The Thirty-Third International
FLAIRS Conference (FLAIRS-33)

Debiased Offline Evaluation of Active Learning in Recommender Systems

Diego Carraro, Derek Bridge
Insight Centre for Data Analytics
University College Cork, Ireland
{diego.carraro, derek.bridge}@insight-centre.org

Abstract

Active Learning (AL) when applied to Recommender Systems (RSs) aims at proactively acquiring additional ratings data from the RS users in order to improve subsequent recommendation quality. AL strategies are typically evaluated offline first, but the classic AL offline evaluation methodology does not take into account the *bias problem* in RS offline evaluation. This problem affects the evaluation of an RS, as brought to light by recent literature. But, we argue, it also affects the evaluation of AL strategies as well. For this reason, in paper, we propose a new AL offline evaluation methodology for RSs which mitigates the bias and thus facilitates a truer picture of the performances of the AL strategies under evaluation. We illustrate our proposed methodology on two datasets and with three simple and well-known AL strategies from the literature. Our experimental results differ from those reported previously in the literature, which shows the importance of our approach to AL evaluation.

Introduction

The items that a Recommender System (RS) recommends to its users are typically ones that it thinks the user will find to be novel and that the user will like. If the user consumes an item, the RS invites the user to rate it. But an RS can use Active Learning (AL) to proactively acquire additional ratings. The idea is to explicitly query the users, asking them to rate items which have not been rated yet. The items that a user will be asked to rate (known as the query items) are selected 'intelligently' by an *active learning strategy*. Different AL strategies take different approaches to identifying the query items (Elahi, Ricci, and Rubens 2016). Query items are different from recommendations. In general, an AL strategy should select items that it believes will be familiar to the user, in order to get a successful response from them. Moreover, it should select items that it believes will improve subsequent recommendation quality.

Offline evaluation of AL strategies can help to narrow the number of strategies that need to be evaluated in costly user trials and online experiments. The most common methodology simulates the ratings elicitation process by using a pre-collected *observed dataset*, which records historical interactions (e.g. clicks, purchases, ratings) between users and items in an RS scenario. However, the RS literature shows that observed datasets are *biased* due to the presence of many confounders (Chaney, Stewart, and Engelhardt 2018; Wang et al. 2018). For example, the way in which items are exposed to users by the RS's user-interface is one source of such bias (Liang et al. 2016). The RS's recommendations themselves are another confounder: users are more likely to rate recommended items than other items. Item popularity is also a confounder: users are more likely to rate popular items (Pradel, Usunier, and Gallinari 2012). Because of these and other confounders, ratings that are missing from an observed dataset are Missing Not At Random (MNAR) (Marlin et al. 2007). Despite this, most offline evaluations of RSs, and all offline evaluations of AL strategies for RS, assume that ratings in the observed dataset are instead Missing At Random (MAR) (Marlin et al. 2007). But treating MNAR data as if it were MAR often leads to misinterpretation of a system's performance. In particular, it will often result in overestimates of the effectiveness of RSs that recommend popular items or that make recommendations to the more active users (Pradel, Usunier, and Gallinari 2012; Cremonesi, Koren, and Turrin 2010).

In this paper, our goal is to assess the impact of bias in the offline evaluation of AL strategies, to complement work already done on its impact on offline evaluation of RSs only, e.g. (Liang, Charlin, and Blei 2016; Cañamares and Castells 2018). We use a general framework for offline evaluation of AL strategies which makes a three part split of an observed dataset into known, hidden and test sets. We compare three evaluation methods. In the first, corresponding to existing evaluations of AL strategies, we do nothing to mitigate bias. In the second, we mitigate bias in the test set; and, in the third, we mitigate bias in both the hidden and test sets. (We explain later why we do not also mitigate bias in the known set.) To mitigate bias, we use a weighted sampling approach, which enjoys low overheads and high generality.

Our experiments on two MNAR datasets show that three simple and well-known AL strategies perform differently according to the three evaluation methods. In particular, under our new proposed evaluation methods, a strategy that asks

users to rate popular items has an impact on recommendation quality that is not much better than a strategy that asks users to rate randomly-chosen items.

Related Work

Offline evaluation of active learning

In offline experiments, AL strategies are evaluated by splitting the dataset into three parts: the known ratings, the hidden ratings and the test ratings. Known ratings are the ones from which the initial recommender model is built, i.e. the ones we assume that the RS has at hand. The hidden ratings are the ones which the simulated users 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. Test ratings are used to measure the performance of the RS both before applying the AL strategy and afterwards. Despite the fact that this setup is widely used in the literature, e.g. (Elahi, Ricci, and Rubens 2014), it has at least two issues, even before we consider matters of dataset bias.

First, offline evaluation of AL generally takes a *user-centric* perspective and on *cold-start* users only, i.e. performance is measured only with respect to the users that are queried, which are the new and low-activity ones. But, in principle, AL is a general approach for acquiring ratings from any kind of user and therefore it can be used to improve recommendations for non cold-start users as well, e.g. (Carenini, Smith, and Poole 2003; Elahi, Ricci, and Rubens 2014; Carraro and Bridge 2018). Moreover, especially for collaborative-filtering recommenders, where new ratings from a subset of the users could affect the recommendations made to other users in the system, measuring the impact only on the queried users is somewhat myopic; measuring also the *system-wide* impact of AL (i.e. the impact on recommendations quality for all the users in the system, regardless of whether they have been queried or not) gives a more rounded perspective (Elahi, Ricci, and Rubens 2014).

Second is the issue of how to set the hyperparameters of the AL strategies. The AL literature has little to say on this, and so it is not clear how to perform this important step which might heavily influence the results of the offline experiment. To the best of our knowledge, only our own previous work explicitly describes in detail how to tune hyperparameters for AL in RS (Carraro and Bridge 2018).

For the reasons above, our evaluation framework is from (Carraro and Bridge 2018), which describes a comprehensive and methodological way of evaluating AL strategies.

Unbiased offline evaluation of RSs

As we have discussed, the ratings data that an RS collects as part of its normal operation is MNAR data, and not MAR data, and therefore cannot be used for unbiased offline evaluation (Marlin et al. 2007).[1] The 'straightforward' approach for coping with bias in offline evaluation of an RS is to separately collect some unbiased data and use it as the test set. This can be done with what is sometimes called a "forced ratings approach" (Cañamares and Castells 2018). In this approach, randomly-selected users are required to rate randomly-selected items. Examples of datasets collected in this way are described in (Marlin et al. 2007) and (Cañamares and Castells 2018) for the music domain, and in (Schnabel et al. 2016) for the clothing domain. If a user is to rate a randomly-selected item, and if that item is not known to her, then she must be able to quickly consume it (or part of it) in order to form an opinion of it. In the music domain, she can listen to a music track; in the clothing domain, she can look at images of the items; in the movie domain, she can watch a movie trailer, perhaps. But in many domains, this approach may be expensive or impracticable.

For this reason, most of the work on unbiased offline evaluation makes use of MNAR datasets but tries to reduce the evaluation bias. One way to cope with the bias in an MNAR dataset is to design an unbiased estimator, i.e. an evaluation metric that compensates for the bias in the dataset. The estimators in (Steck 2010) and (Lim, McAuley, and Lanckriet 2015) are examples. Such estimators can indeed provide unbiased or nearly unbiased measures of RS performance. However, one drawback is that they might require an expensive learning step. This is the case, for example, for one of the propensity-based estimators in (Schnabel et al. 2016).

An alternative way of mitigating bias in offline evaluation is the *intervention approach*. In this approach, we sample the MNAR test set to produce a smaller test set with MAR-like properties. For example, Liang et al. propose SKEW, which samples ratings in inverse proportion to their item popularity, thus generating an intervened test set with reduced popularity bias (Liang, Charlin, and Blei 2016). In (Carraro and Bridge 2020), we define an alternative intervention approach, which we call WTD, which we empirically compare with SKEW. We demonstrate that WTD is more suitable for approximating how an RS would perform on unbiased test data.

In the next section, we will describe in detail how we propose to compare AL strategies in offline experiments using MNAR data but using WTD to mitigate problems of bias.

Our Approach to Unbiased AL Evaluation

In this paper, we mitigate the bias problem in our proposed offline AL evaluation by means of the WTD intervention approach. We will give an overview of WTD and then describe how we use it in AL evaluation.

WTD: debiasing by weighted sampling

In this section, we give an overview of WTD, which can be used to debias a dataset of user-item ratings (Carraro and Bridge 2020).

We denote with $u \in U$ a generic user and with $i \in I$ a generic item. We denote with $D = \{O \in \{0,1\}^{U \times I}, Y \in \mathbb{R}^{U \times I}\}$ a generic observed dataset. The binary matrix O records which ratings have been observed: $O_{u,i} = 1$ if a rating is observed and $O_{u,i} = 0$ otherwise. We also define

[1] In this paper, we use the terms MNAR and "biased" interchangeably, and the same for MAR and "unbiased". What we refer to as MAR, in line with papers such as (Cañamares and Castells 2018; Steck 2010), others refer to as Missing Completely At Random (MCAR), e.g. (Schnabel et al. 2016)

the associated matrix $Y \in \mathbb{R}^{U \times I}$, which records the value of the ratings of the corresponding observed entries in O: we have $Y_{u,i} \neq 0$ where $O_{u,i} = 1$, $Y_{u,i} = 0$ otherwise. We also define the binary random variable $\mathcal{O} : U \times I \rightarrow \{0, 1\}$ over the set of user-item pairs in O as $\mathcal{O} = 1$ if the user-item interaction is observed and $\mathcal{O} = 0$ otherwise. (But later we will use abbreviation $P(\mathcal{O})$ in place of $P(\mathcal{O} = 1)$.)

WTD performs a debiasing intervention on MNAR data $D_{mnar} = \{O^{mnar}, Y^{mnar}\}$. For the purposes of explaining WTD, we will initially assume also the availability of some unbiased MAR data $D_{mar} = \{O^{mar}, Y^{mar}\}$ in addition to the MNAR data. (But, see later, where we explain that actual MAR data is not needed.)

WTD samples from D_{mnar}. The result of this intervention is a dataset $D_S = \{O^S \subset O^{mnar}, Y^S \subset Y^{mnar}\}$, with the objective that D_S has unbiased-like properties. Note that the sampling is totally independent from the values of the ratings in D_{mnar}, i.e. Y^{mnar}. It is only dependent on O^{mnar}. To model such sampling, we denote with $\mathcal{S} : U \times I \rightarrow \{0, 1\}$ the binary random variable that guides the sampling. $\mathcal{S} = 1$ when a particular user-item pair is sampled from O^{mnar}, 0 otherwise. (Again, we will use abbreviation $P(\mathcal{S})$ in place of $P(\mathcal{S} = 1)$.) In practice, the sampling is characterized by the expression of the probability $P_S(\mathcal{S}|u, i), \forall (u, i) \in O^{mnar}$, which is the probability distribution responsible for guiding the sampling on O^{mnar}.

The key idea of WTD is to make the posterior probability of each user-item pair in the sampled O^S, i.e. $P_S(u, i|\mathcal{S})$, approximately the same as the posterior distribution observed for the corresponding user-item pair in O^{mar}, i.e. $P_{mar}(u, i|\mathcal{O})$. Writing this as a formula, we want:

$$P_S(u, i|\mathcal{S}) \approx P_{mar}(u, i|\mathcal{O}) \quad \forall (u, i) \in O^S \tag{1}$$

To obtain this approximation, we adjust the posterior distributions of the sampling space O^{mnar}, i.e. $P_{mnar}(u, i|\mathcal{O})$, using user-item weights $w = (w_{ui})_{u \in U, i \in I}$. We denote the modified weighted MNAR posteriors by $P_{mnar}(u, i|\mathcal{O}, w)$ and we use w to obtain the following equality:

$$P_{mnar}(u, i|\mathcal{O}, w) = P_{mar}(u, i|\mathcal{O}) \quad \forall (u, i) \in O^{mnar} \tag{2}$$

We define and calculate user-specific weights $w = (w_u)_{u \in U}$ and item-specific weights $w = (w_i)_{i \in I}$ instead of weights that are user-item specific. To save space, we will not show how the formulae for calculating the weights are derived; for this, see (Carraro and Bridge 2020). Instead, we will give the formulae and an informal explanation below.

$$w_u = \frac{P_{mar}(u|\mathcal{O})}{P_{mnar}(u|\mathcal{O})} \quad \forall u \in U \tag{3}$$

$$w_i = \frac{P_{mar}(i|\mathcal{O})}{P_{mnar}(i|\mathcal{O})} \quad \forall i \in I \tag{4}$$

We can think of the calculated weights as quantities that measure the divergence between the MNAR distributions of the sampling space and the target MAR distribution. We directly use weights to model the sampling distribution, i.e. $P_S(\mathcal{S}|u, i) = w_u(w_i)$. (In fact, based on previous experiments, we instead use $P_S(\mathcal{S}|u, i) = w_u(w_i)^2$, where we raise the importance of the item weight relative to the user weight.) During the sampling, the effect of the weights is to increase or decrease the probability that a particular user-item pair is sampled depending on how divergent are the user and item posterior probabilities in the MNAR sampling space with respect to MAR distributions.

Up to this point, we have assumed the availability of some MAR-like data in order to approximate the target posteriors. In fact, when we do not have any MAR-like data, we can still use the WTD approach. We know that the posterior probability distribution for MAR data is uniform ($P_{mar}(u|\mathcal{O}) = 1/|U|$, $P_{mar}(i|\mathcal{O}) = 1/|I|$). Therefore, we can use this hypothesized distribution when calculating the weights, avoiding the need for a MAR-like dataset.

Which sets should we debias?

As we explained earlier, AL strategies are evaluated by splitting the dataset into three parts: the known ratings, the hidden ratings and the test ratings. We could potentially debias any of these three sets.

Debiasing the test set It is obvious that, if we want an unbiased evaluation, then we must, at the least, debias the test set. To the best of our knowledge based on the literature, AL strategies have never been evaluated offline on an unbiased test set before.

Debiasing the hidden set In offline evaluation of AL strategies, the hidden set has a big impact on the final performance of an AL strategy. We can imagine, for instance, that an AL strategy that asks the users to rate popular items might have success in eliciting many ratings from the simulated users in an offline evaluation if the ratings in the hidden set are skewed towards popular items (which is typical in an MNAR dataset (Cañamares and Castells 2018)). But that does not mean that the same AL strategy would perform as well in practice. Its performance in practice will only be similar to performance in the offline evaluation if opinions that the user has not revealed to the system have the same distribution as the ones in the hidden set.

A user's unrevealed opinions are unlikely to be MAR. Users are influenced by external confounders. For example, a user is more likely to have opinions about items that she has been exposed to, such as items that are popular in general or that have been suggested by her friends. So, her unrevealed opinions are MNAR. But a user's unrevealed preferences are also unlikely to have the same distribution as the ratings in the RS's observed dataset, even though this is also MNAR. This is because, as we have discussed, the observed dataset is influenced by the RS itself. The RS acts as a source of several confounders: the user-interface makes some items more prominent and therefore more likely to be rated; the RS's recommendations are more likely to be rated than items that it does not recommend; and so on.

If we debias the hidden set, we make it more MAR-like, which, by the reasoning of the previous paragraph, is not necessarily correct. But if we leave it unchanged, then it is distributed like the whole observed dataset, which, again using reasoning from the previous paragraph, is not necessarily correct. We choose to report results from both, i.e. one set of

results where we debias the hidden set (see *INT_HT* below) and one set of results where we do not (see *INT_T* below). True performance should lie somewhere between the two. To the best of our knowledge, our work is the only one in the literature to explore this issue.

Debiasing the known set Finally, we could debias the known ratings dataset also. We know that, if we build a model on the known ratings without debiasing, then both the model and the AL strategy will be biased; for example, the popularity bias in the data might result in a popularity bias in the recommended items or in the items selected as queries. However, this paper is about *evaluation* of AL strategies; it is not about the development of new AL strategies, such as new unbiased strategies. We want to show how our improved approach to evaluation gives more robust insights into the performance of existing AL strategies. Therefore, in this paper, we will not debias the known ratings set.

Evaluation methods

On the basis of the discussion above, we can distinguish three evaluation methods, which differ depending on which sets are debiased. To present them precisely, we will introduce some notation.

We split the MNAR dataset D_{mnar} into three parts: K^{before} (the known ratings), H_{he} (the heldout hidden ratings) and T_{he} (the heldout test ratings). H_{he} and T_{he} are heldout sets, used as the sampling space to generate the final hidden and test sets, which we will designate by H and T, respectively. The three evaluation methods differ in how H and T are generated, as follows:

- *INT_HT*: In this method, we use WTD to intervene on T_{he} to generate an unbiased test set T. The size of T will be $\rho^T \times |T_{he}|$, where $\rho^T \in [0, 1]$ is a sampling rate. In this method, we also use WTD to intervene on H_{he} to generate an unbiased hidden set H. The size of H will be $\rho^H \times |H_{he}|$, similarly. This method aims to mitigate the bias in both the test and hidden sets.
- *INT_T*: This method aims to mitigate the bias in the test set only. In this method, we again use WTD to intervene on T_{he} to generate an unbiased test set T. But in this method, as discussed above, we do not use WTD to debias the hidden set, H_{he}. However, to ensure a fair comparison between *INT_HT* and *INT_T*, we need to make *INT_T*'s hidden set the same size as *INT_HT*'s. Otherwise, differences in the results of experiments might simply be due to *INT_T* having a larger hidden set. Hence, in this method, we randomly sample H from H_{he} using the same sampling rate ρ^H to produce a hidden set H.
- *CLASSIC*: This method corresponds to the classic way of evaluating an AL strategy, where there is no attempt to mitigate the bias in the dataset. To make comparisons between *CLASSIC*, *INT_HT* and *INT_T* fair, we randomly sample from T_{he} and H_{he} to get T and H using the same sapling rates as above, ρ^T and ρ^H, respectively.

When using WTD above, we use formulae 3 and 4 but we must calculate different weights for each different intervention. For the MAR posteriors, we use the hypothesized distributions that we gave earlier (i.e. $P_{mar}(u|\mathcal{O}) = 1/|U| \quad \forall u \in U$ and $P_{mar}(i|\mathcal{O}) = 1/|I| \quad \forall i \in I$). For the MNAR posteriors we use instead the following:

$$P_{mnar}(u|\mathcal{O}) = \frac{|O_u|}{|O|} \quad \forall u \in U \tag{5}$$

$$P_{mnar}(i|\mathcal{O}) = \frac{|O_i|}{|O|} \quad \forall i \in I \tag{6}$$

In the formulae above, O_u and O_i are the observed interactions in O for user u and item i respectively. O is either T_{he} or H_{he}, depending on the intervention that has to be made. For example, to produce an intervened T from T_{he}, then $O = T_{he}$.

If we remove this comment we are perfect with 6 pages] We compare these three methods using our methodology (Carraro and Bridge 2018), which we described earlier. It randomly selects a group of active users, i.e. $U_{Active} \subseteq U$, to whom the active learning will be applied. In the experiments presented in the next section, we select one-third of the users that have at least one rating in K^{before} to be active users. Selecting a group of active users allows us to measure both user-centric and system-wide impacts of AL (see the earlier discussion in the related work section). For fair comparisons, the same set of active users is used across all configurations of the experiments.

Experiments

In the experiments that we report here, our goal is not to find the best AL strategy. Rather, our goal is simply to show that debiasing can affect the results, even to the extent of changing which strategy is the best one.

We use the MovieLens 1M dataset (ML)[2] and the LibraryThing dataset (LT) (Clements, de Vries, and Reinders 2008). Both datasets have ratings on a 1 to 5 scale in steps of 1 for ML and steps of 0.5 for LT. For test sets, we consider a rating to be positive if it is above 3, and negative otherwise. In order to have enough data to work with (in each of the known-hidden-test sets) and to avoid outliers, for both datasets we discard users with fewer than 100 ratings and with more than 500 ratings.

We must train an RS on K^{before} and retrain it after we have acquired new ratings from the simulated users. We use Matrix Factorization with a ranking loss function (Pilászy, Zibriczky, and Tikk 2010).[3]

This RS has hyperparameters. We follow the procedure described in (Carraro and Bridge 2018) to set them. To save space, we do not explain that procedure here.

We compare well-known and easy-to-implement AL strategies from the literature, as follows:

- *Random* (RND): Candidate items are ordered randomly and the top n are selected. These are the query items that the active users are asked to rate.

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

[3] We use the implementation from the RankSys library: https://github.com/RankSys

Table 1: Average number of ratings per user elicited by each strategy for different evaluation methods.

	ML		LT	
	CLASSIC INT_T	INT_HT	CLASSIC INT_T	INT_HT
RND	0.32	0.33	0.02	0.02
POP	3.42	0.49	1.18	0.01
HP	4.50	2.94	2.89	0.87

- *Popularity* (POP): We score each candidate item by the total number of ratings for that item in K^{before}. Items are ordered by decreasing score and the top n are selected.
- *Highest-Predicted* (HP) (Elahi, Ricci, and Rubens 2014): For every active user $u \in U_{Active}$ and for each candidate item, the Matrix Factorization RS predicts the user's rating. Items are then ordered by decreasing predicted rating and the top n are selected.

HP has hyperparameters (the number of latent factors and a regularization term). Again, we follow the procedure described in (Carraro and Bridge 2018) to set them.

Experiments are performed over 10 runs with different random splits, where K^{before} is 60%, H_{he} is 20% and T_{he} is 20% of the observed dataset. To generate H from H_{he} and T from T_{he}, we set $\rho^H = \rho^T = 0.5$. We set $n = 50$. This is the number of query items that the AL strategy will select for each active user. Of course, in practice it is unlikely that an RS would ask a user for 50 ratings. Our choice of $n = 50$ is experimentally motivated by the fact that we need to elicit enough ratings for there to be an appreciable change in the performance of the RS.

Testing the quality of the recommender is the same both before and after the new ratings are acquired. For each method, we compute Recall, Precision and NDCG for top-10 recommendations on T, for each user in U_{Active}. The three metrics show similar results and therefore, for the sake of space, we report only results in terms of Recall.

Results

Table 1 reports the average number of ratings elicited by each strategy per user for the different evaluation methods. To note, *CLASSIC* and *INT_T* have the same values because they share the same K^{before} and H, hence the same sets of ratings get elicited; *CLASSIC* and *INT_T* differ only in their test sets, which affects the Recall results below.

The results in Table 1 are similar for both datasets. As we can see, the RND AL strategy is unaffected by whether we debias the hidden set (*INT_HT*) or not (*CLASSIC*, *INT_T*). But 'intelligent' strategies (POP and HP) are affected. These strategies elicit more ratings per user on average when the hidden set is not debiased (*CLASSIC*, *INT_T*) than when the hidden set is debiased (*INT_HT*). This is what we would expect. Moreover, looking just at the results for debiased hidden sets (*INT_HT*), we see that HP is the strategy which elicits the largest number of ratings; and we have that POP elicits roughly the same small number of ratings as RND. We would expect this too, if we have successfully removed popularity bias from the hidden sets.

Table 2: ML dataset: Percentage change in Recall@10 after elicitation of new ratings.

	CLASSIC	*INT_T*	*INT_HT*
RND	+0.22%	+0.97%	-0.50%
POP	+9.67%	+3.01%	-0.27%
HP	+9.45%	+11.83%	+10.25%

Table 3: LT dataset: Percentage change in Recall@10 after elicitation of new ratings.

	CLASSIC	*INT_T*	*INT_HT*
RND	+0.63%	+3.94%	+0.13%
POP	+5.33%	+1.92%	+0.31%
HP	+10.18%	+20.96%	+11.86%

Tables 2 and 3 report test results on the two datasets. For each AL strategy and for each evaluation method, we measure Recall@10 for users in U_{Active}, both before and after the AL step. The tables show the percentage change in the Recall@10 achieved after the AL step so that we can observe the impact of each AL strategy on these users. The statistical significance of the results is assessed by performing a pairwise comparison test between the AL strategies, using a two-tailed Wilcoxon signed rank test with $p < 0.05$.

The first observation is that AL improves the recommender's performance in almost all the scenarios (except for RND and POP in *INT_HT* on ML). This once again demonstrates that AL helps an RS improve its recommendation quality. For ML, HP and POP show the best improvement over RND (by roughly 8 percentage points) according to the *CLASSIC* method. However, their performances are not statistically significantly different from each other, so they are basically equivalent under this evaluation method. This changes when we look at *INT_T* and *INT_HT*. Under these evaluation methods, POP is no longer the 'same' as HP: for *INT_T*, POP drops to second place in the ranking (and the differences between the three strategies are statistically significant with respect to each other); for *INT_HT*, POP drops even more and its performance is now similar to RND's (indeed they are not statistically significantly different from each other). POP's performance on *INT_T* and *INT_HT* can be explained by the fact that POP acquires ratings for popular items, which might bias the RS towards popular recommendations; such a recommender will perform poorly when evaluated on a debiased test set.

For LT, the outcome is similar. Again HP is the best strategy according to all three methods (and this is statistically significant with respect to all other strategies). According to the *CLASSIC* ranking, POP and RND come second and third respectively. But this does not hold for the debiased methods, *INT_T* and *INT_HT*. In fact, for *INT_T* and *INT_HT*, POP and RND are both at the bottom of the ranking (and they are not statistically significantly different from each other) and they are a long way from HP in terms of percentage performance improvement.

Conclusions

In this paper, we proposed a new offline experimental methodology to evaluate the effectiveness of AL strategies for RS. The evaluation methodology attempts to mitigate the bias introduced by the use of an MNAR observed dataset, which is a problem not considered by the classic evaluation methodology widely used in the literature. We presented two alternative methods for conducting such a debiased evaluation: one debiases only the test set; the other debiases both the test set and the hidden set. In our experiments, these two methods reveal similar insights, and so it remains unclear whether one should be preferred over the other.

Our experiments compare three simple AL strategies from the literature on two widely-used MNAR datasets. Our results confirm that the HP strategy, which predicts items that the users will like and then asks the users to rate these items, is a good strategy. But our results show that the performance of the POP strategy, which asks users to rate popular items, is overestimated by the classic evaluation method used in the literature. In the light of our findings, this suggests the need to reconsider results presented in the literature by using instead a debiased evaluation method.

The results that we are publishing here show the change in Recall for the active users. We also have system-wide results, which show the impact of the new ratings on all users, not just the active ones. We lack the space to show these results, but they add further weight to our conclusions.

In our future work, we plan to use our debiased evaluation methods to assess the effectiveness of more AL strategies. We also want to complement the investigation reported in this paper, by running online experiments with real users to verify our results.

Acknowledgments

This paper emanates from research supported by a grant from Science Foundation Ireland (SFI) under Grant Number 12/RC/2289-P2, which is co-funded under the European Regional Development Fund.

References

Cañamares, R., and Castells, P. 2018. Should I Follow the Crowd?: A Probabilistic Analysis of the Effectiveness of Popularity in Recommender Systems. In *Procs. of the 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, 415–424.

Carenini, G.; Smith, J.; and Poole, D. 2003. Towards More Conversational and Collaborative Recommender Systems. In *Procs. of the 8th International Conference on Intelligent User Interfaces*, 12–18.

Carraro, D., and Bridge, D. 2018. A More Comprehensive Offline Evaluation of Active Learning in Recommender Systems. In *Procs. of the Workshop on Offline Evaluation for Recommender Systems (Workshop Programme of the 12th ACM Conference on Recommender Systems)*.

Carraro, D., and Bridge, D. 2020. Debiased offline evaluation of recommender systems: A weighted-sampling approach. In *Procs. of the 35th ACM/SIGAPP Symposium on Applied Computing*, 1435–1442.

Chaney, A. J. B.; Stewart, B. M.; and Engelhardt, B. E. 2018. How Algorithmic Confounding in Recommendation Systems Increases Homogeneity and Decreases Utility. In *Procs. of the 12th ACM Conference on Recommender Systems*, 224–232.

Clements, M.; de Vries, A. P.; and Reinders, M. J. T. 2008. Optimizing single term queries using a personalized Markov random walk over the social graph. In *Procs. of the Workshop on Exploiting Semantic Annotations in Information Retrieval*, 18–24.

Cremonesi, P.; Koren, Y.; and Turrin, R. 2010. Performance of Recommender Algorithms on Top-n Recommendation Tasks. In *Procs. of the 4th ACM Conference on Recommender Systems*, 39–46.

Elahi, M.; Ricci, F.; and Rubens, N. 2014. Active Learning Strategies for Rating Elicitation in Collaborative Filtering: A System-wide Perspective. *ACM Trans. Intell. Syst. Technol.* 5(1):13:1–13:33.

Elahi, M.; Ricci, F.; and Rubens, N. 2016. A Survey of Active Learning in Collaborative Filtering Recommender Systems. *Comput. Sci. Rev.* 20(C):29–50.

Liang, D.; Charlin, L.; McInerney, J.; and Blei, D. M. 2016. Modeling User Exposure in Recommendation. In *Procs. of the 25th International Conference on World Wide Web*, 951–961.

Liang, D.; Charlin, L.; and Blei, D. M. 2016. Causal Inference for Recommendation. In *Procs. of the UAI Workshop on Causation*.

Lim, D.; McAuley, J.; and Lanckriet, G. 2015. Top-N Recommendation with Missing Implicit Feedback. In *Procs. of the 9th ACM Conference on Recommender Systems*, 309–312.

Marlin, B. M.; Zemel, R. S.; Roweis, S.; and Slaney, M. 2007. Collaborative Filtering and the Missing at Random Assumption. In *Procs. of the 23rd Conference on Uncertainty in Artificial Intelligence*, 267–275.

Pilászy, I.; Zibriczky, D.; and Tikk, D. 2010. Fast ALS-based Matrix Factorization for Explicit and Implicit Feedback Datasets. In *Procs. of the Fourth ACM Conference on Recommender Systems*, 71–78.

Pradel, B.; Usunier, N.; and Gallinari, P. 2012. Ranking with Non-random Missing Ratings: Influence of Popularity and Positivity on Evaluation Metrics. In *Procs. of the 6th ACM Conference on Recommender Systems*, 147–154.

Schnabel, T.; Swaminathan, A.; Singh, A.; Chandak, N.; and Joachims, T. 2016. Recommendations as Treatments: Debiasing Learning and Evaluation. *CoRR* abs/1602.05352.

Steck, H. 2010. Training and Testing of Recommender Systems on Data Missing Not at Random. In *Procs. of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 713–722.

Wang, Y.; Liang, D.; Charlin, L.; and Blei, D. M. 2018. The Deconfounded Recommender: A Causal Inference Approach to Recommendation. *CoRR* abs/1808.06581.