Random Performance Differences Between Online Recommender System Algorithms

Gebrekirstos G. Gebremeskel[1(✉)] and Arjen P. de Vries[2]

[1] CWI, Amsterdam, The Netherlands
g.g.gebremeskel@cwi.nl
[2] Radboud University, Nijmegen, The Netherlands
arjen@acm.org

Abstract. In the evaluation of recommender systems, the quality of recommendations made by a newly proposed algorithm is compared to the state-of-the-art, using a given quality measure and dataset. Validity of the evaluation depends on the assumption that the evaluation does not exhibit artefacts resulting from the process of collecting the dataset. The main difference between online and offline evaluation is that in the online setting, the user's response to a recommendation is only observed once. We used the NewsREEL challenge to gain a deeper understanding of the implications of this difference for making comparisons between different recommender systems. The experiments aim to quantify the expected degree of variation in performance that cannot be attributed to differences between systems. We classify and discuss the non-algorithmic causes of performance differences observed.

1 Introduction

The literature on recommender systems shows that offline and online recommender system evaluations may not concur with each other [1,3,6]. This is to say that recommender systems may behave differently in offline and online evaluations, both in terms of absolute and relative performance. This has a serious implication for recommender system research, because the whole point of offline evaluation is the assumption that at least the relative performance of recommender systems is indicative of their relative online performance and thus an important step for selecting algorithms that can be deployed in a live recommendation setting.

Prior literature has pointed out a variety of explanations for the performance discrepancy between online and offline evaluations [6,7]. First, offline evaluations can only measure accuracy in a static manner, leaving out the differences between resulting from actual user behaviour. Naturally, offline datasets provide only an incomplete and imprecise model of the real world. The abstraction from user behaviour and context by taking a snapshot of recommendations and user responses may deviate too much from reality to allow for a valid comparison between different recommender systems.

N. Fuhr et al. (Eds.): CLEF 2016, LNCS 9822, pp. 187–198, 2016.
DOI: 10.1007/978-3-319-44564-9_15

The online evaluation of recommender systems overcomes some of these limitations, because we can observe the actual user's responses to recommendations originating from a specific system. A drawback of this setup, however, is the additional "randomness" in the evaluation process that will have to be accounted for. As, each recommendation can only be presented to a single user in his or her real context. The research presented here attempts to improve our understanding of how to accommodate for this element of chance, and still make the right inferences from the evaluation data obtained in CLEF NewsREEL. To identify factors that may explain observed performance differences in online recommender system evaluation, we conduct experiments using several algorithms, *two of which are distinct instances of the exact same algorithm.* We use the experimental results obtained to quantify the effect of randomness in online evaluation on the measured performance.

The paper is organized as follows. In Sect. 2, we discuss our approach, followed by experiments in Sect. 3. In Sects. 4 and 5, we discuss the evaluation results, and identify explanations for the performance differences observed. Section 6 summarizes the lessons learned.

2 Approach

In 2015, we participated in the Living Lab setting of the CLEF News Recommendations Evaluation Lab (NewsREEL) [4]. CLEF NewsREEL is a campaign-like online recommender system evaluation, where participants in need of testing their algorithms are connected with real-life online information portals in need of recommendation services.

In order to investigate the effect of the online setting on the performance measurement of recommendation algorithms, we devised several simple but effective algorithms. Among our algorithms, we included two instances of the same algorithm, with the objective to measure the differences in performance that would have to be attributed to randomness - differences between distinct instances of the exact same algorithm, deployed in the same online recommendation scenario, during the exact same period of operation. A direct comparison of the results that should be identical provides us with the opportunity to consider one instance as the baseline, and obtain a quantitative measure of the performance difference that can only originate from non-algorithmic factors. By also logging the recommendation requests, responses, and clicks, we can recreate the recommendation scenario of one algorithm and compare its results to those that would have been given by the other algorithms. Mixing online and offline evaluation methods provides a more controlled way of measuring differences between different recommender systems, that we can use to estimate the part of the difference in performance that should be attributed to chance.

3 Experiments

We experimented with five algorithms, all of them modifications of a straightforward approach to recommendation based on *recency.* The Recency algorithm

takes into account recency and popularity of an item, and it has been shown to be a strong baseline in previous online evaluations. The algorithmic variations that we experimented with are listed below.

Recency: This algorithm keeps the 100 most recently viewed items for each publisher in consideration for being recommended to the user. The most recently read items are returned in response to a recommendation request. We run two instances of this algorithm to get a sense of the randomness involved in the selection of algorithms by the Plista framework [2] and/or clicks on recommendations by users.

RecencyRandom: Instead of recommending the five or six most recently viewed items, this approach returns a random selection from the top 100 most recently viewed items.

GeoRec: The geographical recommender takes the geographical region (states to be specific) of users and the local category of news items into account when generating recommendations. We generate two sets of recommendations, one by the recency recommender and one by a purely geographical recommender. For the purely geographical recommender, we take the 100 most recently viewed items and sort them according to their geographic conditional likelihood scores generated by Eq. 1:

$$r_{u_a,i_k} = P(c_{i_k}|g_{u_a}) \tag{1}$$

where c_{i_k} is a binary corresponding to the local category of item i_k and g_{u_a} is the state-level geographical information of the user u_a, that is, the state the user belongs to. We combine geographical recommendations with recency recommendations as follows. First, we intersect twice the number of recommendations requested from the geographic recommender with the requested number of recommendations from the recency recommender. If the resulting set is smaller than the number of recommendations requested, we append $half - 1$ items from the geographic recommender and another $half + 1$ from the recency recommender.

GeoRecHistory: This modification of the GeoRec recommender excludes items that the user has already visited from recommendation.

We have run recommendation systems that implement these algorithms over a period of 86 days, between April 12th and July 6th, with one exception; the RecencyRandom algorithm was started 12 days later, on April 24th.

4 Results and Analysis

We present two types of performance scores: cumulative and daily click-through rates (CTR). The cumulative CTR is presented in Table 1. We see that the performance differences are small. If we would rank the algorithms based on their performance, however, we see that the GeoRec recommender leads, followed by Recency and GeoRecHistory. Figure 1 shows the performance measurements by day, for the first 53 days. Figure 2 shows cumulative CTR as a function of the number of days, for the same period.

Table 1. Live performance of the five algorithms

Algorithms	Requests	Clicks	CTR(%)
Recency	56,350	478	0.85
Recency2	53,863	420	0.78
GeoRec	54,338	470	0.86
GeoRecHistory	47,001	395	0.84
RecencyRandom	39,616	283	0.71

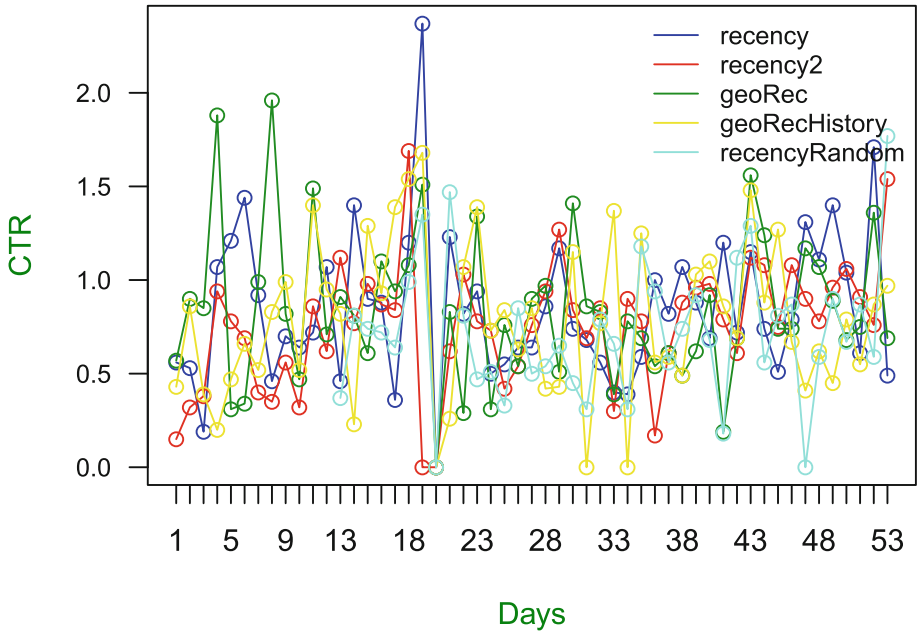

Fig. 1. The daily CTR performances of the five algorithms

From the daily (Fig. 1) and cumulative (Fig. 2) plots, we see that the performance measurements vary considerably. In the cumulative plot, we see that the results for Recency and Recency2 differ considerably during a large part of the evaluation period, although, eventually, converging to a stable situation. If one were to continuously monitor the measured performance of the two algorithms, one might easily conclude (wrongly) that the Recency algorithm is a better approach to recommendation than Recency2.

When observed for a period that is too short, we need appropriate tools to help differentiate the identical recommender systems from their competitors. Imagine for example an experimenter peeking at the experiments every day, to make a decision to choose the best among the competing algorithms. How many times would the experimenter declare statistically significant differences

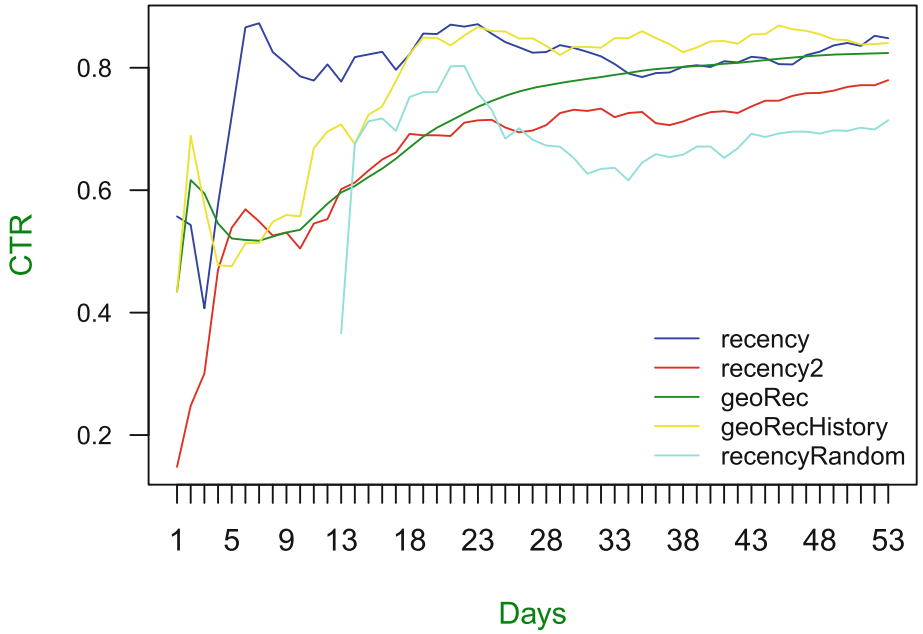

Fig. 2. The cumulative CTR performances of the five algorithms as they progress on a daily basis

between the different algorithms? To compute statistical significance, we used Thumbtack's Abba, a test for binomial experiments [5]. We examined this by using two baselines: the random recommender (RecencyRandom) and Recency2. The results when using the RecencyRandom recommender as a baseline are given in Table 2. Similarly, the results for the baseline of Recency2 are given in Table 3. We see that, when RecencyRandom is used as the baseline, Recency, GeoRec and GeoRecHistory achieve significantly different performance for a majority of the days tested. With Recency2 as the baseline, we see that these percentages are lower; the difference with Recency is considered significant according to the test on two days (perhaps surprising, but a percentage that is in line with the p-value chosen).

The two instances of the same algorithm show large enough differences in performance that there is a chance of concluding one is better than itself. This observation raises questions regarding interpreting the results of the evaluation; it is not so easy to conclude that one algorithm is better than another one based on just an observed difference in performance, even if a statistical test supports that decision. Given the dynamic nature of user-item interactions and the resulting differences in the particular settings that the algorithms operate in, we should be careful when interpreting a small but seemingly significant performance difference. Recommendation evaluations that involve user-item interactions must

Table 2. Statistical significance over the baseline of RecencyRandom. Bracket results are obtained by a recent run.

Algorithm	Days of significance	Percentage (%)
Recency	20	27.4
GeoRec	41	56.2
GeoRecHistory	42	57.5

Table 3. Statistical significance over the baseline of Recency2.

Algorithm	Days of significance	Percentage (%)
Recency	2	2.7
GeoRec	25	34.3
GeoRecHistory	26	35.6

account for some level of randomness, and perhaps a more strict level of statistical significance should be considered than the commonly used 5 %.

5 Causes of Performance Differences

We have seen above that the two instances of the same algorithm can achieve statistically significant difference in performance in an online setting. This is indicative of the extent of performance difference that can arise due to non-algorithmic factors. The two instances of the same algorithm receive different user-item interactions from the evaluation framework. Although they operate in the same recommendation setting, the users and items that they deal with create a unique setting for each instance. We distinguish three types of non-algorithmic factors that may cause the differences in performance: (1) operational differences in the evaluation framework, (2) differences in user-item pairs for which recommendations have been observed, and (3) remaining differences that we consider randomness.

5.1 Operation Causes

By non-algorithmic operational causes, we refer to decisions in the evaluation framework that could affect the observed performance of the recommender systems evaluated. Recommendation systems under evaluation are served requests by a system that distributes the incoming requests in a supposedly "fair" manner. From the perspective of the CLEF NewsREEL participant, fairness of this process is a matter of faith, and difficult to assess. We know that some publishers are more likely to trigger clicks on recommendations than others, such that biases in the distribution of recommendation requests can easily result in performance differences between the algorithms under evaluation. The approach of assigning

recommendation requests to participant systems may exhibit an (implicit) bias with respect to pairing some teams and/or systems with a subset of publishers, or assigning specific users (e.g., those logged-in) to some teams or algorithms, or serving a skewed subset of items from specific categories (e.g., political), or a combination of such factors.

5.2 User-Item Causes

Another source of differences in performance that are not algorithmic could arise due to differences in the sets of items and users that are assigned to the two algorithms. Every algorithm under evaluation receives a different subset of all recommendation requests, resulting in inherent differences in performance if, by chance, certain user-item interactions are incomparable (which would also render the measured results incomparable). In the evaluation of information retrieval systems, for example, it is well known that results obtained on different test collections cannot be compared directly; here, to some extent, we could consider the different performance measurements to result from different test collections, and direct comparison may suffer from the same problems as in the information retrieval evaluation case.

5.3 Random Causes

We refer to all remaining factors that might cause performance differences as random causes, including factors like the user's mood as well as causes that result from idiosyncrasies of the particular datasets (settings, in the online case). Imagine an offline setting with two algorithm (algorithm one and algorithm two) and two datasets (dataset one and dataset two). If on dataset one, algorithm one performs better than algorithm two, but on dataset two the situation is vice versa, the difference between the performance measurements cannot be attributed to the difference in users and items.

One of the advantages of running four algorithms at the same time is that we have datasets that have one big advantage over disparate datasets used for research and that is that we have their online behavior and performance. These logs are, therefore, very important to the performance difference that arises as a result of the random causes in an online setting, as discussed below.

5.4 Overlap in Performance

How can we find out that the random causes (idiosyncrasies of the particular settings) are having an impact on the performance differences of algorithms? To measure the effect of artifacts in evaluation data on performance estimates in an offline setting, we could evaluate two different algorithms on two datasets, and measure the performance differences between the algorithms on each individual dataset. The absolute difference between these two differences can be considered an estimate of the "dataset artifact" on performance. For, if there is no difference, then the measurements are accurate, and both datasets lead to the same

conclusions. However, if a difference is observed, then we would seek the cause for these variations in the differences between the evaluation data. In an online setting, it is not possible to follow this exact procedure, but it is possible to quantify a part of this dataset (setting) artifact using a similar method.

Imagine an ideal world where you can run two algorithms simultaneously in an exactly the same environment for the two algorithms. Users, items, and time are exactly the same. The only things that differ in this ideal world are the recommendations responses by the algorithms. Table 4 shows how different (similar) the recommendation by other algorithms on the different settings would be. The scores are the percentages of shared recommendation over the total number of recommendations. The Table gives two scores for each pair, the first being the exact similarity per recommendation response both in order and content (the number given in each table cell), and the other being the set similarity per recommendation response (order can vary, given between brackets). Each cell corresponds to the similarity measured when the algorithm listed in the column is applied to a dataset constructed from the log obtained when using the algorithm listed in the row. GeoRec-Recency and GeoRec-Recency2 show large similarities, which is not surprising since the GeoRec recommender is only a minor modification of the recency recommender that diversifies its results; which apparently does not diversify the results very much in practice.

Table 4. Shared recommendations. The score in each cell is the percentage of the lists that the two recommendations shared, and the second number, between brackets, is a percentage of the sets of recommendations that the algorithms shared. GeoRec-Recency2 and GeoRec-Recency show the highest similarities.

Algorithms	Recency	GeoRec	RecencyRandom
Recency	100	**85.82(97.96)**	0.0(74.11)
Recency2	100	85.79(97.97)	0.0(73.84)
GeoRec	50.99(91.64)	100	0.0(76.18)
RecencyRandom	0.01(73.28)	0.01(73.40)	100

The idealized system described above would enable us to determine, in the true sense, the algorithm that is the better one; at least, in the evaluation framework in which the algorithms in question are being tested. In practice, such a test would be an approximation, since it does not account to the many factors that can cause performance differences. Obviously such an idealized system is hard to create, but we can create one aspect of that idealized system. That aspect is the overlap in performance that two algorithms would have if they were to be run in the idealized system. The overlap in performance is defined in Eq. 2.

$$Setting_A Overlap_{AB} = \frac{Clicks_{AB}}{Recommendations_{AB}} \tag{2}$$

In Eq. 2, $Setting_A$ is the log generated by running algorithm A, and $Setting_A Overlap_{AB}$ is the overlap in performance of algorithms A and B in

dataset $Setting_A$. $Clicks_{AB}$ and $Recommendation_{AB}$ are counted from intersection of recommended items and the intersections of recommended-and-clicked items respectively of algorithms A and B, when they would be run in an exact online setting that would generate $Setting_A$. The overlap in performance is the ratio of the intersection of recommended-and-clicked items and the intersection of recommended items that two online-deployed algorithms would share if they were to be run in the idealized system. We use this overlap in performance to quantify a part of the performance difference as a result of the random causes by comparing the overlap in performance of two algorithms in two datasets.

To explain how we would obtain the overlap in performance, consider the two algorithms which we used in the NewsREEL challenge. For each algorithm, we have logged the recommendation request, recommendation response, and clicks. If we rerun the other algorithm on the logs of the first algorithm, everything remains the same except the recommendation responses. By determining to what extent the recommendations are the same for the two algorithms, and the ratio of the clicks received by the online-deployed algorithm could also have been obtained by the competing algorithm running on the logs, we obtain the overlap in performance. To obtain the overlap in performance of two algorithms in the idealized system we described, one does not need to run both algorithms online. Running one algorithm online to obtain logs that form a dataset for evaluation, and subsequently running the other algorithm on these logs, is sufficient; for, it is only the overlap of the two algorithms that we are interested, and not the overall performances of the algorithms.

Difference in Overlap. If we have two online-deployed algorithms and record both of their logs, we can determine a measure of overlap between the two algorithms on each of these logs. We call the difference between the two measures of overlap the **difference in overlap**, its definition given by Eq. 3. Note that to compute this difference in overlap, we need to deploy both algorithms and collect their respective logs. If there are no differences in behavior of these algorithms on the same logs, this difference would be zero. The difference in overlap therefore gives us then a measure that quantifies the overall difference in performance that should be attributed to non-algorithmic causes.

$$DiffinOverlap_{Setting_A Setting_B} = |Setting_A Overlap_{AB} - Setting_B Overlap_{AB}| \tag{3}$$

Since we have four algorithms that ran during the complete evaluation campaign (excluding GeoRecHistory), we can quantify differences in overlap between several pairs of algorithms, and, together, these differences in overlap will give us a clue of the extent to which performance differences between algorithms should be attributed to chance. In other words, even though the full difference in overlap cannot be measured, as we can not create the idealized system where two different algorithms would receive the exact same recommendation requests for the exact same user and item combinations, by zooming in on the performance

Table 5. Difference in overlap of our algorithms. Each entry is obtained by subtracting overlap in performance in one dataset of two algorithms from their overlap in performance in another dataset. GeoRec-Recency2 and GeoRec-RecencyRandom show the highest overlap difference

Algorithms	Recency	Recency2	GeoRec	RecencyRandom
Recency	0	0	0.001	0.006
Recency2		0	**0.026**	0.004
GeoRec				**0.026**

overlap we can still obtain an estimate of the level of non-algorithmic differences in the evaluation.

To calculate the difference in overlap, we make one assumption, and that is that we do not take into account the order of the recommended items. If two algorithms have recommended two lists of the same items, but in different order and a click happened on the online deployment, we consider a click happened on the latter too, regardless of the order. Also, the CTR scores were expressed as percentages before any calculations. We take the absolute value as we are interested in the magnitude only. The results are presented in Table 5. To help interpret the Table, the score listed in the cell Recency2-GeoRec corresponds to the difference in overlap between Recency2 and GeoRec obtained as the difference between the overlaps in performances of Recency2 and GeoRec when they ran in two identical online settings (which are represented by the logs of Recency2 and the logs of GeoRec).

The highest differences in overlap observed are between Recency2 and GeoRec and between GeoRec and RecencyRandom, each equal to **0.026**. Given that GeoRec and Recency are closely related algorithms, and Recency and Recency2 are identical, one would expect that the differences in overlaps of GeoRec-Recency, and GeoRec-Recency2 should have been the same, and smaller than the difference in overlap of Georec-RecencyRandom. In an ideal evaluation environment, we would expect the difference in overlap to equal 0, because we would assume that the two settings under which the two algorithms run should affect the two algorithms in similar ways. Why do the two settings then affect the two algorithms in different ways? The positive scores of differences in overlap, we argue, are a results of the idiosyncrasies of the particular settings.

6 Conclusion

We set out to investigate the performance differences in online algorithms. We employed several algorithms among which were two instances of the same algorithm. We demonstrated that two instances of the same algorithms may diverge, and occasionally even to the extent of showing statistically significant differences in performance. The difference in performance seems to indicate that care must be taken to take into account some degree of randomness in recommender

systems evaluation that involve users in a live setting, in addition to statistical significance tests using commonly used statistical significance levels.

We classified and discussed the possible causes of performances differences between online-deployed algorithms and argued that even in the absence of obvious causes of performance differences such as operational biases and the selection of users and items observed in the experiment, performances can vary due to other artifacts in the data collected. These artifacts will also exist in offline datasets, but in the online setting, the researcher is much more susceptible to being mislead by such artifacts, as it involves users and items and their dynamic interactions. We cannot claim that these artifacts are the sole reason for observed significant performance differences between two instances of the same algorithm; and forming an important confounding factor when comparing any two algorithms in general. We may however conclude that we have to take into account these random biases that can only be smoothed out over a sufficiently long evaluation period.

Our results suggest that we should be reluctant in adopting small (statistically significant) improvements as indicative of real performance differences when the evaluation involves real world settings, users and items. We have proposed a new method to quantify the effect of randomness in the evaluation by zooming in on the differences in overlap of the results obtained from two competing algorithms, that are tested on two settings simultaneously. In future work, we plan to develop this approach further to help understand the level of randomness that we should take into account when we compare the performance measurements obtained in an online experiment, to help improve inferences about the quality of different recommender systems.

Acknowledgements. This research was partially supported by COMMIT project Infiniti.

References

1. Beel, J., Genzmehr, M., Langer, S., Nürnberger, A., Gipp, B.: A comparative analysis of offline and online evaluations and discussion of research paper recommender system evaluation. In: Proceedings of the International Workshop on Reproducibility and Replication in Recommender Systems Evaluation, pp. 7–14. ACM (2013)
2. Brodt, T., Hopfgartner, F.: Shedding light on a living lab: the clef newsreel open recommendation platform. In: Proceedings of the 5th Information Interaction in Context Symposium, pp. 223–226. ACM (2014)
3. Garcin, F., Faltings, B., Donatsch, O., Alazzawi, A., Bruttin, C., Huber, A.: Offline and online evaluation of news recommender systems at swissinfo. In: Proceedings of the 8th ACM Conference on Recommender Systems, pp. 169–176. ACM (2014)
4. Hopfgartner, F., Kille, B., Lommatzsch, A., Plumbaum, T., Brodt, T., Heintz, T.: Benchmarking news recommendations in a living lab. In: Kanoulas, E., Lupu, M., Clough, P., Sanderson, M., Hall, M., Hanbury, A., Toms, E. (eds.) CLEF 2014. LNCS, vol. 8685, pp. 250–267. Springer, Heidelberg (2014)
5. Howard, S.: Abba: Frequently asked questions. https://www.thumbtack.com/labs/abba/. Accessed 18 July 2016

6. Kirshenbaum, E., Forman, G., Dugan, M.: A live comparison of methods for personalized article recommendation at forbes.com. In: Flach, P.A., De Bie, T., Cristianini, N. (eds.) ECML PKDD 2012, Part II. LNCS, vol. 7524, pp. 51–66. Springer, Heidelberg (2012)
7. McNee, S.M., Kapoor, N., Konstan, J.A.: Don't look stupid: avoiding pitfalls when recommending research papers. In: Proceedings of the 2006 20th Anniversary Conference on Computer Supported Cooperative Work, pp. 171–180. ACM (2006)