Evaluating Collaborative Filtering Recommender Systems

JONATHAN L. HERLOCKER
School of Electrical Engineering & Computer Science, Oregon State University
and
JOSEPH A. KONSTAN, LOREN G. TERVEEN, AND JOHN T. RIEDL
GroupLens Research Group, University of Minnesota

Recommender systems have been evaluated in many, often incomparable, ways. In this paper we review the key decisions in evaluating collaborative filtering recommender systems: the user tasks being evaluated, the types of analysis and datasets being used, the ways in which prediction quality is measured, the evaluation of prediction attributes other than quality, and the user-based evaluation of the system as a whole. In addition to reviewing the evaluation strategies used by prior researchers, we present empirical results from the analysis of various accuracy metrics on one content domain where all the tested metrics collapsed roughly into three equivalence classes. Metrics within each equivalency class were strongly correlated, while metrics from different equivalency classes were uncorrelated.

Categories and Subject Descriptors: H.3.4 [**Information Storage and Retrieval**]: Systems and Software – *Performance Evaluation (efficiency and effectiveness)*
General Terms: Experimentation, Measurement, Performance
Additional Key Words and Phrases: Collaborative filtering, recommender systems, metrics, evaluation

1. INTRODUCTION

Recommender systems use the opinions of a community of users to help individuals in that community more effectively identify content of interest from a potentially overwhelming set of choices [Resnick and Varian 1997]. One of the most successful technologies for recommender systems, called collaborative filtering, has been developed and improved over the past decade to the point where a wide variety of algorithms exist for generating recommendations. Each algorithmic approach has adherents who claim it to be superior for some purpose. Clearly identifying the best algorithm for a given purpose has proven challenging, in part because researchers disagree on which attributes should be measured, and on which metrics should be used for each attribute. Researchers who survey the literature will find over a dozen quantitative metrics and additional qualitative evaluation techniques.

This research was supported by the National Science Foundation under grants DGE 95-54517, IIS 96-13960, IIS 97-34442, IIS 99-78717, IIS 01-02229, IIS 01-33994, and by Net Perceptions, Inc.

Authors' addresses: J. L. Herlocker, School of Electrical Engineering and Computer Science, Oregon State University, 102 Dearborn Hall, Corvallis, OR, 97331. email: herlock@cs.orst.edu; J. A. Konstan, L. G. Terveen, and J. T. Riedl, Department of Computer Science and Engineering, University of Minnesota, 4-192 EE/CS Building, 200 Union St SE, Minneapolis, MN 55455. email: {konstan, terveen, riedl}@cs.umn.edu.

Evaluating recommender systems and their algorithms is inherently difficult for several reasons. First, different algorithms may be better or worse on different data sets. Many collaborative filtering algorithms have been designed specifically for data sets where there are many more users than items (e.g., the MovieLens data set has 65,000 users and 5,000 movies). Such algorithms may be entirely inappropriate in a domain where there are many more items than users (e.g., a research paper recommender with thousands of users but tens or hundreds of thousands of articles to recommend). Similar differences exist for ratings density, ratings scale, and other properties of data sets.

The second reason that evaluation is difficult is that the goals for which an evaluation is performed may differ. Much early evaluation work focused specifically on the "accuracy" of collaborative filtering algorithms in "predicting" withheld ratings. Even early researchers recognized, however, that when recommenders are used to support decisions, it can be more valuable to measure how often the system leads its users to wrong choices. Shardanand and Maes measured "reversals" – large errors between the predicted and actual rating [1995]; we have used the signal-processing measure of the Receiver Operating Characteristic curve [Swets 1963] to measure a recommender's potential as a filter [Konstan et al. 1997]. Other work has speculated that there are properties different from accuracy that have a larger effect on user satisfaction and performance. A range of research and systems have looked at measures including the degree to which the recommendations cover the entire set of items [Mobasher et al. 2001], the degree to which recommendations made are non-obvious [McNee et al. 2002], and the ability of recommenders to explain their recommendations to users [Sinha and Swearingen 2002]. A few researchers have argued that these issues are all details, and that the bottom-line measure of recommender system success should be user satisfaction. Commercial systems measure user satisfaction by the number of products purchased (and not returned!), while non-commercial systems may just ask users how satisfied they are.

Finally, there is a significant challenge in deciding what combination of measures to use in comparative evaluation. We have noticed a trend recently -- many researchers find that their newest algorithms yield a mean absolute error of 0.73 (on a five-point rating scale) on movie rating datasets. Though the new algorithms often appear to do better than the older algorithms they are compared to, we find that when each algorithm is tuned to its optimum, they all produce similar measures of quality. We – and others – have speculated that we may be reaching some "magic barrier" where natural variability may prevent us from getting much more accurate. In support of this, Hill et al. [1995] have shown that users provide inconsistent ratings when asked to rate the same movie at different times. They suggest that an algorithm cannot be more accurate than the variance in a user’s ratings for the same item.

Even when accuracy differences are measurable, they are usually tiny. On a five-point rating scale, are users sensitive to a change in mean absolute error of 0.01? These observations suggest that algorithmic improvements in collaborative filtering systems may come from different directions than just continued improvements in mean absolute error. Perhaps the best algorithms should be measured according to how well they can communicate their reasoning to users, or with how little data they can yield accurate recommendations. If this is true, new metrics will be needed to evaluate these new algorithms.

This paper presents six specific contributions towards evaluation of recommender systems.

1. We introduce a set of recommender tasks that categorize the user goals for a particular recommender system.
2. We discuss the selection of appropriate datasets for evaluation. We explore when evaluation can be completed off-line using existing datasets and when it requires on-line experimentation. We briefly discuss synthetic data sets and more extensively review the properties of datasets that should be considered in selecting them for evaluation.
3. We survey evaluation metrics that have been used to evaluation recommender systems in the past, conceptually analyzing their strengths and weaknesses.
4. We report on experimental results comparing the outcomes of a set of different accuracy evaluation metrics on one data set. We show that the metrics collapse roughly into three equivalence classes.
5. By evaluating a wide set of metrics on a dataset, we show that for some datasets, while many different metrics are strongly correlated, there are classes of metrics that are uncorrelated.
6. We review a wide range of non-accuracy metrics, including measures of the degree to which recommendations cover the set of items, the novelty and serendipity of recommendations, and user satisfaction and behavior in the recommender system.

Throughout our discussion, we separate out our review of what has been done before in the literature from the introduction of new tasks and methods.

We expect that the primary audience of this article will be collaborative filtering researchers who are looking to evaluate new algorithms against previous research and collaborative filtering practitioners who are evaluating algorithms before deploying them in recommender systems.

There are certain aspects of recommender systems that we have specifically left out of the scope of this paper. In particular, we have decided to avoid the large area of marketing-inspired evaluation. There is extensive work on evaluating marketing campaigns based on such measures as offer acceptance and sales lift [Rogers 2001]. While recommenders are widely used in this area, we cannot add much to existing coverage of this topic. We also do not address general usability evaluation of the interfaces. That topic is well covered in the research and practitioner literature (e.g., [Helander 1988, Nielsen 1994]) We have chosen not to discuss computation performance of recommender algorithms. Such performance is certainly important, and in the future we expect there to be work on the quality of time-limited and memory-limited recommendations. This area is just emerging, however (see for example Miller et al.'s recent work on recommendation on handheld devices [Miller et al. 2003]), and there is not yet enough research to survey and synthesize. Finally, we do not address the emerging question of the robustness and transparency of recommender algorithms. We recognize that recommender system robustness to manipulation by attacks (and transparency that discloses manipulation by system operators) is important, but substantially more work needs to occur in this area before there will be accepted metrics for evaluating such robustness and transparency.

The remainder of the article is arranged as follows:

- Section 2 - We identify the key user tasks from which evaluation methods have been determined and suggest new tasks that have not been evaluated extensively.
- Section 3- A discussion regarding the factors that can affect selection of a data set on which to perform evaluation.
- Section 4 – An investigation of metrics that have been used in evaluating the *accuracy* of collaborative filtering predictions and recommendations. Accuracy has been by far the most commonly published evaluation method for collaborative filtering systems. This section also includes the results from an empirical study of the correlations between metrics.
- Section 5 – A discussion of metrics that evaluate dimensions other than accuracy. In addition to covering the dimensions and methods that have been used in the literature, we introduce new dimensions on which we believe evaluation should be done.
- Section 6 – Final conclusions, including a list of areas were we feel future work is particularly warranted.

Sections 2-5 are ordered to discuss the steps of evaluation in roughly the order that we would expect an evaluator to take. Thus Section 2 describes the selection of appropriate user tasks, Section 3 discusses the selection of a dataset, and sections 4 and 5 discuss the alternative metrics that may be applied to the dataset chosen. We begin with the discussion of user tasks – the user task sets the entire context for evaluation.

2. USER TASKS FOR RECOMMENDER SYSTEMS

To properly evaluate a recommender system, it is important to understand the goals and tasks for which it is being used. In this paper, we focus on end-user goals and tasks (as opposed to goals of marketers and other system stakeholders). We derive these tasks from the research literature and from deployed systems. For each task, we discuss its implications for evaluation. While the tasks we've identified are important ones, based on our experience in recommender systems research and from our review of published research, we recognize that the list is necessarily incomplete. As researchers and developers move into new recommendation domains, we expect they will find it useful to supplement this list and/or modify these tasks with domain-specific ones. Our goal is primarily to identify domain-independent task descriptions to help distinguish among different evaluation measures.

We have identified two user tasks that have been discussed at length within the collaborative filtering literature:

> **Annotation in Context.** The original recommendation scenario was filtering through structured discussion postings to decide which ones were worth reading. Tapestry [Goldberg et al. 1992] and GroupLens [Resnick et al. 1994] both applied this to already structured message databases. This task required retaining the order and context of messages, and accordingly used predictions to annotate messages in context. In some cases the "worst" messages were filtered out. This same scenario, which uses a recommender in an existing context, has

also been used by web recommenders that overlay prediction information on top of existing links [Wexelblat and Maes 1999]. Users use the displayed predictions to decide which messages to read (or which links to follow), and therefore the most important factor to evaluate is how successfully the predictions help users distinguish between desired and undesired content. A major factor is the whether the recommender can generate predictions for the items that the user is viewing.

Find Good Items. Soon after Tapestry and GroupLens, several systems were developed with a more direct focus on actual recommendation. Ringo [Shardanand and Maes 1995] and the Bellcore Video Recommender [Hill et al. 1995] both provided interfaces that would suggest specific items to their users, providing users with a ranked list of the recommended items, along with predictions for how much the users would like them. This is the core recommendation task and it recurs in a wide variety of research and commercial systems. In many commercial systems, the "best bet" recommendations are shown, but the predicted rating values are not.

While these two tasks can be identified quite generally across many different domains, there are likely to be many specializations of the above tasks within each domain. We introduce some of the characteristics of domains that influence those specializations in Section 3.3.

While the *Annotation in Context* and the *Find Good Items* are overwhelmingly the most commonly evaluated tasks in the literature, there are other important generic tasks that are not well described in the research literature. Below we describe several other user tasks which we have encountered in our interviews with users and our discussions with recommender system designers. We mention these tasks because we believe that they should be evaluated, but because they have not been addressed in the recommender systems literature, we do not discuss them further.

Find All Good Items. Most recommender tasks focus on finding some good items. This is not surprising; the problem that led to recommender systems was one of overload, and most users seem willing to live with overlooking some good items in order to screen out many bad ones. Our discussions with firms in the legal databases industry, however, led in the opposite direction. Lawyers searching for precedents feel it is very important not to overlook a single possible case. Indeed, they are willing to invest large amounts of time (and their client's money) searching for that case. To use recommenders in their practice, they first need to be assured that the false negative rate can be made sufficiently low. As with annotation in context, coverage becomes particularly important in this task.

Recommend Sequence. We first noticed this task when using the personalized radio web site Launch (launch.yahoo.com) which streams music based on a variety of recommender algorithms. Launch has several interesting factors, including the desirability of recommending "already rated" items, though not too often. What intrigued us, though, is the challenge of moving from recommending one song at a time to recommending a sequence that is pleasing as a whole. This same task can apply to recommending research papers to learn about a field (read this introduction, then that survey, …). While data mining research has explored product purchase timing and sequences, we are not aware of any recommender applications or research that directly address this task.

Just Browsing. Recommenders are usually evaluated based on how well they help the user make a consumption decision. In talking with users of our MovieLens system, of Amazon.com, and of several other sites, we discovered that many of them use the site even when they have no purchase imminent. They find it pleasant to browse. Whether one models this activity as learning or simply as entertainment, it seems that a substantial use of recommenders is simply using them without an ulterior motive. For those cases, the accuracy of algorithms may be less important than the interface, the ease of use, and the level and nature of information provided.

Find Credible Recommender. This is another task gleaned from discussions with users. It is not surprising that users do not automatically trust a recommender. Many of them "play around" for a while to see if the recommender matches their tastes well. We've heard many complaints from users who are looking up their favorite (or least favorite) movies on MovieLens—they don't do this to learn about the movie, but to check up on us. Some users even go further. Especially on commercial sites, they try changing their profiles to see how the recommended items change. They explore the recommendations to try to find any hints of bias. A recommender optimized to produce “useful” recommendations (for example recommendations for items that the user does not already know about) may fail to appear trustworthy because it does not recommend movies the user is sure to enjoy but probably already knows about. We are not aware of any research on how to make a recommender appear credible, though there is more general research on making websites evoke trust [Bailey et al. 2001].

Most evaluations of recommender systems focus on the recommendations; however if users don’t rate items, then collaborative filtering recommender systems can’t provide recommendations. Thus evaluating if and why users would contribute ratings may be important to communicate that a recommender system is likely to be successful. We will briefly introduce several different rating tasks.

Improve Profile is the rating task that most recommender systems have assumed. Users contribute ratings because they believe that they are improving their profile and thus improving the quality of the recommendations that they will receive.

Express Self. Some users may not care about the recommendations – what is important to them is that they be allowed to contribute their ratings. Many users simply want a forum for expressing their opinions. We conducted interviews with "power users" of MovieLens that had rated over 1000 movies (some over 2000 movies). What we learned was that these users were not rating to improve their recommendations. They were rating because it felt good. We particularly see this effect on sites like Amazon.com, where users can post reviews (ratings) of items sold by Amazon. For users with this task, issues may include the level of anonymity (which can be good or bad, depending on the user), the feeling of contribution, and the ease of making the contribution. While recommender algorithms themselves may not evoke self expression, encouraging self expression may provide more data which can improve the quality of recommendations.

Help Others. Some users are happy to contribute ratings in recommender systems because they believe that the community benefits from their contribution. In many cases, they are also entering ratings in order to express themselves (see previous task). However, the two do not always go together.

Influence Others. An unfortunate fact that we and other implementers of web-based recommender systems have encountered is that there are users of recommender systems whose goal is to explicitly influence others into viewing or purchasing particular items. For example, advocates of particular movie genres (or movie studios) will frequently rate movies high on the MovieLens web site right before the movie is released to try and push others to go and see the movie. This task is particularly interesting, because we may want to evaluate how well the system prevents this task.

While we have briefly mentioned tasks involved in contributing ratings, we will not discuss them in depth in this paper, and rather focus on the tasks related to recommendation.

We must once again say that the list of tasks in this section is not comprehensive. Rather, we have used our experience in the field to filter out the task categories that a) have been most significant in the previously published work, and b) that we feel are significant, but have not been considered sufficiently.

In the field of Human-Computer Interaction, it has been strongly argued that the evaluation process should begin with an understanding of the user tasks that the system will serve. When we evaluate recommender systems from the perspective of benefit to the user, we should also start by identifying the most important task for which the recommender will be used. In this section, we have provided descriptions of the most significant tasks that have been identified. Evaluators should consider carefully which of the tasks described may be appropriate for their environment.

Once the proper tasks have been identified, the evaluator must select a dataset to which evaluation methods can be applied, a process that will most likely be constrained by the user tasks identified.

3. SELECTING DATA SETS FOR EVALUATION

Several key decisions regarding data sets underlie successful evaluation of a recommender system algorithm. Can the evaluation be carried out offline on an existing data set or does it require live user tests? If a data set is not currently available, can evaluation be performed on simulated data? What properties should the dataset have in order to best model the tasks for which the recommender is being evaluated? A few examples help clarify these decisions:

- When designing a recommender algorithm designed to recommend word processing commands (e.g., [Linton et al. 1998]), one can expect users to have experienced 5-10% (or more) of the candidates. Accordingly, it would be unwise to select recommender algorithms based on evaluation results from movie or e-commerce datasets where ratings sparsity is much worse.
- When evaluating a recommender algorithm in the context of the *Find Good Items* task where novel items are desired, it may be inappropriate to use only offline evaluation. Since the recommender algorithm is generating recommendations for items that the user does not already know about, it is probable that the data set will not provide enough information to evaluate the quality of the items being recommended. If an item

was truly unknown to the user, then it is probable that there is no rating for that user in the database. If we perform a live user evaluation, ratings can be gained on the spot for each item recommended.

- When evaluating a recommender in a new domain where there is significant research on the structure of user preferences, but no data sets, it may be appropriate to first evaluate algorithms against synthetic data sets to identify the promising ones for further study.

We will examine in the following subsections each of the decisions that we posed in the first paragraph of this section, and then discuss the past and current trends in research with respect to collaborative filtering data sets.

3.1 Live User Experiments vs. Offline Analyses

Evaluations can be completed using off-line analysis, a variety of live user experimental methods, or a combination of the two. Much of the work in algorithm evaluation has focused on off-line analysis of predictive accuracy. In such an evaluation, the algorithm is used to predict certain withheld values from a dataset, and the results are analyzed using one or more of the metrics discussed in the following section. Such evaluations have the advantage that it is quick and economical to conduct large evaluations, often on several different datasets or algorithms at once. Once a dataset is available, conducting such an experiment simply requires running the algorithm on the appropriate subset of that data. When the dataset includes timestamps, it is even possible to "replay" a series of ratings and recommendations off line. Each time a rating was made, the researcher first computes the prediction for that item based on all prior data; then, after evaluating the accuracy of that prediction, the actual rating is entered so the next item can be evaluated.

Offline analyses have two important weaknesses. First, the natural sparsity of ratings data sets limits the set of items that can be evaluated. We cannot evaluate the appropriateness of a recommended item for a user if we do not have a rating from that user for that item in the dataset. Second, they are limited to objective evaluation of prediction results. No offline analysis can determine whether users will prefer a particular system, either because of its predictions or because of other less objective criteria such as the aesthetics of the user interface.

An alternative approach is to conduct a live user experiment. Such experiments may be controlled (e.g., with random assignment of subjects to different conditions), or they may be field studies where a particular system is made available to a community of users that is then observed to ascertain the effects of the system. As we discuss later in Section 5.5, live user experiments can evaluate user performance, satisfaction, participation, and other measures.

3.2 Synthesized vs. Natural Data Sets

Another choice that researchers face is whether to use an existing dataset that may imperfectly match the properties of the target domain and task, or to instead synthesize a dataset specifically to match those properties. In our own early work designing recommender algorithms for Usenet News [Konstan et al. 1997, Miller et al. 1997], we experimented with a variety of synthesized datasets. We modeled news articles as having a fixed number of “properties” and users as having preferences for those properties. Our data set generator could cluster users together, spread them evenly, or present other distributions. While these simulated data sets gave us an easy way to test algorithms for obvious flaws, they in no way accurately modeled the nature of real users and real data. In their research on horting as an approach for collaborative filtering, Aggarwal et al. used a similar technique [1999], noting however that such synthetic data is "unfair to other algorithms" because it fits their approach too well, and that this is a placeholder until they can deploy their trial.

Synthesized data sets may be required in some limited cases, but only as early steps while gathering data sets or constructing complete systems. Drawing comparative conclusions from synthetic datasets is risky, because the data may fit one of the algorithms better than the others.

On the other hand, there is new opportunity now to explore more advanced techniques for modeling user interest and generating synthetic data from those models, now that there exists data on which to evaluate the synthetically generated data and tune the models. Such research could also lead to the development of more accurate recommender algorithms with clearly defined theoretical properties.

3.3 Properties of Data Sets

The final question we address in this section on data sets is "what properties should the dataset have in order to best model the tasks for which the recommender is being evaluated?" We find it useful to divide data set properties into three categories: *Domain* features reflect the nature of the content being recommended, rather than any particular system. *Inherent* features reflect the nature of the specific recommender system from which data was drawn (and possibly from its data collection practices). *Sample* features reflect distribution properties of the data, and often can be manipulated by selecting the appropriate subset of a larger data set. We discuss each of these three categories here, identifying specific features within each category.

Domain features of interest include

(a) the content topic being recommended/rated and the associated context in which rating/recommendation takes place.

(b) the user tasks supported by the recommender

(c) the novelty need and the quality need

(d) the cost/benefit ratio of false/true positives/negatives

(e) the granularity of true user preferences.

Most commonly, recommender systems have been built for entertainment content domains (movies, music, etc.), though some testbeds exist for filtering document collections (Usenet news, for example). Within a particular topic, there may be many contexts. Movie recommenders may operate on the web, or may operate entirely within a video rental store or as part of a set-top box or digital video recorder.

In our experience, one of the most important generic domain features to consider lies in the tradeoff between desire for novelty and desire for high quality. In certain domains, the user goal is dominated by finding recommendations for things she doesn't already know about. McNee et al. evaluated recommenders for research papers [2002] and found that users were generally happy with a set of recommendations if there was a single item in the set that appeared to be useful and that the user wasn't already familiar with. In some ways this matches the conventional wisdom about supermarket recommenders – it would be almost always correct, but useless, to recommend bananas, bread, milk, and eggs. The recommendations might be correct, but they don't change the shopper's behavior. Opposite the desire for novelty is the desire for high quality. Intuitively, this end of the tradeoff reflects the user's desire to rely heavily upon the recommendation for a consumption decision, rather than simply as one decision-support factor among many. At the extreme, the availability of high-confidence recommendations could enable automatic purchase decisions such as personalized book- or music-of-the-month clubs. Evaluations of recommenders for this task must evaluate the success of high-confidence recommendations, and perhaps consider the opportunity costs of excessively low confidence.

Another important domain feature is the cost/benefit ratio faced by users in the domain from which items are being recommended. In the video recommender domain, the cost of false positives is low ($3 and two-three hours of your evening), the cost of false negatives is almost zero, and the benefit of recommendations is huge (an enormous quantity of movies have been released over the years, and browsing in the video store can be quite stressful – particularly for families). This analysis explains to a large extent why video recommenders have been so successful. Other domains with similar domain features, such as books of fiction, are likely to have datasets similar to the video domain and results demonstrated on video data may likely translate somewhat well to those other domains (although books of fiction are likely to have different sample features – see below). See [Konstan et al. 1997] for a slightly more detailed discussion of cost/benefit tradeoff analysis in collaborative filtering recommender systems.

Another subtle but important domain feature is the granularity of true user preferences. How many different levels of true user preference exist? With binary preferences, users only care to distinguish between good and bad items ("I don't necessarily need the best movie, only a movie I will enjoy"). In such a case, distinguishing among good items is not important, nor is distinguishing among bad items. Note that the granularity of user preference may be different than the range and granularity of the ratings (which is an inherent feature of data sets). Users may rank movies on a 1-10 scale, but then only really care if recommendations are good (I had a good time watching the movie) or bad (I was bored out of my mind!)

Overall, it would probably be a mistake to evaluate an algorithm on data with significantly different domain features. In particular, it is very important that the tasks your algorithm is designed to support are similar to the tasks supported by the system from which the data was collected. If the user tasks are mismatched, then there are likely to be many other feature mismatches. For example, the MovieLens system supported primarily the *Find Good Items* user task. As the result, the user was always shown the "best bets" and thus there are many more ratings for good items than bad items (the user had to explicitly request to rate a poor item in most cases). So MovieLens data is less likely to have many ratings for less popular items. It would probably be inappropriate to use this data to evaluate a new algorithm whose goal was to support *Annotation In Context*. Of course, if an algorithm is being proposed for general use, it is best to select data sets that span a variety of topics and contexts.

Inherent features include several features about ratings:

(a) whether ratings are explicit, implicit, or both;

(b) the scale on which items are rated;

(c) the dimensions of rating; and

(d) the presence or absence of a timestamp on ratings.

Explicit ratings are entered by a user directly (i.e. "Please rate this on a scale of 1-5"), while implicit ratings are inferred from other user behavior. For example, a music recommender may use implicit data such as play lists or music listened to, or it may use explicit scores for songs or artists, or a combination of both. The ratings scale is the range and granularity of ratings. The simplest scale is unary—liked items are marked, all others are unknown. Unary is common in commerce applications, where all that is known is whether the user purchased an item or not. We call the data unary instead of binary because a lack of purchase of item X does not necessarily mean that the user would not like X. Binary items include a separate designation for disliked. Systems that operate on explicit ratings often support 5-point, 7-point, or 100-point scales. While most recommenders have had only a single rating dimension (described by Miller et al. [1997] as "what predictions should we have displayed for this item?"), both research and commercial systems are exploring systems where users can enter several ratings for a single item. Zagat's restaurant guides, for example, traditionally use food, service, and décor as three independent dimensions. Movie recommenders may separate story, acting, and special effects. Data sets with multiple dimensions are still difficult to find, but we expect several to become available in the future. Timestamps are a property of the data collection, and are particularly important in areas where user tastes are expected to change or where user reactions to items likely depend on their history of interaction with other items.

Other inherent features concern the data collection practices:

(e) whether the recommendations displayed to the user were recorded; and

(f) the availability of user demographic information or item content information.

Unfortunately, few datasets recorded the recommendations that were displayed, making it difficult to retrospectively isolate, for example, ratings that could not have been biased by previously displayed predictions. Some logs may keep time stamped queries, which could be used to reconstruct recommendations if the algorithm is known and fully deterministic. The availability of demographic data varies with the specific system, and with the specific data collected. The EachMovie and MovieLens datasets both collected limited demographics. Researchers speculate, however, that a large percentage of the demographic answers may be false (based on user suspicion of "marketing questions"). We would expect greater reliability for demographic data that users believe actually serves a constructive purpose in the recommender (either for recommendation or for related purposes). A film recommender that uses zip code to narrow the theater search, such as Miller et al.'s MovieLens Unplugged [Miller et al. 2003], seems more likely to provide meaningful data.

Finally, we consider:

(g) the biases involved in data collection.

Most data sets have biases based on the mechanism by which users have the opportunity to rate items. For example, Jester [Goldberg et al. 2001] asked all users to rate the same initial jokes, creating a set of dense ratings for those jokes which would not otherwise occur. MovieLens has experimented with different methods to select items to have the first-time user rate before using the recommender system [Rashid et al. 2002], and in the process demonstrated that each method leads to a different bias in initial ratings.

Sample features include many of the statistical properties that are commonly considered in evaluating a data set:

(a) the density of the ratings set overall, sometimes measured as the average percentage of items that have been rated per user; since many datasets have uneven popularity distributions, density may be artificially manipulated by including or excluding items;

(b) the number or density of ratings from the users for whom recommendations are being made, which represents the experience of the user in the system at the time of recommendation; ratings from users with significant experience can be withheld to simulate the condition when they were new users; and

(c) the general size and distribution properties of the data set – some data sets have more items than users, though most data sets have many more users than items.

Each of these sample features can have substantial effect on the success of different algorithms, and can reflect specific policies of the recommender. Density (both individual and overall) reflects both the overall size of the recommender's item space and the degree to which users have explored it. One policy decision that significantly affects density is the level of rating required to participate in the community. Systems that either require an extensive level of start-up rating or require recurring ratings to maintain membership or status levels will generally have greater density than low-commitment recommenders in the same domain. Density also interacts with the type of rating – implicit ratings are likely to lead to greater density, since less effort is needed by the user. Finally, system that allow automated software "agents" to participate may have a significantly higher density than other systems, even if the underlying item space is similar (see for example [Good et al. 1999]). Because software agents are not limited in attention, they can rate much more extensively than humans.

Two particular distribution properties are known to be highly important. The relationship between the numbers of users and numbers of items can determine whether it is easier to build correlations among users or among items—this choice can lead to different relative performance among algorithms. The ratings distribution of items and users also may affect both algorithm and evaluation metric choice. Systems where there is an exponential popularity curve (some items have exponentially more ratings than others) may be able to find agreement among people or items in the dense sub-region and use that agreement to recommend in the sparse space. (Jester, mentioned above, does this directly by creating a highly dense region of jokes rated by all users.) Systems with a more even ratings distribution may be more challenged to cope with sparsity unless they incorporate dimensionality reduction techniques.

To complete the discussion of domain features, inherent features, and sample features, it is important to note that there are significant interactions between these categories of features. For example, the type of task supported by a recommender system (a domain feature) will significantly affect the distribution of ratings collected (a sample feature). However, each of these features represents a dimension which may be useful in explaining differences in evaluation results.

Evaluation of a recommender algorithm on a data set with features that conflict with the end goal of the recommender algorithm could still be useful. By explicitly identifying the features that conflict, we can reason about whether those conflicts will unreasonably bias the evaluation results.

3.4 Past and Current Trends in Datasets

The most widely used common dataset was the EachMovie Dataset (http://research.compaq.com/SRC/eachmovie/). This extensive dataset has over 2.8 million ratings from over 70,000 users, and it includes information such as timestamps and basic demographic data for some of the users. In addition to seeding our MovieLens system (http://www.movielens.org), the EachMovie Dataset was used in dozens of machine learning and algorithmic research projects to study new and potentially better ways to predict user ratings. Examples include Canny's factor analysis algorithm [2002], Domingos and Richardson's algorithm for computing network value [2003], and Pennock et al's work on recommending through personality diagnosis algorithms [2000].

Extracts (100,000 ratings and 1 million ratings) of the MovieLens dataset have also been released for research use; these extracts have been used by several researchers, including Schein et al. [2001] in their investigation of cold-start recommendations, Sarwar et al. [2001] in their evaluation of item-based algorithms, Reddy et al. [2002] in their community extraction research, and Mui et al. [2001] in their work on "collaborative sanctioning."

More recently, several researchers have been using the Jester dataset, which was collected from the Jester joke recommendation web site [Goldberg et al. 2001]. Statistically, the Jester dataset has different characteristics than the MovieLens and Eachmovie data. First of all, there is a set of training items (jokes) that are rated by every single user, providing complete data on that subset of items. Second, in the Jester user interface, the user clicks on a unlabeled scale bar to rate a joke, so the ratings are much less discrete and may suffer from different kinds of biases since it is hard for the user to intentionally create a ranking among their rated items.

The majority of publications related to collaborative filtering recommender algorithms have used one of the three data sets described above. A few other data sets have been used, but most of them are not publicly available for verification. The lack of variety in publicly available collaborative filtering data sets (particularly with significant numbers of ratings) remains one of the most significant challenges in the field. Most researchers do not have the resources to build production-quality systems that are capable of collecting enough data to validate research hypotheses, and thus are often forced to constrain their research to hypotheses that can be explored using the few existing datasets.

With the maturation of collaborative filtering recommender technology, more live systems have been built that incorporate recommender algorithms. As a result, we have recently seen an increased number of studies that have used live systems. Herlocker's explanation experiments [Herlocker et al. 2000] explored the use of 23 different graphical displays to “explain” why each recommendation was given. . Schafer's MetaLens [Schafer et al. 2002] was built to incorporate MovieLens and other systems into a new interface; his evaluation focused entirely on the interface and user experience. Other recent work has combined different evaluations. Our work on "value of information" [Rashid et al. 2002] leads users through different sign-up processes, and then evaluates both the quality of resulting predictions and the subjective user experience.

In the near future, we expect to see a lot more results from live experiments, as recommender algorithms make their way into more production systems. We also hope that new datasets will be released with data from new domains, causing new explosions in collaborative filtering recommender algorithm research similar to what happened with the release of the EachMovie data.

4. ACCURACY METRICS

Establishing the user tasks to be supported by a system, and selecting a data set on which performance enables empirical experimentation – scientifically repeatable evaluations of recommender system utility. A majority of the published empirical evaluations of recommender systems to date has focused on the evaluation of a recommender system’s **accuracy**. We assume that if a user could examine all items available, they could place those items in a ordering of preference. An accuracy metric empirically measures how close a recommender system’s predicted ranking of items for a user differs from the user’s true ranking of preference. Accuracy measures may also measure how well a system can predict an exact rating value for a specific item.

Researchers who want to quantitatively compare the accuracy of different recommender systems must first select one or more metrics. In selecting a metric, researchers face a range of questions. Will a given metric measure the effectiveness of a system with respect to the user tasks for which it was designed? Are results with the chosen metric comparable to other published research work in the field? Are the assumptions that a metric is based on true? Will a metric be sensitive enough to detect real differences that exist? How large a difference does there have to be in the value of a metric for a statistically significant difference to exist? Complete answers to these questions have not yet been substantially addressed in the published literature.

The challenge of selecting an appropriate metric is compounded by the large diversity of published metrics that have been used to quantitatively evaluate the accuracy of recommender systems. This lack of standardization is damaging to the progress of knowledge related to collaborative filtering recommender systems. With no standardized metrics within the field, researchers have continued to introduce new metrics when they evaluate their systems. With a large diversity of evaluation metrics in use, it becomes difficult to compare results from one publication to the results in another publication. As a result, it becomes hard to integrate these diverse publications into a coherent body of knowledge regarding the quality of recommender system algorithms.

To address these challenges, we examine in the advantages and disadvantages of past metrics with respect to the user tasks and data set features that have been introduced in Sections 2 and 3. We follow up the conceptual discussion of advantages and disadvantages with empirical results comparing the performance of different metrics when applied to results from one class of algorithm in one domain. The empirical results demonstrate that some conceptual differences among accuracy evaluation metrics can be more significant than others.

4.1 Evaluation of Previously Used Metrics

Recommender system accuracy has been evaluated in the research literature since 1994 [Resnick et al. 1994]. Many of the published evaluations of recommender systems used different metrics. We will examine some of the most popular metrics used in those publications, identifying the strengths and the weaknesses of the metrics. We broadly classify recommendation accuracy metrics into three classes: *predictive accuracy* metrics, *classification accuracy* metrics, and *rank accuracy* metrics.

4.1.1 Predictive Accuracy Metrics

Predictive accuracy metrics measure how close the recommender system's predicted ratings are to the true user ratings. Predictive accuracy metrics are particularly important for evaluating tasks in which the predicting rating will be displayed to the user such as *Annotation in Context*. For example, the MovieLens movie recommender [Dahlen et al. 1998] predicts the number of stars that a user will give each movie and displays that prediction to the user. Predictive accuracy metrics will evaluate how close MovieLens' predictions are to the user's true number of stars given to each movie. Even if a recommender system was able to correctly rank a user's movie recommendations, the system could fail if the predicted ratings it displays to the user are incorrect[1]. Because the predicted rating values create an ordering across the items, predictive accuracy can also be used to measure the ability of a recommender system to rank items with respect to user preference. On the other hand, evaluators who wish to measure predictive accuracy are necessarily limited to a metric that computes the difference between the predicted rating and true rating such as mean absolute error.

Mean absolute error and related metrics

Mean absolute error (often referred to as MAE) measures the average absolute deviation between a predicted rating and the user's true rating. Mean absolute error (Equation 1) has been used to evaluate recommender systems in several cases [Breese et al. 1998, Herlocker et al. 1999, Shardanand and Maes 1995].

$$\overline{|E|} = \frac{\sum_{i=1}^{N} |p_i - r_i|}{N} \qquad \textbf{(Eq. 1)}$$

Mean absolute error may be less appropriate for tasks such as *Find Good Items* where a ranked result is returned to the user, who then only views items at the top of the ranking. For these tasks, users may only care about errors in items that are ranked high, or that should be ranked high. It may be unimportant how accurate predictions are for items that the system correctly knows the user will have no interest in. Mean absolute error may be less appropriate when the granularity of true preference (a domain feature) is small, since errors will only affect the task if they result

[1] This is a primary reason that many implementations of recommender systems in a commercial setting only display a recommended-items list and do not display predicted values.

in erroneously classifying a good item as a bad one or vice versa; for example, if 3.5 stars is the cut-off between good and bad, then a one-star error that predicts a 4 as 5 (or a 3 as 2) makes no difference to the user.

Beyond measuring the accuracy of the predictions at every rank, there are two other advantages to mean absolute error. First, the mechanics of the computation are simple and easy to understand. Second, mean absolute error has well studied statistical properties that provide for testing the significance of a difference between the mean absolute errors of two systems.

Three measures related to mean absolute error are **mean squared error**, **root mean squared error**, and **normalized mean absolute error**. The first two variations square the error before summing it. The result is more emphasis on large errors. For example, an error of one point increases the sum of error by one, but an error of two points increases the sum by four. The third related measure, normalized mean absolute error [Goldberg et al. 2001], is mean absolute error normalized with respect to the range of rating values, in theory allowing comparison between prediction runs on different datasets (although the utility of this has not yet been investigated).

In addition to mean absolute error across all predicted ratings, Shardanand and Maes [1995] measured separately mean absolute error over items to which users gave extreme ratings. They partitioned their items into two groups, based on user rating (a scale of 1 to 7). Items rated below three or greater than five were considered extremes. The intuition was that users would be much more aware of a recommender system's performance on items that they felt strongly about. From Shardanand and Maes' results, the mean absolute error of the extremes provides a different ranking of algorithms than the normal mean absolute error. Measuring the mean absolute error of the extremes can be valuable. However, unless users are concerned only with how their extremes are predicted, it should not be used in isolation.

4.1.2 Classification Accuracy Metrics

Classification metrics measure the frequency with which a recommender system makes correct or incorrect decisions about whether an item is good. Classification metrics are thus appropriate for tasks such as *Find Good Items* when users have true binary preferences.

When applied to non-synthesized data in offline experiments, classification accuracy metrics may be challenged by data sparsity. The problem occurs when the collaborative filtering system being evaluated is generating a list of top recommended items. When the quality of the list is evaluated, recommendations may be encountered that have not been rated. How those items are treated in the evaluation can lead to certain biases.

One approach to evaluation using sparse data sets is to ignore recommendations for items for which there are no ratings. The recommendation list is first processed to remove all unrated items. The recommendation task has been altered to "predict the top recommended items that have been rated." In tasks where the user only observes the top few recommendations, this could lead to inaccurate evaluations of recommendation systems with respect to the user's task. The problem is that the quality of the items that the user would actually see may never be measured.

In an example of how this could be significant, consider the following situation that could occur when using the nearest neighbor algorithm described in [Herlocker et al. 2002]: when only one user in the dataset has rated an eclectic item I, then the prediction for item I for all users will be equal to the rating given by that user. If a user gave item I a perfect rating of 5, then the algorithm will predict a perfect 5 for all other users. Thus item I will immediately be placed at the top of the recommendation list for all users, in spite of the lack of confirming data. However, since no other user has rated this item, the recommendation for item I will be ignored by the evaluation metric, which thus will entirely miss the flaw in the algorithm.

Another approach to evaluation of sparse data sets is to assume default ratings, often slightly negative, for recommended items that have not been rated [Breese et al. 1998]. The downside of this approach is that the default rating may be very different from the true rating (unobserved) for an item.

A third approach that we have seen in the literature is to compute how many of the highly rated items are found in the recommendation list generated by the recommender system. In essence, we are measuring how well the system can identify items that the user was already aware of. This evaluation approach may result in collaborative filtering algorithms that are biased towards obvious, non-novel recommendations. or perhaps algorithms that are over fitted – fitting the known data perfectly, but new data poorly. In Section 5 of this article, we discuss metrics for evaluating novelty of recommendations.

Classification accuracy metrics do not attempt to directly measure the ability of an algorithm to accurately predict ratings. Deviations from actual ratings are tolerated, as long as they do not lead to classification errors. The particular metrics that we discuss are Precision and Recall and related metrics and ROC. We also briefly discuss some ad hoc metrics.

Precision and Recall and Related Measures

Precision and recall are the most popular metrics for evaluating information retrieval systems. In 1968, Cleverdon proposed them as the key metrics, and they have held ever since. For the evaluation of recommender systems, they have been used by Billsus and Pazzani [1998], Basu et al. [1998], and Sarwar et al. [2000a, 2000b].

Table I. Table showing the categorization of items in the document set with respect to a given information need.

	Selected	Not Selected	Total
Relevant	N_{rs}	N_{rn}	N_r
Irrelevant	N_{is}	N_{in}	N_i
Total	N_s	N_n	N

Precision and recall are computed from a 2x2 table, such as the one shown in **Table I**. The item set must be separated into two classes – relevant or not relevant. That is, if the rating scale is not already binary, we need to transform it into a binary scale. For example, the MovieLens dataset [Dahlen et al. 1998] has a rating scale of 1-5 and is commonly transformed into a binary scale by converting every rating of 4 or 5 to “relevant” and all ratings of 1-3 to “not-relevant.” For precision and recall, we also need to separate the item set into the set that was returned to the user (selected/recommended), and the set that was not. We assume that the user will consider all items that are retrieved.

Precision is defined as the ratio of relevant items selected to number of items selected, shown in Equation 2.

$$P = \frac{N_{rs}}{N_s} \qquad \textbf{(Eq 2)}$$

Precision represents the probability that a selected item is relevant. **Recall**, shown in Equation 3, is defined as the ratio of relevant items selected to total number of relevant items available. Recall represents the probability that a relevant item will be selected.

$$R = \frac{N_{rs}}{N_r} \qquad \textbf{(Eq 3)}$$

Precision and recall depend on the separation of relevant and non-relevant items. The definition of "relevance" and the proper way to compute it has been a significant source of argument within the field of information retrieval [Harter 1996]. Most information retrieval evaluation has focused on an objective version of relevance, where relevance is defined with respect to a query, and is independent of the user. Teams of experts can compare documents to queries and determine which documents are relevant to which queries. However, objective relevance makes no sense in recommender systems. Recommender systems recommend items based on the likelihood that they will meet a specific user's taste or interest. That user is the only person who can determine if an item meets his taste requirements. Thus, relevance is more inherently subjective in recommender systems than in traditional document retrieval.

In addition to user tastes being different, user rating scales may also be different. One user may consider a rating of 3 on a 5-point scale to be relevant, while another may consider it irrelevant. For this reason, much research using multi-point scales (such as in [Hill et al. 1995, Resnick et al. 1994, Shardanand and Maes 1995]) has focused on other metrics besides Precision/Recall. One interesting approach that has been taken to identify the proper threshold is to assume that a top percentile of items rated by a user are relevant [Basu et al. 1998].

Recall, in its purest sense, is almost always impractical to measure in a recommender system. In the pure sense, measuring recall requires knowing whether each item is relevant; for a movie recommender, this would involve asking many users to view all 5000 movies to measure how successfully we recommend each one to each user. IR evaluations have been able to estimate recall by pooling relevance ratings across many users, but this approach depends on the assumption that all users agree on which items are relevant, which is inconsistent with the purpose of recommender systems.

Several approximations to recall have been developed and used to evaluate recommender systems. Sarwar et al [2000a] evaluate their algorithms by taking a dataset of user ratings which they divide into a training set and a test set. They train the recommender algorithm on the training set, and then predict the top N items that the user is likely to find valuable, where N is some fixed value. They then compute recall as the percentage of known relevant items from the test set that appear in the top N predicted items. Since the number of items that each user rates is much smaller than the number of items in the entire dataset (see the discussion on data sparsity at the beginning of this section), the number of relevant items in the test set may be a small fraction of the number of relevant items in the entire dataset. While this metric can be useful, it has underlying biases that researchers must be aware of. In particular, the value of this metric depends heavily on the percentage of relevant items that each user has rated. If a user has rated only a small percentage of relevant items, a recommender with high "true recall" may yield a low value for measured recall, since the recommender may have recommended unrated relevant items. Accordingly, this metric should only be used in a comparative fashion on the same dataset; it should not be interpreted as an absolute measure.

We have also seen precision measured in the same fashion [Sarwar et al. 2000a] with relevant items being selected from a small pool of rated items and predicted items being selected from a much larger set of items. Similarly, this approximation to precision suffers from the same biases as the recall approximation.

Perhaps a more appropriate way to approximate precision and recall would be to predict the top N items *for which we have ratings*. That is, we take a user's ratings, split them into a training set and a test set, train the algorithm on the training set, then predict the top N items *from that user's test set*. If we assume that the distribution of relevant items and non-relevant items within the user's test set is the same as the true distribution for the user across all items, then the precision and recall will be much closer approximations of the true precision and recall. This approach is taken in [Basu et al. 1998].

In information retrieval, precision and recall can be linked to probabilities that directly affect the user. If an algorithm has a measured precision of 70%, then the user can expect that, on average, 7 out of every 10 documents returned to the user will be relevant. Users can more intuitively comprehend the meaning of a 10% difference in precision than they can a 0.5-point difference in mean absolute error.

One of the primary challenges to using precision and recall to compare different algorithms is that precision and recall must be considered together to evaluate completely the performance of an algorithm. It has been observed that precision and recall are inversely related [Cleverdon and Kean 1968] and are dependent on the length of the result list returned to the user. When more items are returned, then the recall increases and precision decreases. Therefore, if the information system doesn't always return a fixed number of items, we must provide a vector of precision/recall pairs to fully describe the performance of the system. While such an analysis may provide a detailed picture of the performance of a system, it makes comparison of systems complicated, tedious, and variable between different observers. Furthermore, researchers may carefully choose at which levels of recall (or search length) they report precision and recall to match the strengths in their system.

Several approaches have been taken to combine precision and recall into a single metric. One approach is the F1 metric (Equation 4), which combines precision and recall into a single number The F1 has been used to evaluate recommender systems in [Sarwar et al. 2000a, Sarwar et al. 2000b]. An alternate approach taken by the TREC community is to compute the average precision across several different levels of recall or the average precision at the rank of each relevant document [Harman 1995]. The latter approach was taken in all but the initial TREC conference. This approach is commonly referred to as Mean Average Precision or MAP. F1 and mean average precision may be appropriate if the underlying precision and recall measures on which it is based are determined to be appropriate.

$$F_1 = \frac{2PR}{P+R} \textbf{ (Eq. 4)}$$

Precision alone at a single search length or a single recall level can be appropriate if the user does not need a complete list of all potentially relevant items, such as in the *Find Good Items* task. If the task is to find all relevant items in an area, then recall becomes important as well. However, the search length at which precision is measured should be appropriate for the user task and content domain.

As with all classification metrics, precision and recall are less appropriate for domains with non-binary granularity of true preference**.** For those tasks, at any point in the ranking, we want the current item to be more relevant than all items lower in the ranking. Since precision and recall only measure binary relevance, they cannot measure the quality of the ordering among items that are selected as relevant.

ROC Curves, Swets' A Measure, and Related Metrics

ROC curve-based metrics provide a theoretically grounded alternative to precision & recall. There are two different popularly held definitions for the acronym ROC. Swets [1963, 1969] introduced the ROC metric to the information retrieval community under the name "relative operating characteristic". More popular however, is the name "receiver operating characteristic," which evolved from the use of ROC curves in signal detection theory [Hanley and McNeil 1982]. Regardless, in both cases, ROC refers to the same underlying metric.

The ROC model attempts to measure the extent to which an information filtering system can successfully distinguish between signal (relevance) and noise. The ROC model assumes that the information system will assign a predicted level of relevance to every potential item. Given this assumption, we can see that there will be two distributions, shown in Figure 1. The distribution on the left represents the probability that the system will predict a given level of relevance (the x-axis) for an item that is in reality not relevant to the information need. The distribution on the right indicates the same probability distribution for items that are relevant. Intuitively, we can see that the further apart these two distributions are, the better the system is at differentiating relevant items from non-relevant items.

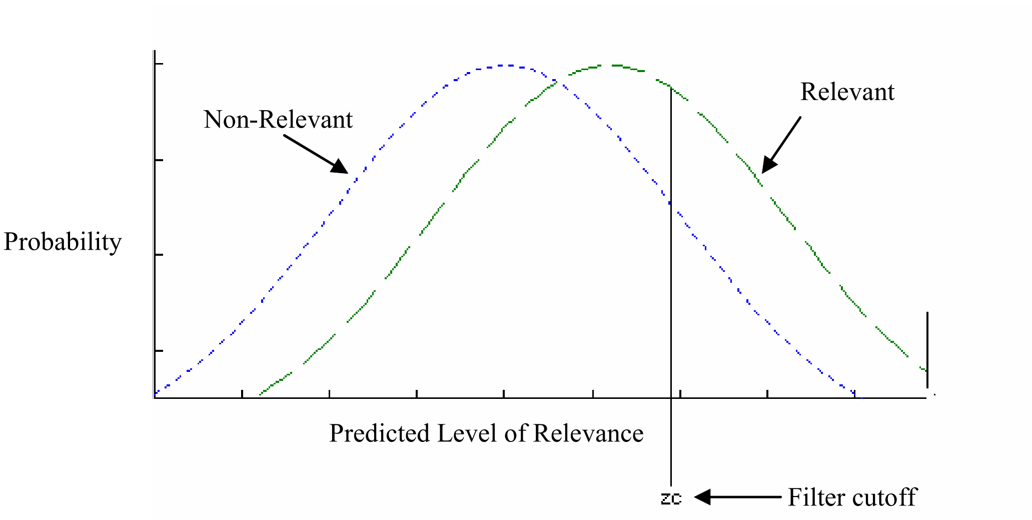

Figure 1. A possible representation of the density functions for relevant and irrelevant items.

With systems that return a ranked list, the user will generally view the recommended items starting at the top of the list and work down until the information need is met, a certain time limit is reached, or a predetermined number of results are examined. In any case, the ROC model assumes that there is a filter tuning value z_c, such that all items that the system ranks above the cutoff are viewed by the user, and those below the cutoff are not viewed by the user. This cutoff defines the search length. As shown in Figure 1, at each value of z_c, there will be a different value of recall (percentage of good items returned, or the area under the relevant probability distribution to the right of z_c) and fallout (percentage of bad items returned, or the area under the non-relevant probability distribution to the right of z_c). The ROC curve represents a plot of recall versus fallout, where the points on the curve correspond to each value of z_c. An example of an ROC curve is shown in Figure 2.

A common algorithm for creating an ROC curve goes as follows:

1. Determine how you will identify if an item is relevant or non-relevant.
2. Generate a predicted ranking of items.
3. For each predicted item, in decreasing order of predicted relevance (starting the graph at the origin):
    a. If the predicted item is indeed relevant draw the curve one step vertically.
    b. If the predicted item is not relevant, draw the curve one step horizontally to the right.
    c. If the predicted item has not been rated (i.e. relevance is not known), then the item is simply discarded and does not affect the curve negatively or positively.

An example of an ROC curve constructed in this manner is shown in Figure 2.

A perfect predictive system will generate an ROC curve that goes straight upward until 100% of relevant items have been encountered, then straight right for the remaining items. A random predictor is expected to produce a straight line from the origin to the upper right corner[2].

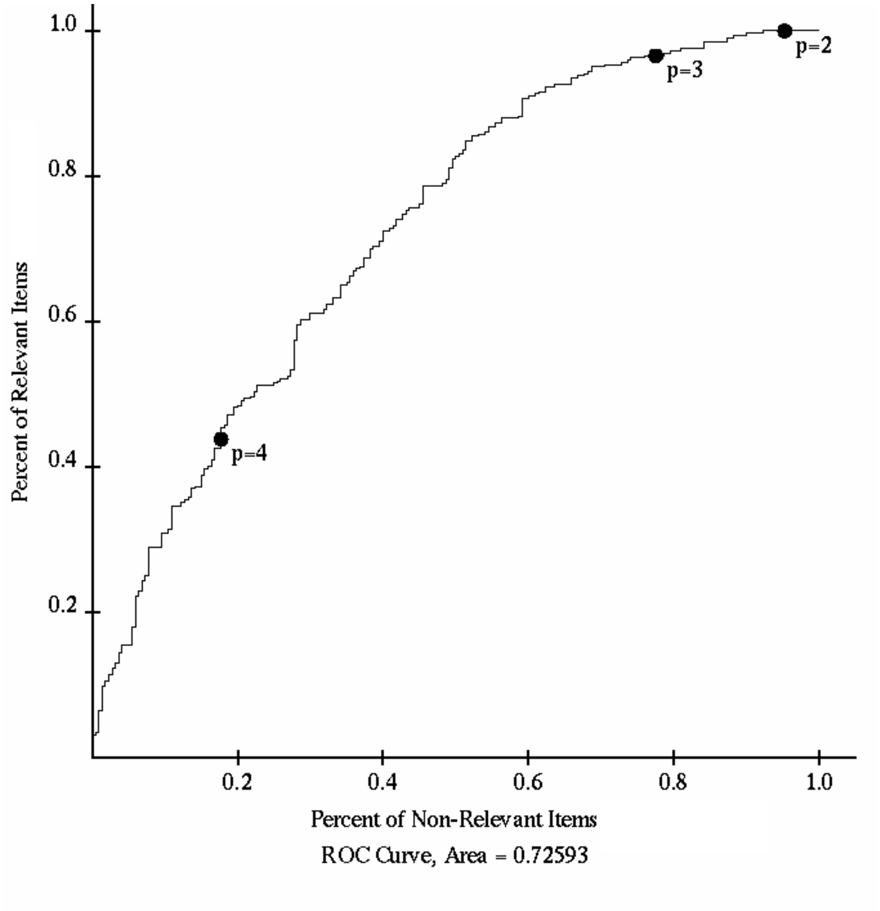

Figure 2. An example of an ROC curve. The p-values shown on the curve represent different prediction cutoffs. For example if we chose to select all items with predictions of 4 or higher, then we experience approximately 45% of all relevant items and 20% of all non-relevant items.

ROC curves are useful for tuning the signal/noise tradeoff in information systems. For example, by looking at an ROC curve, you might discover that your information filter performs well for an initial burst of signal at the top of the rankings, and then produces only small increases of signal for moderate increases in noise from then on.

Similar to Precision and Recall measures, ROC curves make an assumption of binary relevance. Items recommended are either successful recommendations (relevant) or unsuccessful recommendation (non-relevant).

[2] Schein et al. [Schein et al. 2002] present an alternate method of computing an ROC - a *Customer ROC (CROC)*. A CROC measurement applied to a perfect recommender may not produce a perfect ROC graph as described. The reasoning is that some recommender systems may display more recommendations then there exist "relevant" items to the recommender, and that these additional recommendations should be counted as false-positives.

One consequence of this assumption is that the ordering among relevant items has no consequence on the ROC metric – if all relevant items appear before all non-relevant items in the recommendation list, you will have a perfect ROC curve.

Comparing multiple systems using ROC curves becomes tedious and subjective, just as with precision and recall. However, a single summary performance number can be obtained from an ROC curve. The area underneath an ROC curve, also known as Swet's A measure, can be used as a single metric of the system's ability to discriminate between good and bad items, independent of the search length. According to [Hanley and McNeil 1982], the area underneath the ROC curve is equivalent to the probability that the system will be able to choose correctly between two items, one randomly selected from the set of bad items, and one randomly selected from the set of good items. Intuitively, the area underneath the ROC curve captures the recall of the system at many different levels of fallout. It is also possible to measure the statistical significance of the difference between two areas [Hanley and McNeil 1982, Le and Lindren 1995].

The ROC area metric has the disadvantage that equally distant swaps in the rankings will have the same affect on ROC area regardless of whether they occur near the top of the ranking or near the end of the ranking. For example, if a good item is ranked 15 instead of 10, it will have roughly the same affect on the ROC area as if a good item is ranked 200 instead of 195. This disadvantage could be significant for tasks such as *Find Good Items* where the first situation is likely to have a greater negative affect on the user. This disadvantage is somewhat minimized by the fact that relevance is binary and exchanges within a relevance class have no affect (if items ranked 10-15 are all relevant, an exchange between 10 and 15 will have no affect at all). On the other hand, for tasks such as *Find All Good Items*, the discussed disadvantage may not be significant.

Hanley and McNeil [1982] present a method by which one can determine the number of data points necessary to ensure that a comparison between two areas has good statistical power (defined as a high probability of identifying a difference if one exists). Hanley's data suggests that many data points may be required to have a high level of statistical power. The number of required data points for significance becomes especially large when the two areas being compared are very close in value. Thus, to confidently compare the results of different algorithms using ROC area, the potential result set for each user must also be large.

The advantages of ROC area metric are that it (a) provides a single number representing the overall performance of an information filtering system, (b) is developed from solid statistical decision theory designed for measuring the performance of tasks such as those that a recommender system performs, and (c) covers the performance of the system over all different recommendation list lengths.

To summarize the disadvantages of the ROC area metric: (a) a large set of potentially relevant items is needed for each query; (b) for some tasks, such as *Find Good Items* users are only interested in performance at one setting, not all possible settings; (c) equally distant swaps in rankings have the same effect no matter where in the ranking they occur; and (d) it may need a large number of data points to ensure good statistical power for differentiating between two areas.

The ROC area measure is most appropriate when there is a clear binary relevance relationship and the task is similar to *Find Good Items*, where the user wants to see as many of the relevant answers as possible within certain resource limitations.

Ad Hoc Classification Accuracy Measures

Ad hoc measures of classification accuracy have attempted to identify error rates and, in particular, large errors. Error rate can be measured in a manner derived from Precision and Recall. Specifically, the error rate for a system is the number of incorrect recommendations it makes divided by the total number of recommendations. If a system recommends only a few items, it is possible to measure error rate experimentally. Jester, for example, which presents jokes to users, can evaluate the error rate based on the immediate feedback users give to each joke [Goldberg et al. 2001]. More commonly, the error rate computation is limited to the subset of recommended items for which a rating is available; this approach introduces the bias that users commonly avoid consuming (and therefore rating) items that don't interest them, and therefore this approximate error rate is likely to be lower than the true error rate.

Another ad hoc technique specifically identifies large errors. Sarwar et al. measured *reversals* when studying agent-boosted recommendations [1998]. Errors of three or more points on a five point scale were considered significant enough to potentially undermine user confidence, and therefore were tallied separately. Such a measurement mixes aspects of classification and prediction accuracy, but has not been generally used by later researchers. It might be particularly appropriate for the *Evaluate Recommender* task.

4.1.3 Rank Accuracy Metrics

Rank accuracy metrics measure the ability of a recommendation algorithm to produce a recommended ordering of items that matches how the user would have ordered the same items. Unlike classification metrics, ranking metrics are more appropriate to evaluate algorithms that will be used to present ranked recommendation lists to the user, in domains where the user's preferences in recommendations are non-binary.

Rank accuracy metrics may be overly sensitive for domains where the user just wants an item that is "good enough" (binary preferences) since the user won't be concerned about the ordering of items beyond the binary classification. For example, even if the top ten items ranked by the system were relevant, a rank accuracy metric might give a non-perfect value because the best item is actually ranked 10th. By the same token, rank accuracy metrics can distinguish between the "best" alternatives and just "good" alternatives and may be more appropriate for domains where that distinction is important. In such domains it is possible for all the top recommended items to be relevant, but still not be the "best" items.

Ranking metrics do not attempt to measure the ability of an algorithm to accurately predict the rating for a single item – they are not predictive accuracy metrics and are not appropriate for evaluating the *Annotation in Context* task. If a recommender system will be displaying predicted rating values, it is important to additionally evaluate the system using a predictive accuracy metric as described above. We examine several correlation metrics, a half-life utility metric, and the NDPM metric.

4.1.4 Prediction-Rating Correlation

Two variables are correlated if the variance in one variable can be explained by the variance in the second. Three of the most well known correlation measures are Pearson's product-moment correlation, Spearman's ρ, and Kendall's Tau.

Pearson correlation measures the extent to which there is a linear relationship between two variables. It is defined as the covariance of the z-scores, shown in Equation 5.

$$c = \frac{\sum (x-\bar{x})(y-\bar{y})}{n * stdev(x) stdev(y)} \qquad \textbf{(Eq. 5)}$$

Rank correlations, such as Spearman's ρ (Equation 6) and Kendall's Tau, measure the extent to which two different rankings agree independent of the actual values of the variables. Spearman's ρ is computed in the same manner as the Pearson product-moment correlation, except that the x's and y's are transformed into ranks, and the correlations are computed on the ranks.

$$\rho = \frac{\sum (u-\bar{u})(v-\bar{v})}{n * stdev(u) stdev(v)} \qquad \textbf{(Eq. 6)}$$

Kendall's Tau represents a different approach to computing the correlation of the rankings that is independent of the variable values. One approximation to Kendall's Tau is shown in Equation 7. C stands for the number of concondant pairs – pairs of items which the system predicts in the proper ranked order. D stands for the number of discordant pairs – pairs which the system predicts in the wrong order. TR is number of pairs of items in the true ordering (the ranking determined by the user's ratings) that have tied ranks (ie the same rating) while TP is the number of pairs of items in the predicted ordering that have tied ranks (the same prediction value).

$$Tau = \frac{C - D}{sqrt((C + D + TR)(C + D + TP))} \textbf{(Eq. 7)}$$

In spite of their simplicity, the above correlation metrics have not been used extensively in the measurement of recommender systems or information retrieval systems. Pearson correlation was used by Hill et al.[1995] to evaluate the performance of their recommender system.

The advantages of correlation metrics are (a) they compare a non-binary system ranking to a non-binary user ranking, (b) they are well understood by the scientific community, and (c) they provide a single measurement score for the entire system.

There may be weaknesses in the way in which the "badness" of an interchange is calculated with different correlation metrics. For example, Kendall's Tau metric applies equal weight to any interchange of equal distance, no matter where it occurs (similar to the ROC area metric). Therefore, an interchange between recommendations 1 and

2 will be just as bad as an interchange between recommendations 1000 and 1001. However, if the user is much more likely to consider the first five, and will probably never examine items ranked in the thousands, the interchange between 1 and 2 should have a more substantial negative impact on the outcome of the metric.

The Spearman correlation metric does not handle weak (partial) orderings well. Weak orderings occur whenever there are at least two items in the ranking such that neither item is preferred over the other. If a ranking is not a weak ordering then it is called a complete ordering. If the user's ranking (based on their ratings) is a weak ordering and the system ranking is a complete ordering, then the Spearman correlation will be penalized for every pair of items which the user has rated the same, but the system ranks at different levels. This is not ideal, since the user shouldn't care how the system orders items that the user has rated at the same level. Kendall's Tau metric also suffers the same problem, although to a lesser extent than the Spearman correlation.

Half-life Utility Metric

Breese et al. [1998], presented a new evaluation metric for recommender systems that is designed for tasks where the user is presented with a ranked list of results, and is unlikely to browse very deeply into the ranked list. Another description of this metric can be found in [Heckerman et al. 2000]. The task for which the metric is designed is an Internet web-page recommender. They claim that most Internet users will not browse very deeply into results returned by search engines.

This *half-life utility metric* attempts to evaluate the utility of a ranked list to the user. The utility is defined as the difference between the user's rating for an item and the "default rating" for an item. The default rating is generally a neutral or slightly negative rating. The likelihood that a user will view each successive item is described with an exponential decay function, where the strength of the decay is described by a half-life parameter. The expected utility (R_a) is shown in Equation 8. $r_{a,j}$ represents the rating of user a on item j of the ranked list, d is the default rating, and α is the half-life. The half-life is the rank of the item on the list such that there is a 50% chance that the user will view that item. Breese et al. used a half-life of 5 for his experiments, but noted that using a half-life of 10 caused little additional sensitivity of results.

$$R_a = \sum_j \frac{\max\left(r_{a,j} - d, 0\right)}{2^{(j-1)/(\alpha-1)}} \qquad \textbf{(Eq. 8)}$$

The overall score for a dataset across all users (R) is shown in Equation 9. $R_a^{\max}$ is the maximum achievable utility if the system ranked the items in the exact order that the user ranked them.

$$R = 100\frac{\sum_a R_a}{\sum_a R_a^{\max}} \qquad \textbf{(Eq. 9)}$$

The half-life utility metric is best for tasks domains where there is an exponential drop in true utility (one could consider utility from a cost/benefit ratio analysis) as the search length increases, assuming that the half-life α and default vote *d* are chosen appropriately in the utility metric. The utility metric applies most of the weight to early items, with every successive rank having exponentially less weight in the measure. To obtain high values of the metric, the first predicted rankings must consist of items rated highly by the user. The downside is that if the true function describing the likelihood of accessing each rank is significantly different from the exponential used in the metric then the measured results may not be indicative of actual performance. For example, if the user almost always searches 20 items into the ranked recommendation list, then the true likelihood function is a step function which is constant for the first 20 items and 0 afterwards.

The half-life utility metric may be overly sensitive in domains with binary user preferences where the user only requires that top recommendations be "good enough" or for user tasks such as *Find All Good Items* where the user wants to see all good items.

There are other disadvantages to the half-life utility metric. First, weak orderings created by the system will result in different possible scores for the same system ranking. Suppose the system outputs a recommendation list, with three items sharing the top rank. If the user rated those three items differently, then depending on what order the recommender outputs those items, the metric could have very different values (if the ratings were significantly different).

Second, due to the application of the max() function in the metric (Equation 8), all items that are rated less than the default vote contribute equally to the score. Therefore, an item occurring at system rank 2 that is rated just slightly less than the default rating (which usually indicates ambivalence) will have the same effect on the utility as an item that has the worst possible rating. The occurrence of extremely wrong predictions in the high levels of a system ranking can undermine the confidence of the user in the system. Metrics that penalize such mistakes more severely are preferred.

To summarize, the half-life utility metric is the only one that we have examined that takes into account non-uniform utility. Thus it could be appropriate for evaluation of the *Find Good Items* tasks in domains such non-uniform utility is believed to exist. On the other hand it has many disadvantages, in particular when considering standardization across different researchers. Different researchers could use significantly different values of alpha or the default vote – making it hard to compare results across researchers and making it easy to manipulate results. Furthermore, the half-life parameter is unlikely to be the same for all users (different users need/desire different numbers of results).

The NDPM measure.

NDPM was used to evaluate the accuracy of the FAB recommender system [Balabanovíc and Shoham 1997]. It was originally proposed by Yao [1995]. Yao developed NDPM theoretically, using an approach from decision and measurement theory. NDPM stands for "normalized distance-based performance measure". NDPM (Equation 10) can be used to compare two different weakly-ordered rankings.

$$NDPM = \frac{2C^{-} + C^{u}}{2C^{i}} \quad \textbf{(Eq. 10)}$$

C^- is the number of contradictory preference relations between the system ranking and the user ranking. A contradictory preference relation happens when the system says that item 1 will be preferred to item 2, and the user ranking says the opposite. C^u is the number of compatible preference relations, where the user rates item 1 higher than item 2, but the system ranking has item 1 and item 2 at equal preference levels. C^i is the total number of "preferred" relationships in the user's ranking (i.e. pairs of items rated by the user for which one is rated higher than the other). This metric is comparable among different datasets (it is normalized), because the numerator represents the distance, and the denominator represents the worst possible distance.

NDPM is similar in form to the Spearman and Kendall's Tau rank correlations, but provides a more accurate interpretation of the effect of tied user ranks. However, it does suffer from the same interchange weakness as the rank correlation metrics (interchanges at the top of the ranking have the same weight as interchanges at the bottom of the ranking).

Because NDPM does not penalize the system for system orderings when the user ranks are tied, NDPM may be more appropriate than the correlation metrics for domains where the user is interested in items that are "good-enough." User ratings could be transformed to binary ratings (if they were not already), and NDPM could be used to compare the results to the system ranking.

As NDPM only evaluates ordering and not prediction value, it is not appropriate for evaluating tasks where the goal is to predict an actual rating value.

4.1.5 An Empirical Comparison of Evaluation Metrics

After conceptually analyzing the advantages and disadvantages, a natural question is: "which of these advantages and disadvantages have a significant effect on the outcome of a metric?" In an effort to quantify the differences between the above mentioned evaluation metrics, we computed a set of different evaluation metrics on a set of results from different variants of a collaborative filtering prediction algorithm and examined the extent to which the different evaluation metrics agreed or disagreed.

We examined the predictions generated by variants of a classic nearest-neighbor collaborative filtering algorithm formed by perturbing many different key parameters. We used this data for examination of evaluation metrics. There were 432 different variants of the algorithm tested, resulting in the same number of sets of predictions. The parameters of the algorithms that were varied to produce the different results included: size of neighborhood, similarity measure used to compute closeness of neighbors, threshold over which other users were considered neighbors, and type of normalization used on the ratings. (see [Herlocker et al. 2002] for more information on the algorithm)

For each of these result sets, we computed mean absolute error, Pearson correlation, Spearman rank correlation, area underneath an ROC-4 and ROC-5[3] curve, the half-life utility metric, mean average precision at relevant documents and the NDPM metric. For several of the metrics, there are two different variants: *overall* and *per-user*. The

[3] ROC-4 refers to an ROC curve where ratings of 4 and above are considered signal and 3 and below are considered noise.

difference between these two variants is the manner in which averaging was performed. In the overall case, predictions for all the users were pooled together into a single file and then sorted. Likewise the ratings for those items were pooled into a single file and sorted. A ranking metric was then applied once to compare those two files. In the per-user case, predictions were computed for each user, and the ranking metric was computed for each user. Then the ranking metric was averaged over all users.

The experiment was performed with data taken from the MovieLens web-based movie recommender (www.movielens.org). The data were sampled from the data collected over a seven-month period from September 19th, 1997 through April 22nd, 1998. The data consisted of 100,000 movie ratings from 943 users on 1682 items. Each user sampled had rated at least 20 items. For each of the users, 10 rated items are withheld from the training. After training the system with all the other ratings, predictions are generated for the 10 withheld items. Then the predictions were compared to the user's ratings, and the list ranked by predictions was compared to the list ranked by user ratings. The data are freely available from www.grouplens.org, and we encourage researchers using other families of collaborative filtering algorithms to replicate this work using the same data set for comparability.

This analysis is performed on a single family of algorithms on a single dataset, so the results should not be considered comprehensive. However, we believe that the results show evidence of relationships between the metrics that should be investigated further. Our goal is not to provide a deep investigation of the empirical results, which would constitute a entire article by itself.

Figure 3 is a scatter plot matrix showing an overview of all the results. Each cell of the matrix represents a comparison between two of the metrics[4]. Each point of the scatter plot represents a different variant of the recommender algorithm. In the following paragraphs (and figures), we will look more closely at subsets of the results.

[4] Note that because we are displaying the complete matrix, each comparison pair appears twice. However, by having the complete matrix, we can easily scan one metric's interactions with all other metrics in a single row or column.

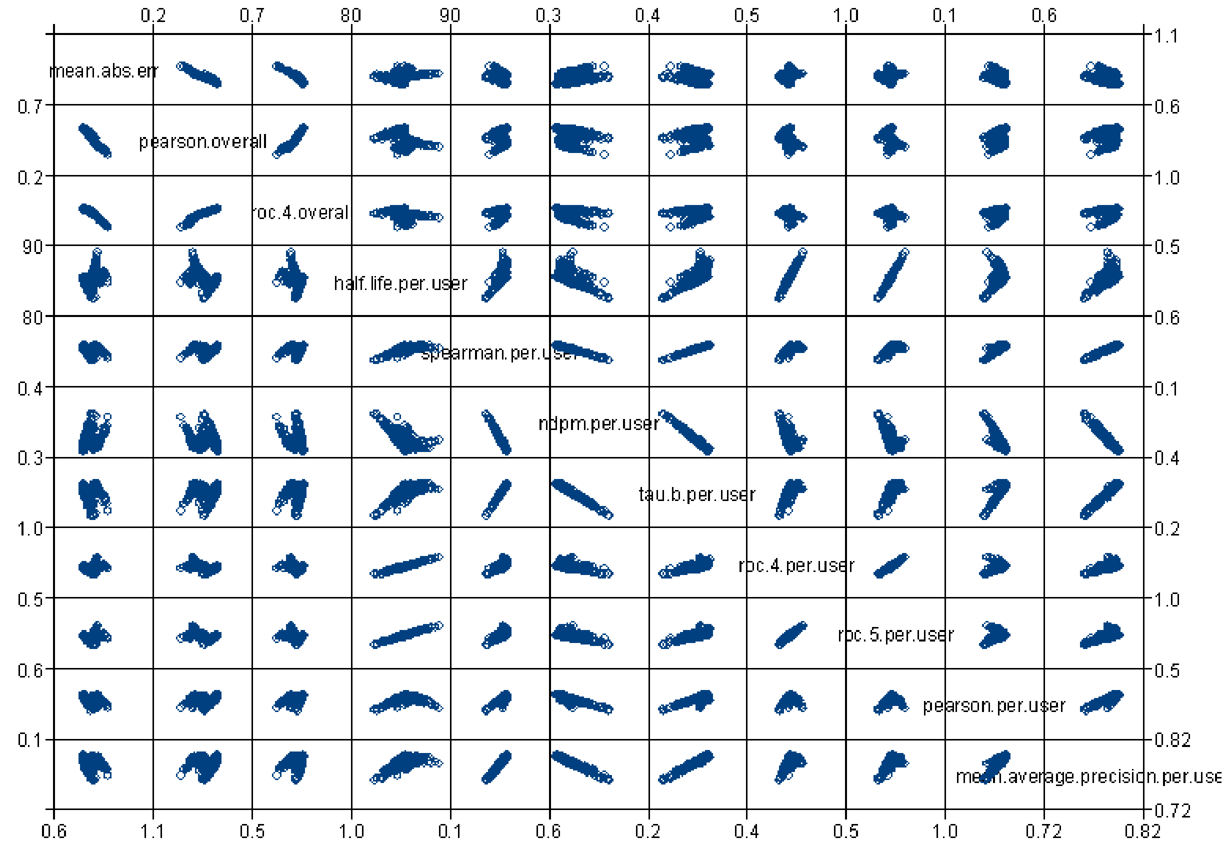

Figure 3. Overview of the comparative evaluation of several different evaluation metrics on 432 different variants of a nearest neighbor-based collaborative filtering algorithm.

In analyzing the data in Figure 3, we notice that there is very strong linear agreement among different subsets of the metrics tested. One subset with strong agreement includes the per-user correlation metrics and mean average precision. This subset is shown expanded in Figure 4. For the data and algorithms tested, Figure 4 suggests that rank correlations do not provide substantially different outcomes from each other or from Pearson correlation. Figure 4 also indicates that mean average precision is highly correlated with the per-user correlation metrics.

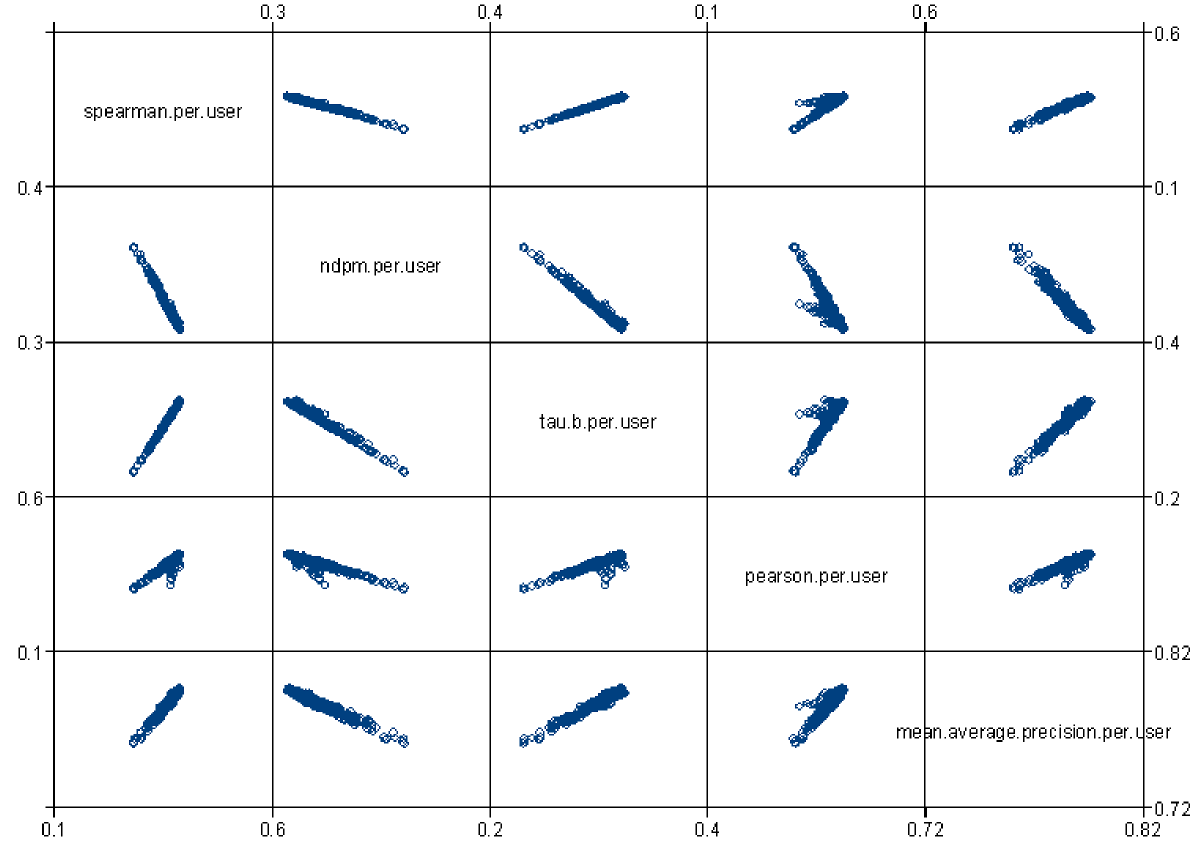

Figure 4. Comparison among results provided by all the per-user correlation metrics and the mean average precision per user metric. These metrics have strong linear relationships with each other.

Figure 5 shows a different, mutually exclusive subset of the evaluation metrics that includes the per-user Half-life Utility metric as well as the per-user ROC-4 and ROC-5 area metrics. We can see that these three metrics are strongly correlated, even more so that the previous subset of metrics.

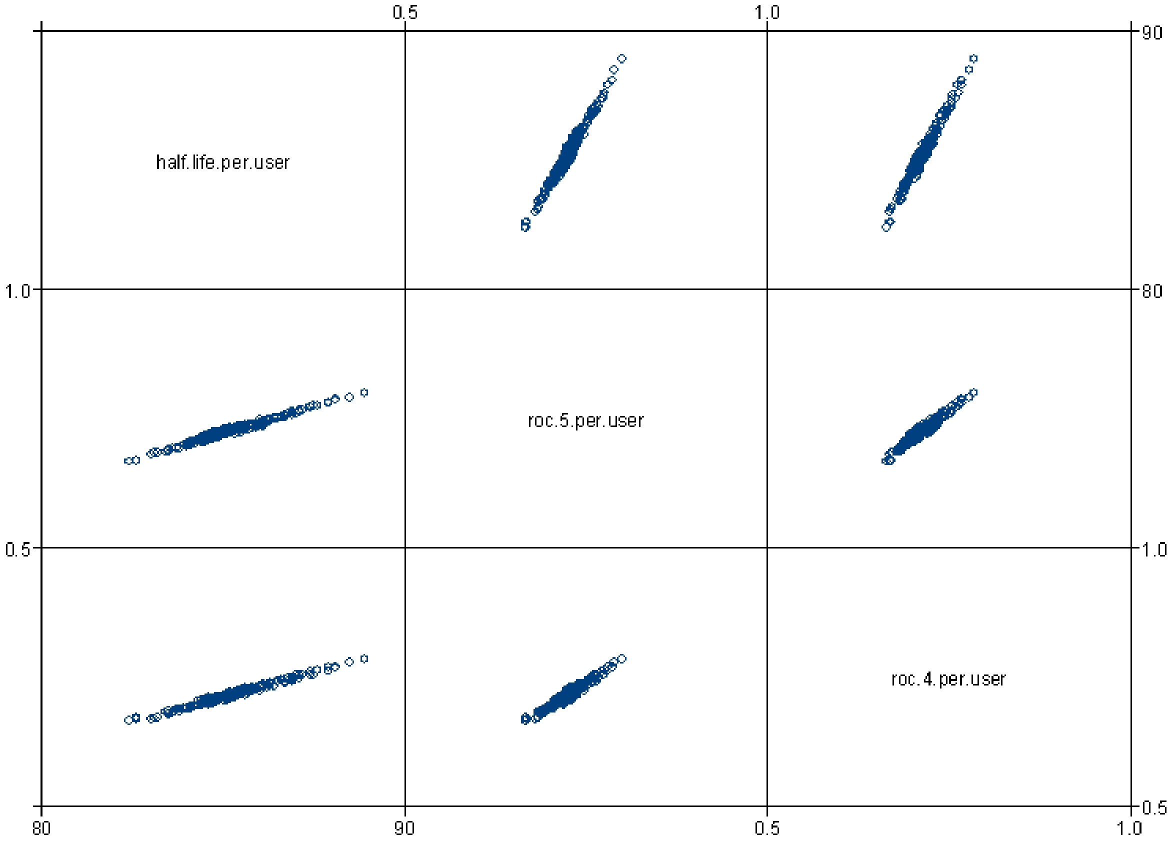

Figure 5. Comparison between results provided by the ROC-4 Area metric, the ROC-5 Area metric, and the Half-life Utility metric. The graphs depict strong linear correlations.

The final subset that we shall examine contains all the metrics that are computed overall as opposed to the metrics depicted in Figure 4 and 5, which are per-user. Figure 6 shows that the metrics computed overall (mean absolute error[5], Pearson Correlation, and ROC-4) have mostly linear relationships.

[5] Note that Mean Absolute Error could produces the same result whether it is averaged per-user or overall. However, it strongly correlates with the overall metrics and not the per-user metrics.

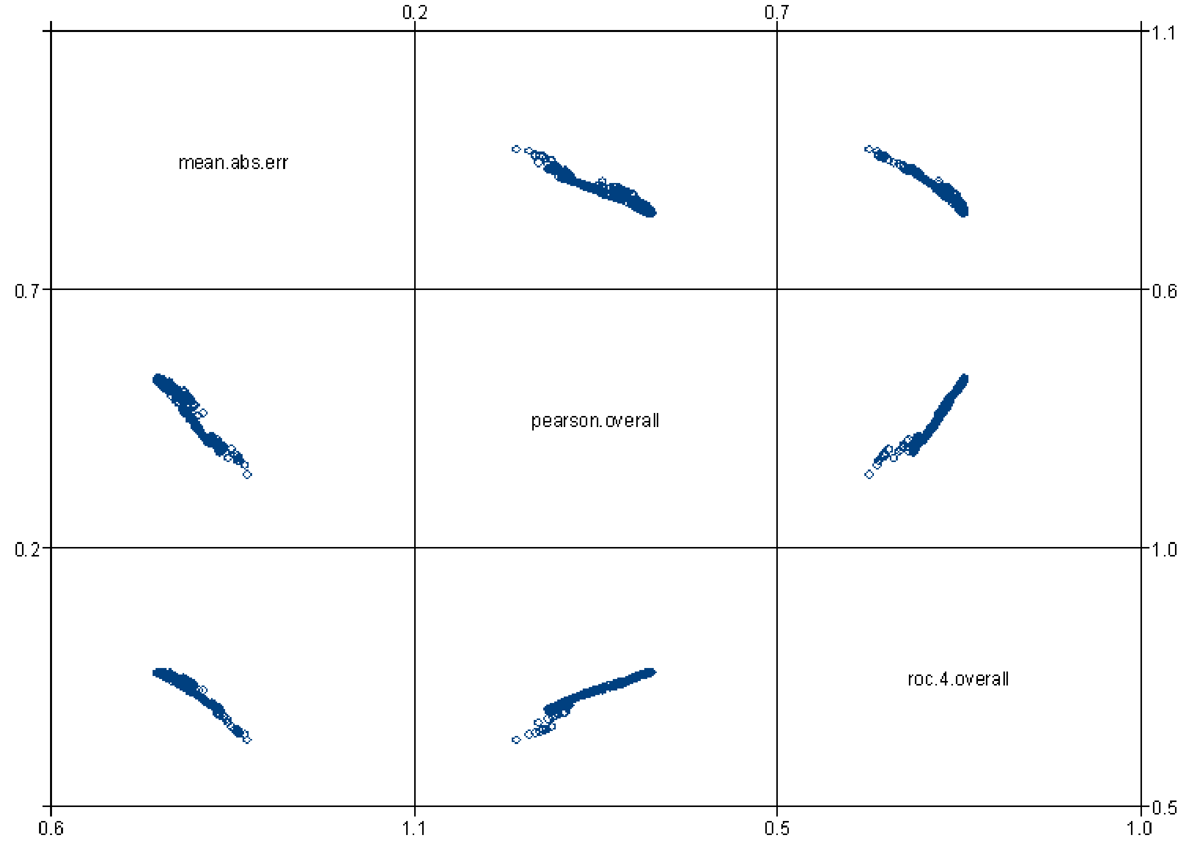

Figure 6. Comparison between metrics that are averaged overall rather than per-user. Note the linear relationship between the different metrics.

To bring the analysis back to the entire set of metrics, we have chosen one representative from each of the subsets that were depicted in Figures 4-6. Figure 7 shows a comparison between these representatives. *Pearson per-user* represents the subset of per-user metrics and Mean Average Precision that are depicted in Figure 4. *ROC-4 per user* represents the ROC-4, ROC-5, and Half-life Utility metrics, all averaged per-user. Mean Absolute Error represents the subset of overall metrics depicted in Figure 6. We can see that while there is strong agreement within each subset of algorithms (as seen in Figures 4-6), there is little agreement between algorithms from different subsets. Algorithms averaged per-user do not seem to agree with algorithms averaged overall. The ROC-4, ROC-5, and Half-life Utility metrics averaged per user do not agree with the other metrics that are averaged per user.

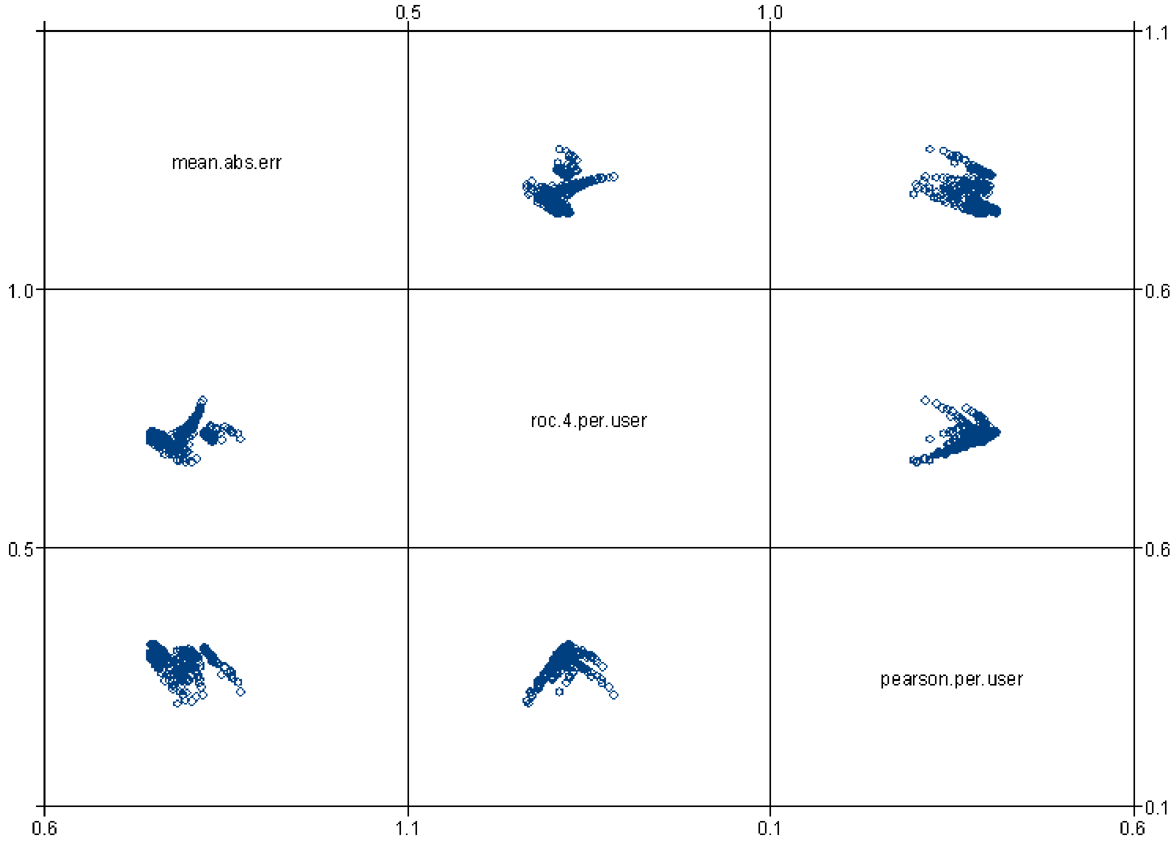

Figure 7. A comparison of representative metrics from the three subsets that were depicted in Figures 4-6. Within each of the subsets, the metrics strongly agree, but this figure shows that metrics from different subsets do not correlate well

Several interesting observations can be taken from that data in Figures 3-7.

- Metrics that are computed per user and then averaged provide different rankings of algorithms that metrics that are computed overall.
- There doesn't appear to be a substantial difference between the Pearson correlation metrics and rank correlation metrics, although a good number of outliers exist.
- Mean Average Precision provides roughly the same ordering of algorithms as the correlation metrics that are computed per user and averaged.
- The ROC area metrics (ROC-4 and ROC-5) when computed overall perform very similar to the other overall metrics, such as Mean Absolute Error and Pearson Overall. However, when they are averaged per user, they provide different rankings of algorithms than other per-user metrics, with the exception of the Half-life Utility metric

- The Half-life utility metric, which is averaged per user provides different rankings of algorithms than the per-user correlation metrics and mean average precision, yet produces rankings similar to the ROC area metrics when computed per user.

In support of these observations, Schein et al [2002] have also observed that overall metrics and per-user metric can provide conflicting results. They observed differences between an overall ROC (which they call Global ROC) and a per-user ROC (which they call a Customer ROC).

One should hold in mind that these empirical results, while based on numerous data points, all represent perturbations of the same base algorithm. The range in rank scores do not vary that much. Future work could extend the comparison of these evaluation metrics across significantly different recommendation algorithms.

4.2 Accuracy Metrics - Summary

We have examined a variety of accuracy evaluation metrics that have been used before to evaluate collaborative filtering systems. We have examined them both conceptually and empirically. The conceptual analysis suggests that certain evaluation metrics are more appropriate for certain tasks. Based on this analysis, there appears to be a potential for inaccurate measurement of certain tasks if the wrong metric is used. Our empirical analysis of one class of collaborative filtering algorithm demonstrates that many of the argued conceptual mismatches between metrics and tasks do not manifest themselves when evaluating the performance of predictive algorithms on movie rating data. On the other hand, we were able to demonstrate that different outcomes in evaluation can be obtained by carefully choosing evaluation metrics from different classes that we identified.

The empirical analysis that we have performed represents only a sample – one class of algorithm and one dataset. A TREC-like environment for collaborative filtering algorithms, with different tracks (and datasets) for different tasks would provide algorithmic results from many different algorithms and systems. These results would provide valuable data for the further verification of the properties of metrics discussed in this section

A final note is that a similar (but very brief) analysis has been performed on metrics for evaluating text retrieval systems. Voorhees and Harman report the strength of correlations computed between different evaluation metrics used in the TREC-7 analysis [Voorhees and Harman 1999]. Instead of showing scatterplots relating metrics, they computed correlation values between different metrics. Their results focused primarily on different variants of precision/recall which we do not discuss here. As the domain features of the document retrieval context are significantly different from the recommender systems context, we do not attempt to incorporate their results here.

5. BEYOND ACCURACY

There is an emerging understanding that good recommendation accuracy alone does not give users of recommender systems an effective and satisfying experience. Recommender systems must provide not just accuracy, but also usefulness. For instance, a recommender might achieve high accuracy by only computing predictions for easy-to-predict items – but those are the very items for which users are least likely to need predictions. Further, a system that always recommends very popular items can promise that users will like most of the items recommended – but a simple popularity metric could do the same thing. (Recommending only very unpopular items and promising that users won't like them is even less useful.)

By recalling that performance of a recommender system must be evaluated with respect to specific user tasks and the domain context, we can deepen the argument for moving beyond accuracy. For example, consider the *Find Good Items* task in a domain where the user wants to select a single item whose value exceeds a threshold – and suppose that the system follows a typical strategy of offering a relatively small, ordered set of recommendations. In this case, it may be best for the system to try to generate a few highly useful recommendations, even at the risk of being off the mark with the others. If the supporting information about the items is good enough, then the user will be able to identify the best recommendation quickly. Turpin & Hersh's study of search engines [2001] provides support for this position. Their subjects were divided into two sets, with half using a simple, baseline search engine, and the others using a state of the art engine. While the latter returned significantly more accurate results, subjects in both cases were about as successful at completing their tasks (e.g., finding the answer to a question such as "Identify a set of Roman ruins in present-day France"). Turpin & Hersh believed that this showed that the difference between (say) 3 and 5 relevant documents in a list of 10 documents was not really material to the user; nor did it matter much if the relevant documents were right at the top of the list or a bit further down. Subjects were able to scan through the titles and brief synopses and quickly locate a relevant document.

This section considers measures of recommender system usefulness that move beyond accuracy to include suitability of the recommendations to users. Suitability includes coverage, which measures the percentage of a dataset that the recommender system is able to provide predictions for; confidence metrics that can help users make more effective decisions; the learning rate, which measures how quickly an algorithm can produce good recommendations; and novelty/serendipity, which measure whether a recommendation is a novel possibility for a user. Finally, we explore measures of recommender system utility based on user satisfaction with and performance on a system.

5.1 Coverage

The coverage of a recommender system is a measure of the domain of items in the system over which the system can form predictions or make recommendations. Systems with lower coverage may be less valuable to users, since they will be limited in the decisions they are able to help with. Coverage is particularly important for the *Find All Good Items* task, since systems that cannot evaluate many of the items in the domain cannot find all of the good items in that domain. Coverage is also very important for the *Annotate In Context* task, as no annotation is possible for items where no prediction is available. Coverage can be most directly defined on predictions by asking "What percentage of items can this recommender form predictions for?" This type of coverage is often called prediction coverage. A different sort of coverage metric can be formed for recommendations, more along the lines of "What percentage of available items does this recommender ever recommend to users?" For an e-commerce site, the latter form of coverage measures how much of the merchant's catalog of items are recommended by the recommender; for this reason we'll call it catalog coverage.

Coverage has been measured by a number of researchers in the past [Good et al. 1999, Herlocker et al. 1999, Sarwar et al. 1998]. The most common measure for coverage has been the number of items for which predictions can be formed as a percentage of the total number of items. The easiest way to measure coverage of this type is to select a random sample of user/item pairs, ask for a prediction for each pair, and measure the percentage for which a prediction was provided. Much as precision and recall must be measured simultaneously, coverage must be measured in combination with accuracy, so recommenders are not tempted to raise coverage by making bogus predictions for every item.

An alternative way of computing coverage considers only coverage over items in which a user may have some interest. Coverage of this type is not usually measured over all items, but only over those items a user is known to have examined. For instance, when the predictive accuracy is computed by hiding a selection of ratings and having the recommender compute a prediction for those ratings, the coverage can be measured as the percentage of covered items for which a prediction can be formed. The advantage of this metric is it may correspond better to user needs, since it is not important whether a system can recommend items a user has no interest in. (For instance: if a user has no interest in particle physics, it is not a disadvantage that a particular recommender system for research papers cannot form predictions for her about particle physics.)

Catalog coverage, expressed as the percentage of the items in the catalog that are ever recommended to users, has been measured less often. Catalog coverage is usually measured on a set of recommendations formed at a single point in time. For instance, it might be measured by taking the union of the top 10 recommendations for each user in the population. Similarly to all coverage metrics, this metric distorts if it is not considered in combination with accuracy. For instance, if there is an item in the catalog that is uninteresting to all users, a good algorithm should never recommend it, leading to lower coverage -- but higher accuracy.

We know of no perfect, general coverage metric. Such a metric would have the following characteristics: (1) It would measure both prediction coverage and catalog coverage; (2) For prediction coverage it would more heavily weight items for which the user is likely to want predictions; (3) There would be a way to combine the coverage measure with accuracy measures to yield an overall "practical accuracy" measure for the recommender system. Recommender systems researchers must continue to work to develop coverage metrics with these properties. In the meantime, we should continue to use the best available metrics, and it is crucial that we continue to report the coverage of our recommender systems. Best practices are to report the raw percentage of items for which predictions can be formed, and to also report catalog coverage for recommender algorithms. Where practical, these metrics should be augmented with measures that more heavily weight likely items. These metrics should be considered experimental, but will eventually lead to more useful coverage metrics. Comparing recommenders along these dimensions will ensure that new recommenders are not achieving accuracy by "cherry-picking" easy-to-recommend items, but are providing a wide range of useful recommendations to users.

5.2 Learning Rate

Collaborative filtering recommender systems incorporate learning algorithms that operate on statistical models. As a result, their performance varies based on the amount of learning data available. As the quantity of learning data increases, the quality of the predictions or recommendations should increase. Different recommendation algorithms can reach “acceptable” quality of recommendations at different rates. Some algorithms may only need a few data points to start generating acceptable recommendations, while others may need extensive data points. Three different learning rates have been considered in recommender systems: overall learning rate, per item learning rate, and per user learning rate. The overall learning rate is recommendation quality as a function of the overall number of ratings in the system (or the overall number of users in the system). The per-item learning rate is the quality of predictions for an item as a function of the number of ratings available for that item. Similarly the per-user learning rate is the quality of the recommendations for a user as a function of the number of ratings that user has contributed.

The issue of evaluating the learning rates in recommender systems has not been extensively covered in the literature, although researchers such as Schein et al [2001] have looked at evaluating the performance of recommender systems in “cold-start” situations. “Cold-start” situations (commonly referred to as the startup problem) refer to situations where there are only a few ratings on which to base recommendations. Learning rates are non-linear and asymptotic (quality can’t improve forever), and thus it is challenging to represent them compactly. The most common method for comparing the learning rates of different algorithms is to graph the quality versus the number of ratings (quality is usually accuracy).

The lack of evaluation of learning rates is due largely to the size of the Eachmovie, MovieLens, and Jester datasets, all of which have a substantial number of ratings. As recommender systems spread into the more data-sparse domains, algorithm learning rates will become a much more significant evaluation factor.

5.3 Novelty and Serendipity

Some recommender systems produce recommendations that are highly accurate and have reasonable coverage -- and yet that are useless for practical purposes. For instance, a shopping cart recommender for a grocery store might suggest bananas to any shopper who has not yet selected them. Statistically this recommendation is highly accurate: almost everyone buys bananas. However, everyone who comes to a grocery store to shop has bought bananas in the past, and knows whether or not they want to purchase more. Further, grocery store managers already know that bananas are popular, and have already organized their store so people cannot avoid going past the bananas. Thus, most of the time the shopper has already made a concrete decision not to purchase bananas on this trip, and will therefore ignore a recommendation for bananas. Much more valuable would be a recommendation for the new frozen food the customer has never heard of -- but would love. A similar situation occurs in a music store around very well known items, like the Beatles' famous White Album. Every music aficionado knows about the White Album -- and most already own it. Those who do not own it already have likely made a conscious decision not to own it. A recommendation to purchase it is therefore unlikely to lead to a sale. In fact, the White Album is an even worse recommendation than bananas, since most people only buy one copy of any given album. Much more valuable would be a recommendation for an obscure garage band that makes music that this customer would love, but will never hear about through a review or television ad.

Bananas in a grocery store, and the White Album in a music store, are examples of recommendations that fail the obviousness test. Obvious recommendations have two disadvantages: first, customers who are interested in those products have already purchased them; and second, managers in stores do not need recommender systems to tell them which products are popular overall. They have already invested in organizing their store so those items are easily accessible to customers.

Obvious recommendations do have value for new users. Swearingen and Sinha [2001] found that users liked receiving some recommendations of items that they already were familiar with. This seems strange since such recommendations do not give users any new information. However, what they do accomplish is to increase user confidence in the system which is very important for the *Find Credible Recommender* task. Additionally, users were more likely to say they would buy familiar items than novel ones. This contrasts with the situation when users were asked about downloading material for free (e.g., as is the case for many technical papers on the Web or for many mp3 music files). Here, users tended to prefer more novel recommendations. The general lesson to take away is that a system may want to try to estimate the probability that a user will be familiar with an item. For some tasks (and perhaps early in the course of user's experience with the system), a greater number of familiar items should be recommended; for others, fewer or none should be included

We need new dimensions for analyzing recommender systems that consider the "non-obviousness" of the recommendation. One such dimension is **novelty**, which has been addressed before in information retrieval literature (see [Baeza-Yates and Ribiero-Neto 1999] for a brief discussion of novelty in information retrieval). Another related dimension is **serendipity**. A serendipitous recommendation helps the user find a surprisingly interesting item he might not have otherwise discovered. To provide a clear example of the difference between novelty and serendipity, consider a recommendation system that simply recommends movies that were directed by the user’s favorite director. If the system recommends a movie that the user wasn’t aware of, the movie will be novel, but probably not serendipitous. The user would have likely discovered that movie on their own. On the other hand, a recommender that recommends a movie by a new director is more likely to provide serendipitous recommendations. Recommendations that are serendipitous are by definition also novel. The distinction between novelty and serendipity is important when evaluating collaborative filtering recommender algorithms, because the potential for serendipitous recommendations is one facet of collaborative filtering that traditional content-based information filtering systems do not have. It is important to note that the term serendipity is sometimes incorrectly used in the literature when novelty is actually being discussed.

Several researchers have studied novelty and serendipity in the context of collaborative filtering systems [Sarwar et al. 2001]. They have modified their algorithms to capture serendipity by preferring to recommend items that are more preferred by a given user than by the population as a whole. A simple modification is to create a list of "obvious" recommendations, and remove the obvious ones from each recommendation list before presenting it to users. A disadvantage of this approach is that the list of obvious items might be different for each user, since each person has had different experiences in the past. An alternative would combine what is known about the user's tastes with what is known about the community's tastes. For instance, consider a hypothetical recommender that can produce a list of the probabilities for each item in the system that a given user will like the item. A naive recommender would recommend the top 10 items in the list -- but many of these items would be "obvious" to the customer. An alternative would be to divide each probability by the probability that an average member of the community would like the item, and re-sort by the ratio. Intuitively, each ratio represents the amount that the given user will like the product more than most other users. Very popular items will be recommended only if they are likely to be exceptionally interesting to the present user. Less popular items will often be recommended, if they are particularly interesting to the present user. This approach will dramatically change the set of recommendations made to each user, and can help users uncover surprising items that they like.

Designing metrics to measure serendipity is difficult, because serendipity is a measure of the degree to which the recommendations are presenting items that are both attractive to users and surprising to them. In fact, the usual methods for measuring quality are directly antithetical to serendipity. Using the items users have bought in the past as indicators of their interest, and covering items one by one to see if the algorithm can re-discover them, rewards algorithms that make the most obvious recommendations

A good serendipity metric would look at the way the recommendations are broadening the user's interests over time. To what extent are they for types of things she has never purchased before? How happy is she with the items recommended? (Does she return a higher percentage of them than other items?) Good novelty metrics would look more generally at how well a recommender system made the user aware of previously unknown items. To what extent does the user accept the new recommendations? We know of no systematic attempt to measure all of these facets of novelty and serendipity, and consider developing good metrics for novelty and serendipity an important open problem.

5.4 Confidence

Users of recommender systems often face a challenge in deciding how to interpret the recommendations along two often conflicting dimensions. The first dimension is the **strength** of the recommendation: how much does the recommender system think this user will like this item. The second dimension is the **confidence** of the recommendation: how sure is the recommender system that its recommendation is accurate. Many operators of recommender systems conflate these dimensions inaccurately: they assume that a user is more likely to like an item predicted five stars on a five star scale than an item predicted four stars on the same scale. That assumption is often false: very high predictions are often made on the basis of small amounts of data, with the prediction regressing to the mean over time as more data arrives. Of course, just because a prediction is lower does not mean it is made based on more data!

Another, broader, take on the importance of confidence derives from considering recommender systems as part of a decision-support system. The goal of the recommendation is to help users make the best possible decision about what to buy or use given their interests and goals. Different short-term goals can lead to different preferences for types of recommendations. For instance, a user selecting a research paper about agent programming might prefer a safe reliable paper that gives a complete overview of the area, or a risky, thought-provoking paper to stimulate new ideas. The same user might prefer the overview paper if she is looking for a paper to reference in a grant proposal, or the thought-provoking paper if she is looking for a paper to read with her graduate students. How can the recommender system help her understand which recommendation will fit her current needs?

To help users make effective decisions based on the recommendations, recommender systems must help users navigate along both the strength and confidence dimension simultaneously. Many different approaches have been used in practice. E-commerce systems often refuse to present recommendations that are based on datasets judged too small[6]. They want recommendations their customers can rely on. The Movie Critic system provided explicit confidence visualization with each recommendation: a target with an arrow in it. The closer the arrow was to the center, the more confident the recommendation. Herlocker et al. [2000] explored a wide range of different confidence displays, to study which ones influenced users to make the right decision. The study found that the choice of confidence display made a significant difference in users' decision-making. The best confidence displays were much better than no display. The worst displays actually worsened decision-making versus simply not displaying confidence at all.

Measuring the quality of confidence in a system is difficult, since confidence is itself a complex multidimensional phenomenon that does not lend itself to simple one-dimensional metrics. However, recommenders that do not include some measure of confidence are likely to lead to poorer decision-making by users than systems that do include confidence. If the confidence display shows users a quantitative or qualitative probability of how accurate the recommendation will be, the confidence can be tested against actual recommendations made to users. How much more accurate are the recommendations made with high confidence than those made with lower confidence? If the confidence display is directly supporting decisions, measuring the quality of the decisions made may be the best way to measure confidence. How much better is decision-making when users are shown a measure of confidence than when not?

5.5 User Evaluation

The metrics that we have discussed so far involve measuring variables that we believe will affect the utility of a recommender system to the user and affect the reaction of the user to the system. In this section we face the question of how to directly evaluate user “reaction” to a recommender system. The full space of user evaluation is considerably more complex than the space of the previously discussed metrics, so rather than examining specific metrics in detail, we broadly review the user evaluation space and past work in user evaluation of recommender systems. In order to better understand the space of user evaluation methods, we begin by proposing a set of evaluation dimensions. We use these dimensions to organize our discussion of prior work, showing the types of knowledge that can be gained through use of different methods. We close by summarizing what we consider the best current practices for user evaluations of recommender systems.

Dimensions for User Evaluation

- *Explicit (ask) vs. implicit (observe)*. A basic distinction is between evaluations that explicitly ask users about their reactions to a system and those that implicitly observe user behavior. The first type of evaluation typically employs survey and interview methods. The second type usually consists of logging user behavior, then subjecting it to various sorts of analyses.

[6] E-commerce managers will say that the first rule for a recommender system is "Don't make me look stupid!"

- Laboratory studies vs. field studies. Lab studies allow focused investigation of specific issues; they are good for testing well-defined hypotheses under controlled conditions. Field studies can reveal what users actually do in their own real contexts, showing common uses and usage patterns, problems and unmet needs, and issues that investigators may not have thought of to consider in a lab study. In particular, tasks such as *Evaluate Recommender* and *Express Self* may require field studies because user behavior may be highly context-sensitive.

- Outcome vs. process. For any task, appropriate metrics must be developed that define what counts as a successful outcome [Newman 1997]. From a systems perspective, accuracy may be the fundamental metric. From a user perspective, however, metrics must be defined relative to their particular tasks. For most tasks (such as *Find Good Items*) a successful outcome requires users to act on the system's recommendations, and actually purchase a book, rent a movie, or download a paper. However, to simply measure whether a goal is achieved is not sufficient. Systems may differ greatly in how efficiently users may complete their tasks. Such process factors as amount of time and effort required to complete basic tasks also must be measured to ensure that the cost of a successful outcome does not outweigh the benefit.

- Short-term vs. long-term. Some issues may not become apparent in a short-term study, particularly a lab study. For example, recall that Turpin & Hersh found that subjects were able to perform information retrieval tasks just as successfully with a less accurate search engine. However, if subjects continually had to read more summaries and sift through more off-topic information, perhaps they would grow dissatisfied, get discouraged, and eventually stop using the system.

We consider several studies to illustrate the use of these methods. Studies by Cosley et al, Swearingen & Sinha, Herlocker et al, and McDonald occupy roughly the same portion of the evaluation space. They are short-term lab studies that explicitly gather information from users. They all do some study of both task and process, although this dimension was not an explicit part of their analysis. Amento et al also did a short-term lab study, but it gathered both implicit and explicit user information and explicitly measured both task outcomes and process. Finally, Dahlen et al did a lab study that used offline analysis to "replay" the history of user interactions, defining and measuring implicit metrics of user participation over the long term.

Most recommender systems include predicted user ratings with the items they recommend. Cosley et al. [2003] conducted a lab study to investigate how these predicted ratings influence the actual ratings that users enter. They presented subjects with sets of movies they had rated in the past; in some cases, the predictions were identical to the subjects' past ratings, in some cases, they were higher, and in some they were lower. Cosley et al. found that the predicted ratings did influence user ratings, causing a small, but significant bias in the direction of the prediction. They also found that presenting inaccurate predictions reduced user satisfaction with the system. Thus, the methods used in this study yielded some evidence that users are sensitive to the predictive accuracy of the recommendations they receive.

Swearingen and Sinha [2002, 2001] carried out a study to investigate the perceived usefulness and usability of recommender systems. Subjects used either 3 movie recommenders or 3 book recommenders. They began by rating enough items for the system to be able to compute recommendations for them. They then looked through the recommendations, rating each one as useful and/or new, until they found at least one item they judged worth trying. Finally, they completed surveys and were interviewed by the experimenters.

The methods used in this experiment let the researchers uncover issues other than prediction accuracy that affected user satisfaction. For example, users must develop trust in a recommender system, and recommendations of familiar items supports this process. Explanations of why an item was recommended also helped users gain confidence in a system's recommendations. Users also face the problem of evaluating a system's recommendations – e.g., a movie title alone is insufficient to convince someone to go see it. Thus, the availability and quality of "supporting information" a system provided – e.g., synopses, reviews, videos or sound samples – was a significant factor in predicting how useful users rated the system. A final point shows that user reactions may have multiple aspects – satisfaction alone may be insufficient. For example, subjects liked Amazon more than MediaUnbound and were more willing to purchase from Amazon. However, MediaUnbound was rated as more useful, most likely to be used again, and as best understanding user tastes. Further analysis showed that Amazon's greater use of familiar recommendations may be the cause of this difference. The general point, however, is that for some purposes, users prefer one system, and for other purposes, the other.

Herlocker et al. [2000] carried out an in-depth exploration of explanations for recommender systems. After developing a conceptual model of the factors that could be used to generate an explanation, they empirically tested a number of different explanation types. They used traditional usability evaluation methods, discovering that users preferred explanations based on how a user's neighbors rated an item, the past performance of the recommender system, similarity of an item to other items the user rated highly, and the appearance of a favorite actor or actress. Specifically, they led users to increase their estimate of the probability that they would see a recommended movie.

McDonald [2001] conducted a controlled study of his Expertise Recommender system. This system was developed within a particular organizational context, and could suggest experts who were likely to be able to solve a particular problem and who were "socially close" to the person seeking help. The notable feature of McDonald's study was that subjects were given a rich scenario for evaluating recommendations, which specified a general topic area and a specific problem. In other words, the study was explicit in attempting to situate users in a task context that would lead them to evaluate the recommendations within that context.

Amento et al [1999, 2003] evaluated their "topic management" system, which lets users explore, evaluate, and organize collections of web sites. They compared their system to a Yahoo-style interface. They gathered independent expert ratings to serve as a site quality metric and used these ratings to define the task outcome metric as the number of high-quality sites subjects selected using each system. In addition, they measured task time and various effort metrics, such as the number of sites subjects browsed. Thus, a key feature of their methods was they were able to measure outcomes and process together; they found that users of their system achieved superior results with less time and effort. Appropriate metrics will vary between tasks and domains; however, often effort can be conceived in terms of the number of queries issued or the number of items for which detailed information is examined during the process of evaluating recommendations.

Dahlen et al [1998] studied the value of jump-starting a recommender system by including "dead data" – that is, ratings from users of another, inactive system. Their experimental procedure involved "replaying" the history of both systems, letting them evaluate the early experience of users in terms of their participation. This was the only course of action open, since it was impossible to survey users of the previous system. They found that early users of the "jump-started" system participated more extensively – they used the system more often, at shorter intervals, and over a long period of time, and they also entered more ratings. We believe that, in general, user contribution to and participation in recommender systems in the long term is quite important and relatively under-appreciated. And we believe that the metrics used in the jumpstarting study can be applied quite broadly.

To summarize our observations on user evaluation, we emphasize that accurate recommendations alone do not guarantee users of recommender systems an effective and satisfying experience. Instead, systems are useful to the extent that they help users complete their tasks. A fundamental point proceeds from this basis: to do an effective user evaluation of a recommender system, researchers must clearly define the tasks the system is intended to support[7]. Observations of actual use (if available) and interviews with (actual or prospective) users are appropriate techniques, since it often is the case that systems end up being used differently than the designers anticipated. Once tasks are defined, they can be used in several ways. First, they can be used to tailor algorithm performance. Second, clearly-defined tasks can increase the effectiveness of lab studies. Subjects can be assigned a set of tasks to complete, and various algorithms or interfaces can be tested to find out which ones lead to the best task outcomes.

We also recommend that evaluations combine explicit and implicit data collection whenever possible. This is important because user preferences and performance may diverge: users may prefer one system to another, even when their performance is the same on both, or vice versa. One advantage of gathering data about both performance and preferences is that the two can be correlated. (This is analogous to work on correlating implicit and explicit ratings [Claypool et al. 2001, Morita and Shinoda 1994]) Having done this, future evaluations that can gather only one of these types of data can have some estimate of what the other type of data would show.

6. CONCLUSION

Effective and meaningful evaluation of recommender systems is challenging. To date, there has been no published attempt to synthesize what is known about the evaluation of recommender systems, nor to systematically understand the implications of evaluating recommender systems for different tasks and different contexts. In this article, we have attempted to overview the factors that have been considered in evaluations as well as introduced new factors that we believe should be considered in evaluation. In addition, we have introduced empirical results on accuracy metrics that provide some initial insight into how results from different evaluation metrics might vary. Our hope is that this article will increase the awareness of potential biases in reported evaluations, increase the diversity of evaluation dimensions examined where it is necessary, and encourage the development of more standardized methods of evaluation.

[7] Whittaker, Terveen, and Nardi (2000) elaborate on this theme to develop a research agenda for Human-Computer Interaction centered around the notion of "reference tasks".

6.1 Future Work

While there are many open research problems in recommender systems, we find four evaluation-related problems to be particularly worthy of attention.

User sensitivity to algorithm accuracy. We know from recent work by Cosley et al. [2003] that user satisfaction is decreased when a significant level of error is introduced into a recommender system. The level of error introduced in that study, however, was many times larger than the differences between the best algorithms. Key questions deserving attention include: (a) for different metrics, what is the level of change needed before users notice or user behavior changes? (b) to which metrics are users most sensitive? (c) how does user sensitivity to accuracy depend on other factors such as the interface? (d) how do factors such as coverage and serendipity affect user satisfaction? If these questions are answered, it may be possible to build a predictive model of user satisfaction that would permit more extensive offline evaluation.

Algorithmic consistency across domains. While a few studies have looked at multiple datasets, no researchers have systematically compared a set of algorithms across a variety of different domains to understand the extent to which different domains are better served by different classes of algorithms. If such research did not find differences, it would simplify the evaluation of algorithms – system designers could select a dataset with the desired properties without needing domain-specific testing.

Comprehensive quality measures. Most metrics to date focus on accuracy, and ignore issues such as serendipity and coverage. There are well-known techniques by which algorithms can trade-off reduced serendipity and coverage for improved accuracy (such as only recommending items for which there are many ratings). Since users value all three attributes in many applications, these algorithms may be more accurate, but less useful. We need comprehensive quality measures that combine accuracy with other serendipity and coverage, so algorithm designers can make sensible trade-offs to serve users better.

Discovering the inherent variability in recommender datasets. We speculate above that algorithms trying to make better predictions on movie datasets may have reached the optimal level of error given human variability. Such variability can be explored using test-retest situations and analyses of taste change over time. If we can find effective ways to analyze datasets and learn the inherent variability, we can discover sooner when researchers have mined as much data as possible from a dataset, and thus when they should shift their attention from accuracy to other attributes of recommender systems.

7. ACKNOWLEDGEMENTS

We would like to express our appreciation to the present and past members of the GroupLens Research Group, our colleagues at AT&T Research, and our current students. We'd also like to thank our many colleagues in the recommender systems community with whom we've had fruitful discussions over the years. We'd also like to acknowledge the financial support to this project provided by the National Science Foundation under grants: DGE 95-54517, IIS 96-13960, IIS 97-34442, IIS 99-78717, IIS 01-02229, and IIS 01-33994, and by Net Perceptions, Inc.

8. REFERENCES

AGGARWAL, C. C., WOLF, J. L., WU, K.-L., YU, P. S. 1999. Horting Hatches an Egg: A New Graph-Theoretic Approach to Collaborative Filtering. In *Proceedings of ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*.

AMENTO, B., TERVEEN, L., HILL, W., HIX, D.,and SCHULMAN, R. 2003. Experiments in Social Data Mining. *To appear in ACM Transactions on Information Systems*.

AMENTO, B., TERVEEN, L., HIX, D., JU, P. 1999. An Empirical Evaluation of User Interfaces for Topic Management of Web Sites. In *Proceedings of the 1999 Conference on Human Factors in Computing Systems (CHI '99),* ACM Press, New York, NY. 552-559.

BAEZA-YATES, R., RIBIERO-NETO, B. 1999. ACM Press / Addison Wesley, New York.

BAILEY, B. P., GURAK, L. J., KONSTAN, J. A. 2001. An Examination of Trust Production in Computer-Mediated Exchange. In *Proceedings of the 7th Conference on Human Factors and the Web,* July 2001.

BALABANOVÍC, M., SHOHAM, Y. 1997. Fab: Content-Based, Collaborative Recommendation. *Communications of the ACM 40*, 66-72.

BASU, C., HIRSH, H., COHEN, W. W. 1998. Recommendation as Classification: Using Social and Content-Based Information in Recommendation. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98),* RICH, C. and MOSTOW, J., Eds. AAAI Press, Menlo Park, CA. 714-720.

BILLSUS, D., PAZZANI, M. J. 1998. Learning collaborative information filters. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98),* RICH, C. and MOSTOW, J., Eds. AAAI Press, Menlo Park, CA. 46-53.

BREESE, J. S., HECKERMAN, D., KADIE, C. 1998. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence (UAI-98),* COOPER, G. F. and MORAL, S., Eds. Morgan Kaufmann, San Francisco. 43-52.

CANNY, J. 2002. Collaborative Filtering with Privacy via Factor Analysis. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information retrieval,* ACM Press, New York, NY. 238-245.

CLAYPOOL, M., BROWN, D., LE, P.,and WASEDA, M. 2001. Inferring User Interest. *IEEE Internet Computing 5*, 32-39.

CLEVERDON, C., KEAN, M. 1968. *Factors Determining the Performance of Indexing Systems*.

COSLEY, D., LAM, S. K., ALBERT, I., KONSTAN, J. A.,and RIEDL, J. 2003. Is Seeing Believing? How Recommender Interfaces Affect Users' Opinions. *CHI Letters 5*.

DAHLEN, B. J., KONSTAN, J. A., HERLOCKER, J. L., GOOD, N., BORCHERS, A., and RIEDL, J. *Jump-starting movielens: User benefits of starting a collaborative filtering system with "dead data"*. TR 98-017. 1998. University of Minnesota.

DOMINGOS, P., RICHARDSON, M. 2003. Mining the Network Value of Customers. In *Proceedings of the Seventh International Conference on Knowledge Discovery and Data Mining,* ACM Press, New York, NY. 57-66.

GOLDBERG, D., NICHOLS, D., OKI, B. M.,and TERRY, D. 1992. Using Collaborative Filtering to Weave an Information Tapestry. *Communications of the ACM 35*, 61-70.

GOLDBERG, K., ROEDER, T., GUPTRA, D.,and PERKINS, C. 2001. Eigentaste: A Constant-Time Collaborative Filtering Algorithm. *Information Retrieval 4*, 133-151.

GOOD, N., SCHAFER, J. B., KONSTAN, J. A., BORCHERS, A., SARWAR, B. M., HERLOCKER, J. L., RIEDL, J. 1999. Combining collaborative filtering with personal agents for better recommendations. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI-99),* HENDLER, J. and SUBRAMANIAN, D., Eds. AAAI Press, Menlo Park. 439-446.

HANLEY, J. A., MCNEIL, B. J. 1982. The Meaning and Use of the Area under a Receiver Operating Characteristic (ROC) Curve. *Radiology 143*, 29-36.

HARMAN, D. 1995. The TREC Conferences. *Hypertext - Information Retrieval - Multimedia: Synergieeffekte Elektronisher Informationssysteme, Proceedings of HIM '95*.

HARTER, S. P. 1996. Variations in Relevance Assessments and the Measurement of Retrieval Effectiveness. *Journal of the American Society for Information Science 47*, 37-49.

HECKERMAN, D., CHICKERING, D. M., MEEK, C., ROUNTHWAITE, R.,and KADIE, C. 2000. Dependency Networks for Inference, Collaborative Filtering, and Data Visualization. *Journal of Machine Learning Research 1*, 49-75.

HELANDER, M. 1988. *Handbook of Human-Computer Interaction*. North Holland, Amsterdam.

HERLOCKER, J. L., KONSTAN, J. A., BORCHERS, A., RIEDL, J. 1999. An algorithmic framework for performing collaborative filtering. In *Proceedings of the 22nd International Conference on Research and Development in Information Retrieval (SIGIR '99),* Aug. 1999, HEARST, M. A., GEY, F., and TONG, R., Eds. ACM Press, New York. 230-237.

HERLOCKER, J. L., KONSTAN, J. A., RIEDL, J. 2000. Explaining Collaborative Filtering Recommendations. In *Proceedings of the 2000 Conference on Computer Supported Cooperative Work,* 241-250.

HERLOCKER, J. L., KONSTAN, J. A.,and RIEDL, J. 2002. An Empirical Analysis of Design Choices in Neighborhood-based Collaborative Filtering Algorithms. *Information Retrieval 5*, 287-310.

HILL, W., STEAD, L., ROSENSTEIN, M., FURNAS, G. W. 1995. Recommending and Evaluating Choices in a Virtual Community of Use. In *Proceedings of ACM CHI'95 Conference on Human Factors in Computing Systems,* ACM Press, 194-201.

KONSTAN, J. A., MILLER, B. N., MALTZ, D., HERLOCKER, J. L., GORDON, L. R.,and RIEDL, J. 1997. GroupLens: Applying Collaborative Filtering to Usenet news. *Communications of the ACM 40*, 77-87.

LE, C. T., LINDREN, B. R. 1995. Construction and Comparison of Two Receiver Operating Characteristics Curves Derived from the Same Samples. *Biom.J. 37*, 869-877.

LINTON, F., CHARRON, A.,and JOY, D. 1998. OWL: A Recommender System for Organziation-Wide Learning. *Proceedings of the 1998 Workshop on Recommender Systems* 65-69.

MCDONALD, D. W. 2001. Evaluating Expertise Recommendations. In *Proceedings of the ACM 2001 International Conference on Supporting Group Work (GROUP'01)*.

MCNEE, S., ALBERT, I., COSLEY, D., GOPALKRISHNAN, P., RASHID, A. M., KONSTAN, J. A., RIEDL, J. 2002. On the Recommending of Citations for Research Papers. In *Proceedings of ACM CSCW 2002*.

MILLER, B. N., ALBERT, I., LAM, S. K., KONSTAN, J. A., RIEDL, J. 2003. MovieLens Unplugged: Experiences with a Recommender Systems on Four Mobile Devices. In *Proceedings of the 2003 Conference on Intelligent User Interfaces*.

MILLER, B. N., RIEDL, J., KONSTAN, J. A. 1997. Experiences with GroupLens: Making Usenet Useful Again. In *Proceedings of the 1997 Usenix Technical Conference*.
MOBASHER, B., DAI, H., LUO, T., and NAKAGAWA, M. *Effective Personalization Based on Association Rule Discovery from Web Usage Data*. 2001. Proceedings of the 3rd ACM Workshop on Web Information and Data Management (WIDM01), held in conjunction with the International Conference on Information and Knowledge Management (CIKM 2001). 2001.
MORITA, M., SHINODA, Y. 1994. Information filtering based on user behavior analysis and best match text retrieval. In *Proceedings of SIGIR '94,* ACM, New York. 272-281.
MUI, L., ANG, C., and MOHTASHEMI, M. *A Probabilistic Model for Collaborative Sanctioning*. Technical Memorandum 617. 2001. MIT LCS.
NEWMAN, W. 1997. Better or just different? On the benefits of designing interactive systems in terms of critical parameters. In *Designing Interactive Systems (DIS97),* ACM Press, New York, NY. 239-246.
NIELSEN, J. 1994. *Academic Press.* Academic Press, San Diego, CA.
PENNOCK, D. M., HORVITZ, E., LAWRENCE, S., GILES, C. L. 2000. Collaborative Filtering by Personality Diagnosis: A Hybrid Memory-and Model-Based Approach. In *Proceedings of the Sixteenth Annual Conference on Uncertainty in Artificial Intelligence (UAI-2000),* Morgan Kaufmann, San Francisco. 473-480.
RASHID, A. M., ALBERT, I., COSLEY, D., LAM, S. K., MCNEE, S., KONSTAN, J. A., RIEDL, J. 2002. Getting to Know You: Learning New User Preferences in Recommender Systems. In *Proceedings of the 2002 Conference on Intelligent User Interfaces (IUI 2002),* 127-134.
REDDY, P. K., KITSUREGAWA, P., SREEKANTH, P., RAO, S. S. 2002. A Graph Based Approach to Extract a Neighborhood Customer Community for Collaborative Filtering. In *Lecture Notes in Computer Science - Databases in Networked Information Systems, Second International Workshop,* Springer, 188-200.
RESNICK, P., IACOVOU, N., SUCHAK, M., BERGSTROM, P., RIEDL, J. 1994. GroupLens: An open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 Conference on Computer Supported Collaborative Work,* FURUTA, R. and NEUWIRTH, C., Eds. ACM Press, New York. 175-186.
RESNICK, P., VARIAN, H. R. 1997. Recommender Systems. *Communications of the ACM 40*, 56-58.
ROGERS, S. C. 2001. *Marketing strategies, tactics, and techniques : a handbook for practitioners*. Quorum Books, Westport, Conn.
SARWAR, B. M., KARYPIS, G., KONSTAN, J. A., RIEDL, J. 2000a. Analysis of Recommendation Algorithms for E-Commerce. In *Proceedings of the 2nd ACM Conference on Electronic Commerce (EC'00),* ACM Press, New York. 285-295.
SARWAR, B. M., KARYPIS, G., KONSTAN, J. A., RIEDL, J. 2000b. Application of Dimensionality Reduction in Recommender System--A Case Study. In *ACM WebKDD 2000 Web Mining for E-Commerce Workshop*.
SARWAR, B. M., KARYPIS, G., KONSTAN, J. A., RIEDL, J. 2001. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th International World Wide Web Conference (WWW10)*.
SARWAR, B. M., KONSTAN, J. A., BORCHERS, A., HERLOCKER, J. L., MILLER, B. N., RIEDL, J. 1998. Using filtering agents to improve prediction quality in the grouplens research collaborative filtering system. In *Proceedings of the ACM 1998 Conference on Computer Supported Cooperative Work (CSCW '98),* ACM Press, New York.
SCHAFER, J. B., KONSTAN, J. A., RIEDL, J. 2002. Meta-recommendation Systems:User-controlled Integration of Diverse Recommendations. In *Proceedings of the 11th International Conference on Information and Knowledge Management,* Nov. 2002, 43-51.
SCHEIN, A. I., POPESCUL, A., UNGAR, L. H.,and PENNOCK, D. M. 2001. Generate Models for Cold-Start Recommendations. *2001 ACM SIGIR Workshop on Recommender Systems*.
SCHEIN, A. I., POPESCUL, A., UNGAR, L. H., PENNOCK, D. M. 2002. Methods and Metrics for Cold-Start Collaborative Filtering. In *Proceedings of the 25th Annual international ACM SIGIR Conference on Research and Development in Information Retrieval,* Aug. 2002, ACM Press, New York, NY.
SHARDANAND, U., MAES, P. 1995. Social Information Filtering: Algorithms for Automating "Word of Mouth". In *Proceedings of ACM CHI'95 Conference on Human Factors in Computing Systems,* ACM, New York. 210-217.
SINHA, R., SWEARINGEN, K. 2002. The Role of Transparency in Recommender Systems. In *CHI 2002 Conference Companion*.
SWEARINGEN, K., SINHA, R. 2001. Beyond Algorithms: An HCI Perspective on Recommender Systems. In *SIGIR 2001 Workshop on Recommender Systems*.
SWETS, J. A. 1963. Information Retrieval Systems. *Science 141*, 245-250.
SWETS, J. A. 1969. Effectiveness of Information Retrieval Methods. *American Documentation 20*, 72-89.
TURPIN, A., HERSH, W. 2001. Why Batch and User Evaluations Do Not Give the Same Results. In *Proceedings of the 24th Annual ACM SIGIR Conference on Research and Development in Information Retrieval,* 17-24.
VOORHEES, E. M., HARMAN, D. K. 1999. Overview of the seventh Text REtrieval Conference (TREC-7). In *NIST Special Publication 500-242,* July 1999, VOORHEES, E. M. and HARMAN, D. K., Eds. NIST, 1-24.
WEXELBLAT, A., MAES, P. 1999. Footprints: History-rich Tools for Information Foraging. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems,* WILLIAMS, M. G. and ALTOM, M. W., Eds. ACM Press, New York, NY. 270-277.
YAO, Y. Y. 1995. Measuring Retrieval Effectiveness Based on User Preference of Documents. *Journal of the American Society for Information Science 46*, 133-145.

Received January 2003, revised June 2003, and accepted September 2003.