Precision-Oriented Evaluation of Recommender Systems: An Algorithmic Comparison

Alejandro Bellogín, Pablo Castells, Iván Cantador
Universidad Autónoma de Madrid, Escuela Politécnica Superior
Ciudad Universitaria de Cantoblanco, 28049 Madrid, Spain
{alejandro.bellogin, pablo.castells, ivan.cantador}@uam.es

ABSTRACT

There is considerable methodological divergence in the way precision-oriented metrics are being applied in the Recommender Systems field, and as a consequence, the results reported in different studies are difficult to put in context and compare. We aim to identify the involved methodological design alternatives, and their effect on the resulting measurements, with a view to assessing their suitability, advantages, and potential shortcomings. We compare five experimental methodologies, broadly covering the variants reported in the literature. In our experiments with three state-of-the-art recommenders, four of the evaluation methodologies are consistent with each other and differ from error metrics, in terms of the comparative recommenders' performance measurements. The other procedure aligns with RMSE, but shows a heavy bias towards known relevant items, considerably overestimating performance.

Categories and Subject Descriptors: H.3.3 Information Search and Retrieval – *information filtering*.

General Terms: Algorithms, Performance, Experimentation.

Keywords: Evaluation, precision metrics, error metrics.

1. INTRODUCTION

The evaluation of Recommender Systems (RS) has been an explicit object of study in the field since its earliest days, and is still an area of active research, where open questions remain [3,10]. The dominant evaluation methodologies in off-line experimentation have been traditionally error-based. There is however an increasing realization that the quality of the ranking of recommended items can be more important in practice (in terms of the effective utility for users) than the accuracy in predicting specific preference values. As a result, precision-oriented metrics are being increasingly often considered in the field. Yet there is considerable divergence in the way these metrics are being applied by different authors, as a consequence of which, the results reported in different studies are difficult to put in context and compare.

In the typical formulation of the recommendation problem, user interests for items are represented as numeric ratings, some of which are known. Based on this, the task of a recommendation algorithm consists of predicting unknown ratings based on the known ones and, in some methods, some additional available information about items and users. With this formulation, the accuracy of recommendations has been evaluated by measuring the error between predicted and known ratings, by metrics such as the Mean Average Error (MAE), and the Root Mean Squared Error (RMSE). Although dominant in the literature, some authors have argued this evaluation methodology is detrimental to the field since the recommendations obtained in this way are not the most useful for users [9]. Acknowledging this, recent works evaluate top-N ranked recommendation lists with precision-based metrics [2,8,5,1], drawing from well-studied methodologies in the Information Retrieval (IR) field.

Precision-oriented metrics measure the amount of relevant and non-relevant retrieved items. A solid body of metrics, methodologies, and datasets has been developed over the years in the IR field to measure this in different ways. There is however a major difference between the RS and IR experimental settings. In ad-hoc IR experiments, relevance knowledge is typically assumed to be (not far from) complete –mainly because in the presence of a search query, relevance is simplified to be a user-independent property. However, in RS it is impractical to gather complete preference information for *each* user in the system. In datasets containing thousands of users and items, only a fraction of the items that users like is generally known. The unknown rest are, for evaluation purposes, assumed to be non-relevant. This is a source of –potentially strong– bias in the measurements depending on how unknown relevance is handled.

To the best of our knowledge, a thorough analysis of precision-oriented methodologies is still missing in the field. In fact, the detailed alternatives in the evaluation procedures are not clearly identified, and the extent to which they are equivalent or provide comparable results has not been studied in depth. Reported results differ in several orders of magnitude for the same metric on similar datasets and similar algorithms in different studies.

In this paper, we present a general methodological framework for evaluating recommendation lists, covering different evaluation approaches, most of them documented in the literature, and some included here for the sake of the systematic consideration of alternatives. The considered approaches essentially differ from each other in the amount of unknown relevance that is added to the test set. We use Precision, Recall, and normalized Discounted Cumulative Gain (nDCG) [10] as representative precision-oriented metrics. The purpose of our study is to assess the differences and potential equivalences resulting from the methodological variants, and to what extent precision-based results relate or differ from error-based metrics. We found that four of the methodologies are consistent with each other in terms of the observed recommenders' performance trend. The other methodology considerably overestimates performance, and suffers from a strong bias towards known relevant items.

2. A GENERAL METHODOLOGY FOR EVALUATING RANKED ITEM LISTS

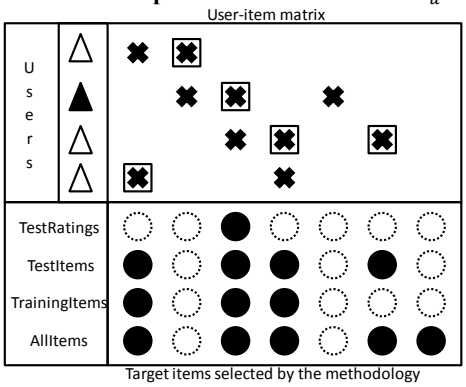
In order to evaluate ranked recommendations for a target user u, we typically select a set L_u of target items the recommender shall rank. For that user and each item i in the list, we request a rating prediction $\hat{r}(u,i)$ from the recommender, and we sort the target items by decreasing order of predicted rating value. In IR terminology, we sort L_u based on the retrieval function $\hat{r}$. In off-line

Copyright is held by the author/owner(s).
RecSys'11, October 23–27, 2011, Chicago, Illinois, USA.
Copyright 2011 ACM 978-1-4503-0683-6/11/10...$10.00.

recommender system evaluation methodologies, a subset of the known ratings is held off from the recommender as ground truth for testing. These ratings play the role of known relevance in the computation of precision metrics: highly rated items are considered relevant, and items with low ratings, or unrated, are taken as non relevant. A (domain-specific) threshold value is often specified to define what a high rating means.

The main substantial difference between how this general evaluation scheme has been brought to practice by different authors lies on how the set L_u of target items is formed. In the following subsections, we present some plausible generation strategies for the list L_u, and, when applicable, we reference works where each strategy has been explicitly reported. Figure 1 summarizes how the first four evaluation procedures differ from each other (note that items already present in the user's profile are excluded in the evaluation). We can see here that all the methodologies (except AllItems) ignore some of the items. One of them (TestRatings) uses a much smaller number of items than the others.

Figure 1. Graphical representation of the four methodologies. Target user is represented with a solid triangle, crosses represent the users' ratings (unboxed ratings denote the training set). Black circles represent items included in L_u.



In the remainder, we will use the following additional notation. $\mathcal{U}$ and $\mathcal{I}$ denote the set of all users and items, and $\mathcal{T} \subseteq \mathcal{U} \times \mathcal{I}$ is the subset of user-item pairs for which the rating is known. $\mathrm{Tr} \subsetneq \mathcal{T}$ and $\mathrm{Te} \subsetneq \mathcal{T}$ denote the training and test subsets into which the set of known ratings is split for evaluation. We shall note as $\mathrm{Te}_u = \{i \in \mathcal{I} \mid r(u,i) \in \mathrm{Te}\}$ the set of items rated by u in the test set, where $r(u,i)$ denotes the known rating for i by u, similarly, we define Tr_u for the set of items rated by u in the training set.

2.1 TestRatings methodology

This procedure takes the same target item sets as error-based evaluation. That is, for each user u, the list L_u consists of items rated by u in the test set, $L_u = \mathrm{Te}_u$. This is the methodology that selects the smallest set of target items for each user, including no unrated items at all. Different from the other procedures, in this methodology, L_u is therefore distinct for each user. The methodology is used in [4] and [5], with a relevance threshold value of 4 in a 1-5 rating scale.

2.2 TestItems methodology

In contrast to the previous methodology, this one adds unrated –therefore non-relevant– items to L_u. Specifically, the list includes for all users, all the items having a test rating by some user –and no training rating by the target user, since it makes no sense to predict known training ratings–, that is $L_u = \bigcup_v \mathrm{Te}_v \setminus \mathrm{Tr}_u$. In this way, this list can be precomputed for all the users at the beginning. An advantage of this approach is that all users are tested on the same set of target items (except for the exclusion of items with training ratings for each target user), which unifies the test conditions for all users. This methodology has been used in [1] to evaluate item-based Collaborative Filtering (CF) recommenders.

2.3 TrainingItems methodology

An adaptation of the previous methodology when no information about the ground truth (test set) wants to be used is defined by selecting all the items belonging to the training set instead of test set. In other words, every item rated by some user in the system is selected –except, again, those rated by the target user–, i.e., $L_u = \bigcup_{v \neq u} \mathrm{Tr}_v$. This methodology may be useful when simulating a real system where no test is available.

2.4 AllItems methodology

A further more general alternative would be to select the whole set of items (except, as before, those already rated by the target user): $L_u = \mathcal{I} \setminus \mathrm{Tr}_u$. Thus, items with no ratings may appear in the recommendation list. This makes no difference with respect to TrainingItems for CF recommenders, since the algorithm would not be able to recommend any item having no training ratings, but could still make a difference for algorithms using other information besides ratings (e.g. content-based). Note that compared to TestItems, AllItems just adds a set of items which have no test ratings –hence they are non relevant items. This should result in a slight precision decrease which may be expected to affect all recommenders evenly.

2.5 One-Plus-Random methodology

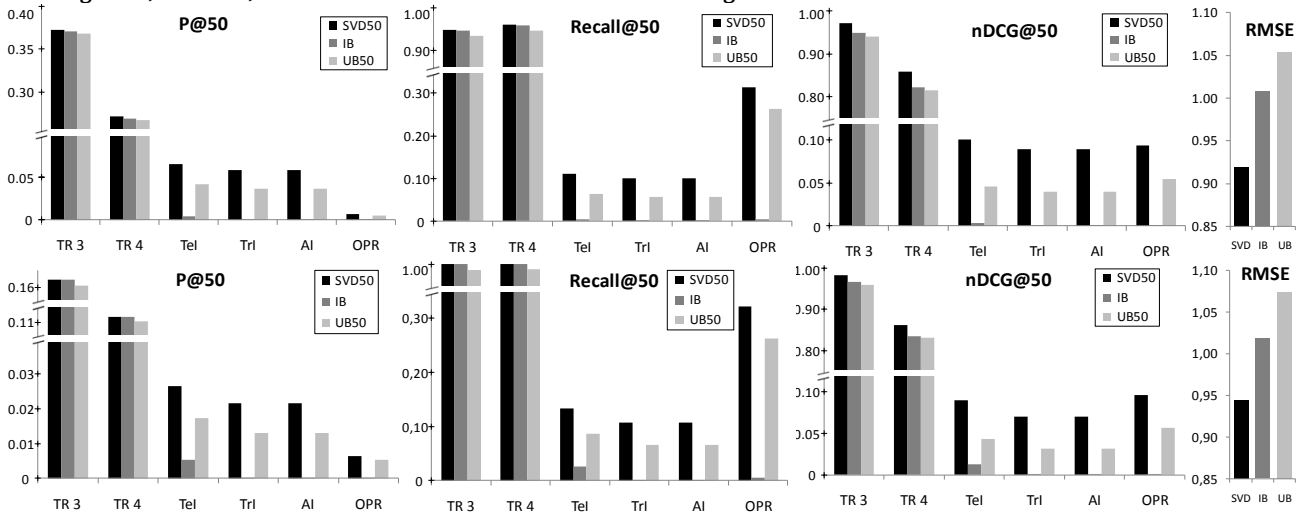
A more elaborate methodology has been recently proposed in [6] and [2]. For each user, a set of highly relevant items is selected among those contained in the test set, that is, $\mathrm{HR}_u \subseteq \mathrm{Te}_u$. Then, a set of non-relevant items is created by randomly selecting N additional items, which we denote as NR_u. In [2], the authors set N = 1000 and clarify that the set NR_u is selected out of those items in the test set not rated by u. Finally, for each item i in HR_u, the recommender is requested to produce a ranked recommendation of the set $L_{ui} = \{i\} \cup \mathrm{NR}_u$. Precision and recall metrics for each user are calculated by averaging the precision and recall values obtained for the rankings associated to L_{ui}, over all items $i \in \mathrm{HR}_u$. The final performance values are simply the average of the values obtained for each user.

In [2], the authors only consider precision and recall, but it is possible to similarly compute any other precision-oriented metric such as nDCG upon this procedure.

2.6 Discussion

Among the aforementioned set of methodologies, we can identify TestRatings as a 'cheap' methodology, in the sense that the evaluation cost is the same as that of evaluating error-based metrics, and the set of recommendations to be computed is minimum compared to the other procedures. Since relevant items tend to be rated much more frequently than non-relevant ones, the recommender's performance is significantly overestimated compared to real-world (fair) situations. TrainingItems and AllItems methodologies represent the opposite end of the spectrum, adding a high number of unrated items. These are considered non-relevant, thus missing a small fraction of unknown positive relevance, and underestimating precision. Finally, TestItems and One-Plus-Random methodologies appear to be somewhat fairer with respect to considering relevant and non-relevant items at the same time. The latter, however, may depend on the number N of non-relevant

Figure 2. Comparison results, D1 above, D2 below. TR_3 and TR_4 stand for the TestRatings evaluation methodology with threshold rating values for relevant items set to 3 and 4, respectively. TeI, TrI, AI, and OPR respectively denote TestItems, TrainingItems, AllItems, and One-Plus-Random evaluation methodologies.



items considered (the higher this number, the smaller the performance values) and, furthermore, on how the highly relevant items are selected. In [6], the authors suggest to choose only those top-rated items. But there may be users who never rate items with that value and, in that case, the performance of those users could not be evaluated. This drawback could be avoided by defining "top-rated" with respect to each user's rating scale.

3. EXPERIMENTS

We now compare empirically the evaluation methodologies presented in the previous section. Specifically, we compare different evaluation metrics applied to some state-of-the-art recommendation algorithms in two different training-test configurations, which differ in how the test rating set is created.

In order to favor repeatability and comparison of results, we use two predefined splits in the Movielens 100K dataset –which includes 943 users and 1682 items. Table 1 shows some statistics of the dataset splits, such as the density and average number of users and items in training and test for each split. The first partition (D1) includes five disjoint random splits for cross-validation. The second dataset (D2) restricts the number of items each user has in the test set, creating a richer situation with respect to the number of users being evaluated and the amount of available training data.

Table 1. Characteristics of the two evaluated training/test sets.

	D1	D2
Split information	80% / 20% 5 splits	10 items per user in test 2 splits
Average users in training / test	943 / 766.2	943 / 943
Average items in training / test	1651.6 / 1410.8	1677.5 / 1137
Average density in training / test	0.051 / 0.020	0.057 / 0.009

All the metric values reported herein have been computed using the *trec_eval* program, considering users in place of queries, and test ratings as relevance judgments (*qrels*). We show results on precision, recall, and nDCG at cutoff 50, though we obtained similar results for different cutoffs such as 5 and 10.

We ran three well-known state-of-the-art CF algorithms as implemented in the Mahout library: a user-based strategy with 50 neighbors, denoted as UB50, an item-based strategy using adjusted cosine, denoted as IB, and a matrix factorization technique with 50 factors, denoted as SVD.

Figure 2 summarizes the obtained results. A first relevant observation is that the comparative results with precision metrics are not quite the same as with error metrics. Specifically, RMSE would suggest that the IB recommender performs better than UB, while in terms of precision metrics, IB appears to clearly underperform in all the experimental configurations except TestRatings, which aligns with RMSE. This suggests that IB better predicts low rating values than UB, but does a poor job with the top user preferences –a nuance that RMSE does not care for, but real users certainly would.

TestRatings results, on the other hand, match error-based metrics, probably because they are computed with the same item set. In particular, as stated in [8], this methodology would create a ranked list consisting of the top *rated* items, which may or may not be related with the recommended items the user would actually get in a real application. Counting non-rated items as non-relevant, in addition to low-rated items, results in an underestimation of the true precision (named "modified precision" in [8]), but may provide a better indicator of the actual user's experience.

We also observe a significant difference in the absolute performance values (for any metric) obtained by the TestRatings as compared to the rest of methodologies. In fact, these results are similar to those reported in [8], since for all the methodologies except TestRatings, performance values are very small.

We also see that the threshold value does make a difference in the performance values –although the relative comparison between recommenders remains the same. Raising the relevance threshold makes items rated as 3 in the top 50 now be considered irrelevant, whereby P@50 naturally decreases. At the same time, the total number of relevant items drops even faster, whereby recall increases. The measured recall values are in fact extremely high, over 0.9, which is clearly far from reflecting a realistic assessment.

Finally, we observe that the methodologies are consistent when we compare the two evaluated splits: the same trend is observed, although different absolute values are obtained. This result is not straightforward, since in these experiments, the number of non-relevant items is usually overestimated (an item which does not appear in the user profile does not necessarily imply that it is non-relevant –perhaps the user was not even aware of its existence). In fact, as discussed in [3], recall depends heavily on the number of relevant items that each user has rated, and therefore, it should only be used for comparison purposes, not interpreted as an abso-

lute measure. Hence, once this situation has been normalized (D2), the performance results might vary, which does not occur in our experiments. In conclusion, the evaluated methodologies are consistent with respect to the test size, since they obtain almost the same results in both situations, whether the test size for each user is random (D1) or fixed (D2).

4. DISCUSSION

As we described above, there are inconsistencies between TestRatings and the rest of the methodologies, mainly because this procedure does not consider user-item pairs without test rating in the L_u target lists, and thus, it only evaluates recommendations over known relevance, which is an unrealistic situation in practice. In terms of absolute performance values, this methodology does not discriminate well among the recommenders, in comparison with the other methodologies.

The One-Plus-Random procedure, on the other hand, is not able to find so many relevant items in the top-50 as the rest of methodologies since, by definition, each produced ranking contains only one relevant item. However, since that one item is highly relevant (a '5' rating value), it results in a nDCG value comparable to those of the rest of methodologies. The relative overall comparison between systems is the same as in the other methodologies, except TestRatings.

We can also observe that TrainingItems and AllItems are completely equivalent, which is natural since there are no unrated items in this dataset and only CF recommenders have been evaluated. Besides, these methodologies always give lower performance values than TestItems since, as discussed earlier, they add non-relevant items in the final ranking list.

Finally, our study also suggests that taking the absolute values of metrics literally may be misleading. We see that most of the time, in our experiments, all the procedures except TestRatings (the less realistic one) result in metric values below 0.1. This does not imply the systems are unacceptably bad as it might appear – these systems get RMSE around or below 1, which is not optimal but acceptable state-of-the-art. Although such precision values are low, they afford a sound relative comparison of systems to each other.

5. CONCLUSIONS AND FUTURE WORK

We have described and compared five evaluation methodologies proposed in the literature on three state-of-the-art recommenders under two evaluation conditions, i.e., different training/test set generation. In the experiments, we have found four methodologies that are consistent with each other, in the sense that the same trend is observed on the performance of the recommenders. The other methodology has proved to overestimate performance values, and leads to a different comparative assessment of the recommenders.

Our experiments lead to questioning again the suitability of error metrics. As in [8], we have found that there is no direct equivalence between results with error-based and precision-based metrics. Common sense suggests that putting more relevant items in the top-N is more important for real recommendation effectiveness than being accurate with predicted rating values, which are usually not even shown to real users. Our study confirms that measured results differ between these two perspectives. An online experiment, where real users' feedback is contrasted to the theoretic measurements, might shed further light for an objective assessment and finer analysis of which methodology better captures user satisfaction.

The suitability of precision-oriented metrics to a typical RS evaluation framework also deserves further and deeper investigation. We have shown different methodologies whose differences arise in how unknown relevance is added to the test set. This is a key point when evaluating RS, in contrast to IR, since we have to define training and test sets, whereas in IR, we would have the whole dataset available, first, for the indexing task, and then, for the retrieval task. In RS, we need to separate the data into training and test; the more the training available, the better the algorithm will learn the users' preferences. However, the smaller the test set, the smaller the confidence about the obtained results. Furthermore, depending on how the recommendation lists are created different performance values are obtained with precision-oriented metrics, as we have presented here. An alternative solution may be to obtain a deeper integration between RS and IR algorithms, like the one presented recently in [1].

On the other hand, we have focused on different methodologies, not considering the metrics themselves. We plan to test how other IR metrics, such as Mean Average Precision (MAP), and Mean Reciprocal Rank (MRR), and other metrics proposed in the context of RS, such as NDPM and ROC curve [10], compare with the results presented here, and whether they are sensitive or not to the experimental configuration, as we have analyzed for precision-oriented metrics. An additional source of divergence in evaluation results is how the training and test sets are created. In this paper, we have experimented with two different splits of a particular dataset: random 5-fold cross-validation partitions, and a data split where the number of items in the test set is fixed. Many further options can be considered, such as leave-one-out, selecting a percentage of data per user, and restricting the number of items in the training set for each user. An important restriction, not commonly taken into account, is that of generating a temporal split, which better reflects real-world conditions in which recommender systems work. Some authors (see [7], among others) have explored this issue in the context of the Netflix dataset. A comprehensive analysis of precision-oriented metrics in that context would be worthwhile.

Acknowledgements. This work was supported by the Spanish Ministry of Science and Innovation (TIN2008-06566-C04-02) and the Community of Madrid (CCG10-UAM/TIC-5877).

6. REFERENCES

[1] A. Bellogín, J. Wang, and P. Castells. Text Retrieval Methods Applied to Ranking Items in Collaborative Filtering. In *ECIR*, Springer, 2011

[2] P. Cremonesi, Y. Koren, and R. Turrin. Performance of Recommender Algorithms on Top-N Recommendation Tasks. In *RecSys*, ACM, 2010

[3] J. L. Herlocker, J. A. Konstan, L. G. Terveen, J. T. Riedl. Evaluating Collaborative Filtering Recommender Systems. *ACM Transactions on Information Systems*, ACM, 2004

[4] T. Jambor and J. Wang. Goal-Driven Collaborative Filtering – A Directional Error Based Approach. In *ECIR*, Springer, 2010

[5] T. Jambor and J. Wang. Optimizing Multiple Objectives in Collaborative Filtering. In *RecSys*, ACM, 2010

[6] Y. Koren. Factorization Meets the Neighborhood: a Multifaceted Collaborative Filtering Model. In *KDD*, ACM, 2008

[7] N. Lathia, S. Hailes, and L. Capra. kNN CF: A Temporal Social Network. In *RecSys*, ACM, 2008

[8] M.R. McLaughlin and J.L. Herlocker. A Collaborative Filtering Algorithm and Evaluation Metric That Accurately Model the User Experience. In *SIGIR*, ACM, 2004

[9] S. M. McNee, J. Riedl, J. A. Konstan. Being Accurate Is Not Enough: How Accuracy Metrics Have Hurt Recommender Systems. In *CHI '06 Extended Abstracts in Computing Systems*, ACM, 2006

[10] G. Shani and A. Gunawardana. Evaluating Recommendation Systems. *Recommender Systems Handbook*, Springer, 2011