 24, 46]. A critical trend in recent years is the exploration of user interfaces that enable more rich interaction with the underlying data of a collaborative filtering system, and CF systems that expose more information about the users from whom recommendations are (or can be) generated. In this section, we describe explanation and social navigation – two of these trends, and why they are so important.

One limitation of the black box approach was that the user interfaces to the CF systems were unable to communicate to the user when predictions were more or less risky than normal. Yet the need for this was common – when users were new or when items were new, predictions are more risky because there is less data on which to base inferences. More generally, the black box approach does not expose the reasoning or the data used in a recommendation. As a result, the user has little data on which to base decisions such as a) should they trust the recommendation process, b) is the current recommendation highly confident – either through trusted sources or overwhelming evidence, or c) is this recommendation appropriate for the user's immediate context or need.

6.1 Explanation

Initial work on the use of explanation in CF recommendations was promising [23], and has more recently been adopted commercially by Amazon.com, which has a link "why was I recommended this item" – the link will list previous ratings or purchases that you made that strongly influenced the recommendation at hand (**Figure 4**). Explanations of CF recommender systems are challenging because the underlying predictive models are complex aggregations of large quantities of data, often with significant probabilistic reasoning. Yet initial research suggests that users are overwhelmed if they are presented with too much data within an explanation [23]. While the current work on recommendations is far from conclusive, promising approaches that have been explored include: showing histograms of a user's neighbors' ratings for the recommended item and showing key items that the user rated that influenced the recommendation.


http://www.amazon.com - Amazon.com: Why was I recommended this? - Mozil...

amazon.com. Help | Close window

Recommended for you

Altered Carbon
by Richard Morgan
Our Price: $10.17
Used & new from $7.48
Add to Cart | Add to Wish List

I Own It
Not interested

Because you rated...

Interface
by Neal Stephenson, J. Frederick George
Use to make recommendations

The Diamond Age
by Neal Stephenson
Use to make recommendations

The Diamond Age : Or, a Young Lady's Illustrated Primer (Bantam Spectra Book)
by Neal Stephenson
Use to make recommendations

Done

Figure 4 : Amazon.com provides customers with list of previous purchases and ratings that strongly influenced a particular recommendation.

There is also a correlation between persuading a user that the recommendation is correct and explaining the recommendation to them. For many contexts, it may be sufficient to supply data from other sources not used in the recommendation that confirms the recommendations – such as reviews from critics. This may help persuade the user that the recommendation is good, yet reveals nothing about the reasoning behind the recommendations.

6.2 Social Navigation

Most of the CF systems we have discussed so far have been systems that use the group as a whole to help each individual user. Such systems tend to ignore the importance of the groups themselves. *Social navigation* systems encompass a variety of techniques that help people work together to help each other by making the aggregate behavior of the community visible. Users can employ this behavior to find their way through often crowded web spaces.

Höök et al. consider one type of social navigation system in which each visitor to a web site leaves "footprints" – telltale signs regarding what information the visitor considered and how frequently or in-depth. These footprints help other users find their way more readily through that same space [27]. This type of visualization has been called "read-wear" or "edit-wear" [25]. Early users leave footprints that help later users make sense of the wealth of alternatives available to them. Later users benefit from the footprint, because they are able to direct their attention to the parts of the site that are most valuable to them. As information spaces become more crowded with users it may become important to have systems that show us only those footprints that are most useful to us.

While these early CF and social navigation systems were clearly "collaborative," they almost always have provided "implicit" collaboration. Users benefited from the ratings and footprints left by other users in an anonymous and virtually untraceable manner. Some second generation collaborative filtering services have begun to experiment with allowing more "explicit" collaboration by exposing more of the identity of the other members of the community whose ratings are being used to generate a user's recommendations.

One example is epinions.com, which is a site designed to help users make purchasing decisions. On epinions.com, users rate and review products that they have purchased and these reviews are made available as recommendations to others. When a user views a recommendation/review, she can also look at the profile of the user who made the review, seeing information such as what other reviews they have written and how other people have responded to those reviews. She can explicitly state that she "trusts" a user as a reviewer. She can also "block" a reviewer, so that user's ratings/reviews are not shown.

Interfaces like epinions.com attempt to mimic more accurately the social process of word-of-mouth recommendations. A user could choose those people whose tastes he agreed with to provide recommendations, yet could choose different people to trust for different contexts. He could base his trust of another user on his observations of their activity within the community (their ratings) or on other's expressed opinions of their value. As users began to rate each other, explicit social networks could be expressed – "webs of trust." Users could then navigate these social networks in their search for items or products that would meet their need.

CF web services that offered this social navigation often evolved to be much more than recommendation sites. Particularly interesting was that the CF aspects of the system would bring together communities of common interest that would then engage in direct social interaction through discussion groups, chat rooms, or email. In theory, this direct social connection is the ultimate rich interface for recommendation. The CF software enables a user to navigate a potentially immense social network and find exactly those people who most closely share their tastes.

For a more detailed look at this topic, refer to *Chapter CH1* [16]

7 Ongoing Challenges to Collaborative Filtering

7.1 Privacy and Security

In order to provide personalized information to users, CF systems need to know things about those users. In fact, the more the system knows about a user, the better predictions it can provide to that user. With this increased information stored by a system often comes an increased concern on the part of the user regarding what information is collected, where and how it is stored, and how it is used. In centralized CF architectures, a single repository stores all user ratings. If the central server becomes compromised or corrupt, a user's anonymity can be destroyed. Users must trust that the CF provider will not use their preferences except for providing ratings and recommendations.

Distributed architectures may deploy ratings or models to each user, risking exposure of information to every peer [46]. To protect against this, researchers have developed security techniques building on encryption and shared keys [10]. In these schemes, a user can encrypt their ratings, and peers can tally encrypted ratings. Once ratings are totaled, distributed agents use shared keys to decrypt the rating tallies, without being able to see the original ratings.

Even systems that maintain the security of their users' ratings can be exploited to reveal personal information, particularly for users with unusual tastes. Ramakrishnan et al [45] use a graph-theoretic framework to explore these concerns. They found that “weak ties” (users who connect clusters of different tastes) are most susceptible to exploitation. Unfortunately, it is often these esoteric users that are most valuable to recommender systems, because they can provide users with unexpectedly novel recommendations. For more on the issue of privacy, see *Chapter CH3* [29].

7.2 Trust

Recommender systems may break trust when malicious users give ratings that are not representative of their true preferences. What happens to a CF system if one or more users decide to “attack” an item by purposefully lowering their rating(s) of the item? What happens if a company bombards a recommender with inflated ratings of its own products (e.g. Sony using quotes from made-up critics to promote its films [6])? There have been many examples of these “shilling” attacks. O’Mahoney et al [42] showed that users could, in fact, artificially raise and lower predicting ratings. User-based algorithms are more susceptible to shilling than item-algorithms, as are new or rarely rated items. Unfortunately, this vulnerability remains a significant challenge to collaborative filtering systems. Methods researchers use to detect attacks are not even sensitive enough to detect harsh attacks [29].

While these shilling attacks may seem slightly benign on the surface, further research has suggested that their effect may be more influential than originally feared. Cosley et al. demonstrated that users may not only perceive biases in ratings, but also adjust their own ratings to match recommenders' biases [13]. This observation indicates that shilling effects may be compounded as having viewed predictions based on the biased ratings potentially skews later users' ratings. More research is needed to understand how to identify attacks and protect systems from them.

8 Open Questions

This section discusses some open questions in the field of collaborative filtering. They are grouped into algorithmic questions (with an emphasis on temporal questions), and questions of broader access to collaborative filtering systems.

8.1 Algorithmic questions

Evaluation metrics. There have been many metrics of recommendation quality proposed [21]. Which ones capture what people perceive as good quality? Which ones are important?

Predicting well and recommending well at the same time. As we discussed in section 2.2, efficient algorithms for recommendation may choose not to produce predicted values at all, or may choose to only store a small amount of information necessary to recommend some items. However, predicting a rating for a given user and item is an appealing application. Are there efficient, scalable algorithms that both recommend and predict well?

Tagging. Social systems such as flickr and del.icio.us, which allow users to tag things (photos and websites, respectively) with keywords, are increasing in popularity and have captured the imagination of many people. These are collaborative filtering systems surely, though without much automation as yet. Other tagging systems have been around for years (e.g., IMDB's movie "keywords"). There are many interesting research questions. How can collaborative filtering algorithms be applied to tags? Can tags be used in conjunction naturally with ratings?

Tags without ratings are missing information. Tagging a movie "high-speed car chase" does not indicate whether that was a good thing or not. Is there a hybrid solution, where tags have associated explicit or inferred ratings?

8.2 Temporal questions

These questions are about the behavior of a collaborative filtering system over time.

1. **Item lifecycle.**
    a) When does an item have enough ratings to be accurately recommendable?

   b) When is an item a rising trend, falling trend, or a fad? Are many items are like that?

2. **User lifecycle.**
   a) When does a user have enough ratings to get good recommendations?
   b) Can one identify the items for which it is possible to give good recommendations for a given user?
   c) At what point do additional user ratings fail to improve his recommendations because the system has built a sufficiently accurate model (diminishing returns)? Can users detect this point and do they change the way they use the system?
   d) Are more ratings useful again as items are added?
   e) How do old ratings affect a user's recommendations, versus new ratings? Do user tastes shift over time? Can we detect it?
3. **Ratings database lifecycle.**
   a) When is a rating "stale" (i.e., no longer reflective of the opinion of the rater)?
   b) When does a database have enough ratings to give good recommendations?
   c) Can one identify which items are likely recommendable?
   d) How does the transition from not enough ratings to enough ratings look? Is there a critical threshold?
   e) Is it useful to expire (not use) ratings for the purposes of recommendations?

8.3 Broader Access

Collaborative filtering systems have been around for at least a decade. However, for the most part only large companies or research labs actually run them, because they require unusual expertise, considerable resources, or both. Many more people might be interested in giving opinions to each other in an automated system if appropriate infrastructure were present, and the range of items, domains, and opinions might be far more diverse. What are other effective ways to access or deliver the power of collaborative filtering?

User interfaces

The most well-known collaborative filtering systems are centralized web-based applications with explicit ratings. Other interfaces are emerging that bring the technology closer to users, who are more likely to use it if it is easy. Wikipedia and SourceForge list several applications with embedded collaborative filtering. For example, Audioscrobbler offers a plug-in to several music players (Winamp, Windows Media Player, iTunes, and several others) that collects data about which songs are played, sends it to a central web site, and produces music recommendations (**Figure 5**.)

Other systems have been proposed, but are not yet well studied. Miller investigates algorithms for portable, user-controlled, accurate recommendations on palmtop-sized devices [38]. These allow the users to remain anonymous and autonomous.

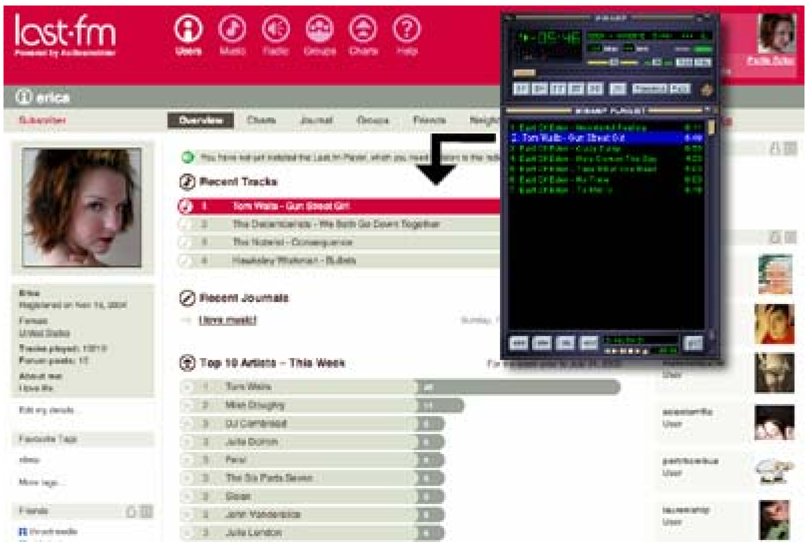


Figure 5 : After installing an Audioscrobbler plugin for your media player (eg: Winamp,) information about every song you listen to on your computer is sent to Last.fm to update your profile.

Libraries or toolkits

Once well understood, a technology can be bundled into a library or toolkit[7] available for embedding into an application. In addition to companies that do this commercially, there are several free, open-source alternatives[8]. However, there are no CF toolkits or libraries that have wide usage, as Apache and Internet Information Server (IIS) do in the web server space. Even though many designers see clear value in recommenders, and there seems to be increasing numbers of them on the web, few tool kits or libraries are gaining wide use. Why is this? What is the right functionality and interface for a toolkit suitable for a wide audience?

Data

Increased public availability of ratings datasets will enable more effective research into collaborative filtering, will allow practitioners to prototype CF system, as well as solve the "cold-start" problem for communities. Organizations often keep that data private, whether for competitive advantage or privacy concerns. Some are starting to open up their data. The EachMovie movie rating dataset was the most popular CF

[7] For more information on collaborative filtering toolkits, consult http://en.wikipedia.org/wiki/Collaborative_filtering#Software_libraries.

[8] Open source toolkits include CoFe (http://eecs.oregonstate.edu/iis/CoFE/), MultiLens (http://www.cs.luther.edu/~bmiller/dynahome.php?page=multilens), and Taste (http://taste.sourceforge.net/).

dataset until it was retired in October 2004. Remaining freely available datasets include MovieLens, Jester, and Book Crossing[9].

9 Summary

Collaborative filtering is one of the core technologies that will power the adaptive web. Content-based personalization can be effective in limited circumstances, but for the most part, it will likely be decades or longer before our hardware and software technology can begin to automatically recognize the subtleties of information that are important to people – particularly aspects of aesthetic taste. Until then, in order to filter information based on such complex dimensions, we need to include people in the loop, who analyze the information and condense their opinions into data that can be easily processed by software – ratings. In this chapter, we have attempted to provide a snapshot of the current understanding of collaborative filtering systems and methods. By necessity, as masses of information become ubiquitously available, collaborative filtering will also become ubiquitous. In the process, we will continue to gain a deeper understanding of the dynamics of collaborative filtering.

References

1. Adomavicius, G. and A. Tuzhilin, *Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions.* IEEE Transactions on Knowledge and Data Engineering, 2005. 17(6): p. 734-749.
2. Aggarwal, C.C., J. Wolf, K.L. Wu, and P.S. Yu. Horting Hatches an Egg: A New Graph-Theoretic Approach to Collaborative Filtering. In Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining. 1999. San Diego, California. ACM Press.
3. Avery, C., P. Resnick, and R. Zeckhauser, *The Market for Evaluations.* American Economic Review, 1999. 89(3): p. 564-584.
4. Balabanovíc, M. and Y. Shoham, *Fab: Content-Based, Collaborative Recommendation.* Communications of the ACM, 1997. 40(3): p. 66-72.
5. Basu, C., H. Hirsh, and W.W. Cohen. Recommendation as Classification: Using Social and Content-Based Information in Recommendation. In Proceedings of the Fifteenth National Conference on Artificial Intelligence. 1998. Madison, Wisconsin. AAAI Press.
6. BBC News Online, "Sony admits using fake reviewer." June 4, 2001 http://news.bbc.co.uk/1/hi/entertainment/film/1368666.stm.
7. Billsus, D. and M.J. Pazzani. Learning collaborative information filters. In Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98). 1998. Menlo Park, CA. Morgan Kaufmann Publishers Inc.
8. Breese, J.S., D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In Proceeding of the Fourteenth Conference on Uncertainty in Artificial Intelligence (UAI). 1998. Madison, Wisconsin. Morgan Kaufmann.

[9] All three data sets are available from http://www.grouplens.org/

9. Burke, R. *Hybrid Web Recomender Systems*, in *The Adaptive Web: Methods and Strategies of Web Personalization*, P. Brusilovsky, A. Kobsa, and W. Nejdl, Editors. 2006, Springer-Verlag:London.
10. Canny, J. Collaborative filtering with privacy via factor analysis. In Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval. 2002. Tampere, Finland. ACM Press.
11. Claypool, M., A. Gokhale, T. Miranda, P. Murnikov, D. Netes, and M. Sartin. Combining Content-Based and Collaborative Filters in an Online Newspaper. In Proceedings of the ACM SIGIR '99 Workshop on Recommender Systems: Algorithms and Evaluation. 1999. Berkeley, California.
12. Condliff, M.K., D. Lewis, D. Madigan, and C. Posse. Bayesian Mixed-Effect Models for Recommender Systems. In Proceedings of the SIGIR-99 Workshop on Recommender Systems: Algorithms and Evaluation. 1999. Berkeley, California.
13. Cosley, D., S.K. Lam, I. Albert, J.A. Konstan, and J. Riedl. Is seeing believing?: how recommender system interfaces affect users' opinions, in Proceedings of the SIGCHI conference on Human factors in computing systems. 2003, ACM Press: Ft. Lauderdale, Florida, USA. p. 585-592.
14. Dahlen, B.J., J.A. Konstan, J. Herlocker, and J.Riedl. Jump-starting movielens: User benefits of starting a collaborative filtering system with "dead data". TR 98-017, University of Minnesota.
15. Delgado, J. and N. Ishii. Memory-Based Weighted Majority Prediction for Recommender Systems. In 1999 SIGIR Workshop on Recommender Systems. 1999. University of California, Berkeley.
16. Dieberger, A. and M. Svensson, *From Social Navigation to the Social Web*, in *The Adaptive Web: Methods and Strategies of Web Personalization*, P. Brusilovsky, A. Kobsa, and W. Nejdl, Editors. 2006, Springer-Verlag:London.
17. Goldberg D, D. Nichols, B.M. Oki, and D. Terry. Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12), pp. 61–70.
18. Goldberg, K., T. Roeder, D. Gupta, and C. Perkins. *Eigentaste: A Constant-Time Collaborative Filtering Algorithm.* Information Retrieval, 2001. 4(2): p. 133-151.
19. Good, N., J.B. Schafer, J.A Konstan, A. Borchers, B. Sarwar, J. Herlocker, and J. Riedl. Combining collaborative filtering with personal agents for better recommendations. In Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI-99). 1999. Orlando, Florida. AAAI Press.
20. Heckerman, D., D.M. Chickering, C. Meek, R. Rounthwaite, and C. Kadie. *Dependency Networks for Inference, Collaborative Filtering, and Data Visualization.* Journal of Machine Learning Research, 2001. 1: p. 49-75.
21. Herlocker, J., J.A. Konstan, L.G. Terveen, and J. Reidl, *Evaluating Collaborative Filtering Recommender Systems.* ACM Transactions on Information Systems, 2004. 22(1): p. 5-53.
22. Herlocker, J.L., J.A. Konstan, A. Borchers, and J. Riedl. An algorithmic framework for performing collaborative filtering. In Proceedings of the 22nd International Conference on Research and Development in Information Retrieval (SIGIR '99). 1999. Berkeley, California. ACM Press.
23. Herlocker, J.L., J.A. Konstan, and J. Riedl. Explaining Collaborative Filtering Recommendations. In Proceedings of the 2000 ACM conference on Computer supported cooperative work. 2000. Philadelphia, Pennsylvania. ACM Press.
24. Hill, W., L. Stead, M. Rosenstein, and G. Furnas. Recommending and Evaluating Choices in a Virtual Community of Use. In Proceedings of ACM CHI'95 Conference on Human Factors in Computing Systems. 1995. Denver, Colorado. ACM Press.

25. Hill, W.C., J.D. Hollan, D. Wroblewski, and T. McCandless. Edit Wear and Read Wear. In Proceedings of the SIGCHI conference on Human factors in computing systems. 1992. Monterey, California. ACM Press.
26. Hofmann, T., *Latent semantic models for collaborative filtering.* ACM Transactions on Information Systems (TOIS), 2004. 22(1): p. 89-115.
27. Höök, K., D. Benyon, and A. Munro, *Footprints in the snow*, in *Social Navigation of Information Space*, K. Höök, D. Benyon, and A. Munro, Editors. 2003, Springer-Verlag: London.
28. Johnson, S.C., *Hierarchical clustering schemes.* Psychometrika, 1967. 32(3): p. 241-254.
29. Kobsa, A. *Privacy Enhanced Web Personalization*, in *The Adaptive Web: Methods and Strategies of Web Personalization*, P. Brusilovsky, A. Kobsa, and W. Nejdl, Editors. 2006, Springer-Verlag:London.
30. Konstan, J.A., B. Miller, D. Maltz, J. Herlocker, L. Gordon, and J. Riedl. GroupLens: Applying collaborative filtering to Usenet news. *Communications of the ACM*, 40(3), pp. 77–87.
31. Lam, S.K. and J. Riedl, Shilling recommender systems for fun and profit, in Proceedings of the 13th international conference on World Wide Web. 2004, ACM Press: New York, NY, USA. p. 393-402.
32. Lin, W., *Association Rule Mining for Collaborative Recommender Systems*, Master's Thesis, Worcester Polytechnic Institute, May 2000.
33. Linden, G., B. Smith, and J. York, *Amazon.com recommendations: item-to-item collaborative filtering.* Internet Computing, IEEE, 2003. 7(1): p. 76-80.
34. Ludford, P.J., D. Cosley, D. Frankowski, and L. Terveen, Think different: increasing online community participation using uniqueness and group dissimilarity in Proceedings of the SIGCHI conference on Human factors in computing systems 2004 ACM Press: Vienna, Austria p. 631-638
35. MacQueen, J. Some methods for classi cation and analysis of multivariate observations. In Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability. 1967.
36. McLaughlin, M. and J. Herlocker. A Collaborative Filtering Algorithm and Evaluation Metric that Accurately Model the User Experience. In Proceedings of the SIGIR Conference on Research and Development in Information Retrieval. 2004.
37. Maltz D, and E. Ehrlich. Pointing the way: Active collaborative filtering. In *Proceedings of ACM CHI'95 Conference on Human Factors in Computing Systems*, ACM, pp. 202–209.
38. Miller, B.N., J.A. Konstan, and J. Riedl, *PocketLens: Toward a personal recommender system.* ACM Trans. Inf. Syst., 2004. 22(3): p. 437-476.
39. Mobasher, B. *Data Mining for Web Personalization*, in *The Adaptive Web: Methods and Strategies of Web Personalization*, P. Brusilovsky, A. Kobsa, and W. Nejdl, Editors. 2006, Springer-Verlag:London.
40. Oard, D.W. and J. Kim. Implicit Feedback for Recommender Systems. In Proceedings of the AAAI Workshop on Recommender Systems. 1998. Madison, Wisconsin.
41. O'Connor, M., D. Cosley, J. A. Konstan, and J. Riedl, PolyLens: A Recommender System for Groups of Users. In Proceedings of ECSCW 2001. 2001. Bonn, Germany.
42. O'Mahoney, M.P., N. Hurley, N. Kushmerick, and G. Silvestre, *Collaborative recommendation: A robustness analysis.* ACM Transactions on Internet Technology, 2003. 4(3).
43. Pazzani, M., and D. Billsus. *Content-based Recommendation Systems*, in *The Adaptive Web: Methods and Strategies of Web Personalization*, P. Brusilovsky, A. Kobsa, and W. Nejdl, Editors. 2006, Springer-Verlag:London.

44. Popescul, A., L.H. Ungar, D.M. Pennock, and S. Lawrence Probabilistic Models for Unified Collaborative and Content-Based Recommendation in Sparse-Data Environments. 2001.
45. Ramakrishnan, N., B.J. Keller, and B.J. Mirza, Privacy Risks in Recommender Systems, in IEEE Internet Computing. 2001. p. 54-62.
46. Resnick, P., N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. GroupLens: An open architecture for collaborative filtering of netnews. In Proceedings of the 1994 ACM conference on Computer supported cooperative work. 1994. Chapel Hill, North Carolina. ACM Press.
47. Sarwar, B., G. Karypis, J.K. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In Proceedings of the 10th international conference on World Wide Web. 2001. Hong Kong. ACM Press.
48. Sarwar, B., G. Karypis, J.K. Konstan, and J. Riedl. Incremental SVD-Based Algorithms for Highly Scaleable Recommender Systems. In Proceedings of the Fifth International Conference on Computer and Information Technology (ICCIT 2002). 2002.
49. Sarwar, B.M., G. Karypis, J.K. Konstan, and J. Riedl. Application of Dimensionality Reduction in Recommender System--A Case Study. In ACM WebKDD 2000 Web Mining for E-Commerce Workshop. 2000. Boston, Massachusetts.
50. Schafer, J.B., J.A. Konstan, and J. Riedl, Meta-recommendation systems: user-controlled integration of diverse recommendations in Proceedings of the eleventh international conference on Information and knowledge management 2002 ACM Press: McLean, Virginia, USA p. 43-51
51. Schein, A.I., A. Popescul, and L.H. Ungar. Generative Models for Cold-Start Recommendations. In Proceedings of the Twenty-third Annual International ACM SIGIR Workshop on Recommender Systems. 2001. New Orleans, Louisiana.
52. Shardanand, U. and P. Maes. Social Information Filtering: Algorithms for Automating "Word of Mouth". 1995. New York. ACM.
53. Smyth, B. *Case-based Recommendation*, in *The Adaptive Web: Methods and Strategies of Web Personalization*, P. Brusilovsky, A. Kobsa, and W. Nejdl, Editors. 2006, Springer-Verlag:London.
54. Swearingen, K. and R. Sinha. Beyond Algorithms, An HCI perspective on Recommender Systems. In 2001 SIGIR Workshop on Recommender Systems. 2001. New Orleans, LA.
55. Torres, R., S.M. McNee, M. Abel, J.A. Konstan, and J. Reidl. Enhancing digital libraries with TechLens+ in Proceedings of the 4th ACM/IEEE-CS joint conference on Digital libraries 2004 ACM Press: Tuscon, AZ, USA p. 228-236
56. Ungar, L.H. and D.P. Foster. Clustering Methods for Collaborative Filtering. In Proceedings of the 1998 Workshop on Recommender Systems. 1998. Menlo Park, California. AAAI Press.
57. Ziegler, C.-N., S.M. McNee, J.A. Konstan, and G. Lausen. Improving Recommendation Lists Through Topic Diversification. In Proceedings of the Fourteenth International World Wide Web Conference (WWW2005). 2005.