Learning Preferences of New Users in Recommender Systems: An Information Theoretic Approach

Al Mamunur Rashid
Department of Computer Science & Engineering, University of Minnesota
Minneapolis, MN-55455
arashid@cs.umn.edu

George Karypis
Department of Computer Science & Engineering, University of Minnesota
Minneapolis, MN-55455
karypis@cs.umn.edu

John Riedl
Department of Computer Science & Engineering, University of Minnesota
Minneapolis, MN-55455
riedl@cs.umn.edu

ABSTRACT

Recommender systems are an effective tool to help find items of interest from an overwhelming number of available items. Collaborative Filtering (CF), the best known technology for recommender systems, is based on the idea that a set of like-minded users can help each other find useful information. A new user poses a challenge to CF recommenders, since the system has no knowledge about the preferences of the new user, and therefore cannot provide personalized recommendations. A new user preference elicitation strategy needs to ensure that the user does not a) abandon a lengthy signup process, and b) lose interest in returning to the site due to the low quality of initial recommendations. We extend the work of [23] in this paper by incrementally developing a set of information theoretic strategies for the new user problem. We propose an offline simulation framework, and evaluate the strategies through extensive offline simulations and an online experiment with real users of a live recommender system.

1. INTRODUCTION

Collaborative Filtering (CF)-based recommender systems generate recommendations for a user by utilizing the opinions of other users with similar taste. These recommender systems are a nice tool bringing mutual benefits to both users and the operators of the sites with too much information. Users benefit as they are able to find items of interest from an unmanageable number of available items. On the other hand, e-commerce sites that employ recommender systems benefit by potentially increasing sales revenue in at least two ways: a) by drawing customers' attention to items that they are likely to buy, and b) by cross-selling items.

Problem statement. When a user first enters into a recommender system, the system knows nothing about her preferences. Consequently, the system is unable to present any personalized recommendations to her. This problem is sometimes referred to as the *cold-start* problem of recommender systems [15; 6; 5; 29; 20][1]. There are cold start problems for both new users and new items. In this paper, we investigate the cold-start problem for new users of recommender systems. We pose our research question as: how can we effectively learn preferences of new users so that they can begin receiving accurate personalized recommendations from the system? A related problem of recommender systems is the *systemic bootstrapping* problem—recommender systems cannot serve anybody with personalized recommendations when the site has just started, devoid of any evaluations from anybody. We assume an existing recommender system with an established member base here.

User profile learning techniques. User Modeling researchers have been investigating finding ways to elicit user preferences on various domains for years [26; 30; 14]. For example, researchers examined if it would be a better idea to unobtrusively learn user-profiles from the natural interactions of users with the system. One way to categorize the methods proposed thus far is by grouping them into *explicit* and *implicit* methods [13]. Implicit preference collection works "by observing the behavior of the user and inferring facts about the user from the observed behavior" [13]. In the recommender systems domain, implicit techniques may be more suitable if the items can be consumed directly within the system. Also, implicit preference elicitations may be the only option where members can only provide implicit feedback or evaluation, such as by listening to or skipping a song, by browsing a web page, by downloading some content, and so on. Explicit techniques, on the other hand, garner "knowledge that is obtained when an individual provides specific facts to the user model" [13]. Examples include users providing explicit feedback or evaluations on some rating scale. A comparative analysis between the explicit and implicit techniques can be found in [18].

Another way to classify the techniques of building user profiles can be based on the interaction process between the users and the system, particularly by looking at who is in control of the interaction process. [2] calls the possible interaction techniques *human controlled*, *system controlled*, and *mixed initiative* [11]. To explain these in the context of the recommender systems, a preference elicitation technique would be a) human controlled, if it is the user herself who selects (by typing the titles, for example) the items to evaluate, b) system controlled, if the system makes the list of items for the user to evaluate, and c) mixed initiative, if there are provisions for both user and system controlled interactions. The user controlled scheme may cause more work on behalf of the users; however the users may feel good being in charge [16]. One potential limitation of the user controlled scheme is that the users may not be able to identify the items to evaluate that express their preferences well. Further, they

[1] Claypool et al. refers to the problem as the *early rater* problem.

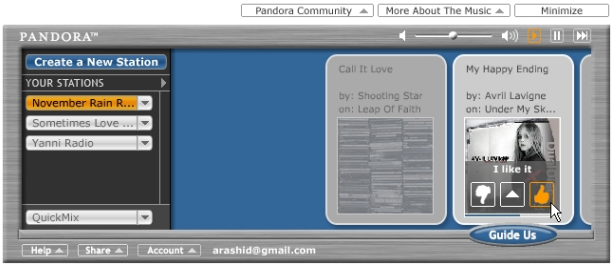


Figure 1: In some systems it is not necessary to be familiar with an item to be able to evaluate it. Shown is a snapshot of the *Pandora* music recommender, which incurs a minimum signup effort for its new members. A member can evaluate a song she has never heard before after she has listened to it on Pandora.

may only remember what they liked the most, not the opposite. An effective system controlled scheme may be able to draw out users' preference information without causing the user to put in a lot of effort. A mixed initiative scheme may have the positive aspects of both of its component schemes. Furthermore, a mixed initiative scheme may be the only option for a large online retail site that contains a wide category of items. A user, for example, may not care about baby products at all, and may not have any opinions about them. Therefore, a user may first help the system filter out the product types she does not care, and then the system can suggest items to evaluate from the remaining item categories.

Desirable criteria of new user preference elicitation strategies. [23] identify a number of aspects a new user preference elicitation strategy should consider. We discuss two important points here. First, *user effort*: a signup process should not seem burdensome to the newcomer—the frustrated user may give up the signup process. Therefore, in some systems, asking users for their opinion about the items they are familiar with would help alleviate the user effort problem. However, in some systems, where the items can be directly consumed within the system, familiarity with the items may not matter. An example is given in figure 1, which shows that the new users of the *Pandora* online music recommender system can be offered the songs they have never heard of and still provide feedback after they listened to the songs from the system. However, in such cases, user effort may be related to the boredom from experiencing a series of items the user does not like. Second, *recommendation accuracy*: the initial quality of recommendations right after the signup process may determine if the user would come back to the site. Therefore, it is very important to provide items that would be able to effectively draw out user preferences, so that the system can compute accurate recommendations for her. In this paper, we consider these points during the selection of the strategies and when we evaluate them.

Our Approach. In this paper, we extend the work of [23] and study the feasibility of a number of item selection measures based on information theory for the new user problem. This involves using each of the measures to find a set of items, and examining how effective the items are in learning profiles of new users. Since the necessary computations to select items are done by the system, and the system prompts the users to provide opinions on a set of items, under the classifications discussed above, our focus can be regarded as *explicit* and *system initiated* approaches. Note that since system controlled approaches are an important component of the mixed initiative approaches as well, this research helps both system controlled and mixed initiative approaches.

We propose an offline simulation framework to investigate how the measures perform for the new user problem. The offline experiments help us to set expectations about the measures for their online deployment performance. Further, we can be warned about a measure that performs poorly in the offline setting and refrain from using this strategy online and bothering actual users. After the offline experiments we investigate the measures with real users online.

2. EXPERIMENTAL PLATFORM

We now introduce the components of the experimental platform we use in this paper. We briefly discuss the CF algorithms we consider, the dataset, and the recommender system site for online experiments.

CF algorithms considered. Researchers have proposed quite a few collaborative filtering algorithms [3; 1] to date. We consider two frequently cited CF algorithms with distinct characteristics for our experiments, namely User-based kNN and Item-based kNN. User-based kNN [10; 25] follows a two step process. First the similarities $w_{u_t,.}$ between the target user u_t and all other users who have rated the target item a_t are computed—most commonly using the Pearson correlation coefficient. Then the prediction for the target item is computed using at most k closest users found from step one, and by applying a weighted average of deviations from the selected users' means: $\overline{R}_{u_t} + \sum_{i=1}^{k}(R_{u_i,a_t} - \overline{R}_{u_i})w_{u_i,u_t} / \sum_{i=1}^{k} w_{u_i,u_t}$. Note that we follow a number of improvements suggested in [10], including *significance weighting* where an attempt is made to lower the similarity between two users if they have not co-rated enough items.

In Item-based kNN [28] similarities are computed between items. To compute a recommendation, all the rated items of the target user u_t are ordered by their similarities with the target item a_t. The recommendation is then a weighted average of the target user's ratings on k most similar items: $\sum_{i=1}^{k}(w_{a_t,a_i} * R_{u_t,a_i}) / \sum_{i=1}^{k}(|w_{a_t,a_i}|)$.

New user signup process in our experimental platform. Our experimental platform is MovieLens[2], an online movie recommender site that uses a collaborative filtering algorithm to deliver movie recommendations to its members. During the MovieLens new user signup process, a user sees one or more pages, where each page contains a list of 10/15 movies. The user rates as many movies as she can from each page and proceeds to the next page until she has rated 15 movies in total. After that, she enters the actual site with all available features. Here she can experience what she came to MovieLens for: recommendations on movies she has not seen yet. The more pages she has to go through to reach the target of the first 15 ratings, the more effort she has to put to scan the movies on each page, and the more frustrating the initial barrier may seem to her.

Data. We extract a dataset from MovieLens. The dataset, let us denote it by D, has about 11,000 users, 9,000 movies,

[2]http://movielens.umn.edu

and 3 million ratings in a scale of 0.5 to 5.0 stars, with an increment of 0.5 star. Therefore, the resulting user×movie matrix is about 97% sparse, typical of recommender system data. In deriving the data we considered only those users who have rated at least 20 movies and logged in at least twice. This is to increase the odds that the ratings collected are from reliable and more active members of MovieLens. We partition D into two sets D_{trn} and D_{tst}, where D_{trn} gets each users' randomly selected 80% of the ratings, and D_{tst} gets the remaining 20%. Therefore, D_{trn} and D_{tst} both contain rating information from each user. As explained further later, D_{trn} is used to compute the heuristics and to carry out offline simulations; evaluations are done using D_{tst}.

3. STRATEGIES TO LEARN NEW USER PROFILES

In this section we incrementally develop a few measures to select items for the goal of learning user profiles effectively. Note that we primarily focus on developing measures based on information theory, since our main goal is to draw *information* about true user preferences.

3.1 Popularity

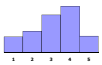
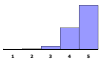
Popularity of an item indicates how frequently users rated the item. Popular items may be good at connecting people with each other as co-raters, since many people are likely to rate popular items. However, depending on the rating distribution, a popular item may or may not be informative. For example, a popular item that is controversial may have about half of the opinions positive and the rest of the opinions negative. This item is deemed to be informative, since the system would at least learn which of the two broad camps of users the rater belongs to. On the other hand, a generally liked popular item may be less informative.

The advantage of Popularity is that it is very easy and inexpensive to compute. A disadvantage of using Popularity measure to elicit preferences, as pointed out by [23], is the possibility of worsening the *prefix bias*—that is, popular items garnering even more evaluations. Unpopular items, lacking enough user opinions, may be hard to recommend. This situation would not improve if the system keeps asking opinions on popular items.

3.2 Entropy

Entropy of an item represents the dispersion of opinions of users on the item. Considering a discrete rating category, entropy of an item a_t, $H(a_t) = -\sum_i p_i lg(p_i)$, where p_i denotes the fraction of a_t's ratings that equals to i. A limitation of entropy is that it often selects very obscure items. For example, an item a_i that has been rated by a moderate number of people (say, 2000 out of the total 6000 members in the system), a rating distribution of (400/2000, 400/2000, 400/2000, 400/2000, 400/2000) corresponding to a rating scale of (1, 2, 3, 4, 5) leads the item to have the maximum entropy score. However, a second item a_f that was rated only 5 times with a rating distribution (1/5, 1/5, 1/5, 1/5, 1/5) possesses the same entropy score. Note that many members may find the former item familiar, and very few members may find the latter item familiar. In general, we cannot infer the rating frequencies or popularities of items from their entropy scores. In fact, figure 2(c), which is a scatter-plot between entropy and popularity (rating frequency, to be exact) of items, shows that entropy and popularity are only slightly correlated (correlation coefficient is only 0.13). A real example demonstrating this limitation of entropy is provided in table 1.

Table 1: Showing a limitation of entropy. While entropy is able to identify the more informative of the two popular films Dumb & Dumber, and Shawshank Redemption, it picks the rarely rated movie Wirey Spindell as the most informative. However, few people will be able to express an opinion about Wirey Spindell.

Film Title	Rating Distrib	# ratings	Entropy
Wirey Spindell		5	**2.32**
Dumb & Dumber		3,918	**2.18**
Shawshank Redemption		6,318	**1.22**

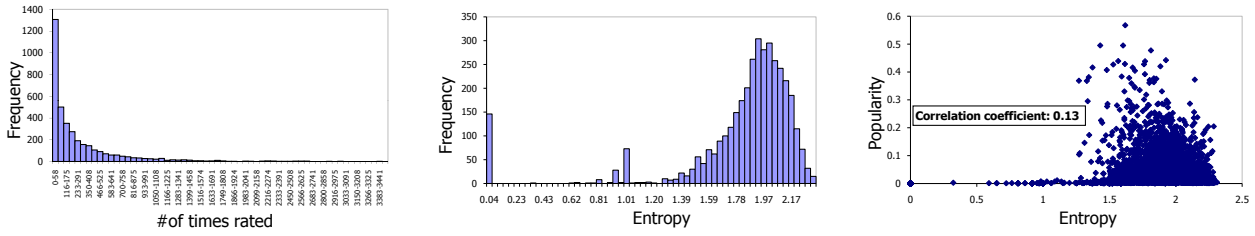
A few other researchers who employed entropy as a measure for informativeness on other domains also report its mixed results. For example, in their work on using information theoretic measures such as entropy to find informative patterns in data, [8] notice that in addition to picking informative patterns, entropy suggests "garbage" (meaningless or not useful) patterns to be useful as well.

In order to modify the behavior of entropy so that in addition to emphasizing the dispersion of user opinions the resulting measure also considers the frequency of user opinions on the item, we next examine two variations of the entropy measure we have discussed so far.

3.3 Entropy0: Entropy Considering Missing Values

In a typical recommender system, most of the items do not receive evaluations from all of the members. This is either because the members have not experienced many of the items, or because the members have not gotten a chance to evaluate them. Therefore, computing entropy might involve a varying number of users' opinions for different items. In order to handle the missing evaluations of an item, we treat the missing evaluations as a separate category of evaluation, for example, a rating value of 0 in the datasets we use, since 1-5 is the usual scale; and fill all the missing evaluations with this new rating category. After this modification, every item has an equal number of user votes, which amounts to the total number of members in the system, and the typical rating scale gets augmented by the new value (0). Furthermore, the frequency of the new rating category (0) of an item indicates how (un)popular the item is—the smaller this value, the more popular this item is.

Note that the new scheme introduces a limitation which can be thought as the reversal of the old limitation. The new scheme might bias frequently-rated items too much. For an example, if the dataset has 6,000 users and an item a that has been rated 200 times, uniformly across the rating category; that is, the rating frequencies are (5800, 50, 50, 50, 50, 50) corresponding to the rating values (0, 1, 2, 3, 4,



(a) (b) (c)

Figure 2: Distribution of items' (a) rating frequencies and (b) entropy scores. (c): a scatter-plot delineating the relationship between items' normalized (to have ranges of values between 0 and 1) rating frequencies and entropy scores.

5), the new scheme yields a score of 0.335. On the other hand, let us consider another item b, which has been rated frequently, say 3,000 times; however, everyone has rated the same way (say 5.0), and the rating frequencies are (3000, 0, 0, 0, 0, 3000). Since everybody agrees on their evaluations, the item b carries no information intuitively. However, the new scheme yields a score of 1.0, even greater than that of the former item, a!

In an attempt to limit the influence of the missing-value category on ENTROPY0, we use a weighted entropy [7] formulation as follows:

$$Entropy0(a_t) = -\frac{1}{\sum_i w_i}\sum_{i=0}^{5} p_i w_i lg(p_i) \quad (1)$$

Using this updated formulation, we can set a smaller weight on w_0 compared to the rest of the weights to lower the effect of the missing-value category of evaluations or the item's rating frequency on ENTROPY0. Note that if we set $w_0 = 0$ and rest of the weights equal to 1.0, ENTROPY0 turns into the basic entropy.

3.4 HELF: Harmonic mean of Entropy and Logarithm of Frequency

We can multiply entropy scores of items with their rating frequencies expecting that an item with a high score of this combined metric would indicate that a) there is a good chance that members would be familiar with it, and b) the user opinions on the item have a high variability. However, the distribution of items' rating frequencies and entropies are very different. As figure 2(a) shows, the distribution of items' rating frequencies is approximately exponential, and the distribution of entropy scores is approximately normal (slightly skewed to the left). Further, figure 2(c) shows that entropy and rating frequency are not correlated at all. Therefore, a straightforward multiplication of the two produces a measure that is heavily related to one of the component measures (as shown in table 2, it is the popularity). By further examining figure 2(a) and 2(b), we find that the scales, as shown in the x-axes, are vastly different. Therefore, it might seem that normalizing both rating frequency and entropy scores so that they remain between 0 and 1 would solve the problem of dominance of rating frequency on the multiplicative measure of the two. However, the shape of the rating frequency distribution remains the same after we normalize the values. We then note that the rating-frequency values have a very wide range—the largest value (about 3,000) is many orders of magnitude larger than the smallest value (0). On the other hand, the entropy scores do not vary that much. As a result, a multiplication between the two varies heavily with the rating frequency scores.

A property of the logarithm function is that it can transform an exponential-like curve into a linear-like curve by compressing large values together. The range of the transformed values, therefore, becomes much smaller. For example, in our dataset, the range of the transformed values becomes: 0-11.5—much smaller than the original. Note that we use a base 2 logarithm (denoted as lg), and treat $0lg0 = 0$.

In [23], the logarithm of rating-frequency is multiplied with the entropy. However, we take a harmonic mean of the two. A widely used evaluation metric in information retrieval utilizing the harmonic mean is the F1 metric, which combines *precision* and *recall* scores [33; 27]. A nice property of the harmonic mean is that it strongly increases or decreases when both of the components increase or decrease [19].

Therefore, the final measure HELF: **H**armonic mean of **E**ntropy and **L**ogarithm of rating **F**requency, can be expressed as below.

$$HELF_{a_i} = \frac{2 * LF'_{a_i} * H'(a_i)}{LF'_{a_i} + H'(a_i)} \quad (2)$$

where, LF'_{a_i} is the normalized logarithm of the rating frequency of a_i: $lg(|a_i|)/lg(|U|)$, and $H'(a_i)$ is the normalized entropy of a_i: $H(a_i)/lg(5)$.

3.5 IGCN: Information Gain through Clustered Neighbors

One issue with information theoretic measures such as entropy and its variants discussed so far is that they are not adaptive to a user's rating history. Depending on the opinions expressed on the items so far, the informativeness of the rest of the items may not be the same for two users, who might have rated a different set of items, or rated the

Combination approach	Corr coeff of the combined measure	
	with Entropy	with rating frequency
Multiplication	0.17	0.99
HELF	0.77	0.55 (0.96 with log(rating frequency))

Table 2: As shown in the case of HELF, applying a log transformation to items' rating frequency helps the combined measure to be related to both of the component measures: entropy and rating frequency. Whereas, in the first approach, the combined measure is only weakly correlated with entropy.

same items in a different manner. In the following, we try to develop an information theoretic measure that takes the items rated so far by a user into account.

In short, our proposed approach IGCN works by repeatedly computing information gain [17] of items, where the necessary ratings data is considered only from those users who match best with the target user's profile so far. Users are considered to have labels corresponding to the clusters they belong to; and the role of the most informative item is treated as helping the target user most in reaching her representative cluster(s). Next we explain how we develop IGCN by taking a few assumptions.

Design decision: Goal of building profiles is to find right neighborhoods. Collaborative filtering-based recommender system algorithms compute recommendations for a user by utilizing the opinions of other users with similar taste, who are also referred to as *neighbors*. Therefore, we can expect that a user will receive the best recommendations if her true like-minded neighbors are found. As a result, the goal of building a good preference-profile of a member can be interpreted as finding the best set of neighbors for her. A key question is: should the set of neighbors of each user be fixed? That is, for the target user, whether we should first find a set of k best neighbors, and use only these neighbors for computations of all her recommendations. Figure 3 demonstrates the limitations of this approach. The best k neighbors might not have an opinion about all the items the target user needs recommendations about. Therefore, dynamically selecting top neighbors from among the users who rated the target item is a better idea for practical reasons. Taking all these dynamic neighbors of the target user together, we may find that the number of neighbors considered is at least a couple of times greater than the value of k.

Design decision: Neighborhoods correspond to user clusters. The objective of clustering is to group entities so that intra-cluster similarities of the entities are maximized, and the inter-cluster similarities are minimized. Therefore, if the same similarity function is used both to find neighbors and to compute clusters, a user cluster can be roughly regarded as a cohesive user neighborhood. However, following the discussion above, the necessary neighbors of a user may come from multiple clusters (proxy for neighborhoods).

Use a decision tree? If we regard user clusters as *classes* of users, and the goal of profile building as finding the right cluster (class) for the target user, a decision tree algorithm such as ID3 [22] can be employed for learning user profiles. The decision tree would have cluster-numbers (class labels) in the leaf nodes; and each internal node would represent a test on an item indicating the possible ways the item can be evaluated by a user. The item on the root node would have the highest information gain, where the information gain of an item a_t can be computed in this context as:

$$IG(a_t) = H(\mathbf{C}) - \sum_r \frac{|\mathbf{C}^r_{a_t}|}{|\mathbf{C}|} H(\mathbf{C}^r_{a_t}) \qquad (3)$$

where $H(X)$ denotes the entropy of a discrete random variable X. $\mathbf{C}$ denotes the distribution of users into classes (clusters), that is, how many users belong to each cluster. $\mathbf{C}^r_{a_t}$ represents the distribution of those users into classes who evaluated the item a_t with the value r. For example, if $r = 4$, $\mathbf{C}^r_{a_t}$ indicates how many of the users who voted a_t a 4-star belong to each cluster. Note that $H(\mathbf{C})$ tells us about the expected information that is required to know which class (cluster) a given user belongs to; and $\sum_r \frac{|\mathbf{C}^r_{a_t}|}{|\mathbf{C}|} H(\mathbf{C}^r_{a_t})$ is essentially the weighted average of the entropies of various partitions of the original class distribution ($\mathbf{C}$) caused by users' ratings of a_t. Thus, the latter term indicates how much expected information is still required to find the class of a given person after rating a_t. Therefore, $IG(a_t)$ essentially expresses the reduction in required information toward the goal of finding the right class by rating a_t.

The goal of the target user would then be to follow a route through the decision tree—starting from the root node and ending at a leaf node. The cluster or class representing the leaf node would imply the user's true class or neighborhood. Unfortunately, this ideal decision tree scenario may not be feasible with most members of a recommender system. The reasons include the following two.

- *Missing values.* Members may not be familiar with some of the items along a path of the tree. Some members may not even know the item on the root of the tree.

- *Inadequate goals.* As explained during the discussion of the assumptions above, depending on the granularity of the user clusters, the goal of finding *one* best cluster may be insufficient.

The *missing value* problem is important in practice for the following two reasons. First, even the most popular items are only rated by a fraction of the users. For example the most popular item in our dataset is rated by only 50% of the users. Second, dealing with the missing values algorithmically is a challenge.

We approach this *missing value* problem by treating the missing evaluations of an item as a separate category (0 in our datasets). As a result, the values of r in our dataset become 0,1,...,5. As in the case of ENTROPY0, however, this introduces a problem in that frequently rated items dominate over infrequently rated ones. Therefore, we incorporate additional weight terms into equation 3 the following way:

$$IG(a_t; \mathbf{W}) = H(\mathbf{C}) - \sum_r \frac{w_r |\mathbf{C}^r_{a_t}|}{E(\mathbf{C}; \mathbf{W})} H(\mathbf{C}^r_{a_t}) \qquad (4)$$

where $E(\mathbf{C}; \mathbf{W}) = \sum_r w_r \frac{|\mathbf{C}^r_{a_t}|}{|\mathbf{C}|}$. This updated equation gives us an opportunity to lower the effect of rating category corresponding to the missing ratings. We can do so by setting a lower weight on w_0 compared to the rest of the weights w_i, for $i = 1 \ldots 5$.

Since a direct application of the decision tree algorithm is not practically feasible in our problem domain, we use an algorithm IGCN, presented in algorithm 3.1 that approximates it. IGCN assumes the following. First, the goal of profile-building is to find a *set* of best clusters, or a number (typically greater than k of kNN) of best neighbors. Second, we assume a system or interface that presents items for evaluation in batches (instead of one item at a time); for example, a web site may list 15 items on a page for a user to evaluate. Third, a user may not know any of the items provided in a step.

Note that IGCN works in two steps. The first step is non-personalized in that the information gain of the items are

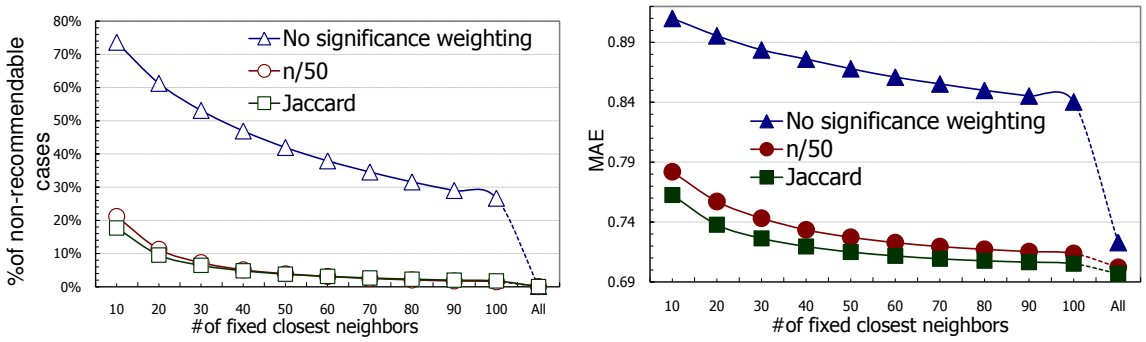


Figure 3: Nearest neighbor CF approaches such as USER-BASED kNN use opinions of up to k closest neighbors to calculate predictions. Should we then find a set of l users ($l \geq k$) whose tastes are most similar to the target user and always use this *fixed* set of neighbors for all predictions? Perils of applying this idea are shown by means of %of non-recommendable items and recommendation error. Further, it gets worse if we do not utilize a *significance weighting* [10] to adjust user-user similarities. Note that the results indicated by *All* also represent the results we get if we use k (=20) dynamic neighbors for each recommendation.

computed considering all users in the system. Once the target user has rated at least some threshold number of items, the personalized step begins. In this step only the best neighbors of the target user are used to compute the information gain of the items.

Algorithm 3.1: IGCN algorithm

- Create c user clusters
- Compute information gain (IG) of the items
- **Non-personalized step:**
  /* *The first few ratings to build an initial profile* */
  REPEAT
  - Present next top n items ordered by their IG scores
  - Add items the user is able to rate into her profile

  UNTIL the user has rated at least i items
- **Personalized step:**
  /* *Toward creating a richer profile* */
  REPEAT
  - Find best l neighbors based on the profile so far
  - Re-compute IG based on the l users' ratings only
  - Present next top n items ordered by their IG scores
  - Add items the user is able to rate into her profile

  UNTIL best l neighbors do not change

Table 3: Table of notations.

Notation	Description
u_t	Target user
D_{trn}	Training dataset
D_{tst}	Test dataset
$D_{u_{t_{trn}}}$	Target user's rating data in the training dataset
$D_{u_{t_{tst}}}$	Target user's rating data in the test dataset
$D_{u_{t_{seen}}}$	Subset of $D_{u_{t_{trn}}}$ corresponding to the movies "seen" by u_t from the "presented" movies

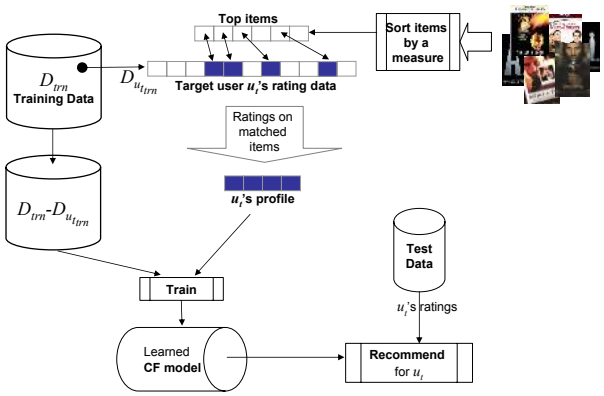


Figure 4: New user signup simulation procedure.

4. OFFLINE EXPERIMENTS

In this section we examine the efficacy of the heuristics presented in section 3 through an offline simulation which mimics the user activity during the MOVIELENS new user signup process. Table 3 lists the explanations of the notations we use.

4.1 Offline Simulation Framework

In order to simulate the MOVIELENS signup process described in section 2, we select one of the heuristics, such as HELF and sort movies in descending order by the measure, and then "present" the top 15, 30, ..., or 75 movies to a new user u_t corresponding to 1, 2, ..., or 5 "pages". We treat u_t's rated movies in the training set as all the movies u_t has watched. We determine which of "presented" movies u_t has seen by taking an intersection between the "presented" movies and movies in $D_{u_{t_{trn}}}$. u_t's rating information on these matched movies constitute u_t's initial profile. We then evaluate the efficacy of this profile by computing predictions for u_t corresponding to her rating information $D_{u_{t_{tst}}}$ found in the test set D_{tst}. Predictions are computed using the entire training data and the new profile, after u_t's old rating information is discarded; that is, using $D_{trn} - D_{u_{t_{trn}}} + D_{u_{t_{seen}}}$. Figure 4 shows the steps we discussed for the offline simulation.

4.2 Procedure

For computations of the heuristics, we used the entire D_{trn}; however, for the simulation, we only used users who have at

least 80 ratings in D_{trn}. There are two reasons for this decision. First, from the historical ratings data, it may be hard to infer what users have seen. For an example, a user who has 20 ratings might have seen more than she has reported. However, many of the movies we "present" may appear unfamiliar to her because of her limited reporting. Second, since we "present" up to 75 movies, we need 75 or more ratings of a user to know how many of the presented movies they have "seen". Selecting users with 80 or more ratings may create a bias in that our findings may apply only to users with many ratings. Note that about 71.5% of the users in D_{trn} had $\geq$80 ratings.

All of the heuristics except the IGCN can be computed prior to the simulation. IGCN requires clustering users in D_{trn}. We use the BISECTING k-MEANS, a variant of k-MEANS clustering algorithm for its simplicity and accuracy [31; 12]. Parameters of the IGCN algorithm are c: number of users clusters, n: number of movies presented at a time, i: number of ratings for the initial profile, and l: number of closest neighbors to re-compute IG. We set values of (c, n, i, l) as (300, 15, 5, 150). Values of c and l are chosen since they yield the best results. The value of n is what is used in MOVIELENS new user signup. For ENTROPY0 we use $w_0 = 1/2$, and $w_i = 1$ for $i = 1 \ldots 5$ since this combination of weights produces the best results.

Note that we do not experiment with basic ENTROPY here, rather directly apply the learning from [23].

4.3 Evaluation Metrics

The first metric we are interested in measures how much users are able to rate movies selected by a strategy. MAE or mean absolute error is another metric we use that indicates recommendation quality. MAE, a widely used metric in the CF domain, is the average of deviations between the recommendations and the corresponding actual ratings. However, a limitation of MAE is that it only considers absolute differences. MAE sees no difference between two pairs of (actual rating, recommendation) for a movie that are (1, 5) and (5, 1). Although users may be unhappy more about the former pair. We, therefore, use another accuracy metric, Expected Utility [24] that tries to penalize *false positives* more than *false negatives*.

4.4 Results

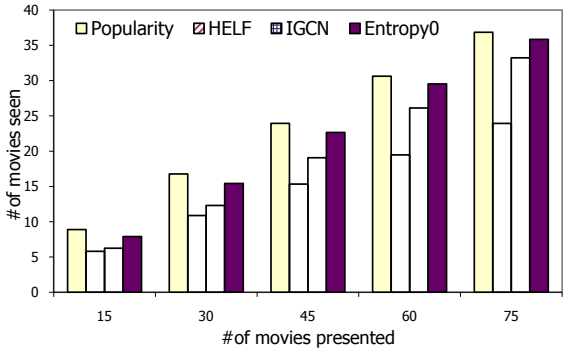


Figure 5: Showing how familiar the movies are to the users as the movies are "presented" in batches according to each of the item selection measures we study here.

In this section we present the results from the offline simulations. Figure 4 shows how well the users are able to rate movies presented by various approaches. We see that the POPULARITY scheme selects items that users find most familiar, and HELF produces the least familiar movies. However, users are able to rate at least one third of the presented movies by each approach, probably reasonable in terms of user effort.

Next we present recommendation accuracy results to compare the effectiveness of the users' new profiles by various measures. Figures 6(a) and 6(b) show the recommendation accuracy results for the USER-BASED kNN CF algorithm. We find that both IGCN and ENTROPY0 perform well by both metrics. We find similar results in figures 6(c) and 6(d) where the CF algorithm used is the ITEM-BASED kNN, although IGCN performs slightly better. The confusing results, however, are from POPULARITY and HELF. According to MAE, HELF and POPULARITY are the bottom performers; however according to EU, HELF is the best measure. We next drill down into the results to better understand the confusing performance of HELF and POPULARITY.

Table 4 shows recommendation accuracy results when the number of "presented" movies is 45 and the CF algorithm used is the ITEM-BASED kNN. Note that identical results found on the USER-BASED kNN CF algorithm are not presented here. For brevity, we treat recommendations and actual ratings to be of two values: positive (≥ 3.0) and negative (< 3.0).

In order to understand the puzzling results of POPULARITY and HELF, we present data about the following: a) of all the true negative ratings (< 3.0) found in the test dataset, what percentage of the recommendations are positive, b) of all the true positive ratings (≥ 3.0) found in the test dataset, what percentage of the recommendations are negative, c) MAE of the recommendations corresponding to the positive actual ratings, and d) MAE of the recommendations corresponding to the negative actual ratings. We find from the table that HELF does the best job and POPULARITY does the worst job in avoiding false positive recommendations. On the other hand, user profiles built by evaluating popular movies help avoid false negative recommendations. Similarly, MAE due to POPULARITY is much better than that of HELF when actual ratings are positive. The performance of this duo reverses when ratings are negative. Since the ratio of actual positive to negative ratings is about 3.6:1, the good MAE of POPULARITY on positive actual ratings helps POPULARITY to be better than HELF. On the other hand, in the expected utility (EU) measure for the movie domain we penalize false positives more than the false negatives; therefore, HELF beats POPULARITY when EU is used as the performance metric.

Note also from table 4 that IGCN performance balances between not doing too bad in recommending either false positives or false negatives. As a result, it consistently performs well on both metrics.

Note that all the results we present are averages over all users. For example, the MAE results shown in the figure ?? are computed as follows. For an influence metric, such as HELF, the MAE is computed for one user. Then this computation is iterated over all users and averaged. This way of computing an average MAE, instead of taking results of all users and then computing an MAE, has the advantage that each user is given an equal importance.

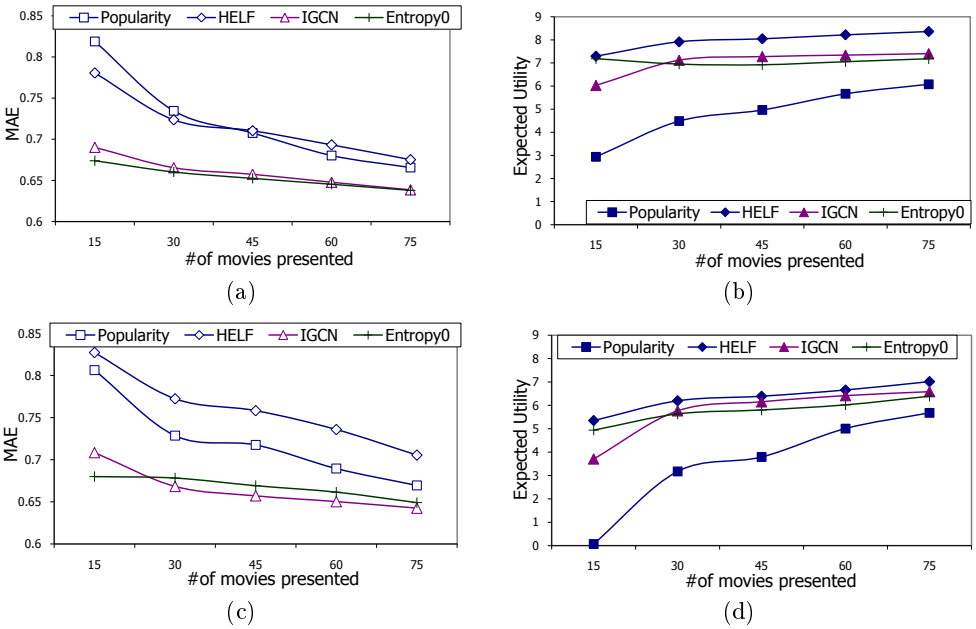


Figure 6: How effective are the learned user profiles? Recommendation accuracy results from the offline simulations. (a)-(b) on the USER-BASED kNN, and (c)-(d) on the ITEM-BASED kNN CF algorithm. The evaluation metrics used are mean absolute error or MAE (the lower, the better) and expected utility or EU (the higher, the better).

Table 4: Showing the strengths and weaknesses of each of the approaches in generating different types of recommendations. Recommendations (and actual ratings) are treated to be of two categories: positive (≥ 3.0) and negative (< 3.0). The table is produced for the ITEM-BASED kNN CF algorithm and considering the case when the number of "presented" movies is 45. The best result in each row is shown in bold-face.

	POPULARITY	HELF	IGCN	ENTROPY0
%of true negatives as false positives	66.25	**32.27**	51.08	53.48
%of true positives as false negatives	**7.34**	28.92	12.09	11.27
MAE for positive actual ratings	0.54	0.72	**0.52**	0.53
MAE for negative actual ratings	1.37	**0.91**	1.14	1.18

5. ONLINE EXPERIMENTS

Online studies are essential to examine if the offline findings hold true in real world settings. In this section we describe the online experiments we performed for the new user problem.

5.1 Design

We perform our online experiments on MOVIELENS recommender system. We define four experimental groups corresponding to the item selection measures we investigate. After a new user is done with the basic registration process, such as providing the user name and password, he or she is randomly assigned to one of the four groups. They see pages full of movies to rate. These movies are selected by the item selection measure corresponding to the subject's group. After she has finished rating a fixed number of movies she is taken to a brief optional survey and that concludes the experiment.

The goal of the survey was to collect user opinions about the signup process. To make it short, we asked their opinions on only the following areas. We asked if they thought that the movies they rated represented their general movie preferences, if they found it easy to rate during the signup, if they thought that the signup process was a barrier to entry for the site, and if they prefer to search for the movies they wanted to rate. There was also an area in the survey where they could express any additional thoughts they had about the signup.

After the signup process the new user enters into the full-fledged MOVIELENS system. There she can do various activities as she prefers to, including receiving personalized recommendations, participating in discussion forums, and so on. The important activity for this experiment is rating movies. In MOVIELENS, a user can rate a movie anytime she sees it either because she searched for it, or she used the "rate more movies" option, a feature available in MOVIELENS that currently lists random movies to rate. These additional ratings after the signup process are important because we check how accurate the recommendations of the movies she rated are. That is, we treat the ratings of a user during the signup process as her profile, and the ratings after the signup process as the test data to evaluate the efficacy of the profile.

MOVIELENS requires each new member to rate at least 15 movies during the signup process to finish registration. In this experiment, however, we raised the threshold to 20, so that initial profiles of the subjects are more mature. However, one may worry that users may find it burdensome

Table 5: Group-wise participations of the subjects

Approach	#of registrants	#of valid subjects	#of pages to finish
IGCN	118	95 (81%)	5.5
POPULARITY	123	95 (77%)	3.6
ENTROPY0	115	95 (83%)	3.0
HELF	112	92 (82%)	4.6

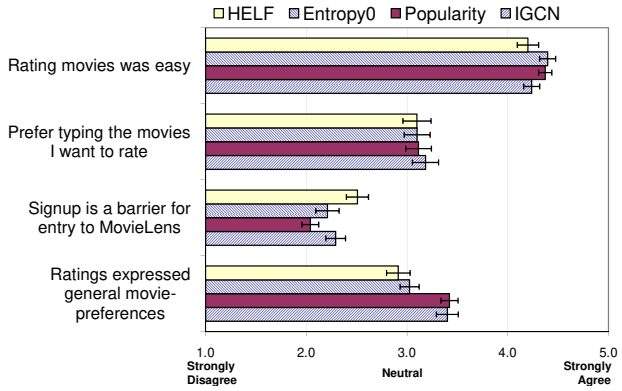


Figure 7: Average survey responses.

to rate 20 movies before they can enter into the system. We asked the users about this in the short survey after the signup, and users reported that they did not find it burdensome. Rather they understood the benefit of more ratings as the only way to teach the system about their preferences. One user wrote about the signup process: *"I understand why it is needed. I know it will make your recommendations more accurate."*

5.2 Results and Discussion

We ran the experiment for 20 days. 468 users tried to join MOVIELENS during that period, who became the subjects of our experiment. 381 subjects finished the signup process—we call them the *valid* subjects. Among these valid subjects, 305 users at least partially participated in the survey. Table 5 shows how the participants and the valid subjects are distributed across the experimental conditions. We perform all our data analysis using the valid subjects.

The last column of table 5 indicates the varying degrees of effort required by the subjects to finish the signup process. Parallel to our findings from the offline simulation, the POPULARITY and ENTROPY0 approaches required users to scan fewer pages than that of the IGCN and HELF approaches. Interestingly, users seemed not to notice this extra effort. Figure 7 supports this fact. Users' self reports indicate that in all experimental conditions they *agreed* that rating movies was easy, and they *disagreed* that the signup process was a barrier to entry. Since users either do not notice or do not get bothered with the variations of effort caused by the item selection measures, the initial recommendation quality can be considered as the deciding factor to judge the approaches.

Table 6 shows the initial recommendation quality from the new user profiles by the four approaches. As described before, we considered the ratings of a subject during the signup process as her profile and her ratings afterwards as the test data. We made two variations of the test data in that one test dataset contains all her ratings after the signup step,

Table 6: Effectiveness of the learned user profiles according to the accuracy of the initial recommendations on two CF algorithms. Recommendations are evaluated using test data collected the following way: (a) all ratings of the users after the signup, and (b) the first 20 ratings of the users after the signup. The best and worst results in each column are shown in bold-face.

(a) USER-BASED kNN CF algorithm

	Test data: #of ratings *after signup*			
	All ratings		First 20 ratings	
Approach	MAE	EU	MAE	EU
IGCN	**0.67**	4.72	**0.70**	**5.67**
POPULARITY	0.73	**3.63**	0.72	**3.27**
ENTROPY0	0.73	6.20	0.71	5.03
HELF	**0.84**	5.63	**0.88**	5.36

(b) ITEM-BASED kNN CF algorithm

	Test data: #of ratings *after signup*			
	All ratings		First 20 ratings	
Approach	MAE	EU	MAE	EU
IGCN	**0.69**	3.78	**0.75**	5.00
POPULARITY	0.73	**3.33**	0.76	**3.75**
ENTROPY0	0.77	**5.55**	0.75	5.23
HELF	**0.85**	5.29	**0.92**	**5.78**

and another test dataset contains at most her first 20 ratings after the signup step. Therefore, the latter dataset is a subset of the former dataset. The subject's initial profile is used to generate recommendations for each of the test datasets by using two CF algorithms: USER-BASED kNN and ITEM-BASED kNN.

IGCN performs the best and HELF performs the worst by the MAE metric on both CF algorithms. However, HELF performs well by the expected utility metric. Overall, user profiles due to the POPULARITY measure resulted in the least accurate recommendations.

Note that we get similar results (similar performance ordering of the approaches) by considering the first 15 ratings of each subject as their initial profiles instead of all of the 20 ratings they made during the signup step.

Educating users. A number of surveys [32; 9] including the one we carried out in this experiment suggest that users of recommender systems generally understand that more ratings help the system learn their tastes better. This user understanding is perhaps the result of the relevant explanations most recommender systems provide on their sites. However, from anecdotal evidence, it seems that in general users do not realize the fact that telling about what they do not like is important in addition to informing the system about what they like. A subject in our experiment was not happy to find a movie we asked to evaluate that he or she did not like: *"Babel is the worst movie I have ever seen!"* Another user commented about how to improve the signup process: *"I think there should be an option to list your 5 alltime favorites. Even better, for those who care to take the time...list favorites in several categories."* Since users keep rating items beyond the signup stage, educated users will be able to express their tastes better and will not be annoyed if the system asks them to rate something they did not like.

Table 7: Summary of various strategies along two important dimensions of the new user problem. (★★★★★: best, ★: worst, considering both offline and online performance.)

Strategy	User effort	Recommentation accuracy
IGCN	★★★★	★★★★★
ENTROPY0	★★★★★	★★★★
HELF	★★★	★★★★
POPULARITY	★★★★★	★★★
ITEMITEM [23]	★★★★★	★★
RANDOM [23]	★	★★
ENTROPY [23]	★	★★

6. CONCLUSION

In this paper we have approached the new user cold start problem of recommender systems with a set of item selection measures. The set of measures we considered here has a root in information theory, and belongs to the category of *system-controlled* techniques for learning user profiles. We believe that research on developing effective system-controlled techniques is important, since system-controlled techniques demand less cognitive and manual effort of the users. Besides, in retail sites deploying recommender systems, where a mixed initiative technique is necessary because of the multitude of categories of items it carries, an effective system-controlled technique can do its part the best possible way.

Through an offline simulation and an online study, we have found that all of the approaches worked well, both in terms of the user effort and the initial recommendation quality. Overall, the POPULARITY measure performed the worst, and the IGCN measure performed the best. Table 7 juxtaposes the performance summary of the measures we presented in this paper with the measures discussed in [23]. We notice from the table that IGCN and ENTROPY0 are two top performers.

The closeness between the results from the online study and the offline simulation suggests that the simulation framework we proposed is effective for the type of new user signup process it mimicked. This simulation framework, therefore, can be used to evaluate future novel approaches before trying with real systems, or where studying with real users is not feasible at all.

Much remains to be done. Given a recommender system, a formal analysis able to express the minimum independent information that is still required to understand a given user's profile is an intriguing direction. [21] sketched some ideas by means of value of information analysis, and [4] implemented some of those ideas. However, more work is needed, due to the limitation of the approach of [4] on sparse datasets, a common scenario for e-commerce recommenders.

Learning a user's profile may be a continuous activity, and preferences of a user may change over time. The possibility of the ephemeral nature of user preferences may require some type of longitudinal adaptive profiling. That is, either old preferences must be discarded, or more weight must be given to recent preferences. The problem of updating profiles by the age of evaluations is another interesting direction for future work.

7. REFERENCES

[1] Member-Gediminas Adomavicius and Member-Alexander Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17(6):734–749, 2005.

[2] James F. Allen. Mixed-initiative interaction. *IEEE Intelligent Systems*, 14(5):14–23, 1999.

[3] Daniel Billsus and Michael J. Pazzani. Learning collaborative information filters. In *Proc. 15th International Conf. on Machine Learning*, pages 46–54. Morgan Kaufmann, San Francisco, CA, 1998.

[4] C. Boutilier, R. Zemel, and B. Marlin. Active collaborative filtering, 2003.

[5] Mark Claypool, Anuja Gokhale, Tim Miranda, Pavel Murnikov, and Dimitry Netes adn Matthew Sartin. Combining content-based and collaborative filters in an online newspaper. In *ACM SIGIR Workshop on Recommender Systems*, 1999.

[6] Nathan Good, Ben Schafer, Joseph Konstan, Al Borchers, Badrul Sarwar, Jon Herlocker, and John Riedl. Combining collaborative filtering with personal agents for better recommendations. In *Proceedings of the 1999 Conference of the American Association of Artificial Intelligence (AAAI-99)*, July 1999.

[7] S. Guiasu. Weighted entropy. In *Reports on Math, Phys.2*, pages 165–179, 1971.

[8] Isabelle Guyon, Nada Matic, and Vladimir Vapnik. Discovering informative patterns and data cleaning. pages 181–203, 1996.

[9] F. Maxwell Harper, Xin Li, Yan Chen, and Joseph A. Konstan. An economic model of user rating in an online recommender system. In *User Modeling*, page 307, Edinburgh, Scotland, 2005. Springer Berlin.

[10] Jon Herlocker, Joseph Konstan, Al Borchers, and John Riedl. An algorithmic framework for performing collaborative filtering. In *Proceedings of the 1999 Conference on Research and Development in Information Retrieval (SIGIR-99)*, August 1999.

[11] Eric Horvitz. Principles of mixed-initiative user interfaces. In *CHI '99: Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, pages 159–166, New York, NY, USA, 1999. ACM Press.

[12] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: A review. *ACM Computing Surveys*, 31(3):264–323, 1999.

[13] Robert Kass and Tim Finin. Modeling the user in natural language systems. *Computational Linguistics*, 14(3):5–22, 1988.

[14] Alfred Kobsa and Wolfgang Wahlster, editors. *User models in dialog systems*. Springer-Verlag New York, Inc., New York, NY, USA, 1989.

[15] D. A. Maltz and K. Erlich. Pointing the way: Active collaborative filtering. In *Proceedings of CHI 95*, pages 202–209. ACM SIGCHI, May 1995.

[16] Sean M. McNee, Shyong K. Lam, Joseph A. Konstan, and John Riedl. Interfaces for eliciting new user preferences in recommender systems. In *User Modeling*, pages 178–187, Johnstown, PA, USA, 2003. Springer Verlag.

[17] Thomas M. Mitchell. *Machine Learning*. McGraw-Hill Higher Education, 1997.

[18] Miquel Montaner, Beatriz López, and Josep Lluís De La Rosa. A taxonomy of recommender agents on theinternet. *Artificial Intelligence Review*, 19(4):285–330, 2003.

[19] Didier Nakache, Elisabeth Metais, and Jean François Timsit. Evaluation and nlp. pages 626–632, 2005.

[20] Seung-Taek Park, David Pennock, Omid Madani, Nathan Good, and Dennis DeCoste. Naïve filterbots for robust cold-start recommendations. In *KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 699–705, New York, NY, USA, 2006. ACM Press.

[21] David M. Pennock, Eric Horvitz, Steve Lawrence, and C. Lee Giles. Collaborative filtering by personality diagnosis: A hybrid memory and model-based approach. In *UAI '00: Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*, pages 473–480, Stanford, CA, 2000. Morgan Kaufmann Publishers Inc.

[22] J. R. Quinlan. Induction of decision trees. In Jude W. Shavlik and Thomas G. Dietterich, editors, *Readings in Machine Learning*. Morgan Kaufmann, 1990. Originally published in *Machine Learning* 1:81–106, 1986.

[23] Al Mamunur Rashid, Istvan Albert, Dan Cosley, Shyong K. Lam, Sean McNee, Joseph A. Konstan, and John Riedl. Getting to know you: Learning new user preferences in recommender systems. In *Proceedings of the 2002 International Conference on Intelligent User Interfaces*, pages 127–134, San Francisco, CA, February 2002.

[24] Al Mamunur Rashid, Shyong K. Lam, George Karypis, and John Riedl. Clustknn: A highly scalable hybrid model- & memory-based cf algorithm. *WEBKDD 2006: Web Mining and Web Usage Analysis*, 2006.

[25] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. GroupLens: An open architecture for collaborative filtering of netnews. In *CSCW '94: Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work*, pages 175–186, Chapel Hill, North Carolina, United States, 1994. ACM Press.

[26] Elaine Rich. User modeling via stereotypes. *Cognitive Science*, 3(4):329–354, 1979.

[27] G. Salton and C. Buckley. Improving retrieval performance by relevance feedback. *Journal of the American Society for Information Science*, June 1990.

[28] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *WWW '01: Proceedings of the 10th International Conference on World Wide Web*, pages 285–295, Hong Kong, 2001. ACM Press.

[29] Andrew I. Schein, Alexandrin Popescul, Lyle H. Ungar, and David M. Pennock. Methods and metrics for cold-start recommendations. In *SIGIR '02: Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 253–260, New York, NY, USA, 2002. ACM Press.

[30] D. Sleeman. Umfe: a user modelling front-end subsystem. *Int. J. Man-Mach. Stud.*, 23(1):71–88, 1985.

[31] M. Steinbach, G. Karypis, and V. Kumar. A comparison of document clustering techniques, 2000.

[32] K. Swearingen and R. Sinha. Beyond algorithms: An hci perspective on recommender systems, 2001.

[33] C. J. Van Rijsbergen. *Information Retrieval, 2nd edition*. Dept. of Computer Science, University of Glasgow, 1979.