Putting Users in Control of their Recommendations

F. Maxwell Harper, Funing Xu, Harmanpreet Kaur,
Kyle Condiff, Shuo Chang, Loren Terveen
GroupLens Research
University of Minnesota
Minneapolis, MN, USA
{harper, xuxx0572, kaurx090, cond0155, schang, terveen}@cs.umn.edu

ABSTRACT

The essence of a recommender system is that it can recommend items *personalized* to the preferences of an individual user. But typically users are given no explicit control over this personalization, and are instead left guessing about how their actions affect the resulting recommendations. We hypothesize that any recommender algorithm will better fit some users' expectations than others, leaving opportunities for improvement. To address this challenge, we study a recommender that puts some control in the hands of users. Specifically, we build and evaluate a system that incorporates user-tuned popularity and recency modifiers, allowing users to express concepts like "show more popular items". We find that users who are given these controls evaluate the resulting recommendations much more positively. Further, we find that users diverge in their preferred settings, confirming the importance of giving control to users.

Categories and Subject Descriptors

H.5.3 [**Group and Organization Interfaces**]: Computer-supported cooperative work; H.1.2 [**User/Machine Systems**]: Human factors; H.3.3 [**Information Search and Retrieval**]: Information filtering

Keywords

recommender systems; collaborative filtering; social computing; user control; personalization; user study; simulation study; MovieLens

1. INTRODUCTION

Recommender systems are usually not user-configurable. Recommendation algorithms personalize their responses for users by measuring behaviors (ratings, clicks, purchases, survey questions, etc.), sometimes matching these with content attributes (popularity, price, author, etc.) Typically, users are given no explanation of how their behaviors affect their recommendations[1]; the system is a "black box". Therefore, while users may *want* a recommender to behave differently ("my recommendations are too obscure", or "my recommendations show too many expensive things"), they are not given any means to *tell* the system to behave differently.

In this research, we explore the idea of giving users control over their recommender. What if the user could tell the recommender to de-emphasize obscure content, or to prioritize affordable content? We hypothesize that such a system could leave users feeling more satisfied with their recommendations and more in control of the process.

To this end, we evaluate a system that gives users control over a single variable that is external to (or subsumed by) the recommendation algorithm. By examining a single variable, we allow users to control the recommender through simple actions like "show less expensive items".

Our recommendation method uses a linear weighted combination of a *personalization variable* (the output of a recommender algorithm such as item-item collaborative filtering) and a *blending variable* (a non-personalized content attribute such as price); we give users control over the weights. This approach has a number of advantages. It can be incorporated into any existing item-ranking recommender system by simply re-ranking result lists. It is computationally cheap, both to recommend items, and to change the recommender's weights. It is simple to understand, both from a user perspective and from a data analytics perspective.

We structure this work around the following research questions:

- RQ1: Do users like having control over their recommendations?
- RQ2: Given control, how different are users' tuned recommendations from their original recommendations?
- RQ3: Do users converge to a common "best tuning" setting?

We focus our analysis on two attributes available in most recommender systems: item *popularity*, and item *age*. Item popularity — the number of users who have interacted with the item — may be operationalized by measuring data such as clicks, ratings, and mentions. Item age may be operationalized by the time when the item was added to the system, or when the item was created/released.

[1]Some users will try to reverse engineer the algorithm [12]

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions@acm.org.
RecSys'15, September 16–20, 2015, Vienna, Austria.
© 2015 ACM. ISBN 978-1-4503-3692-5/15/09 ...$15.00.
DOI: http://dx.doi.org/10.1145/2792838.2800179.

To deepen our understanding of the impact of using popularity and age in item recommendations, we conduct an offline simulation study and an online user study in MovieLens (`http://movielens.org`), a movie recommendation web site with several thousand active monthly users. MovieLens members rate movies on a 5 star scale (with half-star increments). They expect that rating more movies will help the system deliver better recommendations.

We make several research contributions. We describe a functional, computationally efficient recommendation method that can add user-configurability into any existing item recommendation framework. Through a simulation study, we develop several insights concerning popularity and age — two widely-available, easily understood variables for user manipulation. We determine, via a user study, that users who are given control over their recommender evaluate the resulting recommendations more positively.

2. RELATED WORK

There has been substantial research effort on optimizing recommender algorithms for system-determined outcome measures such as prediction accuracy, click-through rate, purchase dollars, and return visits. Some of the best-known approaches are based on nearest-neighbor (e.g., item-item collaborative filtering [19]) or matrix factorization approaches (e.g., SVD++ [14]).

Recently, there has been an emphasis on learning to rank [15], a family of machine learning techniques that are trained on a ranking-oriented loss function. There are several related approaches to this idea that seek to maximize ranking outcomes while remaining computationally feasible (e.g., [6, 4]). Learning to rank has the potential for optimizing recommendation lists by examining many sources of behavioral input, including implicit feedback such as clicks and purchases [18]. Algorithms will continue to improve in their ability to optimize system-determined outcomes. We build on this fact by seeking a method that lets users decide their own outcomes. Our recommendation method can extend any algorithm that scores or ranks items.

In this work, we develop a *hybrid* recommender — an ensemble of multiple recommender algorithms. Hybrid recommenders have been widely applied to improve prediction and recommendation accuracy [3, 13, 23, 21]. Weighted ensembles — linear combinations of multiple recommendation algorithms — are one of the most popular methods in this area [3]. For example, [1] describes introducing a simple linear ensemble model in Netflix, incorporating predicted ratings with many other features. We build on this literature by allowing the user to vary the weights, and by conducting a user study to measure user perceptions of different weights.

We are interested in more than just offline optimizations; we want happy users. Statistical accuracy metrics (e.g., accuracy, recall, and fallout) are not necessarily a good approximation of the recommendation quality perceived by users [8]. Other factors, such as transparency [22] and user control [16, 5] also affect the user experience [9].

Especially relevant to this work are systems that give users more control over their recommended items. Critiquing recommender systems [7, 17] allow users to quickly build a highly-customized, task-specific, transparent taste profile. These systems rely on content attributes and are task-specific; in this work we aim to provide users with control over longer-lived, general-purpose, content-agnostic recommendation algorithms. Research on Tasteweights [2], an interactive recommender system that allows users to directly manipulate many attributes, found that users were more satisfied when they were given control. We extend this idea by evaluating a recommendation method that works on top of existing recommendation methods, and that allows simple user manipulation.

3. RECOMMENDATION METHOD

To build a top-N recommendation list for each user, we build a personalized blending score ($s_{u,i}$) for each item, then sort by that score. The mechanism for computing these scores is kept simple so that we can learn from the results, and flexible so that we can experiment with different blending strategies.

Equation 1 shows the general form of our method — a weighted linear combination of input variables. $r_{u,i}$ is the recommender's score of item i for user u; $f_i^1, ..., f_i^n$ are numeric representations of item features, and $w_u^0, ...w_u^n$ are weights that determine the relative importance of the features.

$$s_{u,i} = w_u^0 \cdot r_{u,i} + w_u^1 \cdot f_i^1 + ... + w_u^n \cdot f_i^n \tag{1}$$

We operationalize this general method as shown in Equation 2.

$$s_{u,i} = w_u^0 \cdot pred_{u,i} + w_u^1 \cdot pop_i + w_u^2 \cdot age_i \tag{2}$$

The specific variables used are:

- $pred_{u,i}$: a prediction of the rating that user u will assign item i, as generated by an item-item collaborative filtering algorithm. We use Lenskit [10], configured to use cosine similarity scoring and a neighborhood size of 20. We clamp the output values to the range of 0.5-5 to match the values used in the live system.

- pop_i: the number of times item i was rated in the past year.[2]

- age_i: 50 minus the number of years since item i was released, clamped to the range of 0-50. We subtract from 50 so that newer items have higher values, which is more consistent with typical user preferences.

Weights $\{w^0, w^1, w^2\}$ may take any value; we experiment with weights in the range of 0-1. In addition, we examine only one of $\{w^1, w^2\}$ at a time, which we accomplish by setting the other variable's weight to zero.

We scale $pred_{u,i}$, pop_i, and age_i to the range of 0-1 by computing their percentile rank before combining them, in order to make their weights comparable. For example, to scale $pred_{u,i}$, we first compute the user's predicted rating for each item on a 0.5-5 scale, then calculate the percentile of each of those scores. In this way, we flatten the distributions in a consistent manner (as compared with simple normalization, which would preserve the existing shape of the distribution). This flattening step is critical for ratings data, such as those found in MovieLens, where the normal-shaped distribution of rating values ($\mu = 3.53, \sigma = 1.05$) causes many closely-clustered predicted rating values, thus dramatically over-weighting differences in other parameters

[2]In our offline analysis, we varied the popularity window across 3, 6, 12, and 24 months, and found no major differences in outcomes.

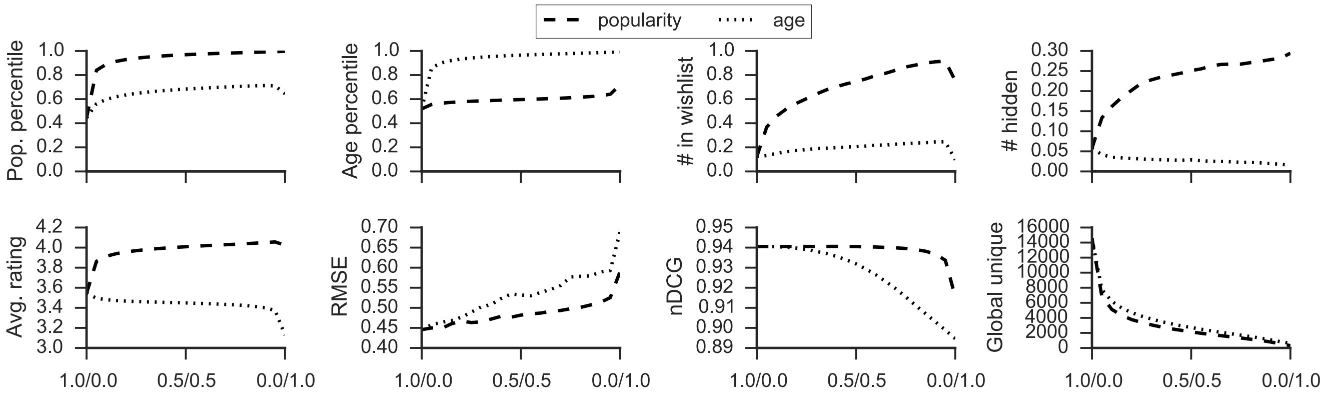


Figure 1: Simulated effects from the offline experiments that blend predicted rating with one of two non-personalized blending variables: popularity and age. The x-axis shows the algorithm weights; values range from (1.0 * predicted rating + 0.0 * blending variable) on the left to (0.0 * predicted rating + 1.0 * blending variable) on the right. That is, the leftmost point represents the unadjusted behavior of the item-item CF algorithm; as the line travels right, it shows the effect of diminishing the importance of CF while increasing the importance of the blending variable. The outcome variables (on the y-axis) are discussed in the text.

for movies in this prediction range. We note that percentile-adjusted distributions are not perfectly flat, since percentiles may have ties.

4. OFFLINE ANALYSIS

To establish the feasibility of our recommendation method, and to guide the design of our user-facing experiment, we use an offline simulation study to understand the impact of different weighting values.

4.1 Methods

This study is based on a dataset of user ratings and movie rating statistics drawn from the MovieLens database on April 2, 2015. We sample users who joined MovieLens in the six months preceding that date. We include users with at least 40 ratings, to ensure that all users could have at least 20 ratings in both their training and test sets. The resulting sample consists of 4,976 users; these users had visited MovieLens a median of 35 times and had rated a median of 253 movies.

We conduct an offline simulation study using a rating-holdout methodology [11] to simulate users' top-N recommendation lists. Specifically, we conduct 5-fold cross-validation experiments, using the LensKit evaluation framework [10], which partitions by user. In each fold, we use four groups of users to train the item-item similarity model; we measure outcomes for the remaining group. For users in that group, we choose 20 ratings at random as their training data (their simulated ratings profile), while their remaining ratings are treated as their test data.

We run two groups of simulations, one to measure the impact of each of the two blending variables: popularity ("pop") and age. For both pop and age, we test simulated outcomes with differently configured recommenders that span the range of weighting. The recommenders' weights span the following values, where the first number in the tuple is the weight assigned to predicted rating, and the second number is the weight assigned to the blending variable (pop or age):

$$(1.00, 0.00), (0.95, 0.05), ..., (0.00, 1.00)$$

We only experiment with one blending variable at a time (the other blending variable is held constant at 0) to isolate its effect.

For each configuration, and for each user, we simulate generating a top-20 list of recommendations. Using the test data for that user, we can then evaluate their ability to interact with those recommendations (e.g., to add the item to their wishlist). All reported metrics only concern these top-20 recommendations (e.g., RMSE is measured on rated items in users' top-20 simulated recommendations).

4.2 Results

See Figure 1 for a visualization of the results of this simulation. Each plot in this figure shows the simulated results in both experiments — popularity and age — on a single outcome variable, averaged across users. The x-axis represents the full range of relative weighting: the far-left point is the result of sorting by predicted rating, while the far-right point is the result of sorting by the blending variable (pop or age). The intermediate points on the x-axis span the range of weighting values as discussed above. Though the far-right point represents a recommendation algorithm that is 0% predicted rating, it is still semi-personalized because we assume in this experiment, as we do in the live system, that we do not display already-rated movies to users.

Manipulation checks. Blending a small amount of popularity strongly increases the average `pop percentile` of top-N lists; blending a small amount of age strongly increases the average `age percentile`. These effects are expected, and are included mostly as sanity checks. However, these charts also show that even small weights on the blending variable will have a dramatic influence on the resulting recommendations.

Actionable items. MovieLens users add items to a "wishlist" to express interest, and "hide" items to express a lack of interest. We simulate the presence of these items in users' top-20 lists with `# in wishlist` and `# hidden`. We find, interestingly, that increasing popularity simultaneously increases both actions, though they express opposite responses to the content. Possibly, this is due to increased familiarity with the items in the list — a user may have stronger preferences for items they have heard of. Increasing the weighting of age has the property of increasing the number of wishlisted items and decreasing the number of hidden items, though the effect is more subtle than with popularity.

Item quality. `Average rating` is a non-personalized proxy for item quality, which we measure on a 0-5 star scale. Adding a small amount of popularity dramatically increases the average rating, while adding age slightly lowers the average rating of items in users' top-20 lists.

Recommender quality. `RMSE` is a prediction accuracy metric; `nDCG` is a rank quality metric [20]. We find that both pop and age have worsening effects on both metrics, though the effects are minimal at low blending values.

Impact on personalization. By blending non-personalized factors into a recommender algorithm, we expect the system to become less personalized. We quantify this cost using `global unique`, which shows the number of unique movies recommended across all users in our sample. We find that even small weights cause a large decline in this metric — even blending in 5% popularity drops the number of unique top-20 movies from 14.5k to 6.8k, while blending 5% age drops the number to 7.7k.

5. USER EXPERIMENT

Offline analysis gives us some general understanding of the effects of blending on top-N recommendations, but does not tell us how users perceive these effects and what amount of popularity or age (if any) they choose to blend in. We therefore conduct a user experiment in which users are able to control the blending themselves.

Our user experiment has two main parts. First, we ask users to "tune" a list of recommendations using a small set of controls. Then, we ask users to complete two surveys (in random order): one about their original list of recommendations, and another (with the same questions) about their adjusted list.

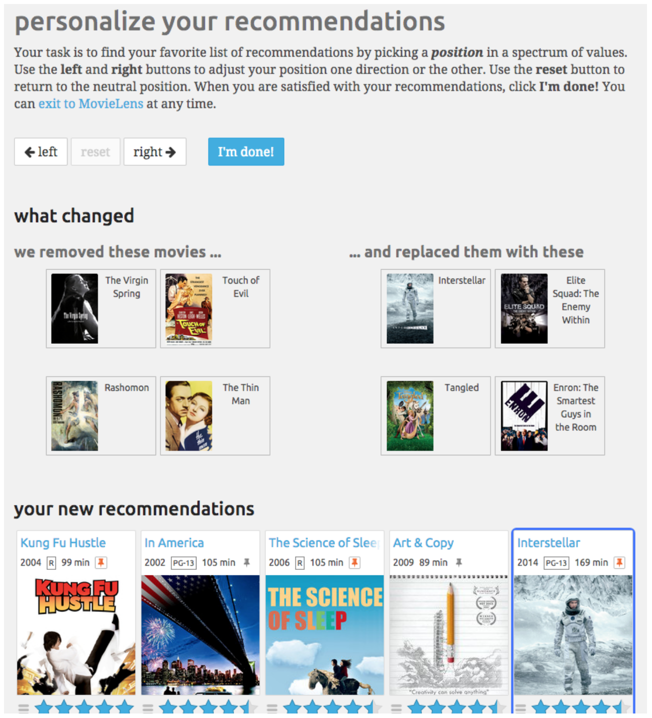
5.1 Methods

We invited 2,023 existing MovieLens users to participate in the user study between March 26 and April 16, 2015. This set of users was chosen randomly among a larger set of users who (a) had logged in during the previous six months, (b) had rated at least 15 movies, and (c) had consented to receive emails from MovieLens.

Our study has a 2x2 design with random assignment. The first variable, *condition*, determines if the user will be given control over the popularity ("pop") variable or the age variable in their recommendations. The second variable, *order*, determines the order of the surveys shown, to test and control for order effects.

Subjects first see the *recommender tuner* interface — a top-24 list of movies and several buttons for changing the contents of the list (see Figure 2). The instructions read:

> Your task is to find your favorite list of recommendations by picking a **position** in a spectrum of values. Use the **left** and **right** buttons to adjust your position one direction or the other. Use the **reset** button to return to the neutral position. When you are satisfied with your recommendations, click **I'm done!**



personalize your recommendations

Your task is to find your favorite list of recommendations by picking a ***position*** in a spectrum of values. Use the **left** and **right** buttons to adjust your position one direction or the other. Use the **reset** button to return to the neutral position. When you are satisfied with your recommendations, click **I'm done!** You can exit to MovieLens at any time.

what changed

we removed these movies ...
The Virgin Spring
Touch of Evil
Rashomon
The Thin Man

... and replaced them with these
Interstellar
Elite Squad: The Enemy Within
Tangled
Enron: The Smartest Guys in the Room

your new recommendations

Kung Fu Hustle
2004 R 99 min

In America
2002 PG-13 105 min

The Science of Slee
2006 R 105 min

Art & Copy
2009 89 min

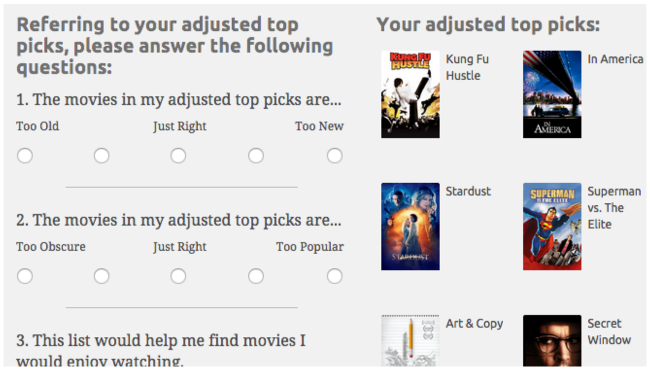
Interstellar
2014 PG-13 169 min

Figure 2: A screenshot of the experimental recommender tuner interface. Users click *left* or *right* to explore different recommender configurations. Each click changes the list by four movies — these are emphasized visually in the "what changed" section.

These instructions intentionally do not reveal how their actions control the recommendations in order to remove any cognitive bias. E.g., a user who believes "I want to watch new releases" may use the feature differently if the controls are labelled "older" and "newer" rather than the more neutral "left" and "right". Note that the offline analysis above evaluates top-20 lists; we choose 24 for this experiment because it matches the system interface.

By clicking the *left*, *right*, and *reset* buttons, the user is able to change the movies shown in the list. At the time the page is loaded (i.e., the *origin*), the top-24 list is the result of 100% predicted rating from MovieLens's item-item CF algorithm. Each subsequent click "farther" from the origin increases the weight of the blending variable (pop or age) relative to the personalization variable. Values to the "right" of the origin represent positive weights on the blending variable. As in the offline analysis, moving farther from the origin simultaneously increases the weight on the blending variable and decreases the weight on the personalization variable (to a maximum of 1.0 blending, 0.0 personalization). Values to the left of the origin represent negative weights (to a maximum of -1.0 blending, 0.0 personalization). Note that the offline analysis above does not model negative weights; we include these so that users can express "more obscure" or "older" if they wish.

Each *step* — a *right* or *left* click — changes four items in the user's top-24 list. This means that the specific amount


Referring to your adjusted top picks, please answer the following questions:

1. The movies in my adjusted top picks are...

Too Old | Just Right | Too New

2. The movies in my adjusted top picks are...

Too Obscure | Just Right | Too Popular

3. This list would help me find movies I would enjoy watching.

Your adjusted top picks:

Figure 3: A screenshot of the survey that asks users to evaluate their top-24 list. Subjects take this survey twice in random order — once for their original recommendations, and once for their hand-tuned recommendations.

by which the weights change per step varies by user. Early testing of the interface revealed that it was difficult to set an increment that worked well for all users (a given change in weights may affect half of the items in the recommendation list for one user, and not change the list for another user). Thus, we pre-computed a set of weights for each subject that would correspond to replacing four items — enough to feel the list changing, but not enough to be overwhelming. We locate a value for each step using a binary search algorithm.

To assist users in determining whether the list is better before or after the change, we show two boxes, one showing the movies that were just removed, and one showing the movies that were just added. We also call attention to the new items in the top-24 list by highlighting them.

After clicking *I'm done!*, we ask survey questions about the subject's original and tuned recommendations (see Figure 3). These questions are designed to assess their perceptions of the differences between the two lists. Several of these questions are derived from [5]. We conclude the survey with several questions concerning the general feasibility of this interface as a permanent feature in MovieLens.

5.2 Results

We consider two groups for analysis. First, for examining use of the recommender tuner interface, we look at users who (a) used both the left and the right buttons, (b) took three or more actions, and (c) clicked *I'm done!* (N=168, 8.3% of those emailed). Second, for examining survey results, we look at users from the first group who also completed the survey (N=148, 7.3% of those emailed). Both groups come from a pool of 381 users (19% of those emailed) who clicked on the email link and were assigned to an experimental condition. These samples skew towards highly-active users (e.g., users who finished the survey have a median of 186 logins and 626 ratings vs. the broader sample of emailed users who have a median of 48 logins and 254 ratings).

Overall, subjects used a median of 10 actions in tuning their list of recommendations (pop condition: median=12; age condition: median=10). 15% of subjects chose the original recommender as their favorite configuration, while the remaining 85% chose a configuration one or more steps from the initial setting. See Figure 4 for a histogram of these

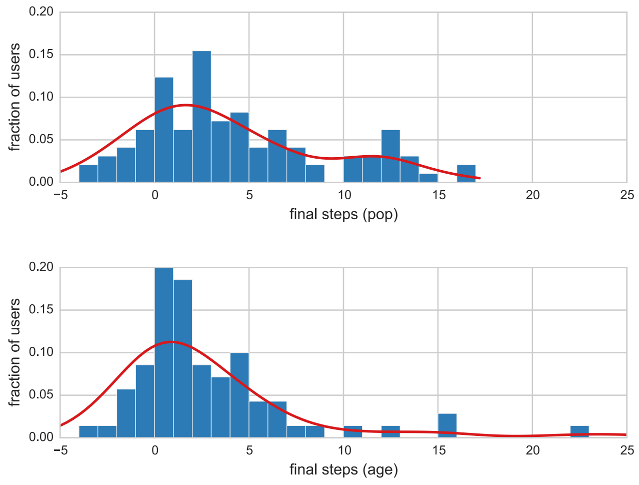


Figure 4: A histogram with a smoothed density overlay (red line) of users' final recommender selections, represented in the number of steps (i.e., clicks) away from the origin of 100% item-item CF. Each step represents a change of four movies out of the 24 shown. Negative values represent final selections with negative weights for the variable (i.e., biasing the list towards obscure or old content).

subjects' final choices (measured in number of clicks left or right from the initial configuration) and Table 1 for descriptive statistics of three outcome measures.

See Table 2 for an overview of how the top-24 movies themselves changed in terms of popularity, age, and average rating. Subjects in the pop condition tuned their recommendations to become more popular (average ratings, last year: 22 ⇒ 100), newer (average release date: 1993 ⇒ 1998), and more highly-rated (average rating: 3.58 ⇒ 3.85). Subjects in the age condition tuned their recommendations to become newer (average release date: 1995 ⇒ 2002); there were not statistically significant differences in the other metrics.

Of the 148 subjects who completed the survey, 85 are in the pop condition, 63 are in the age condition. 72 users took the *original list* survey first, while 76 users took the *tuned list* survey first. We found no order effects in any of the statistical tests described below.

Subjects in both the pop condition and the age condition responded more favorably concerning the properties of the tuned list, as compared with the original list. These results are summarized in Figure 5 and Figure 6. All differences shown in those figures concerning the distributions of responses between *original* and *tuned* are statistically significant ($p<0.001$) using a Wilcoxon rank sum test.

Subjects responded that the tuned list contained more movies they had heard of, and more movies they wanted to watch (Wilcoxon test, $p<0.001$). See Table 3 for descriptive statistics. These differences remain statistically significant when looking only at subjects in the pop condition ($p<0.001$) and the age condition ($p<0.001$).

6. DISCUSSION

RQ1: Do users like having control over their recommendations? Subjects strongly preferred their top-24 recommen-

Table 1: Descriptive statistics of users' tuned list of recommendations, by condition. ***Movies changed from original*** **indicates how many of the original 24 movies were not present in the tuned list.** ***Final step*** **indicates the number of clicks from the origin of the final choice (negative numbers are "left" of the origin).** ***Final coefficient*** **indicates the user's preferred blending weight (this number was not exposed to users).**

		pop	age
movies changed from original	min	0	0
	25%	5	4
	median	12	7
	75%	20	14.5
	max	24	24
final step	min	-4	-4
	25%	0	0
	median	3	1
	75%	6	4
	max	16	25
final coefficient	min	-0.052	-0.012
	25%	0.000	0.000
	median	0.021	0.002
	75%	0.193	0.014
	max	0.990	1.000

dation lists after using the experimental controls to adjust the popularity or age. Subjects in the popularity condition were more likely to say that their recommendations were "just right" in terms of popularity; subjects in the age condition were more likely to say that their recommendations were "just right" in terms of recency. Across both conditions, subjects reported that their adjusted lists better represent their preferences, and would better help them find movies to watch.

Subjects responded positively to a survey question asking if they would use a feature like this if it were a permanent part of the system (median "agree" on a likert scale). However, subjects responded negatively to a survey question asking if they found the interface easy to use (median "disagree" on a likert scale). Based on user feedback, we attribute a large part of this usability problem to our decision to obfuscate the experimental controls (e.g., labeling a button with "right" instead of "newer", asking them to pick "a position in a spectrum of values" instead of "a popularity setting"). Given the strongly positive feedback towards the tuned lists of recommendations, we feel it is worth investing effort in

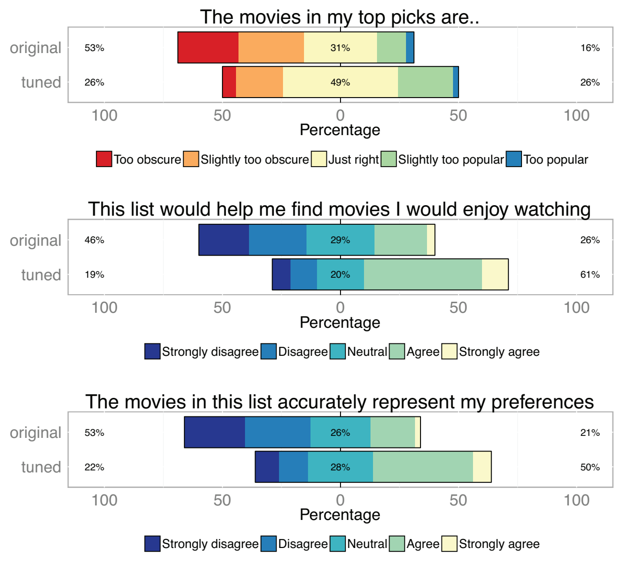


Figure 5: User survey responses from the ***popularity*** **condition. Note that the first plot uses a custom scale where the best values are in the middle ("just right"), while the other two plots use a conventional agree/disagree scale. All differences between original and tuned are statistically significant (Wilcoxon test, p<0.001).**

building usable interfaces for recommender tuning, but this remains future work.

RQ2: Given control, how different are users' tuned recommendations from their original recommendations? Different users used the controls in different ways. The median user in the pop condition changed out 12 (50%) of their original top-24 recommendations, while the median user in the age condition changed out 7 (29%). There were some users who changed out all of their movies (N=13, 7.7%), and some users who did not change any movies (N=13, 7.7%).

Offline simulation predicts several effects of our recommendation method on resulting lists. We are able to measure several of these same effects using data from the user study — popularity percentile, age percentile, and average rating — and find that subjects' recommendation lists moved in the predicted direction. These results show some striking differences in the tuned recommendation. For example, users in

Table 2: Average statistics of subjects' original and tuned top-24 recommendation lists by condition. * denotes a statistically significant difference ($p < 0.001$) using a paired t-test.

condition	metric	original	tuned
pop	pop. percentile	0.80	0.91*
	age percentile	0.51	0.57*
	avg. rating	3.58	3.85*
age	pop. percentile	0.77	0.79
	age percentile	0.53	0.65*
	avg. rating	3.49	3.47

Table 3: User responses concerning their familiarity with and desire to watch the movies in the original and tuned top-24 lists. The difference between original and tuned is statistically significant (Wilcoxon test, p<0.001).

		original	tuned
# heard of	25%	3	7
	median	7	13
	75%	12	18
# want to watch	25%	2	5
	median	5	9
	75%	10	15

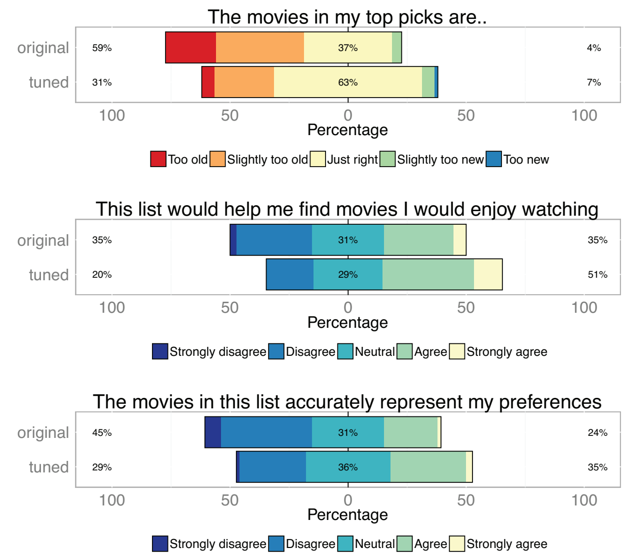


Figure 6: User survey responses from the *age* condition. Note that the first plot uses a custom scale where the best values are in the middle ("just right"), while the other two plots use a conventional agree/disagree scale. All differences between original and tuned are statistically significant (Wilcoxon test, p<0.001).

the popularity condition tuned their lists to contain movies that were rated nearly five times as often in the past year (100 vs. 22), on average.

RQ3: Do users converge to a common "best tuning" setting? There does not appear to be a "one size fits all" tuning value where users converge in their preferences, in either condition. In fact, 15% of users in the pop condition chose weights below 0.0 (to the "left" of the origin, in the interface) to encourage less popular items to appear; 17% of users in the age condition chose negative weights to encourage older items to appear. Given that the majority of user feedback in MovieLens (outside this experiment, in day-to-day operation) indicates recommendations that skew too old or too obscure, the number of users who chose negative value weights is surprising, which underscores the importance of giving the users control.

6.1 Limitations

Our algorithm uses item-item collaborative filtering as the baseline personalization algorithm. This algorithm, optimized for accuracy in the prediction task, may behave extremely differently from other algorithms [9], content-based or machine-learning algorithms in particular. It is possible that user controls would have a more subtle effect in the context of a recommender that was on average better oriented with user goals (e.g., optimized to predict movies that will be added to the wishlist).

Our experiments examine just two attributes — popularity and age — in a single context — movie recommendations. Therefore, our findings may or may not generalize to a broader pool of entity attributes in different domains; different systems will probably need to experiment to find the most useful attributes for providing users with control. In addition, our method lends itself most naturally to attributes that may be linearly scaled; categorical variables (e.g., movie genre) do not fit naturally into this framework.

Our user study was taken by power users of MovieLens. These users may have responded more favorably to our experimental interface than more typical users would have, but we cannot know. We must consider the possibility that a typical user would not be nearly as sensitive to the differences in top-24 recommended items.

Our experimental interface was poorly rated for its usability, and we received several comments about the difficulty of the task. Again, this biases the experimental sample towards users who have a higher tolerance for complexity or navigating unclear tasks.

6.2 Future Work

In this research, we address only item-level recommendations. Our method does not naturally generalize to different types of recommendation tasks, such as recommending categories ("top dark comedies") or recommending similar items ("more movies like this"). Developing and evaluating the feasibility of user-tuned recommenders for these tasks is future work.

We make the assumption that users have relatively static preferences, but in reality, a user's context is always changing (e.g., changing moods, social contexts, or physical locations). A recommender tuning that works one day may be less appropriate the next day. Understanding the relationship between context-sensitive recommenders and user-tuned recommenders — and developing interfaces that merge the two — is an interesting next step.

7. CONCLUSION

In this paper, we build and evaluate a recommender system that incorporates user-tuned popularity and recency modifiers. We find that users who are given these controls evaluate the resulting recommendations much more positively than their original recommendations: they rated the tuned recommendations to be more personalized, and identified more movies that they hoped to watch. Further, we find that there is no globally optimal setting that works for all users. Some used the experimental controls to change out every movie in their recommendation list, while others responded that their original list was optimal.

These results underscore the importance of user control — based on these data, any globally-optimized hybrid recommender we deploy will not match the desires of a large fraction of our users. Recommender systems traditionally offer no explicit control to users, and leave users guessing about the relationship between their actions and the resulting recommendations. Based on the results of this study, we want this to change — users will be happier if they are given control.

8. ACKNOWLEDGMENTS

We would like to thank Joseph Konstan and Daniel Kluver for their contributions, and MovieLens users for their participation. This material is based on work supported by the National Science Foundation under grants IIS-0808692, IIS-0964695, IIS-0968483, and IIS-1111201. This project was supported by the University of Minnesota's Undergraduate Research Opportunities Program.

9. REFERENCES

[1] X. Amatriain. Mining Large Streams of User Data for Personalized Recommendations. *SIGKDD Explor. Newsl.*, 14(2):37–48, Apr. 2013.

[2] S. Bostandjiev, J. O'Donovan, and T. HÃűllerer. TasteWeights: A Visual Interactive Hybrid Recommender System. In *Proceedings of the Sixth ACM Conference on Recommender Systems*, RecSys '12, pages 35–42, New York, NY, USA, 2012. ACM.

[3] R. Burke. Hybrid Web Recommender Systems. In P. Brusilovsky, A. Kobsa, and W. Nejdl, editors, *The Adaptive Web*, number 4321 in Lecture Notes in Computer Science, pages 377–408. Springer Berlin Heidelberg, 2007.

[4] Z. Cao, T. Qin, T.-Y. Liu, M.-F. Tsai, and H. Li. Learning to Rank: From Pairwise Approach to Listwise Approach. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, pages 129–136, New York, NY, USA, 2007. ACM.

[5] S. Chang, F. M. Harper, and L. Terveen. Using Groups of Items for Preference Elicitation in Recommender Systems. In *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work and Social Computing*, CSCW '15, pages 1258–1269, New York, NY, USA, 2015. ACM.

[6] O. Chapelle and S. S. Keerthi. Efficient algorithms for ranking with SVMs. *Information Retrieval*, 13(3):201–215, Sept. 2009.

[7] L. Chen and P. Pu. Critiquing-based recommenders: survey and emerging trends. *User Modeling and User-Adapted Interaction*, 22(1-2):125–150, Oct. 2011.

[8] P. Cremonesi, F. Garzotto, S. Negro, A. V. Papadopoulos, and R. Turrin. Looking for "Good" Recommendations: A Comparative Evaluation of Recommender Systems. In P. Campos, N. Graham, J. Jorge, N. Nunes, P. Palanque, and M. Winckler, editors, *Human-Computer Interaction - INTERACT 2011*, number 6948 in Lecture Notes in Computer Science, pages 152–168. Springer Berlin Heidelberg, 2011.

[9] M. D. Ekstrand, F. M. Harper, M. C. Willemsen, and J. A. Konstan. User Perception of Differences in Recommender Algorithms. In *Proceedings of the 8th ACM Conference on Recommender Systems*, RecSys '14, pages 161–168, New York, NY, USA, 2014. ACM.

[10] M. D. Ekstrand, M. Ludwig, J. A. Konstan, and J. T. Riedl. Rethinking the Recommender Research Ecosystem: Reproducibility, Openness, and LensKit. In *Proceedings of the Fifth ACM Conference on Recommender Systems*, RecSys '11, pages 133–140, New York, NY, USA, 2011. ACM.

[11] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl. Evaluating Collaborative Filtering Recommender Systems. *ACM Trans. Inf. Syst.*, 22(1):5–53, Jan. 2004.

[12] M. Honan. I Liked Everything I Saw on Facebook for Two Days. Here's What It Did to Me, Aug. 2014.

[13] M. Jahrer, A. Toscher, and R. Legenstein. Combining Predictions for Accurate Recommender Systems. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '10, pages 693–702, New York, NY, USA, 2010. ACM.

[14] Y. Koren. Factorization Meets the Neighborhood: A Multifaceted Collaborative Filtering Model. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '08, pages 426–434, New York, NY, USA, 2008. ACM.

[15] T.-Y. Liu. Learning to Rank for Information Retrieval. *Found. Trends Inf. Retr.*, 3(3):225–331, Mar. 2009.

[16] S. M. McNee, S. K. Lam, J. A. Konstan, and J. Riedl. Interfaces for Eliciting New User Preferences in Recommender Systems. In P. Brusilovsky, A. Corbett, and F. d. Rosis, editors, *User Modeling 2003*, number 2702 in Lecture Notes in Computer Science, pages 178–187. Springer Berlin Heidelberg, 2003.

[17] P. Pu and L. Chen. User-Involved Preference Elicitation for Product Search and Recommender Systems. *Ai Magazine*, 29(4):93–103, 2008.

[18] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme. BPR: Bayesian Personalized Ranking from Implicit Feedback. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, UAI '09, pages 452–461, Arlington, Virginia, United States, 2009. AUAI Press.

[19] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based Collaborative Filtering Recommendation Algorithms. In *Proceedings of the 10th International Conference on World Wide Web*, WWW '01, pages 285–295, New York, NY, USA, 2001. ACM.

[20] G. Shani and A. Gunawardana. Evaluating Recommendation Systems. In F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, editors, *Recommender Systems Handbook*, pages 257–297. Springer US, 2011.

[21] J. Sill, G. Takacs, L. Mackey, and D. Lin. Feature-Weighted Linear Stacking. *arXiv:0911.0460 [cs]*, Nov. 2009. arXiv: 0911.0460.

[22] R. Sinha and K. Swearingen. The Role of Transparency in Recommender Systems. In *CHI '02 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '02, pages 830–831, New York, NY, USA, 2002. ACM.

[23] L. Tang, Y. Jiang, L. Li, and T. Li. Ensemble Contextual Bandits for Personalized Recommendation. In *Proceedings of the 8th ACM Conference on Recommender Systems*, RecSys '14, pages 73–80, New York, NY, USA, 2014. ACM.