Methods and Metrics for Cold-Start Recommendations

Andrew I. Schein, Alexandrin Popescul, and Lyle H. Ungar

University of Pennsylvania
Dept. of Computer Science
200 South 33rd St.
Philadelphia, PA 19104-6389 U.S.A.

{ais,popescul,ungar}@gradient.cis.upenn.edu

David M. Pennock

NEC Research Institute
4 Independence Way
Princeton, NJ 08540
USA

dpennock@research.nj.nec.com

General Terms

Algorithms, Experimentation, Performance

Categories and Subject Descriptors

H.1.m [**Information Systems**]: Models and Principals—*Miscellaneous*; I.2.6 [**Computing Methodologies**]: Artificial Intelligence—*Learning*; G.3 [**Mathematics of Computing**]: Probability And Statistics—*Contingency table analysis*

ABSTRACT

We have developed a method for recommending items that combines content and collaborative data under a single probabilistic framework. We benchmark our algorithm against a naïve Bayes classifier on the *cold-start* problem, where we wish to recommend items that no one in the community has yet rated. We systematically explore three testing methodologies using a publicly available data set, and explain how these methods apply to specific real-world applications. We advocate heuristic recommenders when benchmarking to give competent baseline performance. We introduce a new performance metric, the CROC curve, and demonstrate empirically that the various components of our testing strategy combine to obtain deeper understanding of the performance characteristics of recommender systems. Though the emphasis of our testing is on *cold-start* recommending, our methods for recommending and evaluation are general.

Keywords

Recommender systems, collaborative filtering, content-based filtering, information retrieval, graphical models, performance evaluation

SIGIR'02, August 11-15, 2002, Tampere, Finland.

1. INTRODUCTION

Recommender systems suggest items of interest to users based on their explicit and implicit preferences, the preferences of other users, and user and item attributes. For example, a movie recommender might combine explicit ratings data (e.g., Bob rates *Shrek* a 7 out of 10), implicit data (e.g., Mary purchased *The Natural*), user demographic information (e.g., Mary is female), and movie content information (e.g., *Scream* is marketed as a horror movie) to make recommendations to specific users.

Pure *collaborative filtering* methods [3, 12, 15, 23, 30] base their recommendations on community preferences (e.g., user ratings and purchase histories), ignoring user and item attributes (e.g., demographics and product descriptions). On the other hand, pure *content-based filtering* or *information filtering* methods [17, 24] typically match query words or other user data with item attribute information, ignoring data from other users. Several hybrid algorithms combine both techniques [1, 4, 6, 8, 21, 29]. Though "content" usually refers to descriptive words associated with an item, we use the term more generally to refer to any form of item attribute information including, for example, the list of actors in a movie.

One difficult, though common, problem for a recommender system is the *cold-start* problem, where recommendations are required for items that no one (in our data set) has yet rated.[1] Pure collaborative filtering cannot help in a cold-start setting, since no user preference information is available to form any basis for recommendations. However, content information can help bridge the gap from existing items to new items, by inferring similarities among them. Thus we can make recommendations for new items that appear similar to other recommended items. In this paper, we evaluate the performance of two machine learning algorithms on cold start prediction. We present our own probabilistic model that combines content and collaborative information

[1]The phrase *cold start* has also been used to describe the situation when almost nothing is known about customer preferences [9] (e.g., a start-up company has no little or no purchase history). The problem of making recommendations for new users can also be thought of as a cold-start problem. We concentrate on the new-item problem, although the new-user problem is symmetric when we have access to user attribute (e.g., demographic) data. The new-user problem without attribute data essentially falls within the framework of pure information filtering or information retrieval [24].

by using expectation maximization (EM) learning to fit the model to the data. We perform benchmarking on movie ratings data and compare against a naïve Bayes method that has also been proposed for this task [16].

Some key questions in evaluating recommender systems on testbed data are: what to predict, how to grade performance and what baseline to compare with. We identify three useful components to predict on our data set, and show where past work has focussed. In deciding what metric to use in evaluating performance, we have borrowed heavily from the literature in addition to developing our own tool: the CROC curve. For baseline measures of performance we advocate the use of heuristic recommenders: algorithms that are trivial to implement yet give performance that is well above random. We find that heuristic recommenders do surprisingly well: in some cases outperforming more sophisticated methods. Our testing goal is to uncover the most informative characterization of performance for our method and the naïve Bayes algorithm.

2. BACKGROUND AND RELATED WORK

Early recommender systems were pure collaborative filters that computed pairwise similarities among users and recommended items according to a similarity-weighted average [22, 30]. Breese et al. [3] refer to this class of algorithms as *memory-based* algorithms. Subsequent authors employed a variety of techniques for collaborative filtering, including hard-clustering users into classes [3], simultaneously hard-clustering users and items [31], soft-clustering users and items [14, 21], singular value decomposition [26], inferring item-item similarities [27], probabilistic modeling [3, 6, 10, 20, 21, 29], machine learning [1, 2, 18], and list-ranking [5, 7, 19]. More recently, authors have turned toward designing hybrid recommender systems that combine both collaborative and content information in various ways [1, 4, 6, 8, 21, 29]. To date, most comparisons among algorithms have been empirical or qualitative in nature [11, 25], though some worst-case performance bounds have been derived [7, 18], some general principles advocated [7], and some fundamental limitations explicated [19]. Techniques suggested in evaluating recommender system performance include mean average error, receiver operator characteristic (ROC) curves, ranked list metrics [3, 11] and variants of precision/recall statistics [25].

In this work we extend the hybrid recommender system of Popescul et al. [21] to average content data in a model based fashion. In evaluating our method we introduce novel testing strategies and metrics that can discover fine-grain characterization of performance leading to actionable conclusions.

3. THE TWO-WAY ASPECT MODEL

In predicting an association between person p and movie m, we employ a latent class variable framework called the aspect model that has been designed for contingency table smoothing [13]. Figure 1 (a) shows a graphical model description of the aspect model for a person/movie contingency table and Table 1 explains our notation used in the graphical model as well as in other descriptions of the movie recommendation task.

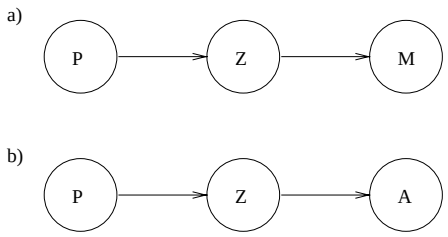

Figure 1: Graphical model of the person/movie aspect model (a) and the person/actor aspect model (b). These graphs can be interpretted precisely as belief networks.

Random Variable	Object	Interpretation
P	p	person
M	m	movie
A	a	actor
Z	z	latent class

Table 1: Notation used in our model descriptions.

3.1 Pure Collaborative Filtering Model

The aspect model of Figure 1 (a) encodes a probability distribution over each person/movie pair. Observations consist of tuples (p, m) recording that person p has seen/rated movie m. We store observations in a *count matrix* or *contingency table* with rows ranging over people and columns ranging over movies (or *vice versa*). Often our data may include multiple observations that are identical (e.g., Lyle saw *Memento* twice). With each observation we increment by one the count of the appropriate contingency table *cell* (or matrix entry). A naïve probability estimate for each cell is simply the observed frequency of events in that cell. However, notice that using this method of assigning probabilities, an empty cell implies that there is zero probability of the corresponding person seeing the corresponding movie, clearly an unrealistic inference.

An *aspect model* hypothesizes the existence of a hidden or *latent* cause z (e.g., an affinity for a particular style of movies) that motivates person p to watch movie m. According to the generative model semantics, person p chooses a latent class z, which in turn determines the movie m watched. The choice of movie m is assumed independent of p given knowledge of z. Since z is hidden, we sum over possible choices to define the distribution over (p, m):

$$\mathsf{P}(p,m) \quad = \quad \sum_z \mathsf{P}(p)\mathsf{P}(z|p)\mathsf{P}(m|z). \qquad (1)$$

Parameters $\mathsf{P}(z|p)$ and $\mathsf{P}(m|z)$ correspond to the processes of p stochastically choosing a latent class z, and z stochastically choosing m. The $\mathsf{P}(p, m)$ values can be thought of as smoothed estimates of the probability distribution of the contingency table. The latent variables perform the smoothing in a manner that maximizes the model likelihood (by keeping estimates of $\mathsf{P}(p, m)$ close to the empirical distribution). The model also creates smoothed estimates for the values $\mathsf{P}(p)$ and $\mathsf{P}(m)$, both taking their interpretations from contingency table analysis. The parameters are calculated using the tempered EM algorithm as described in [13].

We choose the number of latent classes using performance on a partition of training data as the criterion. Recommendations to person p are made using: $\mathsf{P}(m|p) \propto \mathsf{P}(p,m)$. Our own source code for fitting the two-way aspect model is available online [28]. We have found training to take only a few minutes using a Pentium II computer with 512 mb of RAM. Once trained, the method can generate a user's recommendations in a fraction of a second.

3.2 Adding Content Information

The recommender system described so far is a pure collaborative filtering algorithm developed by Hofmann and Puzicha [14]. We will not use the pure collaborative filtering aspect model since we wish to experiment on the cold start problem. The person/actor aspect model of Figure 1 (b) combines collaborative with content data in one model:

$$\mathsf{P}(p,a) \quad = \quad \sum_z \mathsf{P}(p)\mathsf{P}(z|p)\mathsf{P}(a|z). \tag{2}$$

In using this model, we hope that casts of actors can act as surrogates for movies. We recommend movies to a user based on how similar the cast is to movies the user has already rated. We generate a dataset from the collaborative filtering model by taking the collaborative observations (p,m) and creating a set of observations (p,a_i) for each actor i in movie m. These newly formed observations are no-longer independent, breaking an assumption of the aspect model. We have found in our own experiments that the person/actor aspect model has strengths to offset potential disadvantages of broken independence assumptions (for an example, see [21]).

3.3 Folding In

Notice that the person/actor aspect model does not have a movie object in the event space. In order to recommend a movie, we must create a new movie object out of the set of actors that appear in that movie. This pseudo-movie is then placed in the latent space based on the content information. We use Hofmann's [13] *folding-in* algorithm (originally used to fold term-queries into a document-word aspect model). For example, suppose we have fit a person/actor model and want to fold-in a new movie. We create a new set of parameters $\mathsf{P}(z|m)$ and use the actors in the movie $\{(a,m)\}$ as evidence for placing the movie in latent space in a manner that maximizes the likelihood of the movie. All of the original parameters from (2) are held constant during the process. The exact EM algorithm operates as follows:

E-Step:

$$\mathsf{P}(z|a,m) \propto \mathsf{P}(a|z)\mathsf{P}(z|m)$$

M-Step:

$$\mathsf{P}(z|m) \propto \sum_a n(a,m)\mathsf{P}(z|a,m)$$

Recommendations are made using:

$$\mathsf{P}(p|m) = \sum_z \mathsf{P}(p|z)\mathsf{P}(z|m)$$

If we desire an estimated value of $\mathsf{P}(p,m)$, we will first need to estimate $\mathsf{P}(m)$. We are currently experimenting with Bayesian weighting of movie-queries that computes an average of $\mathsf{P}(a)$ for the various actors in the movie.

4. NAIVE BAYES RECOMMENDER

As an alternative to the aspect model, we use the bag-of-words naïve Bayes text classifier developed in [17] applied to person/actor data. For each person a separate naïve Bayes classifier is trained so that no collaborative information is used. Hence, this is a pure content-based method capable of cold-start recommendation. The model is trained using Laplace smoothing.

Ratings prediction is computed with the probability function:

$$\mathsf{P}(c_j|M) \quad = \quad \frac{\mathsf{P}(c_j)}{\mathsf{P}(M)} \prod_{i=1}^{|M|} \mathsf{P}(a_i|c_j) \tag{3}$$

where the class c_j is a rating. Embedded in formula (3) is the belief that all movies in the data set are rated. In other words, the naïve Bayes generative model does not predict that an item will or won't be rated. We have tried to be consistent with the experiments of Melville et al. [16] who applied this classifier to a movie dataset similar to our own; we duplicate some of their testing strategies while adding our own.

5. TESTING METHODOLOGY

Our data primarily comes from the MovieLens data set assembled by the GroupLens project [11] consisting of actual ratings from a group of users. The core of this data set is a list of movie ratings grouped by person. Each person rates at least twenty movies. All MovieLens observations consist of a rating between 1 and 5 inclusive. In order to obtain actor and director information, we downloaded relevant pages from the Internet Movie Database (http://www.imdb.com). We speed up model fitting by considering only actors billed in the top ten and eliminating any actors who appear in only one movie.

In our experiments we randomly split the movies into a training set and a test set. In this manner we create a set of movies that have no observations in the training set. We test only using the 331 movies in the test set (out of 1682 total movies in the data set). There are 943 people we make recommendations for, with an average of 85 observations per person in the entire training set. There are 19,192 observed events for the test set movies out of a possible total of 312,133. 10,640 of the test set observations have rating of 4 or higher. All results reported below come from the same test set in order to facilitate comparison.

In testing various recommender systems on a static data set such as the MovieLens data, it is important to place test results in their proper context for those who may want to implement such systems. We have identified three modes of testing on our data set that correspond to different real-world applications. Statements about the performance of a recommender system should be based on what sort of recommendation task is simulated in testing. The differences in our testing modes are linked to the role of rating versus purchase data in the recommendation problem. We may wish to predict that a customer will purchase an item or that a customer will both purchase and like an item. A final task is to impute (guess) a customer's rating on an item that was purchased. These testing modes are elaborated below.

1. Implicit Rating Prediction

   Implicit rating prediction refers to prediction of data

such as purchase history; a purchase is not necessarily an indication of satisfaction, but a purchase is an indication of some implicit need or desire for an item. For MovieLens data we predict that a person has rated a movie, a task that is analogous to predicting a customer purchase. When evaluating performance, we do not consider the rating itself or events that occur in the training set (for domains where we don't recommend what a user already owns or has rated). implicit rating prediction is most appropriate for domains where explicit rating information is not available and we are satisfied to recommend products that the user is likely to purchase on their own. Past implicit evaluation work includes [21, 29]. We will refer to MovieLens observations stripped of a rating component as implicit rating data throughout this paper.

2. Rating Prediction

   In rating prediction we wish to predict both implicit rating and rating components of an observation simultaneously. In our MovieLens benchmarking we classify each person/movie pair that doesn't occur in the training observations into two groups:

   a) p_i rated $m_j \geq 4$.
   b) p_i did not rate $m_j \geq 4$.

   Condition b could imply that person i did not rate movie j at all.

3. Rating Imputation

   Rating imputation is prediction of ratings for items where we have implicit rating observations. In concrete terms, we ask "given that a person has seen movie x, how likely are they to rate it ≥ 4?" Our prior knowledge that the person has seen x means we have a implicit rating observation to this effect. Our goal is to guess the best rating. We implement rating imputation testing by taking held out observations from the MovieLens data and predicting ratings on this set.

   In real-world applications we may have data sets where implicit rating observations are available in large quantities, but the rating component is missing at random. Rating imputation measures success at filling in the missing values. Rating imputation has been used previously in [3, 11, 16] to evaluate recommender system performance.

6. EVALUATION METRICS

Before deploying an actual recommender system we would need to decide how often to recommend and to whom. For instance, in one application we may need to recommend k products to each customer in a database, where k is a number we may choose according to a desired false-positive rate. In other applications we may be permitted to recommend a greater number of products to customers who we understand better. In this section, we identify metrics that measure success in either mode of recommending and show in the results section how using both metrics on the same data uncovers fine grain recommender system characteristics.

Herlocker et al. [11] suggest using receiver operator characteristic (ROC) curves as one measure to evaluate recommender systems. We call the method of Herlocker et al. a global ROC (GROC) curve in order to distinguish it from the customer ROC (CROC) variation presented shortly. A ROC curve is a curve showing hit/miss rates for different classification thresholds. Currently, machine learning researchers use ROC curves to evaluate binary classification algorithms. We will employ ROC curves in this context (e.g., "our algorithm says a person *will* have rated a movie" is a predicted *positive* outcome). The area under a ROC curve is a performance measure with perfect performance indicated by area of one and random guessing indicated by area 0.5. ROC curves are nearly equivalent to precision/recall curves in the information retrieval community and lift curves of the marketing literature. The terms precision and recall are analogous to specificity and sensitivity respectively, where the latter terms are found on ROC curves. Sensitivity (or hit rate) is equal to the percentage of all positive values found above some threshold. 1 - Specificity (or miss rate) is equal to the fraction of all negative values found above some threshold. Miss rate and hit rate are plotted on the x and y axis respectively. Varying the threshold generates a series of points forming a *curve*. We will not actually connect the points of our plots to form the curve in order to make the figures more readable.

We use a global ROC (GROC) curve to measure performance when we are allowed to recommend more often to some users than others. GROC curves are constructed in the following manner:

```
1.  Order the predictions pred(p_i, m_j) in a list
by magnitude, imposing an ordering:  (p, m)_k.
2.  Pick n, calculate hit/miss rates caused by
predicting the top n (p, m)_k by magnitude, and
plot the point.
```

By selecting different n (e.g. incrementing n by a fixed amount) we draw a curve on the graph.

Customer ROC (CROC) curves measure performance of a recommender system when we are constrained to recommend the same number of items to each user. Unlike the GROC curve, the CROC curve is not a special case of the ROC curve, though it is constructed in an analogous manner:

```
1.  For each person p_i, order the predictions
pred(p_i, m_j) in a list by magnitude imposing an
ordering:  (m)_k.
2.  Pick n, calculate global hit/miss rates
caused by recommending the top predicted n
movies to each person and plot the point.
```

We vary n as in the GROC case.

In a GROC curve, the perfect recommender will generate a curve with area one, but for the CROC curve this is not the case. To see why, imagine using an omniscient recommender on a data set with three people: person a sees four movies, person b sees two movies, and person c sees six movies. When we recommend four movies to each person, we end up with two false-positives from person b, lowering the area of the curve. However, for any particular data set, we can plot the curve and calculate the area of the omniscient recommender in order to facilitate comparison.

Whether using GROC or CROC curves it is important to focus attention on the left hand side of the graph. This is the portion of the graph focusing on a low false positive rate. In

most applications of recommender systems it is desirable to obtain the best possible performance in the low false positive region of the curve.

7. RESULTS

We test the person/actor aspect model against the naïve Bayes method of [17]. We conduct implicit rating and rating imputation since these two methods are most commonly used to evaluate the two recommender systems in the previous literature [16, 29]. Our preliminary experiments with naïve Bayes confirmed earlier observations [16] that naïve Bayes is sensitive to sparsity below 40 rated movies on this type of data (results not shown). In order to better duplicate the prior work in rating imputation [16] as well as present naïve Bayes in the best possible light, we perform rating imputation only for users who have rated 40 or more movies in the training set. We train the models separately according to precedent: the aspect model is trained on implicit rating data, while the naïve Bayes method is trained on the corresponding ratings data.

Implicit in the GROC and CROC curve is the performance of a random recommender: a 45 degree line with area 0.5 underneath. We feel that in many applications a superior baseline can be developed. For instance, we can recommend first to users that on average rate movies higher in order to obtain better-than-random rating imputation GROC performance. We call such recommendation algorithms *heuristic recommenders* since they use simple tricks to obtain above-random performance. Where possible we propose and include heuristic recommenders in the analysis.

7.1 Implicit Rating Testing

Figure 2 shows GROC and CROC plots comparing the person/actor aspect model, the naïve Bayes recommender and (for the GROC case) a heuristic recommender on the implicit rating prediction task. The heuristic recommender for the GROC plot is created by substituting the total number of movies seen by person i for the recommender output for pair: (p_i, m_j). In other words we list users by their relative activity in rating on the MovieLens site and recommend *all* movies to these users in that order. Note that the heuristic recommender performs nearly as well as the aspect model in the region of interest (the far left-hand side of the graph).

Figure 2 (b) shows CROC plots comparing the person/actor aspect and naïve Bayes methods on the implicit rating prediction task without a heuristic recommender. There is no obvious heuristic recommender to use here other than random recommendation due to the cold-start problem. Note that both machine learning methods perform noticeably better than the area 0.5 indicative of random prediction by both CROC and GROC metrics.

7.2 Rating Imputation Testing

Figure 3 shows GROC and CROC plots comparing the person/actor aspect model, the naïve Bayes recommender and (for the GROC case) a heuristic recommender on the rating imputation task. Here we predict that a user rates a movie 4 or higher. This technique of evaluation has been called ROC-4 [25]. The heuristic recommender is created by using the mean rating for each person i as the predicted rating for (p_i, m_j), for all m_j. Surprisingly, the mean rating method outperforms all other methods by GROC standards. As in the implicit rating prediction task, there is no obvious heuristic recommender to deploy in the CROC plot other than the implicit random recommender with area 0.5.

8. DISCUSSION

The aspect model performs better than the naïve Bayes method on the implicit rating prediction task, but the reverse is true on the rating imputation task. This is not surprising since the aspect model is trained solely on implicit rating data, and the naïve Bayes recommender is not designed for implicit rating prediction. Both methods need to be altered in order to optimize performance on the alternative test. Re-designing the aspect model training and test procedure for rating imputation and rating prediction will be a subject of future work. The current experiments test the degree to which implicit rating observations are sufficient for prediction in the rating imputation task by benchmarking the aspect model against naïve Bayes. In addition we test whether rating classification recommenders as embodied by the naïve Bayes method can predict implicit rating observations as well as our proposed aspect model.

Our empirical evaluation methods point out some interesting subtleties that occur in benchmarking recommender systems. On both GROC plots, some very simple methods are competitive with the more sophisticated aspect model and naïve Bayes methods. In the rating imputation case, the mean rating of a user is the single best predictor for rating imputation according to the GROC criteria. Likewise, the number of movies a person has rated is a very good method on the implicit rating prediction GROC plot. The use of intelligent *heuristic recommenders* like the ones described above are a key ingredient to interpreting the performance of a GROC curve. Past performance results using the MovieLens and similar datasets that employ the GROC curve including [11, 16, 29] should be evaluated with this observation in mind.

A natural question to ask is whether both the GROC and CROC curve are needed in evaluation. One hypothesis is that once a suitable heuristic recommender is chosen, relative performance on a GROC curve graph is sufficient to tell the whole story of recommender system performance characteristics. Our results show that this is not the case; in particular our empirical results demonstrate that GROC and CROC curves can often have no predictive power over each other: e.g., they often give conflicting measures. To see why, compare the relative performance of the user activity recommender and the aspect model recommender on the implicit rating prediction task (Figure 2). On the GROC plot, the two methods have nearly identical characteristics. If GROC and CROC metrics were necessarily correlated, we would expect similar behavior in the CROC graph for the implicit rating prediction task. Note that the CROC graph of the implicit rating prediction task does not include the user activity recommender because this method produces perfectly random performance (area 0.5) by CROC standards. In contrast, the aspect model performance has area 0.64: well above random. The GROC and CROC graphs together point out that the aspect model has nearly identical global (GROC) performance to the heuristic recommender while actually recommending to a more diverse group of people. A similar situation is visible in the rating imputation GROC and CROC plots.

Let's say we are deciding between the heuristic recommender and the aspect model for implicit rating prediction.

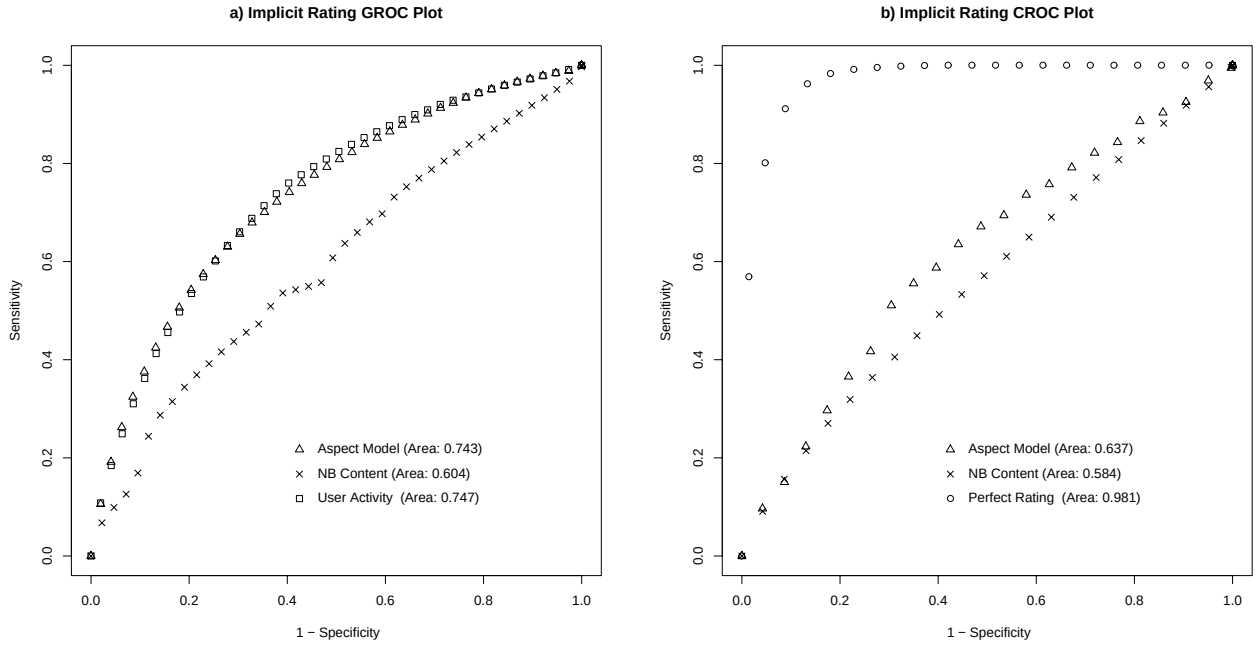

Figure 2: GROC (a) and CROC (b) plots comparing person/actor aspect model, naïve Bayes and a heuristic recommender on the implicit rating prediction task. To generate GROC plot points we increment the number of predictions by 7800. To generate CROC plot points we increment the number of movies predicted for each person by 15.

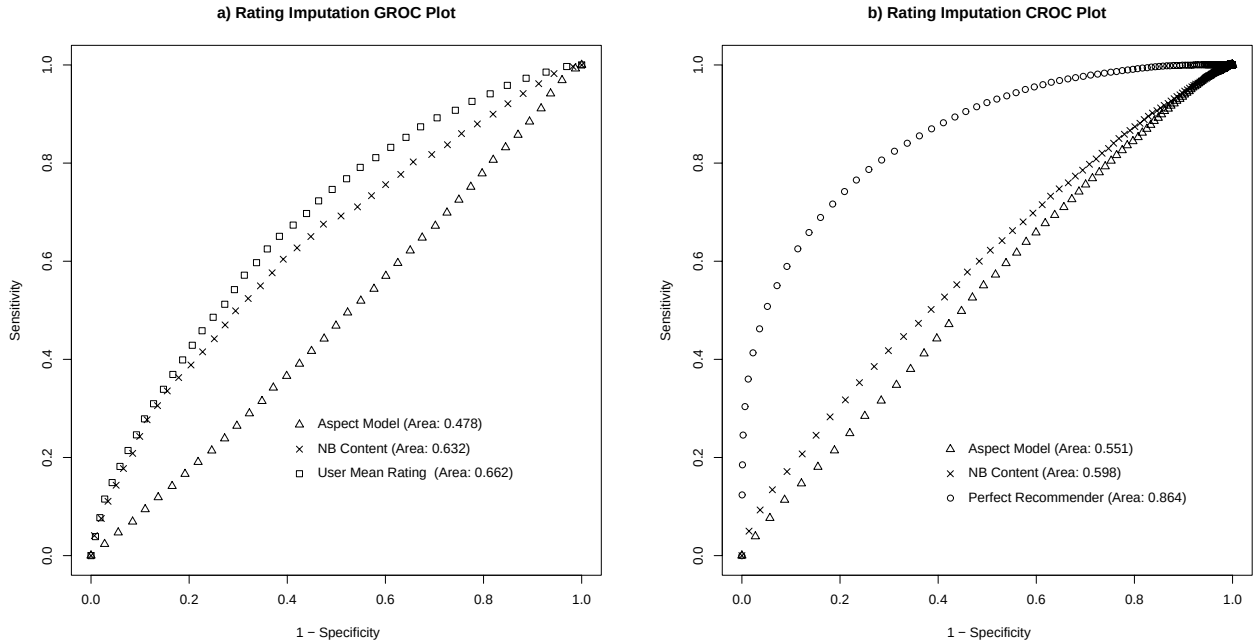

Figure 3: GROC (a) and CROC (b) plots comparing person/actor aspect model, naïve Bayes and a heuristic recommender on the rating imputation task. To generate GROC plot points we increment the number of predictions by 420. To generate CROC plot points we increment the number of movies predicted for each person by 1.

If greater customer coverage is a priority, then we will choose the aspect model based on its CROC plot, even though the two methods perform equivalently on the GROC plot. We will then return to the GROC plot to determine what decision threshold to use based on false positive rate (calculated from miss rate on a GROC plot).

9. CONCLUSIONS

We have proposed the aspect model latent variable method for cold-start recommending. Our aspect model combines both collaborative and content information in model fitting. The folding in algorithm allows us to make predictions for unrated items by using content data: averaging the set of content data (e.g. actors) that associate with an item (e.g. movie). Our method is tested against naïve Bayes and several heuristic recommenders on publicly available movie rating data. We have tested these methods on implicit rating and rating imputation tasks while evaluating performance under two different methods of recommending embodied by GROC and CROC curve metrics.

A surprising outcome of the empirical evaluation is the performance of so-called heuristic recommenders on the GROC curves. These methods are so effective that we feel that all future work using GROC curve plots should include appropriate heuristic predictors in order to give perspective on actual gain in performance caused by using sophisticated machine learning or other techniques. In many cases it is feasible to include heuristic recommenders for CROC curves as well. For instance we could use average rating of a movie in the training set as our rating prediction. However, when operating from a cold start we do not have this sort of information. Fortunately, GROC and CROC curves each have built-in baselines: the random recommender with area 0.5.

The combination of GROC/CROC testing and heuristic recommenders tells us when it makes sense to deploy the aspect model on the cold-start task. On the implicit rating prediction task, we recommend the aspect model over the heuristic recommender for applications when it is important to recommend to as many users as possible. If real-world success is embodied exactly by the GROC curve, we feel that the heuristic recommender is up to the job and easiest to implement. On the rating imputation task, we draw the same conclusion regarding the effectiveness of naïve Bayes method over the corresponding heuristic recommender.

Our results demonstrate the value of using both GROC and CROC curves in evaluating recommender system performance. We can show when our aspect model recommender outperforms and underperforms alternatives and link testing performance to real-world problems. GROC and CROC curves each measure performance in a specific real world task. However, in situations where the evaluation goal is to characterize the performance of a recommender system generally, both GROC and CROC curves are needed since they are often conflicting measures of recommender system performance.

10. ACKNOWLEDGMENTS

Andrew Schein was supported by NIH Training Grant in Computational Genomics, T-32-HG00046. Alexandrin Popescul was supported in part by a grant from the NEC Research Institute.

11. REFERENCES

[1] C. Basu, H. Hirsh, and W. Cohen. Recommendation as classification: Using social and content-based information in recommendation. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, pages 714–720, 1998.

[2] D. Billsus and M. J. Pazzani. Learning collaborative information filters. In *Proceedings of the Fifteenth International Conference on Machine Learning*, pages 46–54, 1998.

[3] J. S. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, pages 43–52, 1998.

[4] M. Claypool, A. Gokhale, and T. Miranda. Combining content-based and collaborative filters in an online newspaper. In *Proceedings of the ACM SIGIR Workshop on Recommender Systems—Implementation and Evaluation*, 1999.

[5] W. W. Cohen, R. E. Schapire, and Y. Singer. Learning to order things. *Journal of Artificial Intelligence Research*, 10:243–270, 1999.

[6] M. K. Condliff, D. D. Lewis, D. Madigan, and C. Posse. Bayesian mixed-effect models for recommender systems. In *ACM SIGIR '99 Workshop on Recommender Systems: Algorithms and Evaluation*, 1999.

[7] Y. Freund, R. Iyer, R. E. Schapire, and Y. Singer. An efficient boosting algorithm for combining preferences. In *Proceedings of the Fifteenth International Conference on Machine Learning*, pages 170–178, 1998.

[8] N. Good, J. B. Schafer, J. A. Konstan, A. Borchers, B. M. Sarwar, J. L. Herlocker, and J. Riedl. Combining collaborative filtering with personal agents for better recommendations. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence*, pages 439–446, 1999.

[9] H. Guo. Soap: Live recommendations through social agents. In *Fifth DELOS Workshop on Filtering and Collaborative Filtering, Budapest*, 1997.

[10] D. Heckerman, D. M. Chickering, C. Meek, R. Rounthwaite, and C. Kadie. Dependency networks for collaborative filtering and data visualization. In *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence*, pages 264–273, 2000.

[11] J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl. An algorithmic framework for performing collaborative filtering. In *Proceedings of the Conference on Research and Development in Information Retrieval*, 1999.

[12] W. Hill, L. Stead, M. Rosenstein, and G. Furnas. Recommending and evaluating choices in a virtual community of use. In *Proceedings of the Conference on Human Factors in Computing Systems*, pages 194–201, 1995.

[13] T. Hofmann. Probabilistic latent semantic indexing. In *22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 50–57, 1999.

[14] T. Hofmann and J. Puzicha. Latent class models for collaborative filtering. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, pages 688–693, 1999.

[15] J. A. Konstan, B. N. Miller, D. Maltz, J. L. Herlocker, L. R. Gordon, and J. Riedl. GroupLens: Applying collaborative filtering to Usenet news. *Communications of the ACM*, 40(3):77–87, 1997.

[16] P. Melville, R. J. Mooney, and R. Nagarajan. Content-boosted collaborative filtering. In *Proceedings of the 2001 SIGIR Workshop on Recommender Systems*, 2001.

[17] R. J. Mooney and L. Roy. Content-based book recommending using learning for text categorization. In *Proceedings of the Fifth ACM Conference on Digital Libraries*, pages 195–204, 2000.

[18] A. Nakamura and N. Abe. Collaborative filtering using weighted majority prediction algorithms. In *Proceedings of the Fifteenth International Conference on Machine Learning*, pages 395–403, 1998.

[19] D. M. Pennock, E. Horvitz, and C. L. Giles. Social choice theory and recommender systems: Analysis of the axiomatic foundations of collaborative filtering. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence*, pages 729–734, 2000.

[20] D. M. Pennock, E. Horvitz, S. Lawrence, and C. L. Giles. Collaborative filtering by personality diagnosis: A hybrid memory- and model-based approach. In *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence*, pages 473–480, 2000.

[21] A. Popescul, L. H. Ungar, D. M. Pennock, and S. Lawrence. Probabilistic models for unified collaborative and content-based recommendation in sparse-data environments. In *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence*, 2001.

[22] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. GroupLens: An open architecture for collaborative filtering of netnews. In *Proceedings of the ACM Conference on Computer Supported Cooperative Work*, pages 175–186, 1994.

[23] P. Resnick and H. R. Varian. Recommender systems. *Communications of the ACM*, 40(3):56–58, 1997.

[24] G. Salton and M. McGill. *Introduction to Modern Information Retrieval*. McGraw Hill, 1983.

[25] B. M. Sarwar, G. Karypis, J. A. Konstan, and J. T. Riedl. Analysis of recommender algorithms for e-commerce. In *Proceedings of the 2nd ACM Conference on Electronic Commerce*, pages 158–167, 2000.

[26] B. M. Sarwar, G. Karypis, J. A. Konstan, and J. T. Riedl. Application of dimensionality reduction in recommender system—A case study. In *ACM WebKDD Web Mining for E-Commerce Workshop*, 2000.

[27] B. M. Sarwar, G. Karypis, J. A. Konstan, and J. T. Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the Tenth International World Wide Web Conference*, pages 285–295, 2001.

[28] A. I. Schein, A. Popescul, and L. H. Ungar. PennAspect: A two-way aspect model implementation. Technical Report MS-CIS-01-25, Department of Computer and Information Science, The University of Pennsylvania.

[29] A. I. Schein, A. Popescul, L. H. Ungar, and D. M.Pennock. Generative models for cold-start recommendations. In *Proceedings of the 2001 SIGIR Workshop on Recommender Systems*, 2001.

[30] U. Shardanand and P. Maes. Social information filtering: Algorithms for automating 'word of mouth'. In *Proceedings of the Conference on Human Factors in Computing Systems*, pages 210–217, 1995.

[31] L. H. Ungar and D. P. Foster. Clustering methods for collaborative filtering. In *Workshop on Recommendation Systems at the Fifteenth National Conference on Artificial Intelligence*, 1998.