CinemaScreen Recommender Agent: Combining Collaborative and Content-Based Filtering

James Salter and Nick Antonopoulos, *University of Surrey*

A film recommender agent expands and fine-tunes collaborative-filtering results according to filtered content elements—namely, actors, directors, and genres. This approach supports recommendations for newly released, previously unrated titles.

Directing users to relevant content is increasingly important in today's society with its ever-growing information mass. To this end, recommender systems have become a significant component of e-commerce systems and an interesting application domain for intelligent agent technology.

Traditionally, recommender systems employ *collaborative filtering*—recommending movies, for example, by matching a user to other users with similar tastes and suggesting movies these others have enjoyed. For example, if Bob and Wendy liked the same movies as you in the past and they both rated *Star Wars* highly, you might like it, too. However, recommender systems that employ purely collaborative filtering can't recommend an item until several users have rated it.

We've developed a film recommender agent (available at www.filmrecommendations.co.uk) that extends predictions based on collaborative filtering into the content space—specifically, to actors, directors, and film genres. This *content-based filtering* lets us include new movies in our recommendations. Experimental results show that our approach also improves on the accuracy of predictions based solely on content.

Recommendation methods

Recommender systems try to simulate the knowledge shopkeepers might develop about their customers' preferences over time.

In collaborative filtering, a recommender agent matches a user to other users who've expressed similar preferences in the past. In content-based filtering, the agent matches items users have previously rated highly to other similar items, presuming that people will like items similar to those they've selected previously. Similarity is based on content characteristics—in movies, for example, on actors, directors, and genres.

If used in isolation, both techniques exhibit certain weaknesses. Collaborative-only solutions suffer from a *cold-start problem*: the system can't produce recommendations until a large number of users have rated items in its databases. Systems relying exclusively on content-based filtering recommend only items closely related to those the user has previously rated. Such systems never reveal novel items that users might enjoy outside their usual set of choices. For example, if a user only rates war movies starring a small set of actors, it's likely that the vast majority of content-based system recommendations will also be war movies starring those actors.

Previous work to combine the positive aspects of both techniques has relied on one of two methods (see the "Related Work in Recommender Systems" sidebar). One method generates two recommendations sets—one from each technique—and then combines the results. The second method, *collaboration-via-content*, expands each user's item ratings into ratings for the item's content elements and then matches to other users through a collaborative-filtering algorithm.

Like collaborative filtering, collaboration-via-content can only generate recommendations for items that other users have already rated. Its content-based techniques generate a set of intermediate scores—for example, a score for each actor, director, and film genre. It then uses these intermediate scores, rather

1541-1672/06/$20.00 © 2006 IEEE
Published by the IEEE Computer Society

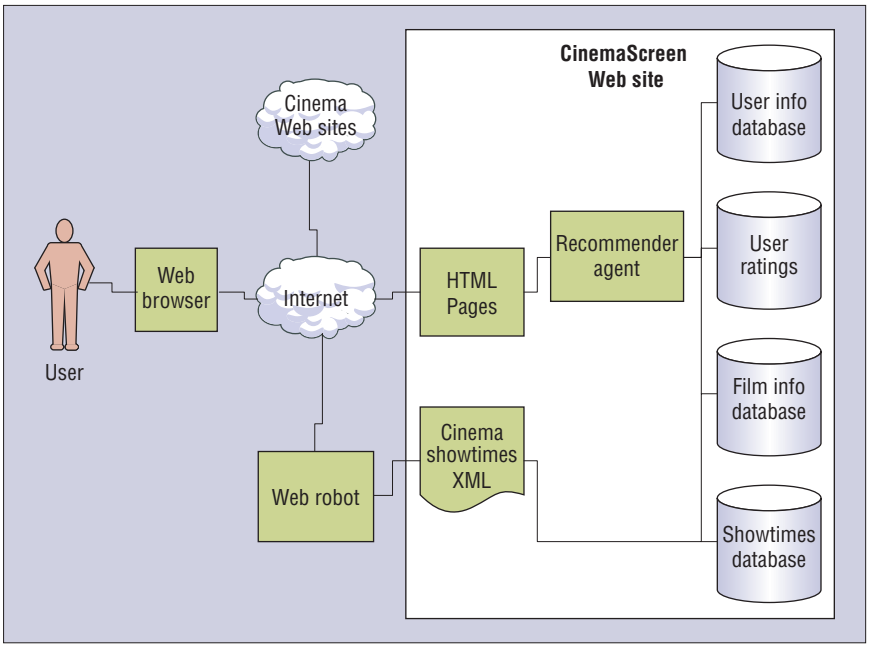

Figure 1. Components of the CinemaScreen recommender system.

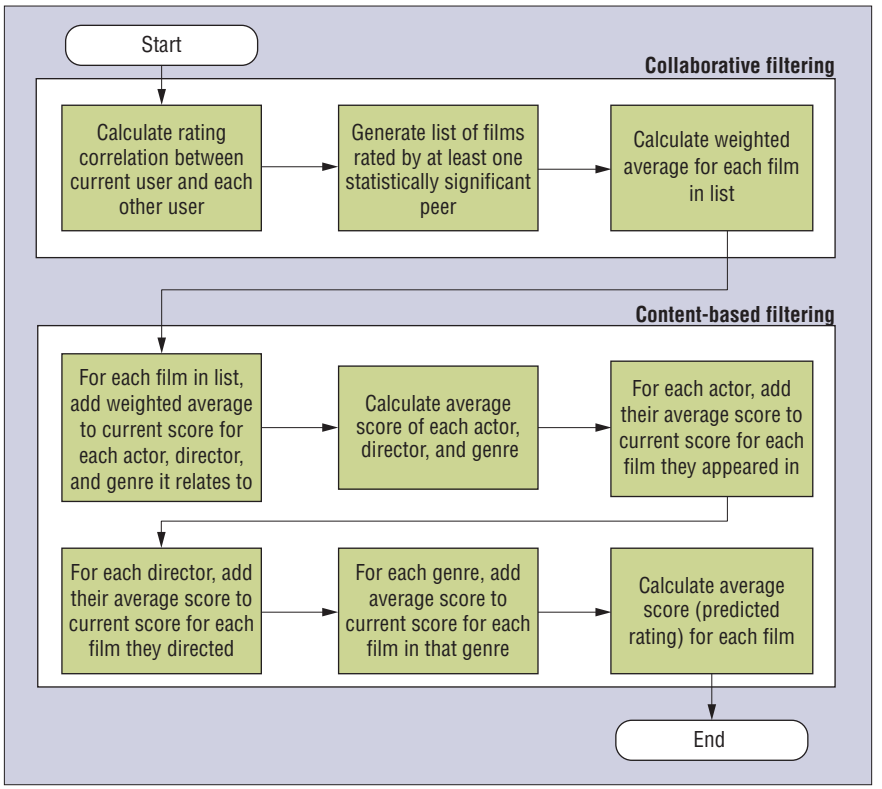

Figure 2. CinemaScreen recommendation process. The collaborative-filtering process feeds a weighted list of films for the current user into the content-based filtering process. The weights reflect ratings by at least one statistically significant peer of the current user.

than film ratings, in collaborative filtering to find users with similar scores.

The CinemaScreen Recommender Agent uses a technique that reverses collaboration via content. It executes content-based filtering on a results set generated through collaborative filtering. By reversing the strategy of filtering content first, the CinemaScreen Recommender Agent results set can include films that haven't yet received any user ratings but do, for example, star actors that have also appeared in films from the collaborative-filtering results.

This approach is crucial for the CinemaScreen site function that lets users select their local cinemas and receive recommendations from the list of films currently showing. The vast majority of these films are new or recent releases, so little or no rating data is available.

CinemaScreen recommender system

We built our system on the preexisting CinemaScreen Web site. The heart of the system is the recommender system (figure 1), comprising the recommender agent and its associated databases. Users interact with the system through a Web browser interface.

The system stores data in several Web server databases. As figure 1 shows, these include information about users and the ratings they've given to films they've seen, as well as film information and showtime databases, which the CinemaScreen Web site generates for use by the site as a whole. A partly automated and partly manual process collects and verifies data for the film information database from multiple sources such as news articles, archive collections, and dedicated content providers.

Figure 2 presents a flow diagram for our recommendation process.

Collaborative filtering first

The top row in figure 2 describes the collaborative-filtering process. It involves first finding a subset of users with film tastes similar to the current user. Comparing the current user's rating history with the history of every other user, the system finds the current user's *potential peers*—that is, other users who have rated films the current user has rated. The system produces a list of films with both rating sets and calculates a correlation measure between them (Pearson's product-moment correlation coefficient, r).[1] We test the Pearson's r value, using a standard significance test for the purpose. If the value is statistically significant, the agent adds the user to the current user's *peer list*.

This discriminating stage distinguishes our system from several others. Other approaches generate recommendations based on all users'

Related Work in Recommender Systems

GroupLens was one of the early recommender system implementations.[1] Unlike our hybrid technique, it used a collaborative-only approach to generate recommendations. The GroupLens research group has since created MovieLens, a film recommender system. Again, it uses only collaborative filtering.

Another system, Ringo,[2] uses collaborative filtering to recommend music to users, but it requires a training set of values to generate an initial user profile. Ringo therefore takes considerable processing time before making recommendations available to users. The correlation method we use is faster,[3] and it doesn't require training, which means it gives near-instantaneous recommendations to users if any are available.

In their PTV system, Barry Smyth and Paul Cotter[4] have used both collaborative and content-based filtering to independently generate two recommendation sets, which are subsequently combined. This approach differs from our agent, which seeks to combine the result sets at a much earlier stage. We believe our approach is better here, because determining how many items from each recommendation set to include in the final recommendation list is difficult, as is determining the order for listing them when they are merged. Other examples of methods similar to PTV include Profbuilder,[5] which asks users to manually select between collaborative and content-based filtering results sets. It doesn't try to automatically combine the two methods.

The Fab system[6] combines collaborative and content-based filtering in its recommendations by measuring similarity between users after first computing a profile for each user. This process reverses ours by running content-based filtering on the results of collaborative filtering. Restaurant recommenders have used this collaboration-via-content approach.[7]

In the context of a Chinese bookstore, Zan Huang and colleagues used recommendations to test a graph-based method of combining collaborative and content-based filtering in a digital library user service.[8] However, their collaborative filtering based user similarity on demographic information only, rather than the more usual technique of matching users based on similar ratings.

Andrew Schein and colleagues proposed an approach to the cold-start problem that used Bayesian techniques.[9] However, results show a naïve Bayes recommender outperforms their aspect model approach in predicting users' ratings for items of known interest to them. The researchers didn't supply any coverage measures of the cold-start recommendations their system generated.

References

1. P. Resnick et al., "GroupLens: An Open Architecture for Collaborative Filtering of Netnews," *Proc. Conf. Computer Supported Cooperative Work* (CSCW 94), ACM Press, 1994, pp. 175–186.
2. U. Shardanand and P. Maes, "Social Information Filtering: Algorithms for Automating 'Word of Mouth,'" *Proc. Human Factors in Computing Systems Conf.* (CHI 95), ACM Press/Addison-Wesley, 1995, pp. 210–217.
3. J.S. Breese, D. Heckerman, and C. Kadie, "Empirical Analysis of Predictive Algorithms for Collaborative Filtering," *Proc. 14th Conf. Uncertainty in Artificial Intelligence*, Morgan Kaufman, 1998, pp. 43–52.
4. B. Smith and P. Cotter, "Personalised Electronic Program Guides for Digital TV," *AI Magazine*, Summer 2001, pp. 89–98.
5. A.M.A. Wasfi, "Collecting User Access Patterns for Building User Profiles and Collaborative Filtering," *Proc. 1999 Int'l Conf. Intelligent User Interfaces*, ACM Press, 1999, pp. 57–64.
6. M. Balabanović and Y. Shoham, "Fab: Content-Based, Collaborative Recommendation," *Comm. ACM*, vol. 40, no. 3, Mar. 1997, pp. 66–72.
7. M.J. Pazzani, "A Framework for Collaborative, Content-Based and Demographic Filtering," *Artificial Intelligence Rev.*, Dec. 1999, pp. 393–408.
8. Z. Huang et al., "A Graph-Based Recommender System for Digital Library," *Proc. ACM/IEEE Joint Conf. Digital Libraries* (JCDL 2002), ACM Press, 2002, pp. 65–73.
9. A.I. Schein et al., "Methods and Metrics for Cold-Start Recommendations," *Proc. 25th Ann. Int'l ACM SIGIR Conf. Research and Development in Information Retrieval* (SIGIR 02), ACM Press, 2002, pp. 253–260.

ratings, even if the correlation with the current user's ratings is statistically insignificant. Although this approach might generate a larger set of films for making recommendations, it would likely also reduce the prediction accuracy.

To make its predictions, our collaborative-filtering process uses the peer ratings and gives a weighted average to each film according to the strength of each peer's correlation with the current user. Conveniently, the value of Pearson's r for each user in the first stage is in the range −1 to 1, where 0 indicates no correlation, 1 indicates a perfect positive correlation, and −1 indicates a perfect negative correlation. The agent can use peers with significant positive correlations to generate predicted ratings.

The weighted mean equates to the predicted rating for the film, and we calculate it as follows:

$$w_f = \frac{\sum_{p \in P} v_{p,f} \times r_p}{n}$$

where w_f is the weighted mean for film f, P is the set of significant peers of the current user, $v_{p,f}$ is the rating given by peer p to film f, r_p is the correlation coefficient calculated for peer p, and n is the current user's number of significant peers.

Once all calculations are complete, the agent stores the list of films and predicted ratings. The system also stores the number of significant peers who rated the film because it gives an indication of the potential recommendation's strength. The system can therefore use this number as a secondary sorting field when it produces recommendation lists.

The system then feeds the predicted ratings into the content-based filtering algorithms.

Content-based filtering on collaborative results

We designed the content-based filtering process (bottom two rows in figure 2) to use information about each film with a content-based rating as input to the process of finding links to other similar films.

There are several ways to find links. For example, you could build film sets starring a particular actor and associate a predicted rating with each set. However, we used a simple scoring mechanism. For every rated film input to the process, the agent queries the film information database for relevant information

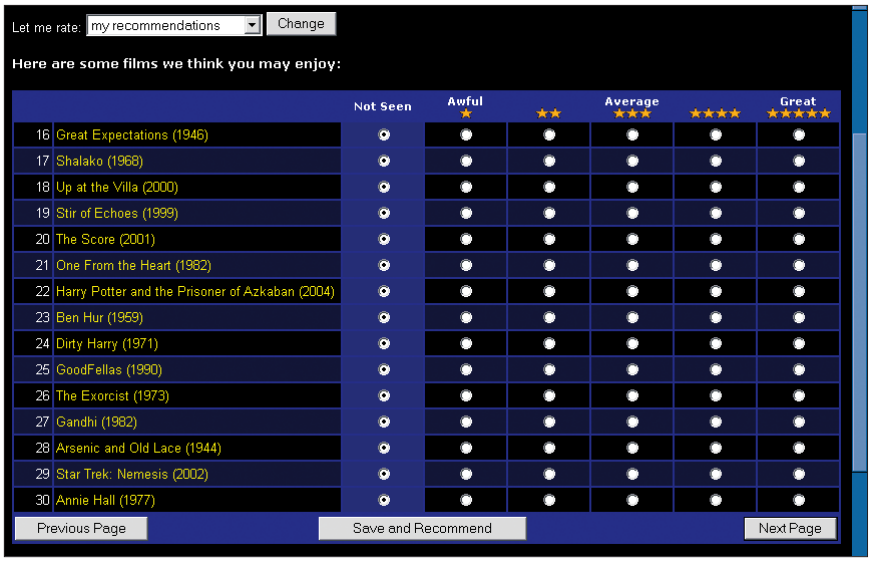

Figure 3. Recommendation display.

(actors, directors, and genres in this implementation). It then adds the film's rating (either predicted or user-given) to the score for each film element. For example, if the predicted rating generated by collaborative filtering for a film were 5, each actor who starred in the film would have 5 added to his or her score. Similarly, the agent would add 5 to the director's score and the score for each genre associated with the film.

Once it completes this process for all ratings, the agent calculates the average score for each actor, director, and genre. This score indicates how much the user likes or dislikes each element.

The agent can then compute the predicted rating for each film. For each element that received an average score, the agent queries the film information database regarding the film. In a process similar to that for finding links, the element's average score is added to the film's score. System administrators who are configuring the recommender system can also assign weights to the elements—for example, giving more weight to the actors than to a movie's genre.

The agent can then compute the predicted rating by dividing the film's total score by the number of elements used to calculate it. The agent can augment the list of films and predicted ratings with any predictions that resulted from the initial collaborative-filtering process but didn't appear in the final prediction set (because of incomplete film information in the database). The agent also records the number of elements for each film as an indicator of the prediction's strength, again so it can use the information as a secondary sort field when it creates recommendation lists.

Displaying recommendations

We designed our system to be flexible, allowing system administrators to turn off one of the filtering algorithms. If the administrator turns off collaborative filtering (or if collaborative filtering generated no recommendations), the content-based filtering module would have no input, so the actual ratings given to films are also used for content-based filtering. Where both a predicted rating from collaborative filtering and an actual user rating are available for a film, the agent uses the actual rating because it more accurately indicates the user's feelings about the film.

Generating recommendations uses a lot of server resources, so recalculating them more often than necessary is undesirable. Instead, after generating a recommendation set, the system caches it for the remainder of the session, unless it encounters recalculation trigger events, such as the user explicitly asking for a recalculation or a counter reaching a trigger value for the number of new items rated since the recommendation set was last generated.

Figure 3 shows the system's Web-based interface for displaying recommendations. Recommendation lists are paged, and buttons at the bottom of the list facilitate access to next and previous pages.

Users can rate any of the recommended films they've already seen by selecting a radio button to the right of the film title. The system saves these new ratings whenever a button is pressed. Users can force the system to produce recommendations at any stage by clicking the Save and Recommend button. By submitting ratings, users are providing implicit feedback about the quality of the recommendations produced.

Users can specify the film types they wish to rate to receive better recommendations by using the "Let Me Rate" drop-down menu at the top of the page. Items in the list include films from a particular decade, films within a particular genre, and a complete listing of films the user has not yet rated.

Cinema-based recommendations

The system includes a Web robot to retrieve showtimes from cinema Web sites so users can optionally restrict recommendations to films showing at cinemas near them.

The robot crawls highly targeted Web sites, as we require it to retrieve only explicit information (cinema name and location, film titles, and dates and times of showings) from specified cinema sites. A series of detailed templates define the crawl, giving the robot exact directions about what parts of each Web page to extract. These templates give us confidence in the retrieved data's quality (for example, the name of the town where the cinema is playing really is the name of the town and not part of a film title).

In the UK, system users select their local cinemas through an extension of the Web interface. After selecting their postal code area (defined as the first one or two letters at the beginning of their postcode) and the date they wish to visit the cinema, the system presents a list of available cinemas. Users can deselect any cinemas they don't wish to visit.

The system generates a list of films showing at the selected cinemas and extracts matching films from the full list of generated recommendations. It displays the list in an enhanced version of the standard results interface, as shown in figure 4. For each film in the list, the display shows the cinema where it's playing. The list includes films the user has previously rated if they meet certain rating strength conditions.

System performance tests

To test the system's performance, we used a data set consisting of 100,000 film ratings by 943 users. The University of Minnesota's GroupLens Research Group makes this data set publicly available for use in recommender system testing. The group collected the data between September 1997 and April 1998 on

their MovieLens Web site. The data set contained ratings given on an integer scale between one star (poor) and five stars (excellent). Recently, a new version of the MovieLens data set consisting of one million ratings by 6,000 users was released; we will test our system with it in the near future.

Our tests involved automatically generating recommendations for a random sample of 200 users. We removed a number of ratings from the data set for each user and attempted to generate predicted ratings. We wanted to compare our technique with four others:

- collaborative filtering used in isolation,
- content-based filtering used in isolation,
- collaboration-via-content filtering, and
- running the results of content-based filtering through collaborative filtering.

Numerous metrics are available for evaluating recommender systems.[2] Accuracy metrics evaluate how well a system can predict a rating for a specified item, which is a key measure of a recommender system's success. We chose to use precision and recall in our testing because they're popular, well-established metrics from the information retrieval community. Precision measures the probability that the system's selected films will be relevant to the user, while recall measures the probability that the system will select the entire set of relevant films.

We don't believe recall to be as important as precision for our recommender. Because a visit to the cinema involves a financial outlay, users should prefer a recommendation for a film they are sure to enjoy (precision) over a recommendation for all the films they might enjoy (recall). Erroneous recommendations from low system precision could decrease confidence in the system to the point where users might not trust any advice it gives.

Nevertheless, accuracy alone isn't sufficient proof of a recommender's usefulness. For example, a recommender might be highly accurate but produce rating predictions for only a small number of items. Therefore, we also use *coverage metrics* to indicate the number of films our system can produce predictions about. We used three types of coverage in our testing:

- *Standard coverage* measures the average number of films for which the system can produce predictions for each user. We calculate this as a percentage of the total films (6,459) in the database.

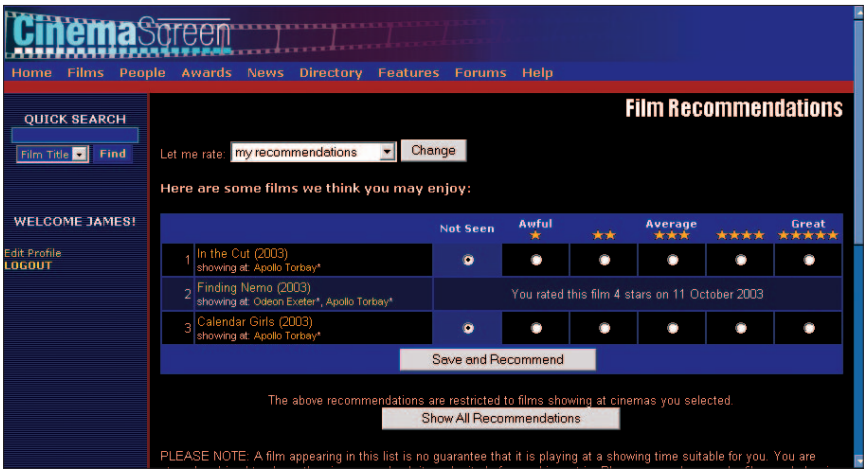

Figure 4. Interface to recommendations for films playing at local cinemas.

- *Catalog coverage* shows the percentage of films in the database for which the system ever generates predictions. We use it as a measure of recommendation diversity.
- *Prediction coverage* shows how many of the films removed from the user's rating set in our experiment to test for unrated (new) films could be regenerated by each recommendation technique. By listing the predictions generated for films the user rated above each star threshold, we show how many predictions the technique generates for films the user would actually be interested in watching. We express prediction coverage as a percentage of the total number of ratings removed from the test data set (50 for each user).

Figure 5 plots each technique's precision and recall at various thresholds—that is, the point on a five-star rating system that marks the border between a "good" and "bad" film. Observing the overall precision, our technique is slightly—but not significantly—better than others, excepting content-based filtering. However, we can't consider content-based filtering to have the best overall performance, because it's recall is the worst at thresholds below four stars and misses approximately a third of recommendations at the one-star threshold.

If the user's threshold is set at 3.1, our agent's precision and recall are equivalent to any other technique, whereas content-based filtering could recall less than 60 percent of movies. A 3.1 threshold is significant, as it separates between one, two, and three stars ("not relevant") and four and five stars ("relevant"). We could also choose 3.5 as a sensible threshold; at this point, our technique shows equivalent precision and recall of around 10 percent less than the best-performing technique—collaboration via content.

The recall graph shows our recall dropping at a higher rate than other methods beyond

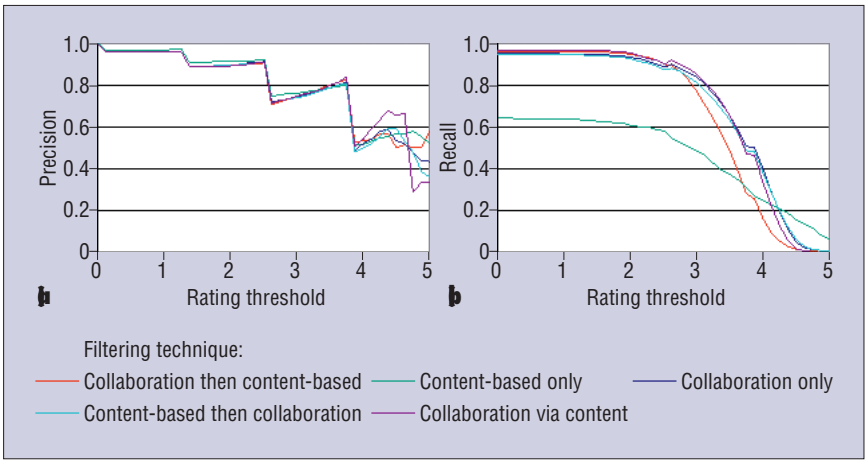

Figure 5. Comparison of five recommender techniques for (a) precision and (b) recall.

Table 1. Coverage of predictions using different recommendation techniques.

Filtering technique	Standard coverage (percent)	Catalog coverage (percent)	Prediction coverage (percent, for *n* stars and above) 1	2	3	4	5
Collaborative then content-based	67.5	71.8	97.6	97.7	98.1	98.3	99.1
Content-based only	40.9	67.2	65.2	65.7	66.5	68.3	70.4
Collaborative only	11.9	14.2	96.5	96.6	97.2	97.3	98.0
Content-based then collaborative	11.8	15.5	96.3	96.5	97.0	97.1	97.4
Collaboration via content	13.3	14.2	97.9	98.0	98.3	98.3	98.5

Table 2. Coverage of predictions using different recommendation techniques for new movies.

Filtering technique	Standard coverage (percent)	Catalog coverage (percent)	Prediction coverage (percent, for *n* stars and above) 1	2	3	4	5
Collaborative then content-based	64.4	66.4	98.8	98.8	98.8	98.7	98.9
Content-based only	52.6	62.4	76.7	76.3	75.4	74.7	76.2
Collaborative only	12.0	12.7	0	0	0	0	0
Content-based then collaborative	11.8	12.7	0	0	0	0	0
Collaboration via content	12.4	12.7	0	0	0	0	0

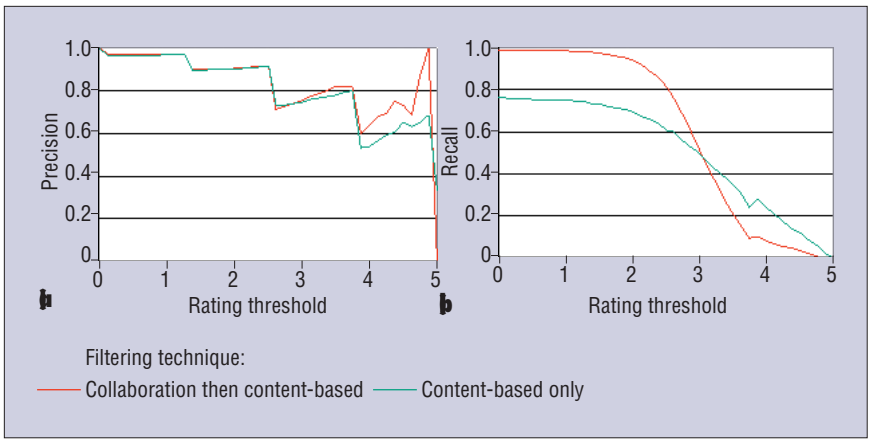

Figure 6. Comparison of two techniques for recommending new movies: (a) precision and (b) recall.

three stars. Having high precision but lower recall shows that our agent can select four- and five-star films as well as other techniques but has less capability in distinguishing between these films. Our agent made conservative recommendations, clustering films that users in the test data set had rated as four or five stars closer to three stars. The content-based filtering algorithm causes this averaging effect because it treats actors, directors, and genres independently. For example, a user might give five-star ratings to three comedies starring Robin Williams, but dislike two dramas he appeared in, giving them both one star. The overall effect would be to lower the predicted rating for a fourth Robin Williams comedy, even though the user appears to always enjoy his comedies.

Table 1 summarizes coverage measures for each recommendation technique. Recommender algorithms involving collaborative filtering in their final stage exhibit standard coverage approximately six times lower than our technique. This is because collaborative filtering can only make predictions for films that at least a few users have rated. Catalog coverage measurements support this, with the last three techniques in the table also exhibiting catalog coverage over only a sixth of the database's films. Catalog coverage is a useful guide to the novelty of recommendations; a system with low catalog coverage might only be recommending blockbuster films that all users are likely to enjoy rather than selecting films the user might not be aware of.

Coverage is particularly important for our agent, which users employ to recommend movies from a small list of those showing at the local cinema. If coverage was low, the system wouldn't be able to make a prediction. With the exception of content-based filtering, each technique had roughly comparable prediction coverage. Content-based filtering's prediction coverage is low because not all films in the database have sufficient information about directors, genres, and actors for the algorithm to deduce relationships among them.

Because we designed our system to let users view recommendations of movies at their local cinema, both coverage and prediction accuracy for new movies is important. It's highly probable that no users will have rated new movies at the time the system needs to generate predictions for them. To test this, we modified our initial experiment. First, we removed all ratings for a set of films, thus simulating new movies that users hadn't yet rated. Then we used the remaining rating information for each user and attempted to regenerate predictions for this film set.

The prediction coverage is zero for the last three techniques in table 2 because none of them can make a prediction about a given movie unless some users have rated it. Our system's prediction coverage is high because the content-based algorithm has enough information about the films to deduce relationships.

Figure 6 shows the precision and recall results from the revised experiment for our system and content-based filtering—the only

two techniques that produced predictions for the simulated new movies.

Our agent's precision is significantly higher than content-based filtering for thresholds above three stars, with recall becoming worse after three and a half stars. However, the precision isn't lower than in the first experiment, which means we can be confident in our system's ability to generate predictions for new movies of a similar quality as for other movies in the database. Again, with a user's threshold set at 3.1, our technique classifies films as well as content-based filtering. At a threshold of 3.5, our precision and recall are identical to those of content-based filtering. When these scores are coupled with increased coverage, our technique is clearly better than others for recommending new movies.

The second experiment confirms that our technique suffers only in the capability of distinguishing between four- and five-star films to gain increases coverage and precision. Some precision improvements are marginal but others are significant under certain conditions.

Overall, our testing results show that our agent can make more recommendations than collaborative-filtering techniques and better recommendations than content-based filtering in terms of precision.

All of our measurements assume that users rate movies randomly rather than rating a movie immediately after seeing it. If this is not the case, the value of our enlarged standard and catalog coverage is decreased, but the value of our prediction coverage and recall will increase. Recall will tend toward a measure of the total number of relevant films that the system would recommend and the user would like. If the system's true recall were 100 percent, further increasing coverage would have no impact on the number of relevant recommendations the system could produce.

Improving recommendations

We believe we can further improve our system's prediction accuracy by improving the content-based filtering algorithm. The algorithm appears to give an averaging effect, clustering predicted ratings around the midpoint. We will investigate methods to reduce this effect by creating more complex groupings of actors, directors, and genres, rather than treating each as an independent entity. However, generating groupings of every combination of elements for each movie would have a devastating effect on the agent's time complexity. The diversity of recommendations would also suffer, as only films sharing several common elements would be considered similar. Finally, this would eliminate the possibility of recommending a new film in, for example, the user's favorite genre if it doesn't have any directors or actors in common with previously rated movies. Therefore, we must investigate alternative methods that complement rather than interfere with the existing behavior.

The current agent is an incomplete model of the film recommendation process. It implements only some of the most obvious aspects. For example, the agent doesn't consider changes in user tastes and attitudes over time. Another nontrivial parameter is a model of the way an actor or director's career evolves. For example, a user might dislike certain actors in their early films but rate them higher as their skills developed in later films. The agent doesn't recognize these patterns; it creates an overall average score for each actor.

The agent has no mechanism for determining which actors played the main roles in a film and which made shorter appearances. For example, the system establishes a list of the user's preferred actors at the start of the session. The agent could recommend a new film based on its inclusion of one or more of these preferred actors. However, these actors might only appear in short cameo roles and in reality have little bearing on the user's enjoyment of the film.

We will use the results and experience from this system's development to further investigate the integration of content-based and collaborative-filtering techniques. We plan to explore two avenues of further research.

First, we want to enable the assignment of different weightings to each filtering technique's results according to certain parameters. We also want to have the option of reversing the order of the filtering techniques used. For example, if the users' similarity is based on a small number of ratings, we'd like to assign higher weights to content-based filtering recommendations because the collaborative-filtering results include a high level of uncertainty. We want to test machine learning methods for calculating these weightings. Such methods might also be appropriate for dynamically altering the component weightings within the content-based filtering algorithm. For example, a highly rated director might have more influence on a film recommendation than a highly rated actor.

Second, we want to apply and evaluate our hybrid recommendation method to other domains and emerging technologies. Grid computing and peer-to-peer networking rely on users being able to find resources. Recommender technology could help users find potential matches, including those in hidden resources available on the network.

Mobile computing is another application area. Current 3G mobile phones provide a viable platform for delivering rich content to mobile devices. However, users must pay for the bandwidth they consume and the limited screen size makes it cumbersome to select items from large lists. Recommender technology could help narrow the choices users must make and enable the selection of appropriate high-quality content. ■

The Authors

James Salter is a PhD student in the Department of Computing at the University of Surrey. His research interests include recommender systems and resource discovery in peer-to-peer networks. He has a BSc in computing and information technology from the University of Surrey. Contact him at the Dept. of Computing, Univ. of Surrey, Guildford, Surrey, GU2 7XH, UK; j.salter@surrey.ac.uk.

Nick Antonopoulos is currently lecturer and director of MSc Programmes in the Department of Computing at the University of Surrey. His research interests include emerging technologies, such as Web services, peer-to-peer networks, software agent architectures, security, communication, and mobility management. He received his PhD in agent-based Internet computing from the University of Surrey. Contact him at the Dept. of Computing, Univ. of Surrey, Guildford, Surrey, GU2 7XH, UK; n.antonopoulos@surrey.ac.uk.

References

1. J. Crawshaw and J. Chambers, *A Concise Course in A-Level Statistics*, 3rd ed., Stanley Thornes, 1994, pp.658–664.

2. J.H. Herlocker et al., "Evaluation Collaborative Filtering Recommender Systems," *ACM Trans. Information Systems*, vol. 22, no. 1, 2004, pp.5–53.