Generative Models 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}@seas.upenn.edu

David M. Pennock

NEC Research Institute
4 Independence Way
Princeton, NJ 08540
USA

dpennock@research.nj.nec.com

ABSTRACT

Systems for automatically recommending items (e.g., movies, products, or information) to users are becoming increasingly important in e-commerce applications, digital libraries, and other domains where mass personalization is highly valued. Such *recommender systems* typically base their suggestions on (1) collaborative data encoding which users like which items, and/or (2) content data describing item features and user demographics. Systems that rely solely on collaborative data fail when operating from a *cold start*—that is, when recommending items (e.g., first-run movies) that no member of the community has yet seen. We develop several generative probabilistic models that circumvent the cold-start problem by mixing content data with collaborative data in a sound statistical manner. We evaluate the algorithms using MovieLens movie ratings data, augmented with actor and director information from the Internet Movie Database. We find that maximum likelihood learning with the expectation maximization (EM) algorithm and variants tends to overfit complex models that are initialized randomly. However, by seeding parameters of the complex models with parameters learned in simpler models, we obtain greatly improved performance. We explore both methods that exploit a single type of content data (e.g., actors only) and methods that leverage multiple types of content data (e.g., both actors and directors) simultaneously.

Keywords

Recommender systems, collaborative filtering, automated recommendation, cold start problem, content-based filtering, information retrieval, generative probabilistic models, expectation maximization

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, 18, 25, 32] 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 [19, 26] 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, 7, 9, 23]. 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, either explicitly or implicitly.[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 recommendations can be made for new items that appear similar to other recommended items. In this paper, we explore several generative probabilistic models that combine content and collaborative information by using expectation maximization (EM) learning to

[*]In Proceedings of the SIGIR-2001 Workshop on Recommender Systems. New Orleans, LA.

[1]The phrase *cold start* has also been used to describe the situation when almost nothing is known about customer preferences [10] (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 [26].

fit the model to the data. These models exploit collaborative data from old items (unlike pure information filtering methods), yet are also able to make informed predictions when operating from a cold start (unlike pure collaborative filtering methods).

In Section 3, we introduce the precise models that we are considering for the cold-start recommendation problem. In Section 4, we describe a technique for seeding the parameters of complex models with values learned from simpler models, resulting in dramatically improved recommendations. We discuss the pros and cons of several evaluation metrics in Section 5, and we look at performance results comparing the models in Section 6. In Section 7, we conclude with observations about these experiments and ideas about future directions.

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 [24, 32]. 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 [33], soft-clustering users and items [17, 23], singular value decomposition [28], inferring item-item similarities [29], probabilistic modeling [3, 7, 12, 22, 23], machine learning [1, 2, 20], and list-ranking [5, 8, 21]. More recently, authors have turned toward designing hybrid recommender systems that combine both collaborative and content information in various ways [1, 4, 7, 9, 23]. To date, most comparisons among algorithms have been empirical or qualitative in nature [13, 27], though some worst-case performance bounds have been derived [8, 20], some general principles advocated [8], and some fundamental limitations explicated [21].

Our algorithms utilize a set of generative probabilistic models that generalize Hofmann and Puzicha's [16, 17] two-way aspect models, and extend our previous work on three-way aspect models [23]. Similar techniques are used in natural language processing applications under the name of aggregate Markov models [30], and in social science applications (e.g., analyzing contingency tables) under the names of structural equation models, modified path models, and log-linear models with latent variables [11]. Our methods build most closely on Hofmann and Puzicha's model-based clustering definitions [16, 17], Hofmann's folding-in algorithm [15], and Cohn and Hofmann's event-space combination models [6].

Many recommender systems implicitly make a *closed-world assumption* that all of the items that exist in the world are observed at least once during training. In the aspect model, this assumption manifests itself in the probability estimates for the training-set contingency table that sum to one (ruling out the existence of other items). In other words, there is no probability mass left over for a new item that is released to the public after building the contingency table estimates. This mistaken assumption contributes to the cold-start problem of modeling items that we have no collaborative data for. Fortunately, the probabilistic interpretation of the aspect model allows us to use attribute information in a principled manner to make good cold-start recommendations.

Random Variable	Object	Interpretation
P	p	person
M	m	movie
D	d	director
A	a	actor
Z	z	latent class
T	t	event type

Table 1: Notation used in our model descriptions.

3. THE MODELS

We employ a class of *generative probabilistic models*, or models that encode probability distributions that generate or reconstruct the observed data, in a sense describing or explaining the data. Model parameters are learned using expectation maximization (EM) in an attempt to find the model most likely to explain the data.

Our models are mainly extensions and generalizations of *aspect models* [15, 16, 17]. Such models are useful for soft-partitioning data, naturally incorporating the ability to, for example, soft-cluster people by movies (collaborative filtering) or people by content attributes like actors (hybrid collaborative/content-based filtering). We divide the models into three categories: (1) Hofmann and Puzicha's [16, 17] original two-way aspect models, (2) three-way aspect models [23] that extend the two-way aspect model to incorporate a third observed component of an event, and (3) mixed event-space models (similar to [6]) that capture generative processes with multiple types of events. Section 3.1 describes the two-way aspect models. Section 3.2 explains how we can make recommendations for items that are not explicitly part of the model, by creating pseudo-items. Sections 3.3 and 3.4 present our three-way aspect models and mixed event-space models, respectively. Table 1 summarizes some of the notation used in this section and throughout the paper, while Figure 1 gives graphical description of the various models we will employ.

3.1 Two-Way Aspect Models

Our first four models, denoted M1 through M4, are models for two-way co-occurence data. For example, consider collaborative data in a movie domain, where observations consist of tuples (p, m) recording that person p has seen 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.

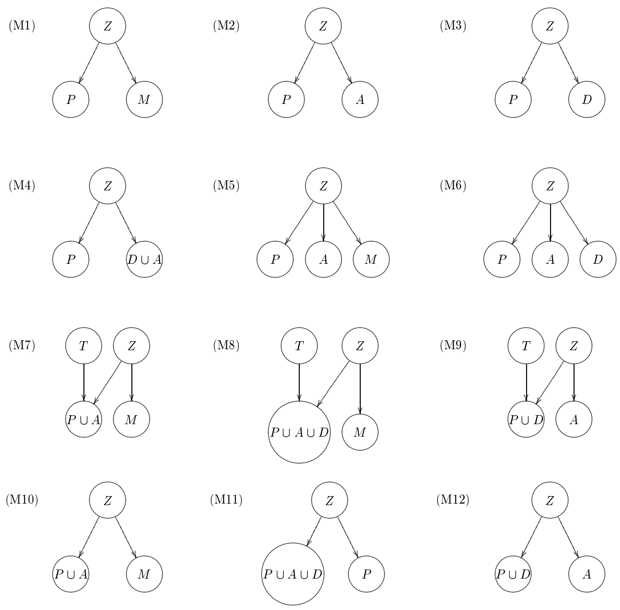

Figure 1: Bayesian networks of generative models for recommending movies.

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):

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

Parameters $\mathsf{P}(z|p)$ and $\mathsf{P}(m|z)$ correspond to the Markov 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.

Equation 1 is an asymmetric formulation of the aspect model. Using Bayes rule, we can re-write the equation in an equivalent symmetric form:

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

The generative interpretation of Equation 2 is that the latent class z is chosen according to $\mathsf{P}(z)$, then p and m are chosen independently according to $\mathsf{P}(p|z)$ and $\mathsf{P}(m|z)$, respectively. Equation 2, which we refer to as model M1, is drawn as a Bayesian belief network in Figure 1.

For model M1, the likelihood of the observed data given the model is:

$$\mathsf{P}(\mathrm{Data}|\Theta) = \prod_{p,m} \mathsf{P}(p, m)^{n(p,m)}, \quad (3)$$

where Θ is the set of model parameters, and $n(p, m)$ is the number of times person p watched movie m (i.e., the value of the cell (p, m) in the contingency table). The model parameters are chosen in an attempt to maximize the likelihood (3) using EM, an iterative method that increases (3) at each step until a local maximum is reached. Each iteration consists of two steps: and expectation step (E-step) and a maximization step (M-step). In this case, the two steps operate as follows:

E-Step

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

M-Step:

$$\mathsf{P}(p|z) \propto \sum_m n(p, m)\mathsf{P}(z|p, m)$$

$$\mathsf{P}(m|z) \propto \sum_p n(p, m)\mathsf{P}(z|p, m)$$

$$\mathsf{P}(z) \propto \sum_{p,m} n(p, m)\mathsf{P}(z|p, m)$$

Our specific implementation uses a simulated-annealing-type variant of EM called *tempered EM* [15]. Our own source code [31] for fitting the two-way aspect model is available upon request. We have found that this algorithm converges in fewer then 100 iterations on most of our data sets. Once the model is fit, we can recommend movies to a person p by ranking movies that the person has not yet seen according to the formula $\mathsf{P}(m|p) \propto \mathsf{P}(p, m)$.

Models M2 and M3 have the same structure as M1, substituting actors and directors, respectively, for movies. In model M4, we pool actors and directors into a single "content" category, so that the dimensionality of the contingency tables remains two. Models M2 through M4 are pictured as Bayesian networks in Figure 1. In the cases of models M2 through M4, movies are recommended using a *folding-in* technique described in the next section.

3.2 Folding In

Notice that models M2 through M4 do not have a movie object, as they are primarily content-based methods. In order to recommend a movie, we must create a new movie object out of the set of actors (or directors) that appear in that movie. This pseudo-movie is then placed in the latent space based on the content information. We use Hofmann's [15] *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 M2 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 M2 are held constant during the process. The exact EM algorithm operates as follows [14]:

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.

3.3 Three-Way Aspect Models

Models M5 and M6 are three-way aspect models, depicted as Bayesian networks in Figure 1. An observation consists of a triple, for example (p,m,a) in model M5. Three-way models correspond to three-way contingency tables (in the the case of M5, containing people, movies, and actors). As in two-way contingency tables, we have empirical estimates for the contingency table probabilities that are calculated by counting. However, by hypothesizing the existence of latent variables z, we can smooth these estimates to infer the likelihood of events that have not occurred. The three-way model extends the two-way model by assuming all three observable objects (people, movies, and actors) are distributed independently given the latent class z. The exact distribution over observations is

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

giving our data set the likelihood

$$\mathsf{P}(\mathrm{Data}|\Theta) = \prod_{p,m,a} \mathsf{P}(p,m,a)^{n(p,m,a)}.$$

The corresponding generative process chooses a class z, then chooses a person, a movie, and an actor independently. The EM model-fitting procedure for this class of models is similar to the two-way model, and is described in detail in our earlier work [23].

3.4 Mixed Event-Space Models

Our final class of models are *mixed event-space models* that encode a generative process consisting of multiple types of events. Our formulation is similar to the definitions of Cohn and Hofmann [6], as we will elaborate shortly. An example of a two-space model is M7 (pictured in Figure 1) which has a movie/actor space denoted $\mathcal{A}$ and a movie/person space denoted $\mathcal{P}$. A random variable T determines the type of event (t) that will be generated. In the generative process we pick z and t independently. The movie is picked according to distribution $\mathsf{P}(m|z)$. Assuming $t = \mathcal{P}$, a person is chosen using $\mathsf{P}(p|z,\mathcal{P})$. If $t = \mathcal{A}$, the probability of generating a person is zero. A symmetric generative process exists for the $\mathcal{A}$ space of events. We say that an event of type $\mathcal{P}$ occurs with probability β and an $\mathcal{A}$ event occurs with probability $1-\beta$. The maximum likelihood estimate for β is calculated by counting events in the training data, turning this value into a constant that disappears from the model-fitting algorithm.

The likelihood of our data set is:

$$\begin{aligned} \mathsf{P}(\mathrm{Data}|\Theta) &= \prod_{p,m\in\mathcal{P}} (\beta\mathsf{P}(p,m|\mathcal{P}))^{n(p,m)} \\ &\quad \prod_{a,m\in\mathcal{A}} ((1-\beta)\mathsf{P}(a,m|\mathcal{A}))^{n(a,m)} \end{aligned} \tag{4}$$

where

$$\begin{aligned} \mathsf{P}(p,m|\mathcal{P}) &= \sum_z \mathsf{P}(z)\mathsf{P}(p|z,\mathcal{P})\mathsf{P}(m|z) \\ \mathsf{P}(a,m|\mathcal{A}) &= \sum_z \mathsf{P}(z)\mathsf{P}(a|z,\mathcal{A})\mathsf{P}(m|z) \end{aligned}$$

The EM algorithm for estimating the model parameters is:
E-Step:

$$\begin{aligned} \mathsf{P}(z|p,m) &\propto \mathsf{P}(z)\mathsf{P}(p|z,\mathcal{P})\mathsf{P}(m|z) \\ \mathsf{P}(z|a,m) &\propto \mathsf{P}(z)\mathsf{P}(a|z,\mathcal{A})\mathsf{P}(m|z) \end{aligned}$$

M-Step:

$$\begin{aligned} \mathsf{P}(z) &\propto \sum_{p,m} n(p,m)\mathsf{P}(z|p,m) \\ &\quad + \sum_{a,m} n(a,m)\mathsf{P}(z|a,m) \\ \mathsf{P}(a|z,\mathcal{A}) &\propto \sum_m n(a,m)\mathsf{P}(z|a,m) \\ \mathsf{P}(p|z,\mathcal{P}) &\propto \sum_m n(p,m)\mathsf{P}(z|p,m) \\ \mathsf{P}(m|z) &\propto \sum_p n(p,m)\mathsf{P}(z|p,m) \\ &\quad + \sum_a n(a,m)\mathsf{P}(z|a,m) \end{aligned}$$

Model M7 is closely related to the joint probabilistic model of Cohn and Hofmann. In the likelihood function below, we take their model and substitute movies for documents, people for citations, and actors for words:

$$\begin{aligned} \mathsf{P}(\mathrm{Data}|\Theta) &= \prod_{p,m} \mathsf{P}(p|m,\mathcal{P})^{\frac{\alpha\, n(p,m)}{\sum_{p'} n(p',m)}} \\ &\quad \prod_{a,m} \mathsf{P}(a|m,\mathcal{A})^{\frac{(1-\alpha)n(a,m)}{\sum_{a'} n(a',m)}} \quad \text{where} \\ \mathsf{P}(p|m,\mathcal{P}) &= \sum_z \mathsf{P}(p|z)\mathsf{P}(z|m) \quad \text{and} \\ \mathsf{P}(a|m,\mathcal{A}) &= \sum_z \mathsf{P}(a|z)\mathsf{P}(z|m). \end{aligned} \tag{5}$$

The main difference between equations (4) and (5) is that the former is symmetric while the latter is asymmetric. This distinction is roughly analogous to the difference between Equations (1) and (2). Additionally, $\mathsf{P}(m)$ does not appear in Cohn and Hofmann's model. If we choose to give a generative interpretation to their model, the absence of the $\mathsf{P}(m)$ parameters corresponds to a uniform distribution over movies. The uniform constants $\mathsf{P}(m)$ can be pulled out of the equation since they do not effect the likelihood surface. A uniform distribution makes sense only if we want each movie to be weighted equally in its contribution to the likelihood, ignoring the unequal number of observations associated with different movies. Note that Cohn and Hofmann do not include the β and $(1-\beta)$ parameters in their formulation. However, if we assume β to be any fixed constant in (0,1), we can pull β and $1-\beta$ out of the likelihood formula for (4) without affecting the maximum likelihood estimates for the model parameters. The observation weights α and $(1-\alpha)$ from (5) do not appear in our model, and could make a significant difference in the likelihood surface. Our parameter seeding techniques (described below in Section 4) give an alternative (but less general) method for weighting observation types. When α is set to 0.5 and $\mathsf{P}(m)$ is constrained to be uniform in M7, the two models have equivalent likelihood functions.

Model M8 is a natural extension of M7 that accommodates both actors and directors. We define mixing proportions $\beta_1, \beta_2, \beta_3$ such that $\sum_{i=1}^{3} \beta_i = 1$ and a class of events $\mathcal{D}$ having the form (d, m). The likelihood of the data is

$$\mathsf{P}(\text{Data}|\Theta) = \prod_{p,m \in \mathcal{P}} (\beta_1 \mathsf{P}(p, m|\mathcal{P}))^{n(p,m)} \prod_{a,m \in \mathcal{A}} (\beta_2 \mathsf{P}(a, m|\mathcal{A}))^{n(a,m)} \prod_{d,m \in \mathcal{D}} (\beta_3 \mathsf{P}(d, m|\mathcal{D}))^{n(d,m)} . \quad (6)$$

The EM algorithm for M8 is analogous to that of M7.

In order to make a prediction for a new movie m and person p, we fold-in the content just as for M2. The use of both actors and directors requires a variant of the folding-in EM algorithm:
E-Step

$$\mathsf{P}(z|a, m) \propto \mathsf{P}(a|z, \mathcal{A})\mathsf{P}(z|m)$$
$$\mathsf{P}(z|d, m) \propto \mathsf{P}(d|z, \mathcal{D})\mathsf{P}(z|m)$$

M-Step

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

We elaborate on mixed event-space model, M13, in Section 4 below.

3.5 Mixed Event-Space Models as Two-Way Models

In models M7 through M9 the random variable T can be eliminated to create an ordinary two-way aspect model similar in structure to M4. Probability estimates related to T exist implicitly within the newly created two-way model, and can be calculated explicitly if necessary. This equivalence suggests that models M7 though M9 (and also Cohn and Hofmann's model [6] under certain conditions) may be unnecessarily complex. We define models M10 through M12, pictured in Figure 1, to test this hypothesis.

Our insights regarding the structure of mixed event-space models put Cohn and Hofmann's observation weighting in a new perspective. In a two-way model, when one object is constrained to have uniform probability, we can weight certain subclasses of the second object differently to maximize an objective function. In intuitive terms, the weighting corresponds to relaxing our desire to fit the empirical distribution of a contingency table for certain rows (or columns) in order to obtain a better fit on more important rows (or columns). Open research problems include (1) generalizing the observation-weighting technique to models without a uniform distribution constraint, and (2) determining when weighting is beneficial.

4. PARAMETER SEEDING FOR BETTER PERFORMANCE

Preliminary model-fitting revealed that the models M7 and M8 over-fit on the first iterations of EM. We measure overfitting in this instance by checking log-likelihood of held-out data. Though tempered EM helps, the overall recommendation performance suffers from a lack of good parameter estimates. We found that parameter-seeding can greatly improve performance. We use the fitted parameters from the various two-way models and fold-in additional objects to obtain a richer model. To optimize model M7 performance, we take fitted parameters from M1, hold them constant, and fold-in the actors based on actor/movie co-occurrence. For model M8 we perform the same steps required for M7, but fold-in the directors (based on movie co-occurrence) in addition to the actors. We create a new model M13 where we take fitted parameters from M2 and fold-in the directors based on actor co-occurrence.

We did not find the over-fitting effect when training M10, and so this model is fit without parameter seeding.

5. PERFORMANCE CRITERIA

We propose three different performance metrics for evaluating the effectiveness of our models: the R value, the GROC curve, and the CROC curve. There are many ways to think about performance in a recommender system, and hence a need for multiple metrics. A good metric should tell us several things. For instance, we want to know how well a recommender does in comparison to a divine or omniscient recommender. We also want a sense of how well a recommender performs in the high specificity area: customers will become impatient with a recommender system that makes many bad recommendations.

One useful metric is Breese et al.'s [3] *rank scoring metric R*:

$$R = 100 \frac{\sum_u R_u}{\sum_u R_u^{max}} \text{ where } R_u = \sum_j \frac{\delta(u, j)}{2^{(j-1)/(\alpha-1)}}$$

where the δ function is the indicator function informing whether person u saw the movie ranked j and α is a decay constant used to emphasize the performance of the highest ranked recommendations. R_u^{max} is the R value of the omniscient recommender for user u—that is, an R value of 100 corresponds with perfect accuracy.

A receiver operating characteristic (ROC) curve is a plot showing true-positive/false-positive trade-offs for different classification thresholds. These curves were originally designed for evaluating performance in deciphering radar signals, but have found common usage in the medical literature for comparing diagnosis techniques. ROC curves are essentially equivalent to precision/recall curves in the information retrieval community [26] and lift curves of in the Marketing literature. The terms precision and recall are analogous to specificity and sensitivity respectively, where the latter terms are found on ROC curves. Herlocker at al. [13] recommend the use of ROC curves as one measure to evaluate recommender systems.

An alternative approach to the R value for measuring recommendation performance is an ROC curve based on global rankings and outcomes of the (p_i, m_j). We call this a Global ROC (GROC) curve, and the metric we compare is area under the curve for the specificity region of interest. The omniscient recommender has an ROC curve with area one under the curve. From the point of view of implementing a recommender system, a GROC curve is ideal when we intend to make the recommendations we are most certain are correct to anyone in our data set. In particular, we may choose not to recommend anything to person p if we have low confidence in recommendations for this person. On the other hand, we are willing to recommend more often to people for whom we can recommend with greater confidence.

Obviously, the area under the GROC curve is not the right metric for applications where we desire to recommend some movies to everybody. Imagine a scenario where we are considering whether to recommend 5, 10, or k movies to each of our customers. At each selection of k, we can plot the global sensitivity and specificity on an ROC curve. We call such a curve a Customer ROC (CROC) curve. Unfortunately, the omniscient recommender does not give any fixed maximal area such as an area of one. 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.

6. RESULTS

6.1 Benchmark Data

Our data primarily comes from the MovieLens data set assembled by the GroupLens project [13]. In order to obtain actor and director information, we web-crawled the Internet Movie Database (http://www.imdb.com). We speed up model-fitting by considering only actors billed in the top ten and throwing out any actors who appear in only one movie.

The data forms a boolean matrix pm_{ij} where a '1' indicates that p_i rated m_j on the MovieLens web site. We call any pm_{ij} with a '1' an *observation*. We take this to mean that p_i has been 'observed' to watch m_j. Entries with '0' indicate no observation. In our experiments we randomly split the movies into a training set and a test set, and then evaluated performance based on our ability to predict the test set boolean matrix pm^t_{ij}. We test only using the 665 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 106 observations per person in the entire data set boolean matrix. There are 41,379 observed events for the test set movies out of a possible total of 627,095. All results reported below come from the same test set in order to facilitate comparison.

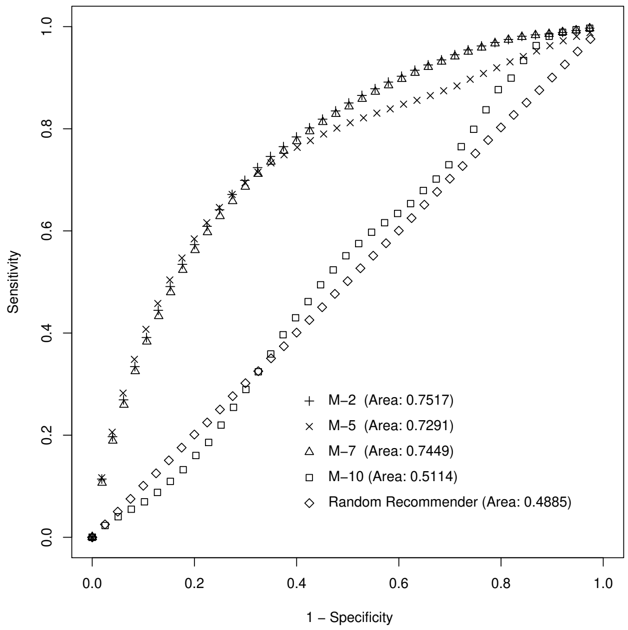

Figure 2: A GROC curve comparing naive guessing, M2, M5, M7, and M10 on the test set. Points are drawn after making $0, 15677, 31354, \ldots$ recommendations (out of 627,095 possible).

Approximately sixty percent of the movies in our data set are the debut of a director. That is, according to our data, these are the only movies that these directors have directed. We want to be able to make recommendations for all movies in our test set, and so we choose actors as the first piece of evidence to consider. Virtually all movies in the test set have actors that have appeared in the training set. After we obtain baseline accuracy measurements for predictions based on actor data, we combine director data with actor data with hopes of improving overall recommendation performance.

6.2 Cold-Start Recommendations with Actor Information

In order to assess the efficacy of using actor information for making cold-start recommendations we fitted models M2, M5, M7, and M10. GROC curves (Figure 2) and CROC curves (Figure 3) demonstrate that we can obtain performance that is well-above random. For instance, the CROC curve tells us that using recommender M5, we can recommend 40 movies to each person and expect about 10% of the test set observations to be correctly predicted with a false-positive rate of 0.57. In contrast, the random recommender will correctly predict only 5.9% percent of the test set observations with false-positive rate 0.60. The ideal recommender can predict 61% of the test set observations with a false-positive rate of 0.02 when recommending 40 movies to each person.

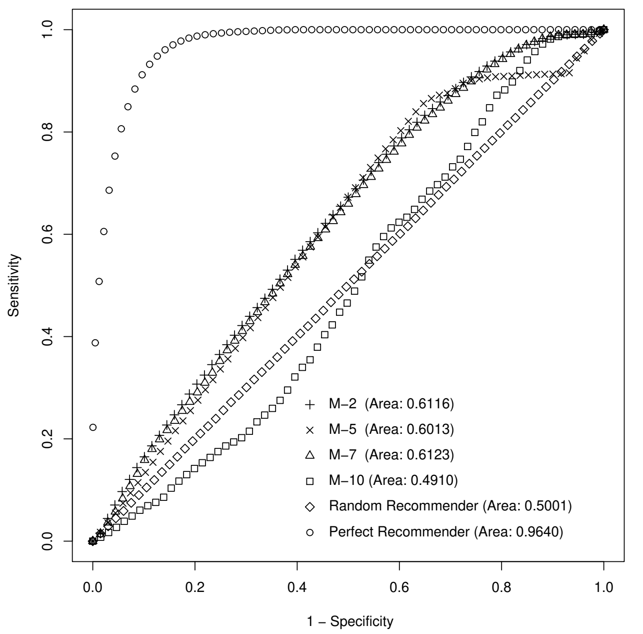

Figure 3: A CROC curve comparing naive guessing, an omniscient recommender, M2, M5, M7, and M10 on the test set. Points are drawn after recommending 0,10,20,... items to each person in the data set.

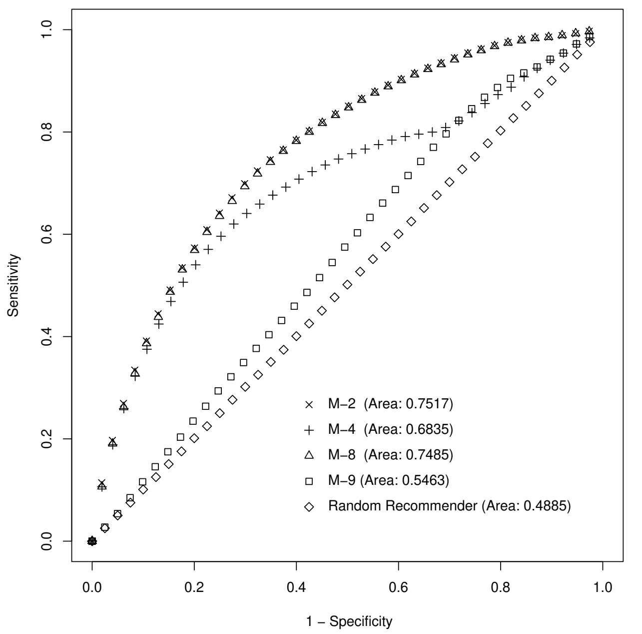

Figure 4: A GROC curve comparing naive guessing, M2, M4, M8, and M9 on the test set. Points are drawn after making $0, 15677, 31354, \ldots$ recommendations (out of 627,095 possible).

Model	M2	M4	M5	M7	M8	M9	M10	Ran.
R val.	7.4	8.2	7.8	5.9	5.0	4.6	3.7	7.0

Table 2: The ranked scoring (R) values for the various models including the random recommender. The decay term α was set to 5.0

In examining the GROC and CROC curves, we see some interesting characteristics of the models. For example, we can see that solid performance on the left side (low false positive (fp) side) of the graph does not lead to good performance on the right side (high sensitivity side). M5 is an example of a model that gives leading GROC performance in the low fp region, but performs less impressively if we desire very high sensitivity. It is our opinion that recommender systems are best evaluated in the low fp region of the ROC curve, and so we feel that M5 gives the strongest performance by the GROC criteria. In the CROC curve, M2 slightly outperforms its rivals in the low-fp region. As a general-purpose recommender, we prefer M2; M2 is a simple model that performs competitively by both GROC and CROC standards.

6.3 Using Directors to Improve Recommendations

We attempted to improve recommendations in Figures 2 and 3 by adding director information. We note that about 80% of the observations in the test data are for movies directed by people who direct more than once. Therefore, the additional information provided by directors should improve performance somewhat. The results in Figures 4 and 5 are mixed. We include M2 as a baseline model. R values for all models benchmarked are shown in Table 2.

The GROC curve performance of the models that combine actor and director evidence are disappointing: none of the models considered here outperform M2. Comparing M8 area (0.7485) to M7 results in Figure 2 we obtain only a marginal improvement over M7's 0.7449 area. In the GROC criteria, we can recommend mostly to the people with large numbers of observations in the training set (these are people with large estimates for $\mathsf{P}(p)$). Under these circumstances the director information helps little or not at all. On the other hand, the CROC performance for model M4 in Figure 5 is noticeably better than M2 in the low false-positive region. We interpret this difference between GROC and CROC performance to mean that director information can help for people with fewer observations (lower $\mathsf{P}(p)$ estimates).

7. CONCLUSIONS AND FUTURE WORK

We proposed several probabilistic models for use as recommender systems while addressing the cold-start problem, in particular. Our benchmark results demonstrate that cold-start recommendation of new items is a feasible enterprise in the presence of adequate content information. In the movie recommendation domain, actors can provide enough information to recommend a movie that nobody has seen before. In order to recommend a new movie based on its content, we need to place the movie into a previously fitted model. The folding-in method finds the coordinates $\mathsf{P}(z|m)$ in latent space for movie m by maximizing the likelihood of the new movie's content information.

Our benchmark analysis relies on three different measures of performance: R values, GROC curves, and CROC curves. We have noted that recommender systems can be applied in

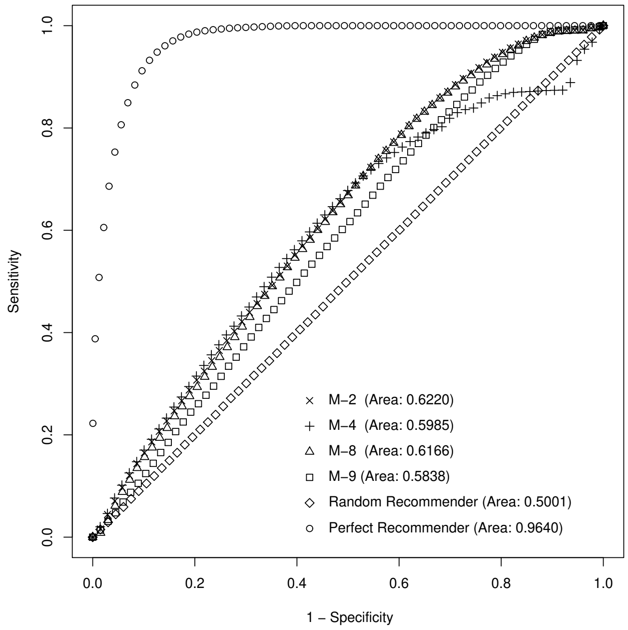

Figure 5: A CROC curve comparing naive guessing, an omniscient recommender, M2, M4, M8, and M9 on the test set. Points are drawn after recommending 0,10,20,... items to each person in the data set.

various different settings, and so we adapt a standard technique (the ROC curve) to measure two specific performance criteria that are relevant to recommender system deployment. In the CROC case we intend to recommend k movies to each customer in the database, while in the GROC case we intend to make the k recommendations we are most certain of. Our benchmarking results suggests that three-way aspect model M5 is a good choice for optimizing GROC criteria, while two-way aspect M2 will perform better by CROC standards. The difference in GROC and CROC definitions are helpful in analyzing the value of combining multiple forms of item-attribute information, such as actor and director data. We see that by GROC standards, directors do not help our models fitted with actor information. The CROC curves tell a different story; director data leads to a marginal improvement in recommender performance. The difference between GROC and CROC results suggest that auxiliary content data can improve recommendations for the customers with less purchase/rating history, and therefore lower estimates of $\mathsf{P}(p)$.

Our calculations and experiments with mixed event-space models illustrate the relationship between the standard two-way model, the event-space combination models of Cohn and Hofmann, and our own mixed event-space models. Our mixed event-space models over-fit in the first EM iterations unless we i) seed the models with the parameters of simpler models or ii) remove the constraining random variable T and reformulate the mixed event-space model as a two-way aspect model. Our best performance with mixed event-space models occurs with M7 where we seed the parameters from the purely collaborative filtering model M1. This may at first seem surprising, since we can fit the likelihood of competitor M10 directly through EM. A comparison of M10 and M7 benchmarking reveals that the joint probabilistic models do not have the right likelihood function for recommender applications; the seeding technique used for M7 overcomes this limitation. In other application domains, M10's likelihood may be a better optimization objective function.

8. 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. Software and hardware donated by Turbolinux, Inc. was helpful in these investigations.

9. 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] D. Cohn and T. Hofmann. The missing link—A probabilistic model of document content and hypertext connectivity. In *Advances in Neural Information Processing Systems*, volume 13. The MIT Press, 2001.

[7] 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.

[8] 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.

[9] 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.

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

[11] J. A. Hagenaars. *Loglinear Models with Latent Variables*. Quantitative Applications in the Social Sciences. Sage Publications, Inc., 1993.

[12] 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.

[13] 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.

[14] T. Hofmann. Personal communication.

[15] T. Hofmann. Probabilistic latent semantic analysis. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, pages 289–296, 1999.

[16] T. Hofmann and J. Puzicha. Statistical models for co-occurrence data. Technical report, Aritificial Intelligence Laboratory and Center for Biological and Computational Learning, MIT, 1998.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] 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, to appear*, 2001.

[24] 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.

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

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

[27] 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.

[28] 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.

[29] 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.

[30] L. Saul and F. Pereira. Aggregate and mixed-order Markov models for statistical language processing. In C. Cardie and R. Weischedel, editors, *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing*, pages 81–89. Association for Computational Linguistics, Somerset, New Jersey, 1997.

[31] 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. In Progress.

[32] U. Shardanand and P. Maes. Social information filtering: Algorithms for automating 'word of mouth'. In *Proceedings of Computer Human Interaction*, pages 210–217, 1995.

[33] 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.