Getting to Know You: Learning New User Preferences in Recommender Systems

Al Mamunur Rashid, Istvan Albert, Dan Cosley,
Shyong K. Lam, Sean M. McNee, Joseph A. Konstan, John Riedl

GroupLens Research Project
Department of Computer Science and Engineering
University of Minnesota
Minneapolis, MN 55455 USA
{arashid, ialbert, cosley, lam, mcnee, konstan, riedl}@cs.umn.edu

ABSTRACT

Recommender systems have become valuable resources for users seeking intelligent ways to search through the enormous volume of information available to them. One crucial unsolved problem for recommender systems is how best to learn about a new user. In this paper we study six techniques that collaborative filtering recommender systems can use to learn about new users. These techniques select a sequence of items for the collaborative filtering system to present to each new user for rating. The techniques include the use of information theory to select the items that will give the most value to the recommender system, aggregate statistics to select the items the user is most likely to have an opinion about, balanced techniques that seek to maximize the expected number of bits learned per presented item, and personalized techniques that predict which items a user will have an opinion about. We study the techniques thru offline experiments with a large pre-existing user data set, and thru a live experiment with over 300 users. We show that the choice of learning technique significantly affects the user experience, in both the user effort and the accuracy of the resulting predictions.

Keywords

Recommender systems, collaborative filtering, information filtering, startup problem, entropy, user modeling.

INTRODUCTION

People make decisions every day. "Which movie should I see?" "What city should I visit?" "What book should I read?" "What web page has the information I need?" We have far too many choices and far too little time to explore them all. The exploding availability of information that the web provides makes this problem even tougher.

IUI'02, January 13-16, 2002, San Francisco, California, USA.

Recommender systems help people make decisions in these complex information spaces. Recommenders suggest to the user items that she may value based on knowledge about her and the space of possible items. A news service, for example, might remember the articles a user has read. The next time she visits the site, the system can recommend new articles to her based on the ones she has read before.

Collaborative filtering is one technique for producing recommendations. Given a domain of choices (*items*), users can express their opinions (*ratings*) of items they have tried before. The recommender can then compare the user's ratings to those of other users, find the "most similar" users based on some criterion of similarity, and recommend items that similar users have liked in the past.

When new users come along, however, the system knows nothing about them. This is called the *new user problem* for recommender systems [1, 2, 6]. The system must acquire some information about the new user in order to make personalized predictions. The most direct way to do this is to ask for ratings directly by presenting items to the user.

However, the system must be careful to present useful items that garner information. A food recommender, for instance, probably should not ask whether a new user likes vanilla ice cream. Most people like vanilla ice cream, so knowing that a new user likes it tells you little about the user. At the same time, the recommender should ask about items the user is likely to have an opinion about. A travel recommender would probably not benefit by asking a new user if she liked Burkina Faso, for instance. The recommender system is likely to learn only that, like most people, she has not visited Burkina Faso, which is of little value in forming future travel recommendations.

The choice of exactly what questions to ask a new user, then, is critical. An intelligent recommender interface will minimize a new user's effort and get him to the fun part—using the system and seeing recommendations—while still learning enough to make good recommendations.

In this paper we explore approaches for choosing which items to present to new users for rating. We consider this problem in the general case of recommender systems, illustrating strategies and performing experiments using the

MovieLens movie recommender. We first survey related work in the areas of decision theory and recommender systems, then consider approaches for selecting movies to present to users. We test these approaches on historical data drawn from the 7.5 million-rating MovieLens dataset. We also test three of the most promising strategies on over 300 new MovieLens users. We then discuss the results and suggest directions for future work.

RELATED WORK

We briefly mention related work in the field of decision theory and survey work that has been done on the new user problem in the area of recommender systems.

Decision theory and entropy

Decision theory has proved useful in determining models for re-ordering search results [4]. This application of utility functions has also been used in recommender systems [13, 14].

Analysis of data for entropy—its theoretical information content—has been a standard technique used in information retrieval [10], medical diagnostic systems [9], and sequential classification problems [3] for many years. Lately, researchers have extended the use of entropy into areas such as probabilistic models for information retrieval [7] and value-of-information analysis [16].

We apply decision theory techniques to a new problem: choosing the items to first present to a new user of a recommender system. Our problem is in some ways the converse of the cited research; we are selecting items as questions to present to the user, rather than choosing which answers to present for a user's question.

Recommender systems and the new user problem

There has been little work in solving the new user problem by analyzing ratings data to make smart decisions. Pennock and Horvitz proposed the use of a "value-of-information" calculation to discover the most valuable ratings information to next gather from a user [14]. To our knowledge, they have not published any implementations or evaluations of their calculations.

Kohrs and Merialdo make use of entropy and variance in their ratings data in order to generate more accurate predictions for new users [12]. Our work expands their results by using a number of strategies that we consider as being more suitable than variance or entropy. We also have a much larger dataset for our offline experiments and verify our findings in a live experiment.

Another approach to solving the new user problem creates pre-made user categories and quickly assigns new users to one of them. The partitioning can be accomplished by asking the user pre-determined questions that build a user preference structure. This helps jump-start the user into the system without requiring a substantial number of ratings [8, 13]. This class of approaches addresses the question of what to present first by starting with a small set of preference models (e.g. demographic models, models based on attributes of items) and asking questions that help choose an appropriate model for a user. When these models are accurate they can be quite useful, but the premise of personalized recommender systems and collaborative filtering is that a person's preferences are a better predictor of other preferences than other attributes. Category and demographic models are thus less general than the methods we present; they apply only to certain domains, and require domain-specific expertise.

Filterbots are a technique to overcome the startup problem for new *items* in a collaborative filtering system by injecting ratings agents that rate every item in the system according to their algorithmic analysis of the content of the item [6]. Filterbots can make sure that every item in the system has many ratings to help users find the items they are most interested in. However, filterbots do not directly attack the new *user* problem.

Others have integrated agents into a collaborative filtering environment to extract user preference information transparently [17]. This method has the advantage of collecting implicit information in addition to explicitly provided ratings, and should gather data for new users more rapidly. Using implicit data in addition to explicit data is a promising approach, and is complementary to our approach of carefully selecting which explicit data to collect.

STRATEGIES FOR SELECTING ITEMS TO PRESENT

There are trade-offs to be made when choosing a strategy for presenting items. As discussed in the introduction, requiring too much effort of the user will cause some users to give up, while not asking enough questions will result in poor recommendations. We identify four dimensions that a strategy might choose to support: (a) *User effort*: how hard was it to sign up? (b) *User satisfaction*: how well did the user like the signup process? (c) *Recommendation accuracy*: how well can the system make recommendations to the user? (d) *System utility*: how well will the system be able to serve all users, given what it learns from this one?

We choose to focus on user effort and accuracy. We chose these two is because they are easy to measure and can be measured in both off-line and on-line experiments. User satisfaction studies are difficult to do off-line from historical data, and we believe that user satisfaction will rise as user effort falls. While we touch on a few issues related to system utility, such as the danger of introducing biases into a system's ratings database when using certain strategies, we do not focus on it since our primary focus is on factors that directly influence the user's experience.

We consider several types of strategies for presenting items, ranging from random selection, through strategies that exploit aggregate properties of the system's database such as choosing popular items, to strategies that tune themselves to individual users.

Random strategies

Random strategies avoid bias in the presentation of items. We consider two variants.

Random. Select items to present randomly with uniform probability over the universe of items.

MovieLens Classique. For each page of movies presented, select one movie randomly from an ad hoc manual list of popular movies and the rest randomly from all movies.

Discussion. Random strategies have the advantage of collecting user preference data over the entire universe of items. If the distribution of ratings is not uniform, however, users will likely not have seen many of the randomly selected movies. MovieLens Classique tries to boost the chance that users will have at least one success per page.

Popularity

In MovieLens, the number of movies with a given number of ratings decreases in an approximately exponential manner, deviating from this exponential form for the least- and most-rated movies.

Popularity. Rank all items in descending order of number ratings. Present movies in descending popularity order.

Discussion. This strategy is easy to calculate and should make it easy for users to rate movies. However, popular movies may be widely liked; if this is true, then their ratings carry little information. If everyone likes *Titanic*, and I say I like it too, what can the system learn from that?

Another concern when using the popularity strategy is the possibility of exacerbating the *prefix bias*. Popular movies are easier for the system to recommend, because similar users are more likely to have seen them. Since users can rate any movie the system recommends, popular movies garner still more ratings. Unpopular movies suffer from the same problem in reverse: they are hard to recommend, so users see them and rate them less often than they otherwise might. This bias may explain the deviation from exponential form for the popularity distribution of movies.

Pure entropy

An alternative approach is to ask users about movies that will give us the most information for each rating. Informally, a movie that has some people who hate it and others who like it should tell us more than a movie where almost everyone liked it. Kohrs and Merialdo used both variance and entropy to get at this notion [12].

Pure Entropy. For each movie, calculate its entropy using the relative frequency of each of the five possible ratings. Sort the movies in descending order of entropy. Present the movies with the highest entropy that the user has not seen.

Discussion. There are several choices to make when using an entropy-based strategy. The first is how to handle missing ratings. We choose to ignore them in the calculation because the information content of a missing rating is hard to measure: a user may have chosen not to see it, never heard of it, or seen it but not thought to rate it. Another choice is whether to compute the entropy over each rating individually, or whether to convert ratings into a binary "like vs. dislike" model where ratings of 4 or 5 indicate like while ratings of 1 to 3 indicate dislike.

Finally, "most information" in the technical sense meant by entropy does not necessarily translate into information usable by the system. A movie with only two ratings, a 1 and a 5, has high entropy but little value in finding similar users or making recommendations. Similarly, a recommender may present high entropy movies, but if the user has not seen any of them, the system will gain no information at all. We performed a pilot study for the on-line experiment using a pure entropy strategy, and it turned out to be unusable. Two users had to view several hundred movies each before finding ten to rate.

Balanced strategies

Popularity-based strategies tend to get many ratings from users, but each rating may have low information-theoretic value to the recommender system. Conversely, entropy-based techniques get a lot of value from each rating, but users may find relatively few items to rate. In this section we consider balanced techniques that attempt to obtain many ratings, each of which has a relatively high value. In a sense, these techniques are working to obtain as many *expected* bits of information as possible from each item presented for the user to possibly rate.

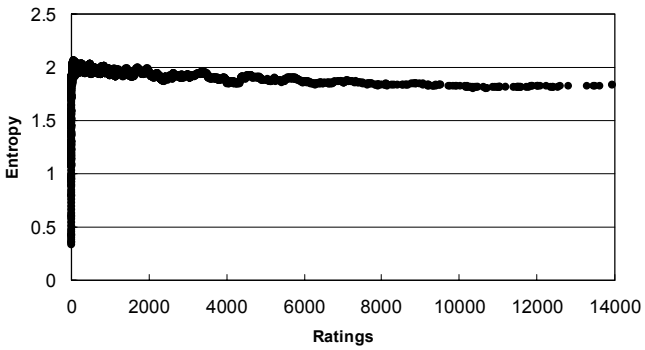

Figure 1. Entropy vs. the popularity of a movie, smoothed by using a moving average entropy of the previous 50 most popular movies.

*Popularity*Entropy (Pop*Ent)*. Rank items by the product of popularity and entropy. Entropy is the number of bits of information if the user rates the item, and popularity is the probability that this user will rate this item. Using Bayes' theorem in this way assumes that popularity and entropy are independent, which is unlikely to be strictly true, but the approach is likely to be a good approximation to expected number of bits. Further, our experience with MovieLens ratings suggests that popularity and entropy are not strongly correlated. Figure 1 plots the average entropy of movies with a given popularity ranking using a moving average over the 50 prior most popular movies. This figure

shows that there is little correlation between a movie's popularity and entropy except for movies with few ratings.

*Log Popularity*Entropy (log Pop*Ent).* As above, but take the log of the number of ratings before computing popularity. In studying Entropy and Popularity we observed that their distributions in our dataset were such that Popularity almost completely dominated Pop*Ent, with both strategies producing nearly the same sets of rankings. Taking the log of the ratings nearly linearized popularity, making it a better match for entropy.

Discussion. Figure 1 suggests that entropy alone may not be an effective strategy, since entropy is nearly independent of popularity. Thus, entropy alone will sometimes choose items that users have low probability of having an opinion about. Balanced techniques directly combine entropy and popularity to increase both the odds that a user will be able to rate movies that the recommender presents and the expected value of that rating.

Personalized

The strategies above all use aggregate statistics. However, the overall popularity of an item is only a rough approximation for the chance that a particular user has seen it. Ideally, the movies presented to a user could be tailored to that user as soon as we have some information about that user. Once we know that a user has rated *Ghostbusters*, we might want to show other movies rated by people who have rated *Ghostbusters*. The goal is to hone in on movies that the user is likely to have seen in order to make the signup process easier and require less user effort. A simple personalized strategy uses *item-item similarity.*

Item-Item personalized: Present movies using any strategy until the user has given at least one rating. Then use a recommender that computes similarity between items to select other items that the user is likely to have seen. Update the list of similar movies whenever the user submits more ratings, remembering movies that the user has already seen so that they are not presented again. For our experiments, we present the initial screen of movies as a random selection from the top 200 movies ranked by the log Pop*Ent strategy.

Discussion. Personalizing the movies we present is similar to, but not the same as, recommending movies. When we recommend movies, we try to identify movies the user will *like*; when presenting movies, we only care whether he has *seen* a movie. The SUGGEST recommender, used as the item-item recommender in our experiments, was developed with e-commerce in mind and uses binary ratings (e.g. the user bought the item) [11]. It accepts a list of items the user has bought and returns a list of other items the user would be most likely to buy. This is exactly the task we face: given a list of movies the user has seen, what other movies is he most likely to have seen.

One possible disadvantage for the item-based personalized strategy is that seeing a movie is probably correlated with liking a movie. The average rating in the MovieLens dataset, for example, is close to 4 on a 1 to 5 scale. This means that we may get mostly positive ratings for the new user, which is not as useful as knowing both some movies that the user likes and some that she dislikes.

Other plausible strategies

There are a number of other plausible strategies that we do not consider in this paper. The system might ask attribute-based questions of the user, although as mentioned earlier such strategies are domain-dependent. The system might also ask for the names of items a user likes. CDnow.com does this, explicitly asking for album titles the user liked. A recommender system could also preferentially display new items, or items that have recently been added to its database. Finally, it could perform a more sophisticated analysis of entropy and personalization than we attempt in this paper and try to select items with high independent entropy. We focus on domain-independent strategies, and within these on the simplest ones; exploring more complex strategies would be fertile ground for future work.

OFFLINE EXPERIMENTS

We decided to first explore the strategies mentioned above by performing off-line experiments that use historical data to simulate the signup process for new MovieLens users. The benefit of these offline experiments is that we can quickly test a variety of strategies without bothering actual users with strategies that turn out in practice to work poorly. A disadvantage of these offline experiments, described in detail below, is that biases in our existing data may bias the results for or against particular approaches. We identify the biases as carefully as we can, and interpret our results in that context. Still, these experiments were invaluable to us in ruling out several algorithms that would have been painful for actual users.

Experimental Design

To build the dataset for the off-line experiments, we took a snapshot of the MovieLens ratings database and eliminated users who had fewer than 200 ratings. This left 7,335 users, 4,117 movies and more than 2.7 million ratings. The cutoff of 200 is both high and somewhat arbitrary. However, we needed a large number of ratings for each user as in the historical data it is hard to know which movies the user might have seen other than through their ratings. We needed many ratings for each user so we had a good sample of movies they were able to rate.

We tested the Pure Entropy, Random, Popularity, Pop*Ent, and Item-Item personalized strategies. We did not test the MovieLens Classique strategy because the historical data were gathered with the Classique strategy and we feared possible bias.

To mimic the on-line sign-up process, we used each strategy to "present" a total of 30, 45, 60, or 90 movies to each user. We varied the number of movies presented so that we could see how the strategies performed as the system attempted to gather more information. When we

started a run, we withheld all of that user's ratings from the system. As we presented the movies, users "rated" the movies they had "seen" (i.e. those for which we had ratings for in the database).

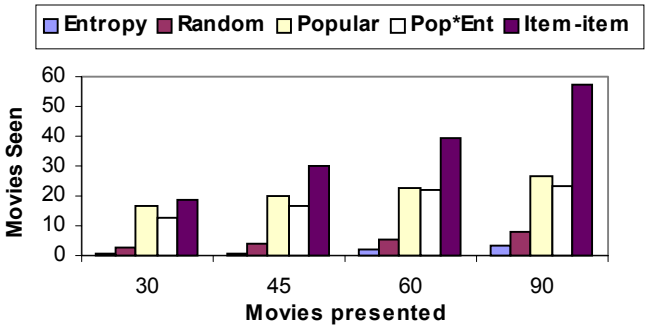

Figure 2. Number of movies seen versus number of movies presented to a user.

Once we had presented the correct number of movies, we counted the number of movies the user was able to actually rate. More ratings implied that we did a better job of showing items the user could rate. This is good: it means that we wasted less of the user's time looking at unratable items and that we can present fewer items to get the information the system needs, saving the user effort. After counting the rated movies, we used these as training data for that user and made predictions for all of the other movies the user had rated in the original dataset. We then calculated the Mean Absolute Error (MAE) for these predictions. MAE is the sum of the absolute differences between each prediction and corresponding rating divided by the number of ratings. We performed the entire procedure for each strategy for every user in the test set, and computed an average MAE across all users. Computing average MAE in this way counts all users equally, rather than biasing the results towards users with more ratings.

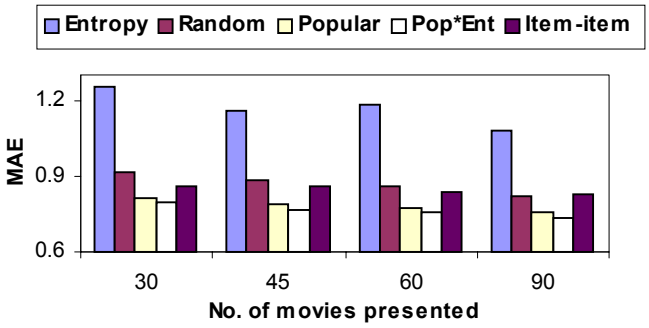

Figure 3. Mean Absolute Error (MAE) vs. the number of movies presented by each strategy.

Biases in the reduced dataset

The reduced dataset inherits several biases from the full MovieLens dataset. In particular, it has the prefix bias, where popular movies are easier to recommend and are shown (and rated) more often. This might give strategies that incorporate popularity an advantage in the number of movies they allow a user to rate. Our decision to remove users with less than 200 ratings also introduces possible bias. One bias is that our results may be most meaningful for active users. It is also possible that removing users with fewer ratings might artificially impact prediction accuracy. Excluding these users also resulted in a denser data set.

Results

Figure 2 shows that the Item-Item personalized strategy did the best in picking movies users can rate, while Pure Entropy was the worst.

Figure 3 shows the effect different strategies have on MAE. Pop*Ent performs best for a given number of presented movies, with Popularity close behind. Again, Pure Entropy is shockingly poor.

The poor performance of Pure Entropy in both metrics is directly related. Figure 1 shows a slight increase in entropy for less popular movies. Since popularity directly relates to the chance that a new user has seen a movie, this strategy presents movies that users are less likely to have seen, resulting in poor performance in the movies-seen metric. Moreover, with fewer rated movies to base predictions on, the MAE for Pure Entropy also suffered.

The Item-Item personalized strategy has the most interesting behavior. We expected it to win in the movies-seen metric, and it in fact trounced the competition. This did not translate into better recommendations, however. It was hard to believe that the Random strategy could get an error rate with eight ratings as training data comparable to the item-item personalized strategy with 57 ratings.

One possible reason is that the item-item strategy presented movies that it could otherwise have made accurate predictions for. Imagine that the system presents *Star Trek 23: the Bowels of Kirk* to a Star Trek fan, who rates it. The system looks and finds that most people who have seen *Bowels* have also seen *Star Trek N, 0<N<23*, and presents those movies next. Someone who has seen all of the *Star Trek* movies has probably rated most of them highly. If the system had only presented one of them, it would have had a good shot of finding other Trekkies in that user's neighborhood, and been able to make a number of accurate predictions, thus lowering MAE.

The problem seems to be that the item-item personalized strategy does not do a good job of sampling the entire space of movies. Item-item methods tend to find loose clusters of similar items and hone in on those clusters. This may cause poor exploration of the universe of items: the recommender may become an expert on *Star Trek* movies at the expense of others. This also helps explain why the random strategy

does well despite finding many fewer movies, as it samples from all genres and all levels of popularity.

We will further discuss the merits of the strategies after we present the results of our online experiment.

ONLINE EXPERIMENT

We followed up our off-line experiment by deploying several strategies on the live MovieLens site. By using live users, we could verify the results of the off-line experiment while removing the bias induced by only considering users who had at least 200 ratings. We also wanted to compare these strategies to the MovieLens Classique strategy.

We had planned to investigate all of the strategies in our online experiments. However, after our pilot study, we decided against the Random and Entropy strategies as the average number of movies a user would have to see before rating enough to get recommendations would be prohibitively high. Reading through hundreds of movie titles can be a frustrating process that would surely turn many users away. The pilot study also lead us to use the log Pop*Ent strategy instead of Pop*Ent, since Pop*Ent and Popularity alone chose almost the same set of movies.

Experimental Design

When a new user joins MovieLens, the system presents pages of ten movies until the user rates ten or more movies. We altered the signup process to ask users if they were willing to let the system use an experimental method for selecting movies for them to rate. Users who consented were assigned to one of three groups, which used the Popularity, log Pop*Ent, or Item-Item personalized strategy to present movies. Those who did not consent received the MovieLens Classique strategy. This self-selection introduces a bias, so we use the Classique strategy only as a baseline.

MovieLens had a total of 351 new users during the ten-day experimental period. Table 1 shows the number of users in each experimental group. Some users gave up before completing the sign-up process. Our results below are based on the users who completed the signup process.

Table 1. Population of the experimental groups.

Strategy	*Total Users*	*Dropouts*	*Completed*
Popularity	91	10	81
Item-item	92	10	82
logPop*Ent	92	13	79
Classique	76	16	60
Total	351	49	302

Our primary goal for the online experiment is to measure the effectiveness of the signup process: how many pages of movies must a user see before they have rated enough to get started? We believe this is a suitable proxy for the effort we require of the user, with fewer pages equaling less effort. We would like to measure prediction accuracy as well, but we do not have a good basis for computing MAE immediately after a user signs up. We could compute it on the movies they rated during the signup process (MovieLens logs predictions for these movies to support retrospective analysis). However, since the purpose of the signup process is to gather information, judging error during the signup process does not make much sense.

User interaction is quite difficult to foresee let alone quantify. Some users have rated all the movies on the first page of a random sample, a highly unexpected event, while others have waded through dozens of pages with popular movies, seemingly not being able to rate a single one from them. We included all of these users without prejudice.

Expectations

Both the Popularity and the log Pop*Ent approaches are expected to show a slow decrease in the number of movies matched per page, up to the point where most of the users finish with the signup. This is a natural consequence of the fact that we pre-sorted the movies and presented them in descending order of the corresponding parameter. We also expected the Item-Item personalized strategy to perform no better than log Pop*Ent on the first page since it uses that strategy to select the initial set of movies. We did expect Item-Item to outstrip the other strategies on subsequent pages, showing that it was successfully finding movies that users had seen.

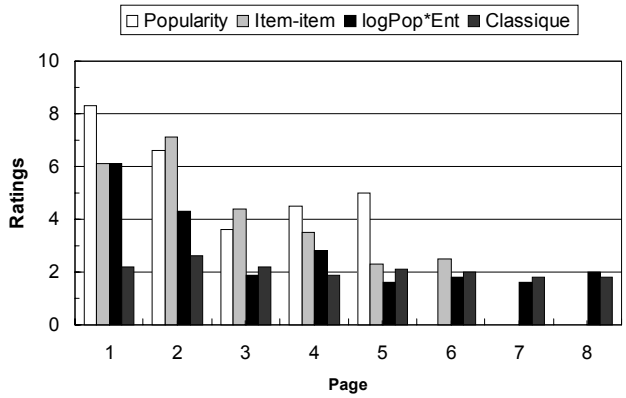

Figure 4. Number of movies users could rate per page using different movie presentation strategies.

Results

Figure 4 shows the number of movies per page an average user was able to rate with each of the strategies. Popularity and log Pop*Ent exhibit the decay we expected, although they both rose slightly after three pages. When the Item-Item recommender kicks in on the second page the users are able to rate more movies than with any other strategy, and more movies than they did on the first page. Classique was approximately constant across all pages.

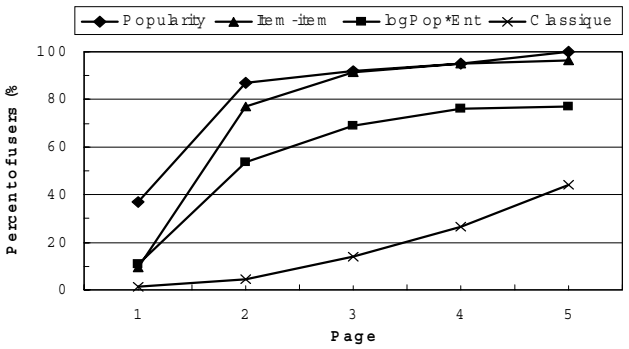

Figure 5. Cumulative percentage of users who finished signing up after a given number of pages.

From a user's point of view, the ease of the signup process is probably best expressed as the number of pages of ten movies he must see before starting to get recommendations. The mean number of pages varied from around two for the Popularity (1.9) and Item-Item (2.3) strategies, then rising for the log Pop*Ent (4.0) and Classique (7.0) strategies. Figure 5, which plots the cumulative percentages of users ending their signup on the *n*th page, shows that these means hide long tails, especially in the case of the log Pop*Ent strategy. This figure shows that the Popularity and Item-Item strategies are by far the most effective strategies with over 90% of users being able to sign up in five pages or less. The other two strategies fare much worse, with a number of users requiring more than five pages. Since we only consider users who completed the signup process, all four strategies eventually reach 100 percent; we truncated the graph at five pages because all of the strategies except for Popularity had outliers that viewed over 200 movies before they found ten movies to rate.

Table 2. Evaluation of strategies over both experiments on user effort and accuracy metrics.

Strategy	User Effort	Accuracy
Random/Classique	★	★★
Popularity	★★★★★	★★★★
(log) Pop*Ent	★★★	★★★★★
Item-Item	★★★★★	★★

DISCUSSION

We consider both the on-line and the off-line results in this section. In evaluating the techniques we focus on two dimensions of the user experience: user effort and recommendation accuracy. The best strategy for eliciting information from users depends on the dimension along which you wish to optimize. Some algorithms do well at minimizing user effort, at the cost of accuracy, while other algorithms provide very good accuracy, at the cost of additional effort from the user. The best algorithms perform well by both measures. Popularity, Pop*Ent, and Item-Item personalized strategies all give reasonable performance on both metrics and provide the system designer an easy way to choose trade-offs. Popularity provides a good balance between effort and accuracy. Pop*Ent trades effort for more accuracy; Item-Item personalized trades accuracy for less effort. Item-Item does sacrifice more in accuracy than the other methods.

The results of the off-line and on-line experiments support each other. Random, Entropy, and Classique performed poorly at helping users rate movies; Popularity performed well in both cases, and Item-Item successfully found movies for users to rate in both experiments. Table 2 compares the overall performance of our algorithms on our two primary dimensions of minimizing user effort and making good predictions. Choosing an intelligent strategy for presenting items to rate can dramatically improve usability. The Classique strategy required over three times the effort of the best strategies, Popularity and Item-Item personalized, and based on the off-line results for the Random strategy, probably delivers worse recommendations.

These results should generalize to any set of ratings where the popularity of an item decays exponentially and the relative entropy of most items is in a fairly narrow range. We expect that most real-world ratings data sets have these properties.

An application's requirements also matter. An e-commerce recommender might have to start making recommendations with no data at all about the current user [15]. In this case, we suggest recommending the most popular items rather than the highest-rated ones, and then using Item-Item strategies to personalize the recommendations as quickly as possible.

We also have anecdotal evidence about another dimension of user experience: users in our research group much preferred using techniques that allowed them to rate several movies per page, especially compared to the techniques that required them to go many pages between ratings. The reaction was so strong that we modified our experimental design to include only one technique with very low ratings density (Classique). Exploiting intelligence about the user may lead to improved satisfaction.

However, using methods that exploit intelligence about the user may induce biases in ratings distributions. The Popularity strategy might exacerbate the bias we described earlier, where more popular movies get more chances to be recommended and rated. Over time, the system might become a "winner takes all" recommender that only recommends generically popular items.

The Item-Item strategy might create the opposite problem. Each user would see a set of items to rate that predicted to be of interest to him. Over time, users may become clustered in small groups with very little overlap, leading to the balkanization of the user population.

Both of these potential long-term dangers can be combated in practice by including some randomness in the set of

items suggested for rating. Too much randomness leads to excessive user effort, but a small amount of randomness may help to extend the space over which the recommender understands the user's interests and ensure that all items are occasionally presented to users.

CONCLUSION AND FUTURE WORK

We conclude that the proper strategy for eliciting information from users depends on the dimension of user experience along which you are trying to optimize. In general, strategies that make good guesses about what items a user is likely to be able to rate do well with both reducing user effort and producing acceptable recommendations. We believe these results will hold for many similar recommender systems.

We studied the techniques we considered in three ways: through analysis, through simulation studies on previously collected user data, and through live user trials. We found the three methods complementary. The analysis helped suggest techniques that might be useful. The simulation studies enabled us to consider a very large number of users quickly, and to explore techniques that would have been frustrating for live users. The live study helped avoid the problems of data bias in our simulations, and increased our confidence in the applicability of the results to real systems. We believe that all three techniques are important in successfully developing intelligent user interfaces.

In this paper we focused on minimizing user effort while still being able to make accurate predictions. It would be useful to perform a more thorough investigation of the system's needs for diverse ratings across all items, and how to balance these needs with the user experience. More direct measurements of user satisfaction, such as longer-term statistics on usage and surveying users, would complement our attempts to minimize user effort.

ACKNOWLEDGEMENTS

We thank members of the GroupLens Research Group, past and present, for their contributions to this research. We would also like to thank the members of the MovieLens system for their support in our research efforts, and our anonymous reviewers for their comments on this paper.

This work was supported by grants from the National Science Foundation (IIS 96-13960, IIS 97-34442, and IIS 99-78717) and by Net Perceptions, Inc.

REFERENCES

1. Avery, C., Resnick, P., and Zeckhauser, R. The Market for Evaluations. *American Economic Review*, 89(3): 564-584.
2. Balabanovic, M., and Shoham, Y. 1997. Fab: Content-based, collaborative recommendation. *Communications of the ACM*, 40(3), 66-72.
3. Ben-Bassat, M. Myopic Policies in Sequential Classification. *IEEE Transactions on Computers*, 27(2), 170-174.
4. Glover, E. J., and Birmingham, W. P. Using Decision Theory to Order Documents. *Proceedings of ACM Digital Libraries 1998*, 285-286.
5. Goldberg, K., Roeder, T., Gupta, D., and Perkins, C. Eigentaste: A Constant Time Collaborative Filtering Algorithm. *Information Retrieval Journal*, 4(2), 133-151.
6. Good, N., Schafer, J.B., Konstan, J., Borchers, A., Sarwar, B., Herlocker, J., and Riedl, J. Combining Collaborative Filtering with Personal Agents for Better Recommendations. *Proceedings of AAAI-99,* 439-446.
7. Greiff, W. R., and Ponte, J. The Maximum Entropy Approach and Probabilistic IR Methods. *ACM Transactions on Information Systems* 18(3) 246-287.
8. Ha, V., and Haddawy P. Towards Case-Based Preference Elicitation: Similarity Measures on Preference Structures. *Proceedings UAI 1998*, 193-201.
9. Horvitz, E., Heckerman D., Ng, K, Nathwani, B. Towards Normative Expert Systems: Part I, Pathfinder Project. *Methods of Information in Medicine*, 31, 90-105.
10. Kantor, P. B., and Lee, J. J. The Maximum Entropy Principle In Informational Retrieval. *Proceedings of ACM SIGIR 1986*, 269-274.
11. Karypis, G. Evaluation of Item-Based Top-*N* Recommendation Algorithms. *Proceedings of CIKM 2001*.
12. Kohrs, A., and Merialdo, B. Improving Collaborative Filtering for New Users by Smart Object Selection, *Proceedings of International Conference on Media Features (ICMF) 2001* (oral presentation).
13. Nguyen, H., and Haddawy, P. The Decision-Theoretic Video Advisor. *Proceedings of AAAI Workshop on Recommender Systems*, 76-80, 1998.
14. Pennock, D., and Horvitz, E. Collaborative Filtering by Personality Diagnosis: A Hybrid Memory- and Model-based Approach. *Proceedings of UAI 2000*, 473-480.
15. Schafer, J.B., Konstan, J., and Riedl, J., Electronic Commerce Recommender Applications. *Journal of Data Mining and Knowledge Discovery*, January 2001.
16. Vetschera, R. Entropy and the Value of Information, *Central European Journal of Operations Research* 8, 2000 S. 195-208.
17. Wasfi. A. M. A. Collecting User Access Patterns for Building User Profiles and Collaborative Filtering. *Proceedings of IUI 1999*, 57-64.