A More Comprehensive Offline Evaluation of Active Learning in Recommender Systems

Diego Carraro
Insight Centre for Data Analytics
University College Cork, Ireland
diego.carraro@insight-centre.org

Derek Bridge
Insight Centre for Data Analytics
University College Cork, Ireland
derek.bridge@insight-centre.org

ABSTRACT

Offline evaluation of Active Learning in recommender systems involves simulated users who, when prompted by the Active Learning strategy, may reveal ratings that were previously hidden from the recommender system. Where the literature describes offline evaluation of Active Learning, the evaluation is quite narrow: mostly, the focus is cold-start users; the impact of newly-acquired ratings on recommendation quality is usually measured only for those users who supplied those ratings; and impact is measured in terms of recommendation accuracy. But Active Learning may benefit mature users, as well as cold-start users; in recommender systems that use collaborative filtering, the newly-acquired ratings may have an impact on recommendation quality even for users who did not supply any ratings; and the new ratings may have an impact on aspects of recommendation quality other than accuracy (such as diversity and serendipity).

In this paper, we present the offline method that we are using to evaluate Active Learning. For reproducibility and to provoke discussion, we make its details as explicit as possible. Then we use this evaluation method in a case study to demonstrate why offline evaluation needs to be more comprehensive than it has been up to now. With just a single dataset and a few very simple Active Learning strategies, we are able to show trade-offs between strategies that would not be revealed otherwise.

KEYWORDS

Recommender systems, Active Learning, Offline evaluation, Beyond-Accuracy

ACM Reference format:
Diego Carraro and Derek Bridge. 2018. A More Comprehensive Offline Evaluation of Active Learning in Recommender Systems. In *Proceedings of Workshop on Offline Evaluation for Recommender Systems at the ACM Conference on Recommender Systems, Vancouver, Canada, October 2018 (REVEAL 2018),* 8 pages.

1 INTRODUCTION

Personalized recommender systems acquire user profiles; from the profiles, they build a model; using the model, they select and rank candidate items to decide which items to recommend to individual users. Other things being equal, the better the profiles, the better the model; and the better the model, the better the recommendations. Profiles are typically populated during a sign-up process, where a new user states her preferences for a small subset of the items. Subsequently, profiles grow either by observing the user's interactions with items (in the case of implicit ratings) or by acquiring the user's opinion of an item after she has consumed it (in the case of explicit ratings). But it is also possible for a recommender system to grow a user's profile by using Active Learning (AL). In AL, the recommender system proactively solicits the user's opinions for selected items.

Different AL strategies take different approaches to identifying which items to ask the user about. In general, a good AL strategy asks the user for as little information as possible while obtaining the most benefit. In selecting items to ask about, a good AL strategy will choose items which it predicts are familiar to the user (since there is little point in asking about items the user is unlikely to be able to rate) and items whose ratings will improve the quality of the recommendations. Most AL strategies take the form of simple heuristics, but there are more complex strategies that involve building a model from the existing ratings.

There is a growing body of work on AL strategies for recommender systems; see survey papers such as [6], for example. But the area is still under-explored. We identify at least the following three opportunities to extend the scope of AL in recommender systems.

First, despite being a general approach to the acquisition of additional ratings, in recommender systems AL has mostly been used in situations where there is a clear lack of data, i.e. to solve the specific problems of cold-start users — new and low-activity users whose profiles contain little data. However, AL's generality suggests that it could also be employed for more mature users (by which we mean users who have more data in their profiles), e.g. to regularly acquire fresh data and stimulate new recommendations.

Second, the benefit of AL to recommender systems has almost exclusively been measured in terms of its effect on recommendation accuracy for these cold-start users. But, it is well-known by now that satisfaction with recommender systems is not just a question of their accuracy. In some cases, satisfaction can be increased by ensuring that a set of recommendations is diverse or that it contains serendipitous items, for example. For both cold-start users and more mature users, we want to know how existing AL strategies impact on these other measures of satisfaction, and we may even want to design new AL strategies that target these measures. For example, there are new approaches to diversification, known as intent-aware approaches, which produce more diverse sets of recommendations by making recommendations that cover the user's different interests (as revealed by the user's profile) [16]. In a recommender that uses intent-aware diversification, an AL strategy that acquires more ratings in order to make a user's profile more representative of her interests may result in a more satisfactory level of diversity.

REVEAL 2018, Vancouver, Canada
2018.

Third, the benefit of AL to recommender systems has almost exclusively been measured in terms of its effect on recommendation accuracy for the users who are asked to provide new ratings. But, in collaborative filtering recommenders, new ratings might influence the recommendations made to other users too, not just those participating in the AL. This is why in [5] Elahi et al. measure the system-wide impact of AL. In practice, both must be measured: there is a time cost, and perhaps a cognitive cost, to providing a rating, so those who participate must generally benefit, but those who are not participating should also benefit or, at least, see no worsening of performance.

It is clear that when evaluating an existing AL strategy, or designing and evaluating a new strategy, we need to do so as comprehensively as possible: we need to look at different kinds of users (cold-start versus mature, those who participate in the AL versus those who do not, for example) and a wider range of performance measures (both accuracy and beyond-accuracy).

As with the evaluation of recommender systems, preliminary evaluation of AL strategies for recommender systems must be done offline, with some kind of simulation on a pre-collected dataset. Promising strategies can be further evaluated both in user trials and online in, for example, A/B experiments within a deployed recommender system. The offline evaluation can help to narrow the number of strategies that need to be evaluated in costly user trials and online experiments.

Offline evaluation of recommender systems typically involves a snapshot, measuring how one or more models, built from training data, perform on one or more test sets. Offline evaluation of AL strategies for recommender systems, on the other hand, requires that we simulate the behaviour of users and measure recommender performance both before and after the application of the strategy to see how performance changes. This requires that the dataset be split into at least three parts: the known ratings, the hidden ratings and the test set. The known ratings are the ones on which the initial recommender model is built; the known ratings are also the ones on which the AL strategy generally operates. The hidden ratings are the ones which the simulated user might reveal to the system if prompted to do so by the AL strategy. Subsequently, these elicited ratings can be added to the known ratings, and a new recommender model can be built. The performance of the initial recommender model and the new recommender model are measured against the ratings in the test set. We refer to this as a single-step AL evaluation, since it measures the effect of a single application of the AL strategy, and this is the focus in this paper. In the last section of the paper, we discuss multi-step AL evaluation, which simulates more than one application of the strategy.

The first contribution of this paper is to present the offline evaluation method that we are using. It draws on existing methods, especially [5]. But, for reproducibility and to provoke discussion, we make more of the details explicit than is commonly done. The design of an offline evaluation of AL strategies is not trivial. Done incorrectly, it can introduce bias in the results.

Our second contribution is to demonstrate why offline evaluation needs to be more comprehensive (as described above) than it has been up to now. With just a single dataset and a few very simple AL strategies, we are able to show that the fuller picture afforded by being more comprehensive may change the choice of AL strategy or strategies to adopt. By extension, more comprehensive evaluation of this kind is likely to be even more valuable for more nuanced strategies and other datasets.

The rest of this paper is structured as follows: Section 2 reviews the literature on offline evaluation of AL strategies for recommender systems. Section 3 formally describes the experimental methodology that we use for offline evaluation of AL strategies. Section 4 presents a sample of some of the results we have obtained by using the methodology to compare three simple AL strategies. It takes the form of a case study that illustrates the importance of taking a more comprehensive approach. Finally, Section 5 draws conclusions, discusses some remaining issues, and offers ideas for future work.

2 RELATED WORK

Most of the published work on Active Learning in recommender systems focuses on presenting new AL strategies. Since this paper is about offline evaluation of AL, we will not summarize any of these strategies.

In early work, AL was used to improve recommendations either for users who had small user profiles (cold-start users) or new users (sometimes known as extreme cold-start users). For these new users, AL has been used in the sign-up process to assist the user in building her initial profile [14]. AL approaches for cold-start users include conversational [2], Bayesian [8], personality-based [7] and decision-tree-based [10]. Again, we will not describe their details because our focus is on offline evaluation. But this does highlight the point we have already made, that a lot of the work focuses on cold-start users and it neither targets, nor evaluates, the impact of AL on more mature users. To the best of our knowledge, only Carenini et al. [2] and Elahi et al. [5] consider non-cold-start users. Carenini et al. describe a conversational approach to be used *after* the sign-up process. However, even this work is quite narrow since its evaluation is confined to users who have 50 ratings; no other scenarios are considered. By contrast, in the work of Elahi et al. the AL strategy is applied to all the users, irrespective of the sizes of their profiles. Therefore, Elahi et al.'s evaluation includes cold-start users but more mature users too. Unfortunately, they do not then break their results down. We cannot therefore discern whether a strategy has a different impact on cold-start users from the one it has on more mature users.

Elahi et al. also make the point that all of the offline evaluations of AL strategies that are reported in the literature (apart from their own) take what they call a *user-centric* approach: results are reported only for the subset of simulated users who participate in the AL (typically again, just the cold-start users). Elahi et al. pioneer the idea of a *system-wide* evaluation, in which they report the impact of the AL on the whole user population. This is important in recommender systems that use collaborative-filtering where new ratings might influence the recommendations made to other users, not just those participating in the AL. In Section 3.3, we break down the sets of users even further.

Finally, we note that in early work, the impact of the AL strategy on the quality of the recommendations is measured by computing prediction error on the test set using metrics such as MAE or RMSE,

e.g. [15]. In the same way that prediction error has been displaced in recommender system evaluation by metrics such as precision, recall and nDCG, the same has happened in the evaluation of AL strategies, e.g. [5]. A few of the evaluations that are reported in the literature employ other metrics such as average popularity [7], coverage [5], and spread [7, 13]. But it also recognized nowadays that satisfaction with recommendations is not just a question of their accuracy [12]. It may be desirable for a set of recommendations to be diverse or for the recommendations to be serendipitous. A wide range of metrics has been proposed to measure these 'beyond-accuracy' aspects of recommendation quality, especially for offline evaluation, e.g. [9]. To the best of our knowledge, none of the evaluations of AL that are reported in the literature has ever used these measures.

We conclude by observing that the literature does contain reports of user trials of AL. For example, [14] and [3] both report user trials that ran in association with the live MovieLens system. The focus in [14] was new users (extreme cold-start); the focus in [3] was "recently-active users".

3 EVALUATION METHODOLOGY

In this section we formally describe the more comprehensive offline evaluation methodology that we use.

3.1 Setup

The first step consists in preparing the dataset for the experiment's execution. We will assume a dataset of explicit ratings (like the one we use for our experiments in Section 4). However, the principles should easily adapt to implicit ratings datasets.

Let U be the set of users and I the set of items; let R be the ratings matrix with r_{ui} being the rating that $u \in U$ gives to item $i \in I$ or $r_{ui} = \perp$ if u has not rated i; and let R_u be u's ratings (the ones in her profile), i.e. $R_u = \{r_{ui} \mid r_{ui} \neq \perp\}$.

We start by eliminating users who have too few ratings. Each user must have enough ratings so that we can meaningfully partition them for the purposes of the experiment. In Section 4, we use 75 as the minimum number of ratings, but it will differ from dataset to dataset.

Next, we randomly partition the ratings R into training (R^{TR}) and test sets (R^{TE}). We do this in a user-based way. For each user, we randomly partition R_u so that a fixed proportion p are in R_u^{TR} and the rest are in R_u^{TE}. Then $R^{TR} = \bigcup_{u \in U} R_u^{TR}$ and $R^{TE} = \bigcup_{u \in U} R_u^{TE}$. By this approach, every user has a set of test ratings. We do this in order to allow for system-wide evaluation, i.e. measuring the impact of an AL strategy on recommendations on all users U, not just those who participate in the AL. For example, in Section 4, we use $p = 0.8$.

We want to evaluate what we previously called a single AL step, i.e. performance before and after using an AL strategy. If we do this for just one user, we are unlikely to see any impact. In even the best of circumstances, so few new ratings will be acquired that the difference in recommender performance will be negligible. So, in the same style as previous work [8, 13], we randomly select a *group* of users to whom the AL strategy will be applied. Let's call these the Active users, $U_{Active} \subseteq U$. For example, in Section 4, we select one-third of users to be Active users.

For each active user in $u \in U_{Active}$, we now need to define a set of ratings initially known by the recommender (R_u^K) and a set of ratings hidden from the recommender (R_u^H), i.e. the ones which might be elicited by means of active learning. Therefore, for each active user $u \in U_{Active}$, we randomly partition u's training set ratings R_u^{TR} into R_u^K and R_u^H. The proportion p_u of R_u^{TR} that are placed in R_u^K is user-specific. In other words, we choose a different value p_u for each u. In Section 4, we choose with uniform probability values from $[0.15, 0.85]$. By using user-specific values p_u, we vary the proportions of known and hidden ratings. This ensures that the simulation contains users in different states: from small to large profiles and, relative to those profile sizes, small or large number of known ratings versus hidden ratings.

To complete the setup, we define $R^{before} = \left(\bigcup_{u \in U_{Active}} R_u^K\right) \cup \left(\bigcup_{u \notin U_{Active}} R_u^{TR}\right)$. This is the entire set of ratings known by the recommender at the initial stage of the experiment. It includes all the known ratings of the Active users and all the ratings (except test set ratings) of the non-Active users.

3.2 An AL step

Using similar notation to that in [5], we define an AL strategy S_{AL} as a function of four arguments, $S_{AL}(u, n, G, P_u)$, which for user u returns a set of items Q_u of size no more than n. The items it returns, Q_u, are selected from a pool of candidates items, $Q_u \subseteq P_u$. Usually, these candidate items are ones for which u's rating is not currently known by the recommender, i.e. $P_u = \{i \in I \mid r_{ui} \notin R^{before}\}$. The strategy also has a parameter G, which designates the data that the AL strategy can use when selecting Q_u from P_u. Most commonly, G is simply the ratings that are currently known by the recommender, i.e. $G = R^{before}$. It is conceivable, however, that G contains additional data (e.g. personality data [7]). The only constraint is that it must be data that the recommender has acquired before this AL step.

In an online evaluation, if Q_u is non-empty, the items in Q_u would be presented to the user and she would be invited to rate as many of them as she cared to. In offline evaluation, this is where we use the hidden ratings R_u^H. The simulated user provides to the system her hidden ratings for items in Q_u. We will call these the elicited ratings, $R_u^E = \{r_{ui} \in R_u^H \mid i \in Q_u\}$. Note, by this approach, if the simulated user has the rating, she supplies it. This may seem unrealistic: real users are likely to ignore at least some of the requests made by an AL strategy or, even if willing to engage, may not be familiar enough with the items Q_u and thus unable to provide ratings. This is anticipated in the design of this offline evaluation because the overlap between the items in Q_u and the ones whose ratings are available in R_u^H will typically be small, and often there will be no overlap.

Now that we have established the notation, we can show how to evaluate the impact of an AL strategy S_{AL}. We must measure recommender performance before and after the acquisition of ratings by the strategy, as follows:

Before:

(1) Build a recommender model from R^{before}.
(2) Test the model (see below).

AL:

(1) For every Active user $u \in U_{Active}$,
   (a) Compute the items whose ratings the strategy wants to acquire, $Q_u = S_{AL}(u, n, G, P_u)$.
   (b) Obtain the ratings for these items $R_u^E = \{r_{ui} \in R_u^H \mid i \in Q_u\}$.
(2) Update the recommender with the elicited ratings $R^{after} = R^{before} \cup \bigcup_{u \in U_{Active}} R_u^E$.

After:

(1) Build a new recommender model from R^{after}.
(2) Test the new model (see below).

Testing the quality of the recommender (above) is the same both before and after. For every user $u \in U$, we use the recommender model to obtain a set of recommendations for u and then compute the value of each evaluation metric (see Section 3.3) by comparing the recommendations with u's test set R_u^{TE}.

Note that where there is more than one strategy to compare, the activities labeled *Before* are run just once. But the *AL* and *After* activities are run once per strategy.

There is, of course, the usual danger that the results obtained are specific to this particular split of the data. Hence, we partition again and repeat the AL step multiple times (repeated holdout), and report the average results. For example, in Section 4, we average over 10 runs.

3.3 Evaluation metrics

As we said in Section 1, there is good reason to evaluate AL strategies using a wider range of metrics than has been used up to now. We do not have space to survey the metrics. Instead, we will simply list the ones that we use in our case study in Section 4:

- For accuracy: Precision, Recall and nDCG.
- For diversity: Expected Intra-List Distance (EILD) and α-nDCG.
- For novelty: Expected Popularity Complement (EPC), Expected Free Discovery (EFD) and Expected Profile Distance (EPD).
- For serendipity: Content-Based Surprise (CBS).

Others that we compute but do not show in Section 4 include Average Popularity, Subtopic-recall and Expected Reciprocal Rank. For the definition of CBS, see [9]; for the rest, see [16]. Note that, for several metrics, [16] offers two definitions, depending on whether recommendations are compared with all test set ratings, or just ratings for 'relevant' items. In Section 4, we use the latter definitions.

But there is the additional issue, also raised in Section 1, of whether we report the average values of the metrics for all users or just for certain subsets of the users. For example, we can report the average values for the following groups:

- *All users*, U.
- *Active users*, U_{Active}, i.e. just those users who are randomly selected to be the ones to whom the AL strategy is applied.
- *Solicitees*, $\{u \in U_{Active} \mid Q_u \neq \{\}\}$, i.e. just those Active users for whom Q_u is non-empty. These are users whose ratings are being solicited.
- *Respondents*, $\{u \in U_{Active} \mid Q_u \neq \{\} \wedge R_u^E \neq \{\}\}$, i.e. just those Solicitees who provide at least one rating.
- *Solicitees-less-Respondents*, i.e. users who were asked for ratings but did not provide any.
- *U-less-Respondents*, i.e. everyone who did not provide a rating either because they were not asked for one or because they were asked for ratings but did not provide any.

In the literature, Elahi et al. give results averaged over U, referring to these as *system-wide* results. Other work in the literature takes a *user-centric* approach, which focuses on users involved in the AL, typically giving results for the Active users. We will see in Section 4 that it can be useful to give results for some of the other subgroups too. For example, suppose that Respondents benefit hugely from the AL but that the impact on all the other users is sometimes negative. If the positive benefit to Respondents outweighs the negative impact on the other users, then system-wide results (for U) may be misleading. If, on the other hand, we also show results for U-less -Respondents, we will see the negative impact that the strategy has.

3.4 Setting the values of hyperparameters

Both the recommender model and the AL strategy may have hyperparameters whose values need to be tuned. The AL literature says little on this, and we want to distinguish our work by providing a more rigorous exposition.

We begin by deciding what metric we wish to optimize in hyperparameter tuning. For example, for the recommender model we typically want hyperparameter values that give highest precision across all users. For now, we assume that hyperparameters for the AL strategy should also optimize precision across all users.

Suppose a dataset has been prepared in the way described in Section 3.1. In particular, each active user's ratings have been partitioned into known, hidden and test. Hyperparameter values must be chosen before running the AL step (Section 3.2). The only data that can be used for this is R^{before} (the active user's known ratings and the non-active user's training ratings). If we use the active user's hidden ratings or any user's test ratings, then we have leakage.

We further split R^{before} in the same way as described in Section 3.1. In other words, for every user we split off a set that will be used for testing. Then for the active users, we split their remaining ratings into known and hidden. It may help to visualize as follows: an active user's rating were divided into three, R_u^K, R_u^H and R_u^{TE}. Now, for hyperparameter tuning purposes, R_u^K has been split into three again, R_u^{valK}, R_u^{valH} and R_u^{valTE}.

We tune the recommender model's hyperparameters first. For each configuration of hyperparameter values, we train a recommender on ratings that are not part of any hidden sets or test sets. We measure recommender performance using R_u^{valTE} for each user. We perform the splits again (repeated holdout), compute the average and select the hyperparameter values that optimize the metric we chose earlier (e.g. precision). In Section 4, for example, we average results over 5 runs.

Now we can tune the AL strategy's hyperparameters. For each configuration of hyperparameter values, we run an AL step but using R_u^{valK}, R_u^{valH} and R_u^{valTE}. We run an AL step for each different split of the repeated holdout and we choose the hyperparameter

Table 1: Dataset statistics (after filtering)

~93% sparsity	~9K ratings
3482 users	avg. ~256 ratings per user
3675 items	avg. ~243 ratings per item

values that optimize the metric we chose previously (e.g. precision across all users).

4 CASE STUDY

In a more conventional paper, this section would compare the authors' new AL strategy with existing and baseline strategies, perhaps on multiple datasets, in an effort to convince the reader of the value of the new strategy. But that is not our intention here. Instead, we present a sample of our results, obtained using three very simple strategies on a single dataset, and our intention is to show the value of a more comprehensive evaluation. We want to illustrate, for example, that it can be misleading to conduct only a system-wide or only a user-centric evaluation; it can be misleading to fail to distinguish cold-start from more mature users; and it can be misleading to ignore metrics other than accuracy.

4.1 Preliminaries

The dataset we use is the MovieLens 1M dataset.[1] As discussed in Section 3.1, we discard users who have fewer than 75 ratings. But recall that $p = 0.8$ so, for all users, 20% of those ratings will be test set ratings, and for Active users p_u is chosen from $[0.15, 0.85]$, so between 15% and 85% of the training ratings will be known ratings, the remainder being hidden ratings. Hence, even though we start with users who have quite a lot of ratings (minimum 75), we will still end up with some Active users with very few known ratings (effectively becoming cold-start users). Table 1 summarizes the characteristics of the dataset after filtering.

For the recommender, we use the RankSys implementation of Matrix Factorization.[2]

Of the AL strategies that we have implemented, the ones that appear in this case study are:

- *popularity*: We score each item by the total number of ratings for that item in R^{before}. Items are ordered by decreasing score and the top n are selected. This strategy is not personalized: if two users have the same pool of candidates P_u, then their Q_u will be the same.
- *highest-predicted* [5]: For every active user $u \in U_{Active}$ and for each item in P_u, the Matrix Factorization recommender predicts the user's rating. Items are then ordered by decreasing predicted rating and the top n are selected. This strategy is personalized, choosing items which the recommender thinks the user will like.
- *binary-predicted* [5]: The matrix R^{before} is binarized to give a new matrix B, where $b_{ui} \in B$ is 1 if $r_{ui} \in R^{before}$ is 1 and b_{ui} is 0 if r_{ui} is $\perp$. An implicit Matrix Factorization model is built from B. Then, the *highest-predicted* strategy is applied, and the top n items are selected. The strategy is personalized, choosing items that are likely to be familiar to the user.

[1] https://grouplens.org/datasets/movielens/1m/
[2] https://github.com/saulvargas

Table 2: # of elicited ratings per Solicitee

# elicited per user	
binary-predicted	2.45
highest-predicted	2.41
popularity	1.75

In our experiments, the maximum number of items selected, n, is 5 and P_u (the pool of candidates) are ones whose ratings are not known ($P_u = \{i \in I \mid r_{ui} \notin R^{before}\}$). The number $n = 5$ is chosen as a reasonable approximation to what might fit on a screen. On portable devices, it might be slightly fewer; on other devices, it might be slightly more.

We run the experiments for 10 iterations of repeated holdout and we average results over the 10 runs. In each run, we randomly select one-third of the users to be Active users (about 1100 users). To the best of our knowledge, no best practice exists in the literature to fix this proportion. We chose one-third in order to have enough Active users and non-Active users to be able to show both user-centric and system-wide results.

The Matrix Factorization has one main hyperparameter, the number of latent factors. We use 5 iterations of repeated holdout to select the value from $\{10, 15, 20, 25, 30, 35, 40\}$ that maximizes precision for top-10 recommendations across all users. Among the AL strategies, only *binary-predicted* has a hyperparameter, again the number of latent factors. This too is selected from $\{10, 15, 20, 25, 30, 35, 40\}$ to maximize top-10 recommendations across all users.

When testing recommendation quality, we make 10 recommendations and we consider a test set item to be 'relevant' if its rating is above 3 stars. To compute scores for diversity, novelty and serendipity, we use the 18 MovieLens genres as item features.

4.2 System-wide versus user-centric results

Consider the accuracy results in Figure 1 first. In every subplot, we report the percentage improvement in each accuracy metric (Precision, Recall and nDCG). Subplot (a) shows the results for Respondents, i.e. users who have provided at least one new rating to the system. All three strategies enjoy a substantial improvement for all three metrics and this is a demonstration of the utility of AL. All three metrics also agree on how to rank the strategies: *highest-predicted* is the best strategy (including a notable 37% improvement in terms of nDCG), then *binary-predicted* and last *popularity*. Table 2 shows the number of elicited ratings. From this, we note that *highest-predicted* and *binary-predicted* gather about the same amount of ratings. But, *highest-predicted* must on average be acquiring more useful ratings, since the ratings it acquires result in higher accuracy. However, we will not investigate the reasons behind these results because this is not the purpose of this case study.

Subplot (b) shows the system-wide performances (i.e. averaged over all $u \in U$). The strategies have the same ranking. Careful inspection of the scales on the y-axes shows that the improvements are significantly lower, but this is expected because these results include non-Active users.

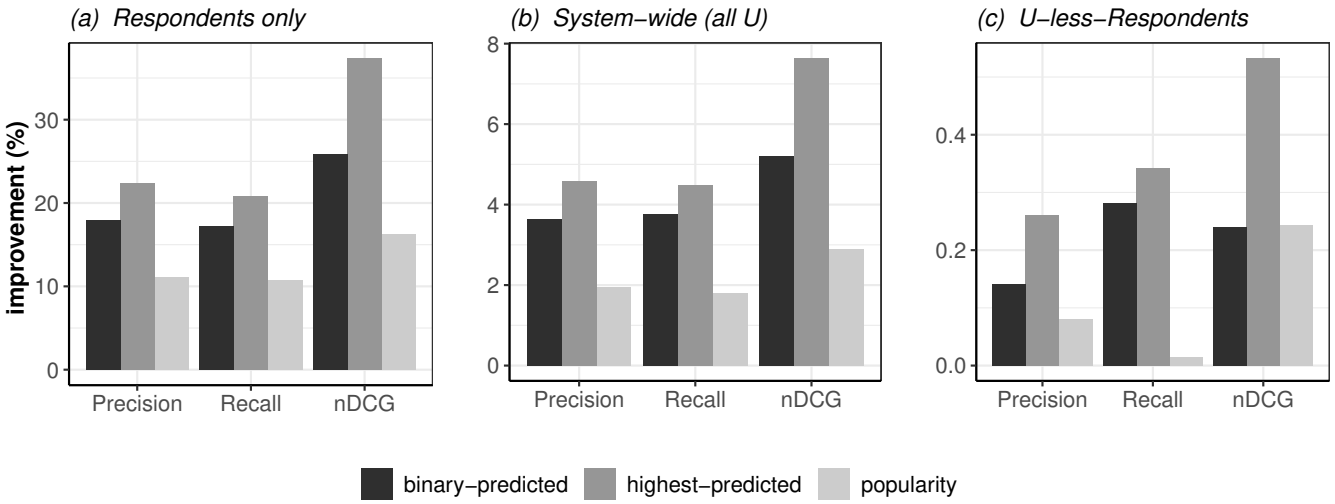

Figure 1: Accuracy results, user-centric vs. system-wide

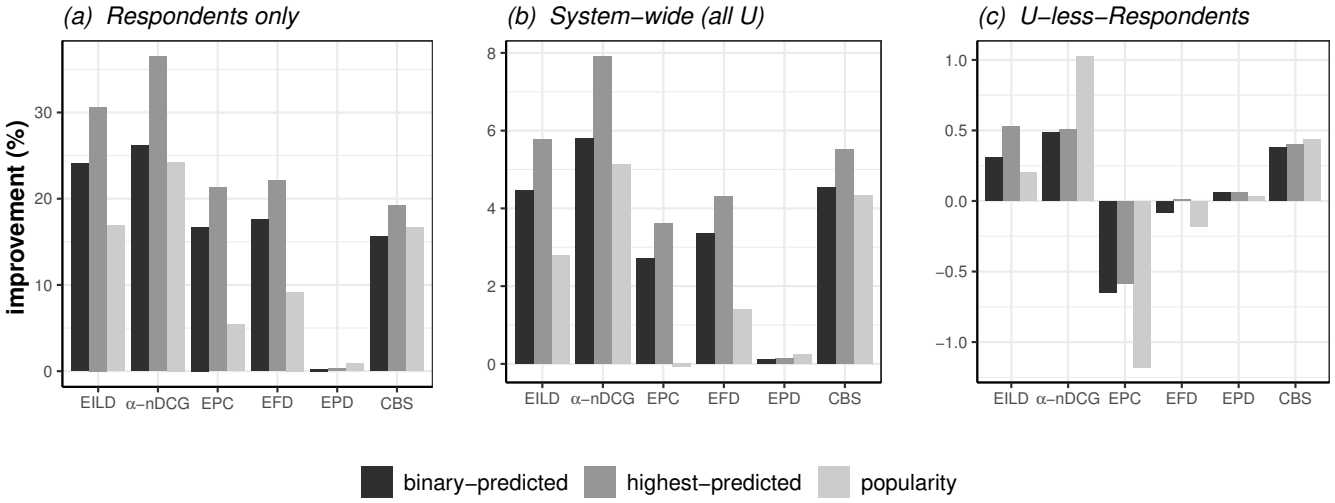

Figure 2: Beyond-accuracy results, user-centric vs. system-wide

Subplot (c) shows the results for U-less-Respondents. The ranking of the strategies remains the same but the improvements are very low. This is a useful feature of a more comprehensive method: it confirms that in this collaborative-filtering case study, active learning on average does have accuracy benefits even for the users who do not have new incoming data. Subpot (b) alone would not confirm this because it includes results for Respondents.

Now, consider the beyond-accuracy results in Figure 2. Subplot (a) shows that Respondents enjoy improvements in all measures, although for the EPD novelty measure the improvements are negligible. This is good because it means that the new data makes recommendations more diverse, serendipitous and, according to most of the novelty metrics, more novel. *highest-predicted* is the winning strategy for all the metrics except for EPD, where the winner is *popularity*. Subplot (b) shows the same trend system-wide. There is one small difference with subplot (a): in the system-wide results, *popularity* does not improve the EPC novelty metric (it is slightly negative).

But it is subplot (c) that most illustrates the advantage of averaging over different user groups. For users who do not provide any rating (U-less-Respondent), *popularity* is now a winning strategy in terms of α-nDCG and CBS. Furthermore, the EPC novelty is on average negative for all three strategies and the EFD is negative for two strategies. The dis-improvements here are small and apply only to two metrics and so a system designer might decide to sacrifice a little recommendation novelty in exchange for the benefits across the other metrics. But, without subplot (c), this trade-off would be hidden.

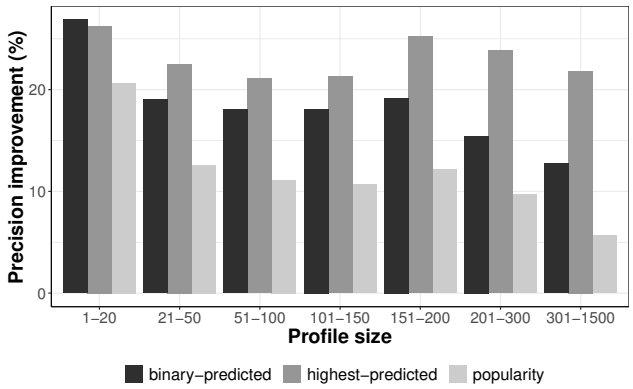

Figure 3: Precision results for Respondents by profile size

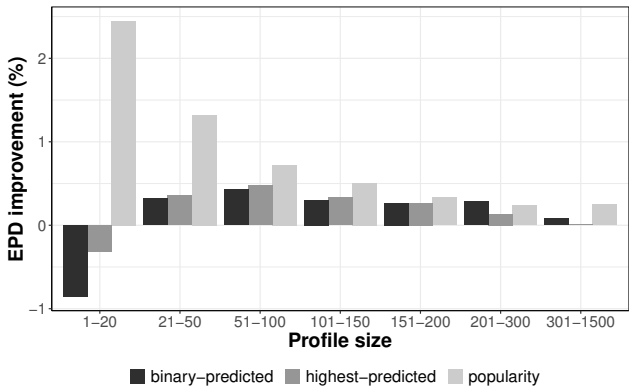

Figure 4: EPD results for Respondents by profile size

4.3 Results by profile-size

In this section, we concentrate on Respondents but we show results by profile size. This enables us to compare the impact of AL on cold-start users and more mature users. To do this, we put each Respondent u into a bucket based on her profile size, $|R_u^K|$. The design of the buckets in this case study is somewhat heuristic. The first bucket comprises profiles that contain 1–20 ratings. This is inspired by the literature (e.g. [7, 11]), where a user is labeled as a cold-start user if her profile is in this specific range. The subsequent buckets are chosen in a way that ensures each contains a substantial number of users.

We do not have space to show all the metrics, so we select just three: Precision (Figure 3), EPD (Figure 4) and CBS (Figure 5).

Figure 3 shows improvements in Precision. For all except the first bucket, we see the same trend we have seen before: *highest-predicted* is the best strategy followed by *binary-predicted* and then *popularity*. However, for cold-start users (profiles of size 1–20), *binary-predicted* is the best strategy. This might suggest to a system designer the need for a hybrid strategy, or even an adaptive hybrid. This result is hidden if the only results are like those in Figure 1.

Figure 4 and 5 shows improvements in EPD and CBS respectively. Again this level of analysis reveals interesting trade-offs. For EPD (Figure 4), *popularity* is generally the best strategy, but it is

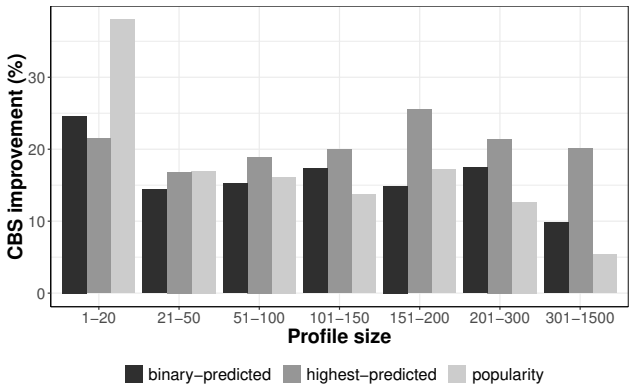

Figure 5: CBS results for Respondents by profile size

particularly good for cold-start users whereas *highest-predicted* and *binary-predicted* have a negative impact on cold-start users. For CBS (Figure 5), for cold-start users, *popularity* is the best strategy, with *binary-predicted* second and *highest-predicted* last, but for other users the usual ranking of the strategies applies.

5 CONCLUSIONS, DISCUSSION AND FUTURE WORK

In this paper, we have presented as explicitly as possible the methodology that we use for offline evaluation of Active Learning (AL) strategies. We presented it at a level of detail that allows for reproducibility and detailed discussion. We have argued that a full picture only emerges if AL offline evaluation results are analyzed: (a) for different groups of users (e.g. all users U, Respondents only, U-less-Respondents, etc.), (b) by profile size, and (c) using a wide range of metrics, both accuracy and beyond-accuracy. A small case study illustrated this. The purpose of the case study was not to choose the best strategy, nor to analyze why one strategy outperforms another. The purpose was to illustrate that, even for three very simply AL strategies, trade-offs only become visible with the level of analysis that we propose.

We want at this point to discuss three aspects of the method that may be open to criticism. The first is the proportion of users who are randomly chosen to be Active users. We select one-third for the pragmatic reason that it gives a reasonable number of both Active users and non-Active users. But the balance between the two affects the results. If there were fewer Active users, for example, then system-wide results (for all U) might not look so good: with fewer Active users there can be fewer new ratings and thus a reduced impact. Of course, in some ways, this further supports our position that one should not just look at system-wide results; one should look at results for other user subgroups too. Nevertheless, we are doing some experiments using different proportions of Active users to quantify its effect.

The second discussion point concerns the distribution of rating values. In recommender systems that use explicit ratings, positive ratings (e.g. 4s and, to some extent, 5s) often predominate; certainly, this is true of the MovieLens datasets. This motivates work on recommender models that recognize that missing ratings are

missing-not-at-random [18]. But our concern here is how this affects offline evaluation. For example, it means that positive ratings predominate in hidden sets. Could this mean that a strategy such as *highest-predicted*, which targets items that it thinks the user will like, enjoys an advantage in the offline evaluation that it would not enjoy in a deployed system? In fact, we took steps to check this. We ran a version of our experiment on a modified version of the MovieLens dataset in which we discarded 'excess' positive ratings to ensure that hidden sets were not dominated by them. There were no noticeable differences in the results that we obtained. In particular, *highest-predicted* was still the winning strategy as often as before. Nevertheless, we continue to think about ways of testing for, and avoiding, biases of this kind.

The third discussion point is that we are not using multi-step evaluation, where AL is applied repeatedly to simulate the evolution of a recommender system. There are many questions about whether multi-step evaluations do, in fact, model system evolution. Some of these are addressed in [4, 6]. Unless these issues can be resolved satisfactorily, it may be best to resort to user trials and online evaluation as soon as single-step evaluation identifies the promising AL strategies.

For the future, there are several lines of research. One is to apply the method more widely, e.g. to other datasets and to more AL strategies. It may be useful to extend the analysis of results, so we can see results not just by profile size but also perhaps by rating variance and profile diversity. We should also include analyses of AL costs: there are time costs and cognitive costs to users; there are time and space costs in building and running recommender models, building and running AL models, and in re-training recommender models. Moreover, at the moment, the method takes a user-perspective: it assumes that AL acquires new ratings for the benefit of users. But AL can be used for cold-start items [1, 17]. A similar evaluation method can be designed for situations where AL is used to boost the ratings of new items and more mature items.

Finally, we want to use our method to help us design new AL strategies that are better targeted to the needs of different kinds of users. We could imagine, for example, personalized or adaptive strategies that help cold-start users rapidly improve recommendation relevance but which place an emphasis on discovery for more mature users.

ACKNOWLEDGMENTS

This paper emanates from research supported by a grant from Science Foundation Ireland (SFI) under Grant Number SFI/12/RC/2289 which is co-funded under the European Regional Development Fund.

REFERENCES

[1] Michal Aharon, Oren Anava, Noa Avigdor-Elgrabli, Dana Drachsler-Cohen, Shahar Golan, and Oren Somekh. 2015. ExcUseMe: Asking Users to Help in Item Cold-Start Recommendations. In *Proceedings of the 9th ACM Conference on Recommender Systems (RecSys '15)*. ACM, New York, NY, USA, 83–90.

[2] Giuseppe Carenini, Jocelyin Smith, and David Poole. 2003. Towards More Conversational and Collaborative Recommender Systems. In *Proceedings of the 8th International Conference on Intelligent User Interfaces (IUI '03)*. ACM, New York, NY, USA, 12–18.

[3] Shuo Chang, F. Maxwell Harper, and Loren Terveen. 2015. Using Groups of Items for Preference Elicitation in Recommender Systems. In *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing (CSCW '15)*. ACM, New York, NY, USA, 1258–1269.

[4] Mehdi Elahi, Francesco Ricci, and Neil Rubens. 2012. Adapting to Natural Rating Acquisition with Combined Active Learning Strategies. In *ISMIS*.

[5] Mehdi Elahi, Francesco Ricci, and Neil Rubens. 2014. Active Learning Strategies for Rating Elicitation in Collaborative Filtering: A System-wide Perspective. *ACM Trans. Intell. Syst. Technol.* 5, 1, Article 13 (Jan. 2014), 33 pages.

[6] Mehdi Elahi, Francesco Ricci, and Neil Rubens. 2016. A Survey of Active Learning in Collaborative Filtering Recommender Systems. *Comput. Sci. Rev.* 20, C (May 2016), 29–50.

[7] Ignacio Fernández-Tobías, Matthias Braunhofer, Mehdi Elahi, Francesco Ricci, and Iván Cantador. 2016. Alleviating the new user problem in collaborative filtering by exploiting personality information. *User Modeling and User-Adapted Interaction* 26, 2 (01 Jun 2016), 221–255.

[8] Abhay S. Harpale and Yiming Yang. 2008. Personalized Active Learning for Collaborative Filtering. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '08)*. ACM, New York, NY, USA, 91–98.

[9] Marius Kaminskas and Derek Bridge. 2016. Diversity, Serendipity, Novelty, and Coverage: A Survey and Empirical Analysis of Beyond-Accuracy Objectives in Recommender Systems. *ACM Trans. Interact. Intell. Syst.* 7, 1, Article 2 (2016), 42 pages.

[10] Rasoul Karimi, Alexandros Nanopoulos, and Lars Schmidt-Thieme. 2015. A supervised active learning framework for recommender systems based on decision trees. *User Modeling and User-Adapted Interaction* 25, 1 (01 Mar 2015), 39–64.

[11] Daniel Kluver and Joseph A. Konstan. 2014. Evaluating Recommender Behavior for New Users. In *Proceedings of the 8th ACM Conference on Recommender Systems (RecSys '14)*. ACM, New York, NY, USA, 121–128.

[12] Sean M. McNee, John Riedl, and Joseph A. Konstan. 2006. Being Accurate is Not Enough: How Accuracy Metrics Have Hurt Recommender Systems. In *CHI '06 Extended Abstracts on Human Factors in Computing Systems (CHI EA '06)*. ACM, New York, NY, USA, 1097–1101.

[13] Roberto Pagano, Massimo Quadrana, Mehdi Elahi, and Paolo Cremonesi. 2017. Toward Active Learning in Cross-domain Recommender Systems. *CoRR* abs/1701.02021 (2017). http://arxiv.org/abs/1701.02021

[14] Al Mamunur Rashid, Istvan Albert, Dan Cosley, Shyong K. Lam, Sean M. McNee, Joseph A. Konstan, and John Riedl. 2002. Getting to Know You: Learning New User Preferences in Recommender Systems. In *Proceedings of the 7th International Conference on Intelligent User Interfaces (IUI '02)*. ACM, New York, NY, USA, 127–134.

[15] Al Mamunur Rashid, George Karypis, and John Riedl. 2008. Learning Preferences of New Users in Recommender Systems: An Information Theoretic Approach. *SIGKDD Explor. Newsl.* 10, 2 (Dec. 2008), 90–100.

[16] Saúl Vargas Sandoval. 2015. *Novelty and diversity evaluation and enhancement in recommender systems.* Ph.D. Dissertation. Universidad Autónoma de Madrid, Spain.

[17] Martin Saveski and Amin Mantrach. 2014. Item Cold-start Recommendations: Learning Local Collective Embeddings. In *Proceedings of the 8th ACM Conference on Recommender Systems (RecSys '14)*. ACM, New York, NY, USA, 89–96.

[18] Harald Steck. 2010. Training and Testing of Recommender Systems on Data Missing Not at Random. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '10)*. ACM, New York, NY, USA, 713–722.