On Bootstrapping Recommender Systems

Nadav Golbandi
Yahoo! Labs, Haifa, Israel
nadavg@yahoo-inc.com

Yehuda Koren
Yahoo! Labs, Haifa, Israel
yehuda@yahoo-inc.com

Ronny Lempel
Yahoo! Labs, Haifa, Israel
rlempel@yahoo-inc.com

ABSTRACT

Recommender systems perform much better on users for which they have more information. This gives rise to a problem of satisfying users new to a system. The problem is even more acute considering that some of these hard to profile new users judge the unfamiliar system by its ability to immediately provide them with satisfying recommendations, and may be the quickest to abandon the system when disappointed. Rapid profiling of new users is often achieved through a bootstrapping process - a kind of an initial interview - that elicits users to provide their opinions on certain carefully chosen items or categories. This work offers a new bootstrapping method, which is based on a concrete optimization goal, thereby handily outperforming known approaches in our tests.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*Data Mining*

General Terms

Algorithms

Keywords

collaborative filtering, new user, recommender systems, user cold start

1. INTRODUCTION

Modern consumers are inundated with choices. Electronic retailers and content providers offer huge selections of products, with unprecedented opportunities to meet a variety of special needs and tastes. Matching consumers with the most appropriate products is not trivial, yet is key to enhancing user satisfaction and loyalty. This motivates the study of recommender systems, which analyze patterns of user interest in items or products to provide personalized recommendations of items that will suit a user's taste.

One particular challenge that recommender systems face is handling new users; this is known as the *user cold start problem*. The quality of recommendations strongly depends on the amount of data gathered from the user, making it difficult to generate reasonable recommendations to users new to the system. In order to quantify this point, Fig. 1 shows how the error on Netflix test data decreases as users provide more ratings. Users that have vested many ratings in the system, can enjoy error rates around 0.85, whereas new users, with just a few known ratings, are served with a significantly higher error rate (around 1). Yet, new users are crucial for the recommendation environment, and providing them with a good experience is essential to growing the user base of the system. Pleasing these new users is all the more challenging, as they often judge the system's value based on their first few experiences. This is particularly true for systems based on explicit user feedback, where users are required to actively provide ratings to the system in order to get useful suggestions. The essence of bootstrapping a recommender system is to promote this interaction, encouraging users to invest in a low-effort initial interaction that will lead them to an immediately rewarding experience.

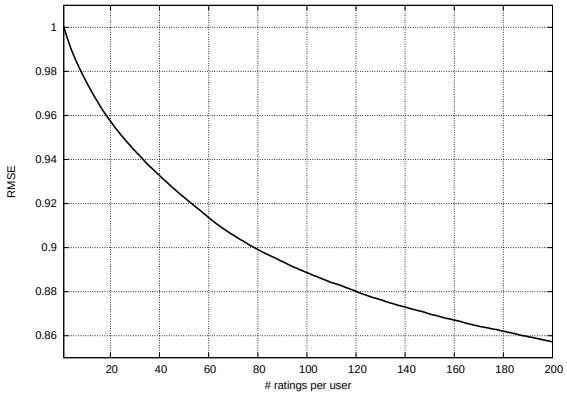

Figure 1: The test error rate vs. number of train ratings per user on the Netflix data. Lower y-axis values represent more accurate predictions. The x-axis describes the exact number of ratings taken for each user. When the x value equals k, we are considering only users that gave at least k ratings. For each such user, we sort the ratings in chronological order and take the first k ratings into account. Results are computed by the factorized item-item model [5].

This paper introduces a method for eliciting information from new users by asking their feedback on a few deliberately chosen items. The method involves creating a seed set of items based on optimizing a formally defined cost function, thereby handily outperforming previous approaches.

CIKM'10, October 26–30, 2010, Toronto, Ontario, Canada.

2. NOTATION AND DATASET

We are given ratings for m users within user set $\mathcal{U} = \{1, \ldots, m\}$ and n items within item set $\mathcal{I} = \{1, \ldots, n\}$. We reserve special indexing letters to distinguish users from items: for users u, v, and for items i, j. A rating r_{ui} indicates the preference by user u of item i, where high values mean stronger preference. For example, values can be integers ranging from 1 (star) indicating no interest to 5 (stars) indicating a strong interest. Usually the data is sparse and the vast majority of ratings are unknown. For example, in the Netflix data, 99% of the possible ratings are missing because a user typically rates only a small portion of the movies. We distinguish predicted ratings from known ones, by using the notation $\hat{r}_{ui}$ for the predicted value of r_{ui}. The set of users rating item i (in the training set) is denoted by $\mathrm{R}(i)$.

Our test bed is a large movie rating dataset released by Netflix as the basis of a well publicized competition [1]. The dataset contains more than 100 million date-stamped ratings performed by about 480,000 anonymous Netflix customers on 17,770 movies between Nov 11, 1999 and Dec 31, 2005. Ratings are integers ranging between 1 and 5. We evaluate our methods on a separate test set ($\mathcal{T}$) designed by Netflix, which contains over 2.8 million ratings (also known as the "Qualifying set"). The quality of the results is measured by their root mean squared error (RMSE) $\sqrt{\sum_{(u,i)\in\mathcal{T}}(r_{ui} - \hat{r}_{ui})^2/|\mathcal{T}|}$.

3. RELATED WORK

A popular approach to building recommender systems is *Collaborative Filtering* (CF), a term coined by the developers of the first recommender system - Tapestry [2]. CF relies only on past user behavior, e.g., their previous transactions or product ratings. It analyzes relationships between users and interdependencies among products, in order to identify new user-item associations. A major appeal of CF is that it is domain free, yet it can address aspects of the data that are often elusive and difficult to profile.

The two primary areas of CF are *neighborhood methods* and *latent factor models*. Neighborhood methods compute the relationships between items or, alternatively, among users. The item-item approach [7, 10] evaluates the preference of a user to an item based on ratings of "neighboring" items by the same user. An item's neighbors are items that tend to be scored similarly when rated by the same user. Latent factor models are an alternative approach that explains ratings by characterizing both items and users on factors inferred from the pattern of ratings. One of the most successful realizations of latent factor models is based on *matrix factorization*, e.g., [6].

3.1 Incremental collaborative filtering

When bootstrapping a recommender system, each user profile starts with no information, and gradually accumulates more ratings. This greatly favors CF methods that can handle incremental user input, i.e. methods that can easily accommodate new ratings entered by the user, without requiring to recompute the underlying model or any other major adaptation. Indeed, not all CF methods have this quality. For example, methods that require explicit representation of user-related parameters are less suitable, as any newly received user ratings would require modifying those parameters. Accordingly, methods that compute user-user correlations are less appropriate, because they require a constant recomputation of those correlations as the system accumulates user interactions. Similarly, the most common forms of the aforementioned matrix factorization method are less desirable, as they rely on associating a user factor vector with each user, which must change with the growing number of user ratings. On the other hand, some of the more popular CF methods do not require a parameterization of user-dependent values. Among the best known examples is the item-item neighborhood approach, whose parameters depend solely on the items in the system. Item characteristics are usually stable and barely affected by the few ratings added by a user, thereby making the underlying item-item model robust to incremental growth of user profiles. Past efforts on item-item models focused on computing the correlations between items [7, 10]. Later works achieved significantly better performance by a more principled modeling approach. In particular, we concentrate on the factorized item-item approach [5], which combines close to the best known accuracy with significant space and time-complexity improvements. This allows us to compare our methods to one of the best performing ones.

3.2 Providing recommendations to new users

Relatively few prior works deal with handling of users new to the recommender system. An early attempt was [4]. A comprehensive study was conducted by the GroupLens team [8], and was later enhanced [9]. The main theme of these works is the construction of a specialized item set on which new users will be asked to provide their ratings. We refer further to these works in the next section.

4. BASELINES FOR SEED SET SELECTION

Learning the preferences of new users can be efficiently achieved by a short interview during which they are asked to rate a few, carefully selected items of a *seed set*. One might intuitively expect the items in such a seed set to have certain properties, leading to various selection criteria which we discuss in the followings.

Popularity. A prime guideline is that the seed set should be biased toward familiar items, since asking users to rate obscure items is mostly futile. Thus, a reasonable strategy is to ask the user about the most popular items, i.e., those that have received the most ratings in the past [8]. In the following we dub this approach *Popularity*.

Contention. Items in the seed set should not only be familiar to the users, but also indicative of their tendencies. After all, finding that a user liked an item that is also liked by everyone else, provides less insight than discovering that a user likes a more controversial item. Two common measures to quantify the contention associated with an item are the variance and entropy of its ratings [4, 8, 9]. For example, a seed set can be populated with the items of highest entropy (*Entropy*).

However, as identified in [8], maximizing a contention-related criterion with disregard to item popularity is unwise. There is little point in presenting highly controversial, yet lesser known items, which are highly indicative only with respect to a narrow group of users that are familiar with them. Furthermore, it has been claimed [8] that the widest spread in rating pattern (indicating contention) tends to be associated with those items receiving fewer ratings, while popular items are less controversial. In this sense, contention is negatively correlated with popularity.

Therefore, contention is only useful when combined with popularity. Among top performing methods in [9] are hybrids of popularity and entropy such as *Entropy0* and *HELF*. Another measure amplifies variance by multiplying it with the square root of the item popularity: $\sqrt{|\mathrm{R}(i)|} \cdot \mathrm{Var}(i)$[1]. This contention measure will be henceforth dubbed *Var*.

Coverage. Items are useful when they possess predictive power

[1]See a post in The Netflix Prize Forum on that measure – `http://www.netflixprize.com/community/viewtopic.php?id=164&p=1`

on other items. Controversial items are not necessarily best at this. For example, within the Netflix movie rating dataset, most contention metrics identify *Napoleon Dynamite* as a highly controversial movie, arising much disagreement among users. Yet, it is well known that ratings of Napoleon Dynamite are very weakly correlated with ratings of other movies. This is an example of a movie which, despite being ranked high on the contention and familiarity axes, is less useful within the seed set, as it is not very instructive on the perceived quality of other items. We tried measuring the predictive ability of an item in different ways. The one we picked, which we call *coverage*, is defined as: $\text{coverage}(i) = \sum_j n_{ij}$. Here, n_{ij} denotes the number of users that rated both items i and j. The intuition is that CF systems infer patterns across co-rated items. Hence, items that are more heavily co-rated than others allow systems to better understand users. Note that this measure also accounts for popularity, as it is biased toward items receiving many ratings.

5. SEED SET SELECTION AS AN OPTIMIZATION PROCESS

The approaches discussed so far use various criteria to select a small seed set of items to be presented to a new user. Most of these criteria were suggested in prior works, and usually involve a combination of contention and popularity of the shown items. Yet, we would like to criticize certain aspects of these approaches:

1. *Arbitrariness of selection criteria.* The criteria presented above come from different, sometimes conflicting, desiderata concerning the seeding items. While popularity, contention and coverage are all sensible criteria, they are still detached from any reasonable end goal of optimizing user experience. Furthermore, to make matters worse, note that none of those desired principles is adequate by itself. This gave rise to arguably less natural combinations of different measures, which essentially lead to a less principled process. Instead, we target a process driven by a formal yardstick, which is well tied with end-user experience, and ideally is both simple and natural.

2. *Independence of selected items.* All above criteria score the utility of each single item independently, and consequently select those of highest utility. No consideration is given to how all these items play in tandem. For example, the second selected item might merely be a version (or, a sequel) of the first - after all, it enjoys the same fundamental properties that got the first item selected. Instead, we desire a system that optimizes the utility of the full set of seed items.

To resolve these issues, we devise a principled approach without a need to balance between conflicting criteria.

5.1 A new error aware approach

The ultimate goal of seed set construction is to maximize user satisfaction due to the subsequent generated recommendations. The literature quantifies user satisfaction by various formal measures. Most common are convenient error metrics like the aforementioned RMSE and the closely related MAE (mean absolute error). Other useful measures evaluate the accuracy of the top-K suggestion, such as recall, precision, and area under the ROC curve. (See [3] for a discussion on measuring performance of recommender systems.) In what follows, the exact optimization metric is not important, as long as it can be computed efficiently from the test set.

Formally, let $\mathcal{A}$ denote the prediction algorithm, and fix the train set. Algorithm $\mathcal{A}$ is parameterized by the k-item seed set S. That is, it is making predictions of user preferences by considering, for each individual user, only her train ratings of the items in S. The performance measure (on the train set) is denoted by the function $\mathcal{F}$ (we assume, without loss of generality, that the goal is *minimizing* this measure). We seek a seed item set S such that:

$$S = argmin_{S \subset \mathcal{I}, |S|=k} \mathcal{F}\left(\mathcal{A}(S)\right) \quad (1)$$

Concretely, this work considers the prediction algorithm $\mathcal{A}$ to be the factorized item-item model [5]. The cost function that we minimize is RMSE (on train set). The seed set S is incrementally grown by a greedy algorithm, which iteratively adds to S the item: $argmin_{i \in \mathcal{I}-S} \mathcal{F}\left(\mathcal{A}(S \cup \{i\})\right)$.

We call this method *GreedyExtend*. In comparison to the approaches described in the previous section, GreedyExtend explicitly accounts for the end goal of optimizing prediction accuracy during the construction of the seed set. It adopts a well defined and user-meaningful optimization criterion, rather than balancing and patching together multiple construction guidelines. Desired properties of the seed items —contention, popularity, coverage and list-diversity— are a side outcome of the selection process, to the extent they actually improve the utility of the set.

6. COMPARISON OF ITEM SELECTION CRITERIA

In order to evaluate the performance of the selection criteria listed above, we tested them on the Netflix test data. Efficient experimentation is facilitated by the good performance of the aforementioned factorized item-item model, which allows to quickly process new user ratings without needing to recompute the model.

For each of the discussed item ordering criteria, we ordered all movies in the Netflix data, and picked each of the first 200 prefixes (of length 1 to 200), as a set of movies to be presented to new users. For each of the 200 resulting seed sets, we predict test ratings for every single user, while accounting only for that user's train ratings that intersect with the seed set. Error rates of the different criteria are depicted in Fig. 2. As expected, prediction accuracy improves (RMSE drops) as seed sets increase in size. However, the different ordering criteria fare differently.

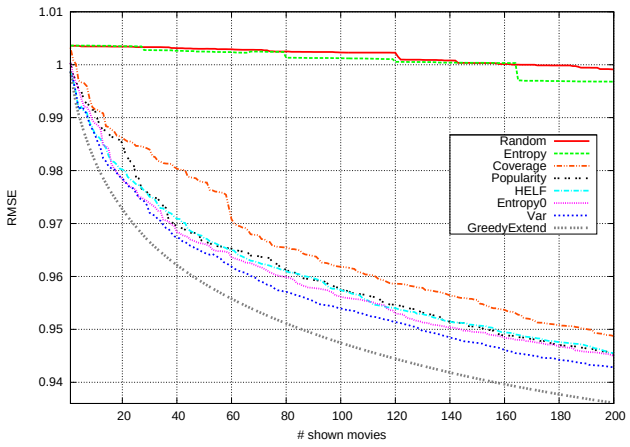

Figure 2: The test error rate vs. number of displayed items (=size of seed set), for various methods of selecting seed set items. Methods that disregard item popularity (Random and Entropy) significantly lag in performance. GreedyExtend delivers the best performing seed sets by guiding the set creation process with a suitable cost function. Note that the legend orders methods by their performance.

rank	Movie title	RMSE	Seed-set size required by Var
1	The Royal Tenenbaums	0.99642	3
2	Miss Congeniality	0.99210	4
3	Pearl Harbor	0.98897	8
4	Lost in Translation	0.98625	11
5	Sweet Home Alabama	0.98383	13
6	Pulp Fiction	0.98171	15
7	The Day After Tomorrow	0.97992	18
8	Independence Day	0.97845	20
9	Maid in Manhattan	0.97694	23
10	Pretty Woman	0.97566	26
11	Gone in 60 Seconds	0.97449	28
12	Being John Malkovich	0.97344	29
13	Mr. Deeds	0.97253	30
14	Kill Bill: Vol. 1	0.97166	31
15	How to Lose a Guy in 10 days	0.97082	33

Table 1: The 15-movie seed set produced by GreedyExtend on the Netflix data, along with the test error achieved by eliciting ratings on each prefix of the seed set. The right-most column compares GreedyExtend's performance to that of its closest competitor - *Var*. The column reports the size of seed sets that Var requires in order to match GreedyExtend's RMSE in each line.

Our proposed GreedyExtend method, which is the only one based on a formal cost function, consistently improves test set prediction accuracy over other methods.

As for the other methods, pure entropy, which disregards popularity, performs only slightly better than a random choice, and markedly worse than the other criteria. This confirms the claims [8] on the necessity to integrate popularity with contention. While all other orderings perform much better, we observe that Coverage is under-performing the measures based on contention. This shows the value of emphasizing the more controversial items, which Coverage disregards. The two entropy based orderings – Entropy0 and HELF– perform in line with the simple Popularity ordering. Finally, the Var ordering is the best performer among contention-based methods.

While we believe running time is not a crucial factor in a one-time preparation of a seed set, we note that GreedyExtend is significantly slower to run compared to the ordering criteria mentioned in Section 4. On the Netflix data, it took 8 hours for our hardware to compute the seed sets of sizes 1 to 200. This is compared to an almost instantaneous execution for the other ordering criteria. One can adopt several strategies to accelerate running time, if necessary. First, the greedy set extension operation is highly parallelizable. Second, unlikely items can be pruned from the candidate item set $\mathcal{I}$. For example, we empirically observed that the method avoids selecting unpopular items, which constitute the majority of the items, so their exclusion from the process significantly speeds run time.

Table 1 reports the actual ordered list of top 15 seed movies selected by GreedyExtend, together with the RMSE values corresponding to each prefix of the list. One familiar with the movies would recognize that they actually conform to the design criteria suggested in the previous section, despite those criteria having not been explicitly employed here. The listed movies are indeed well familiar, most are controversial and constitute a quite diverse list. Interestingly, a movie uncorrelated with others, yet popular and controversial, such as Napoleon Dynamite, did not make it at all into the top-200 list (unlike in previously described methods). For comparison we also report the sizes of seed sets produced by Var that yield comparable error rates. Note that by eliciting feedback for 10 movies, GreedyExtend achieves RMSE=0.97566, a bar that its closest competitor, Var, meets only after presenting as many as 26 movies.

7. CONCLUSIONS

Introducing a new user into a recommender system should require a low effort bootstrapping process, after which the user can immediately appreciate the value provided by the system. System designers trying to impress their new users, who might be the most judgmental ones, should look at methods providing best prediction accuracy at minimal distraction to the user. This usually proceeds by conducting a short interview with the user, where she is asked to evaluate certain products or categories. The prior few works dealing with the design of such an interview concentrated on several, possibly conflicting properties expected by the evaluated items. This work has shown that accuracy of the process significantly improves when it is driven, from its very beginning, by optimizing a natural cost function. To get a feeling of the achieved improvement, an error rate that could have been previously achieved after soliciting 30 user evaluations, becomes achievable with just 13 user evaluations.

8. REFERENCES

[1] J. Bennett and S. Lanning. The Netflix Prize. *Proc KDD Cup and Workshop*, 2007.

[2] D. Goldberg, D. Nichols, B. M. Oki and D. Terry. Using Collaborative Filtering to Weave an Information Tapestry. *Communications of the ACM* 35:12, 61–70, 1992.

[3] J. Herlocker, J. Konstan, L. Terveen and J. Riedl. Evaluating Collaborative Filtering Recommender Systems. *ACM Transactions on Information Systems* 22:1, 5–53, 2004.

[4] A. Kohrs and B. Merialdo. Improving Collaborative Filtering for New Users by Smart Object Selection. *Proc. International Conference on Media Features (ICMF)*, 2001.

[5] Y. Koren. Factor in the Neighbors: Scalable and Accurate Collaborative Filtering. *ACM Transactions on Knowledge Discovery from Data* 4:1, 2010.

[6] Y. Koren, R. Bell and C. Volinsky. Matrix Factorization Techniques for Recommender Systems. *IEEE Computer* 42:8, 30–37, 2009.

[7] G. Linden, B. Smith and J. York. Amazon.com Recommendations: Item-to-item Collaborative Filtering. *IEEE Internet Computing* 7:1, 76–80, 2003.

[8] A. M. Rashid , I. Albert , D. Cosley , S. K. Lam , S. M. Mcnee , J. A. Konstan and J. Riedl. Getting to Know You: Learning New User Preferences in Recommender Systems. Proc. 7th international conference on Intelligent user interfaces (IUI), 127–134, 2002.

[9] A. M. Rashid, G. Karypis and J. Riedl. Learning Preferences of New Users in Recommender Systems: An Information Theoretic Approach. *SIGKDD Explorations* 10:2, 90–100, 2008.

[10] B. Sarwar , G. Karypis , J. Konstan and J. Riedl. Item-based Collaborative Filtering Recommendation Algorithms. *Proc. 10th international conference on WWW*, 285–295, 2001