A Top-N Recommender System Evaluation Protocol Inspired by Deployed Systems

Alan Said
alan@cwi.nl

Alejandro Bellogín
alejandro.bellogin@cwi.nl

Arjen de Vries
arjen@acm.org

Centrum Wiskunde & Informatica
Amsterdam, The Netherlands

ABSTRACT

The evaluation of recommender systems is crucial for their development. In today's recommendation landscape there are many standardized recommendation algorithms and approaches, however, there exists no standardized method for experimental setup of evaluation – not even for widely used measures such as precision and root-mean-squared error. This creates a setting where comparison of recommendation results using the same datasets becomes problematic. In this paper, we propose an evaluation protocol specifically developed with the recommendation use-case in mind, i.e. the recommendation of one or several items to an end user. The protocol attempts to closely mimic a scenario of a deployed (production) recommendation system, taking specific user aspects into consideration and allowing a comparison of small and large scale recommendation systems. The protocol is evaluated on common recommendation datasets and compared to traditional recommendation settings found in research literature. Our results show that the proposed model can better capture the quality of a recommender system than traditional evaluation does, and is not affected by characteristics of the data (e.g. size. sparsity, etc.).

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval - Information filtering, Retrieval models, Selection process; H.3.4 [**Information Technology and Systems Applications**]: Decision Support; H.5.1 [**Multimedia Information Systems**]: Evaluation/ methodology

General Terms

Algorithms; Design; Experimentation; Measurement; Performance

Keywords

Evaluation; Benchmarking; Experimental Settings; Replication; Evaluation Protocol; Deployed Systems

LSRS '13, October 13, 2013, Hong Kong, China

1. INTRODUCTION & RELATED WORK

Recommender system research has seen a large increase in the development of algorithms, scalability and areas of application during the last two decades. One aspect of recommender systems that has not evolved as fast is their evaluation. Most early recommendation approaches were inspired by related concepts from machine learning and/or information retrieval. Today, many evaluation concepts are brought verbatim from these and related fields [6–8]. We emphasize that one should take into consideration the conceptual differences between *an item being recommended* to a user (recommendation) and *a query being posted* in order to find a relevant item (information retrieval), and other similar aspects.

Traditional evaluation of recommender systems is based on the training/test paradigm, where a fraction of the available data is used for algorithm training, and the remaining part is used to evaluate, or *test*, the quality of said recommendation algorithm. Methods to mitigate the effects of overfitting, like n-fold (or random) cross validation [6, 16], are often applied in conjunction with this. More elaborate techniques where the data is merged or altered to produce better results also exist, e.g. the *"One+Random"* evaluation methodology by Cremonesi et al. [5] which proposes the addition of randomly selected items to the one most relevant item for each user. However, even with these mitigation techniques, there remains a discrepancy in the *offline* evaluation protocols, and the *online* deployment and accuracy estimate of the algorithms in a real-life setting. In several cases it has even been shown that high predictive accuracy can be detrimental to the users' perceived quality of the systems [4, 11, 15]. Several methods attempt to overcome this discrepancy, including multiple user-centric evaluation approaches, A/B-testing, guidelines and frameworks [9, 12, 13]. Many of these require that the systems are evaluated through time-consuming user studies and surveys, often involving the prerequisite of a recommender system already in use.

In this paper we present an empirical evaluation protocol for recommender systems based on offline evaluation concepts such as *top-N* and *leave one out*. The protocol is developed with the *"Find good items"* and *"Find all good items"* user tasks in mind, as defined by Herlocker et al. [6], it is however general enough to be applied to other defined user tasks, e.g. *"Recommend Sequence"* and *"Just Browsing"*. The protocol attempts to mimic the data properties in live recom-

mender system evaluation by creating more realistic training and test data splits allowing the recommender system to be evaluated in a setting which is more similar to a deployed (production) system.

Our results show that, in comparison to a traditional evaluation approach, the introduced model gives more insight into the quality of recommendations in different scenarios (e.g. the length of the recommended list), and may even lower the evaluation time in some evaluation scenarios.

The main contributions of this paper are an *evaluation protocol* simplifying the comparison and replication of the results obtained in a recommender system evaluation and *a model for more accurately capturing the quality of a recommender system* for different types of users.

2. EVALUATION PROTOCOL

The proposed evaluation protocol attempts to capture aspects from real, product-oriented recommendation systems which interact with users, e.g. services such as Netflix [1], Movielens[2], etc. It does so by specifying what data will be available for the training and for the evaluation of the system, as well as the type of items the system recommends. As stated above, the protocol has been developed specifically with a "find good items" scenario in mind [6], which is reflected in the method for training and test splitting, candidate item selection and candidate user filtering below.

2.1 Personalized Training & Test Datasets

When considering a deployed recommendation system, it should be safe to assume that the *complete* interaction history (the given ratings or previous clicks/purchases) of each user is available for training the recommendation algorithm. In a traditional evaluation setting, like those described in [6], the training and test split is performed on the complete population of users. Usually this is done regardless of how many ratings (or other interactions) are recorded for each user. The strategy comes with the risk of creating a training model with users having no or very few items in the training or test sets. This aspect is however commonly overlooked, instead a random sample of the candidate user's interactions is selected as a test set, and the rest is kept for training [16]. This type of training and test split is visualized in Fig. 1a which shows an $80\% - 20\%$ split where white and black squares represent the training and test sets respectively.

In order to mimic the recommendation accuracy of a deployed system, we need to take several factors into consideration. For instance, the most accurate recommendations for each user will only be found if all of the available data is used to train the recommendation algorithm. To achieve this, personalized training and test splits must be created. This allows the algorithm to be trained on all available ratings or interactions, with the exception of the *candidate* user's test set. Fig. 1b shows an example of this splitting strategy; black and white squares represent the test and training items for one user respectively. The actual splitting can be based on some predefined conditions, e.g. a rating value, at random, or based on a temporal threshold such that only item interactions up to a certain point in time are selected into the test set [2], as illustrated in Fig. 1c. The end effect of this splitting procedure is that instead of one global training and test split, we have one instance of the training and test split per user. These personal data splits are then used to train and evaluate each user separately.

2.2 Candidate Item Selection

Before performing the training/test splits described above, two aspects of our test sets (the candidate items) need to be considered. First, whenever the evaluation metrics are calculated at a given cutoff point, e.g. precision@N, the calculation of an accuracy at N when there are fewer than N potentially *good* recommendations is problematic. The reason for this is that even an optimal recommender that "cheats" and peeks into the test set – an *oracle* – will not be able to achieve a perfect precision score (i.e. 1.0). As a means of overcoming this issue, we propose that each test set contains exactly N relevant items for the candidate user. When precision is used as the evaluation metric, the effect will be precision at the level of recall, i.e. *R-precision* [10].

The second aspect to be considered ties to the type of evaluation we want to perform. If the focus is "find good items", a lower bound on the ratings given to potential test items can be employed, we refer to this approach as the *relevance threshold*. The definition of relevance can be tailored to the underlying data and/or recommendation scenario, e.g. a global or personalized rating, a random selection, a selection based on item popularity, etc.

2.3 Candidate User Filtering

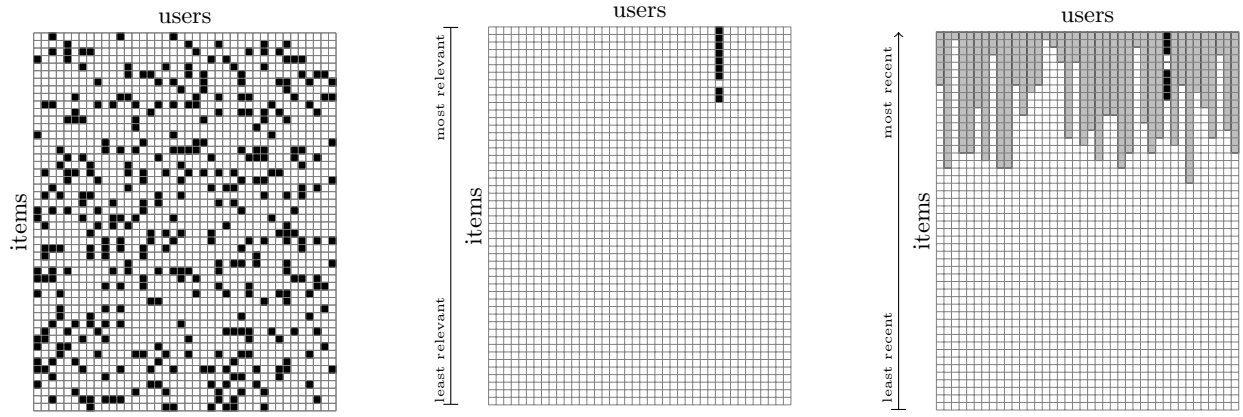
Similarly to the candidate item selection described above, in order to ensure a realistic measure of recommendation accuracy, there are some constraints on our candidate users as well.

Let us consider the evaluation of a recommendation algorithm using precision at N as the accuracy metric. We can assume that all users will not have rated the same amount of items. More realistically, this will likely range from some users having rated very few items to some users having rated very many items [3, 14, 17]. If we select a low N, users with many ratings in the test set will most likely achieve a higher level of accuracy than users with few ratings in the test set, and vice versa. In order to mitigate the effect of this, we suggest that a *minimum rated items* constraint, M, is introduced. M needs to be larger than N to allow a non-empty training set. This constraint is used for selecting which users will be available for accuracy evaluation at different N's. The ratio between M and N specifies how many items the candidate user needs to have in the training set with respect to the test set, since predicting recommendations for users with no or few items in the test set will result in low accuracy scores.

2.4 Proposed Evaluation Setting

To ensure more realistic accuracy values, we propose that the evaluation process be adapted to the deployed system concept as well. With this goal in mind, we instantiate the proposed evaluation protocol as presented in Algorithm 1. Here, we can observe that different training and test splits

[1] `http://www.netflix.com`
[2] `http://www.movielens.org`



(a) A random $80\% - 20\%$ (machine learning type) training and test data split for all users. The split is random, items are not sorted. Only one split is performed for the complete user population.

(b) The proposed training and test data split for one user. Items can be sorted by random, or by relevance (per user) selecting only the n most relevant items to the test set.

(c) The proposed split for one user when considering time. Items in the gray area correspond to interactions by other users in the same time span the candidate user has interacted with the black items.

Figure 1: The traditional training test split for the whole population of users Fig. 1a and the proposed split for one user Fig. 1b as well as for one user when taking temporal aspects into consideration Fig. 1c. The white area corresponds to the training set, the black area corresponds to the test set and the gray area corresponds to unused items.

are generated for each user $u \in \mathcal{U}$, where $\mathcal{U}$ is the complete set of users. Furthermore, to select candidate items for user u we use a time-agnostic relevance threshold (Line 16 of Algorithm 1) where only items having ratings higher than u's mean rating $\mu(u)$ plus a fraction of u's standard deviation ($\sigma(u)$) are deemed relevant. It should be noted that due to user's ratings being "...aspirational rather than reflecting [their] daily activity ..." [18] this relevance threshold might not be suitable in all cases. Additionally, we acknowledge that taking time into consideration when performing the split is more realistic, it would however make the comparison to a traditional evaluation approach impractical and is thus omitted (Section 3). Finally, we constrain the test set size to exactly N items.

Should u not have N ratings above the threshold, the weighting factor on $\sigma(u)$ is iteratively lowered until N items are found. The lowest value the threshold will take is $\mu(u)$. In Algorithm 1 the weighting factor is set to 0.5 and is exponentially decreased in each iteration. Should the lowered threshold result in more than N items, a random selection of the items with the lowest, good, relevance values is returned instead. In a "find good items" scenario, this threshold guarantees that the recommender system is tuned to the specific task at hand, i.e. finding good items. We propose to ensure that $M \geq 2N$ in order to allow enough items in the training set to expect a reasonable prediction quality.

The rationale for this evaluation setting is threefold;

i all available data for each user is used when training the algorithm, as described in Section 2.1

ii any recommender system could theoretically reach a maximum accuracy level when there are enough items considered relevant (good) for each user, as described in Section 2.2

Algorithm 1 Training and test set creation.

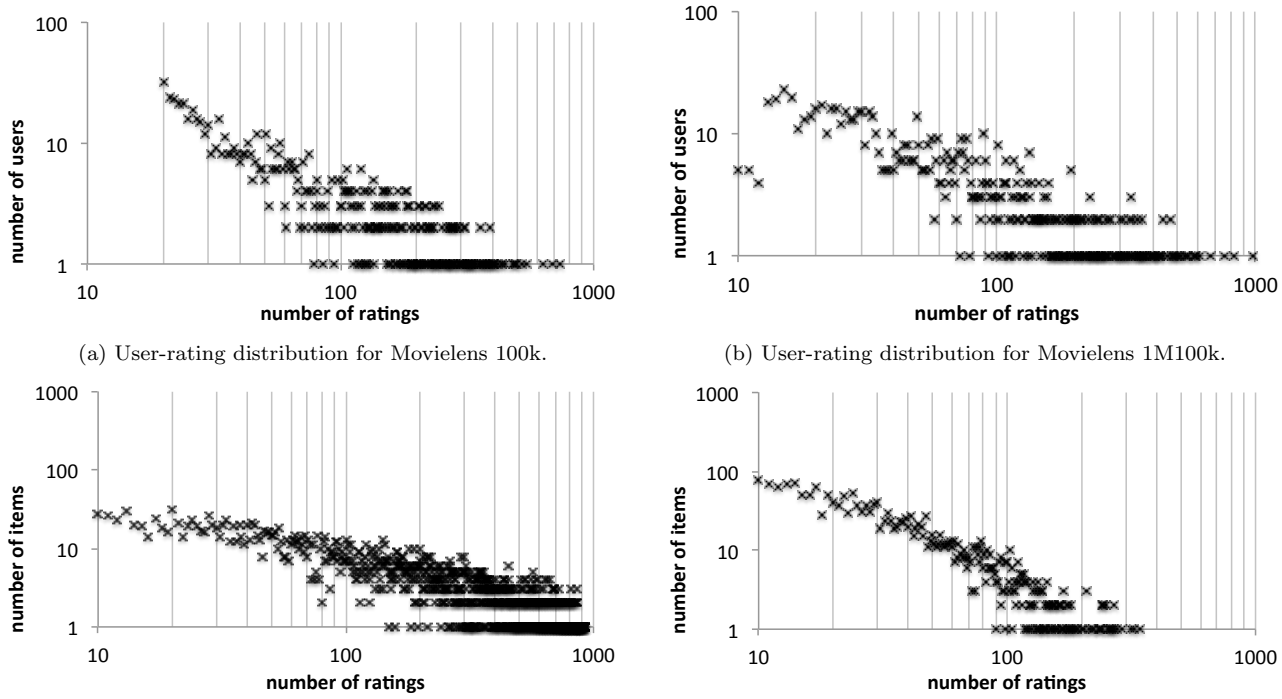
procedure DATASPLIT
Input: $\mathcal{R}, N, M$ where $\mathcal{R}$ is the set of all ratings r_{ui} given by users $u \in \mathcal{U}$ to items $i \in \mathcal{I}$ in the dataset, N the size of the list of recommendations, and M is the minimum number of ratings required for user u.
Output: The training sets $\{\mathcal{T}_u\}_{u \in \mathcal{U}}$ and validation sets $\{\mathcal{V}_u\}_{u \in \mathcal{U}}$, where each $\mathcal{T}_u$ and each $\mathcal{V}_u$ are the training and validation sets for each user u respectively
4: **for all** $u \in \mathcal{U}$ and $|\mathcal{R}_u| \geq M$ **do**
▷ Note: ($\mathcal{R}_u$ is the subset of user u's ratings)
$step = 1$
$\mathcal{V}_u = \{\}$
8: **while** $|\mathcal{V}_u| \neq N$ **do**
$\mathcal{W} \leftarrow \{i \notin \mathcal{V}_u : r_{ui} \geq \text{RELEVANCE}(u, step)\}$
$\mathcal{V}_u \leftarrow \mathcal{V}_u \cup random(\mathcal{W}, N - |\mathcal{V}_u|)$
$step++$
12: **end while**
$\mathcal{T}_u \leftarrow \mathcal{R} \setminus \mathcal{V}_u$
end for
end procedure

16: **function** RELEVANCE(u, q)
return $0.5^q \times \sigma(u) + \mu(u)$
end function

iii accuracy values will not be biased by users with many items in the test set and inaccurately affect the overall result for low N's; besides it ensures that enough data is fed to the algorithm in the training phase, as described in Section 2.3

3. EXPERIMENTS

To illustrate the quality of the proposed evaluation protocol, a set of experiments using an SVD recommender (EM factorizer, 50 factors, 50 iterations), a user-based collaborative filtering (CF) recommender (Pearson, k-NN, $k = 50$) and an item-based CF recommender (Pearson) were per-



(a) User-rating distribution for Movielens 100k.

(b) User-rating distribution for Movielens 1M100k.

(c) Item-rating distribution for Movielens 1M.

(d) Item-rating distribution for Movielens 1M100k.

Figure 2: The number of users vs. the number of ratings in the Movielens 100k dataset and the Movielens 1M100k dataset sample in Figs. 2a and 2b and the number of items vs. the number of ratings in the full Movielens 1 million dataset and the the Movielens 1M100k sample in Figs. 2c and 2d. The latter pair serve as a comparison between the distributions of the sample and the full dataset in order to affirm the validity of the sample. Note that the Ml1M100k sample (Fig. 2b) contains users with fewer than 20 ratings which will be the case in a realistic scenario.

Dataset	Users	Items	Ratings	Density
ML100k	943	1,682	100,000	6.3%
ML1M100k	1,000	3,294	100,000	3.0%

Table 1: The dimensions of the datasets used in the experiments. Note that the ML1M100k dataset is significantly sparser than the ML100k dataset, this is an effect of users in the latter having a minimum of 20 ratings each.

formed. Each algorithm was evaluated according to the proposed protocol (Section 2.4), and in a traditional evaluation setting where only one training test split was performed for the complete dataset.

The algorithms were evaluated using precision@N and R-precision@N values for $N \in \{1, 5, 10, 20, 50, 100\}$. The evaluation was performed using two datasets: a sample of 100 thousand ratings for 1,000 users from the Movielens 1 million dataset (ML1M100k), and the Movielens 100k dataset (ML100k). The reason for this sampling was to generate a more realistic dataset than the Movielens datasets as users in those have 20 ratings as a minimum (e.g. see Fig. 2 for a comparison of the two datasets) which could cause synthetically high precision values for low N's (i.e. $N < 20$). Both dataset's properties are given in Table 1.

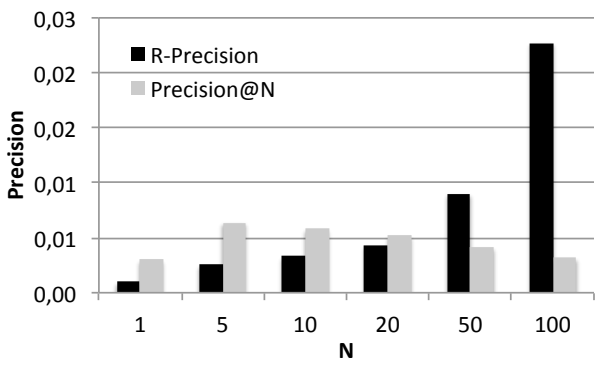
For R-precision values, the proposed test and training splits from Section 2.4 were used, whereas for the precision values (the traditional evaluation approach) only one test and one training test were created for all users and values of N. The latter test set consisted of up to 20% of the items each user had rated, these items needed to have been rated with at least a rating value $r = 3$ in order to be included in the test set. The remaining items were used for training.

4. RESULTS & DISCUSSION

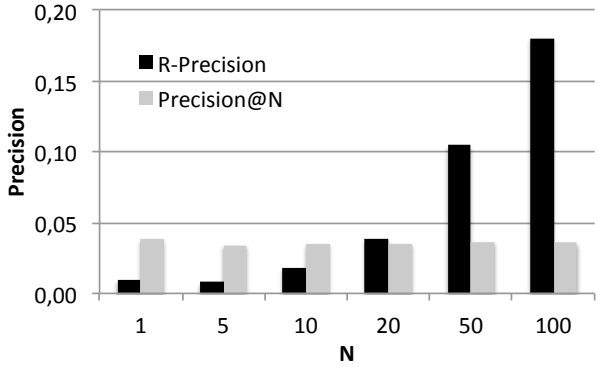
This section summarizes the results obtained from the experiments, both in terms of recommendation accuracy as well as in the time necessary to perform the evaluation.

4.1 Accuracy

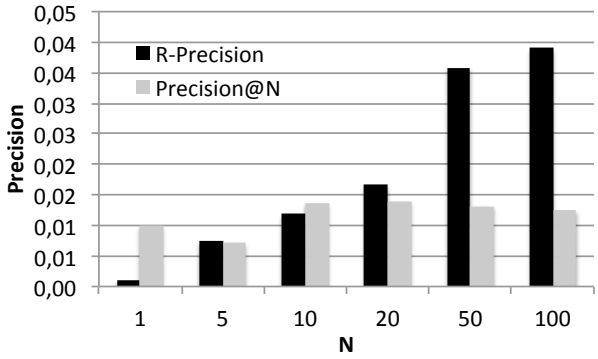
As mentioned in Section 2, the proposed evaluation protocol creates more accurate quality estimates than the traditional evaluation approach. Especially when considering a "first page size", i.e. the length of the recommended set, the protocol more accurately captures the quality of each "page" in respect to its size. Fig. 3 summarizes the results of the evaluation performed on both datasets with the three recommendation algorithms. For small values of N, a traditional evaluation approach receives higher precision values than the R-precision values received through the proposed protocol, as shown in Fig. 3. However, as N grows larger, the proposed method receives higher R-precision values. This is intuitive considering the nature of R-precision (larger number of true positive candidate items), it should nevertheless be noted that we are now able to estimate the *actual* quality of



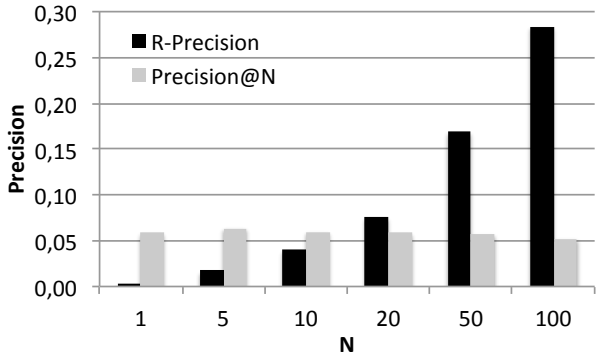
(a) Item-based CF (ML1M100k).



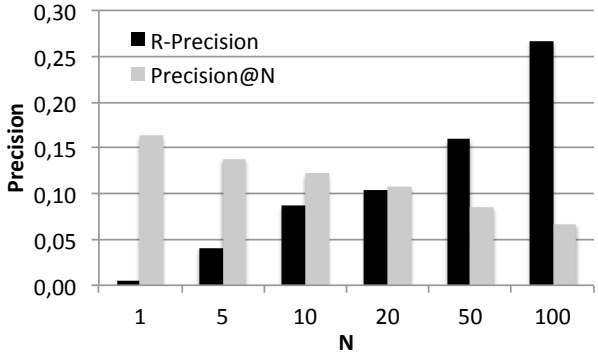
(b) User-based CF (ML1M100k).



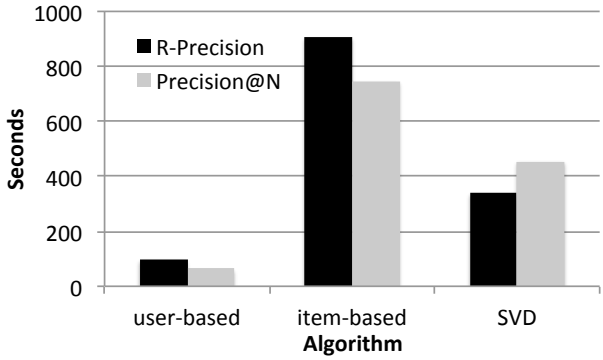
(c) SVD (ML1M100k).



(d) User-based CF (ML100k).



(e) SVD (ML100k).



(f) Time required to evaluate (ML1M100k).

Figure 3: Results for the three algorithms on the ML1M100k dataset (Figs. 3a to 3c), the accumulated time to run and evaluate each algorithm (Fig. 3f), and the results for the ML100k dataset (Figs. 3d and 3e).

the recommender algorithm, i.e. *how the algorithm performs for those users we are able to evaluate at relevant sizes of* N. Using the traditional approach however, results do not vary regardless the value of N, indicating that the evaluation method does not capture the differences in the recommendation quality depending on the "page size". More specifically, Figs. 3a to 3c show the comparison of results using the proposed protocol and a traditional evaluation approach on the sampled dataset. These figures allows us to estimate the quality of the recommendations for users with very few to very many ratings respectively. Additionally, the figures show the sensitivity of the proposed evaluation method to the number of "good" items, i.e. the more possible "good" recommendations, the more of these are recommended, and result in a higher quality estimate. In contrast, the standard evaluation setting fails to capture this and estimates the quality similarly, regardless of N, i.e. the precision levels in Figs. 3a to 3c are very similar for each recommender at different Ns.

Figs. 3d and 3e show the results on the ML100k dataset, where each user has at least 20 ratings. The traditional evaluation of the user-based recommender fails to capture this aspect of the underlying data (Fig. 3d), whereas R-precision has a noticeable improvement between $N = 5$ and $N = 10$, the point where the evaluation could potentially reach the maximum value, R-precision= 1.0. The consecutive improvements (for $N > 10$) are smaller, indicating that the evaluation method was sensitive to this aspect in the dataset. Fig. 3e summarizes the evaluation results of the SVD recommender on the ML100k dataset. Here, traditional evaluation and the proposed model appear to have an almost inverse relation, e.g. low precision values coincide with high R-precision values. This is likely the effect of proportionally fewer "good" items recommended for longer lists of recommended items, e.g. evaluating precision at high levels of N for users with $N >>$ "good" items.

The results of the item-based CF recommender have been omitted as they exhibited very low precision results on this dataset, a finding in accordance to other works, e.g. Bellogín et al. [1].

4.2 Evaluation Time

The time needed to perform an according to the proposed protocol presents another facet of recommender system evaluation. The protocol specifies that during the training phase the recommenders are to be trained individually for each user. In traditional evaluation, this is usually skipped in favor of only training the recommender once to minimize evaluation time. Fig. 3b shows the accumulated time (in seconds) required to sequentially run and evaluate all 6 recommendation approaches ($N = \{1, 5, 10, 20, 50, 100\}$) on a computer with a 3GHz Dual Core Intel i7 CPU and 8GB of memory (the time required for training the ML100k dataset has been omitted as the results exhibited similar characteristics). As shown, the required time to evaluate according to the proposed protocol is similar to the traditional evaluation approach. In fact, the time is actually lower for the SVD recommender. From this, we can infer that it is not the training of the algorithm that is time consuming, rather the evaluation itself. The rationale for this being that if the training process was costly, the proposed approach should have a far higher accumulated running time due to the many more recommenders trained compared to the traditional setting (1 vs. 1 per user at every N). Nevertheless, as the motivation of evaluation is to estimate the quality of a system, decreasing the accuracy of said estimate in order to improve an aspect which has no effect on the sought quality estimate seems contradictory to the initial motivation for evaluation. However, as the evaluation protocol dictates that only users with at least N "good" items should be evaluated at N, the resulting time spent on evaluation can actually be decreased as compared to traditional evaluation. Thus, the time saved by training the recommender model only once should not be seen as a motivation to create less accurate quality estimates.

To summarize our discussion, we know that precision at L has negative effects of the quality estimate when there are many users with less than L items. As such, precision at the level of recall, or R-precision, should provide a better estimate of the quality. As a side effect, R-precision can also show if a recommendation algorithm fails, or starts performing badly, at certain levels of recall. In information retrieval systems it can be said that a user has a higher usefulness of, e.g. a good item being retrieved at $L + 1$ than not having the item retrieved at all. In recommender systems, it commonly does not need to be the case that a recommendation at $L+1$ will actually be presented to the user. Thus making precision at the level of recall better suited to express the quality than precision at any level L.

5. CONCLUSIONS & FUTURE WORK

In this work, we have presented a new evaluation protocol for top-N recommender systems based on R-precision and compared the results obtained when evaluating three recommendation algorithms using two datasets to those obtained when using traditional top-N recommenders. The proposed model stipulates that the evaluation has an *attainable* optimal value, the evaluation better reflects the quality *per-user*, and that accuracy values are not biased by the number of items users have interacted with.

We have shown that the proposed model is more sensitive to the number of possible "good" recommendations, and is able to more accurately reflect the quality of the recommender algorithms, should they be deployed in a production environment. As a side effect of, the experiments also show that this evaluation approach, even though it involved more steps than traditional evaluation, can save evaluation time. As future work, we intend to extend the proposed evaluation approach to other metrics, e.g. RMSE, nDCG, etc. as well as investigate the correlation between the protocol and the experienced quality of a recommender system from the users' perspectives. Additionally we intend to further explore possible strategies for minimizing the accumulated evaluation time without compromising the validity of the evaluation itself.

6. ACKNOWLEDGMENTS

This work was carried out during the tenure of an ERCIM "Alain Bensoussan" Fellowship Programme. The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement n°246016.

7. REFERENCES

[1] A. Bellogin, P. Castells, and I. Cantador. Precision-oriented evaluation of recommender systems: an algorithmic comparison. In *Proceedings of the fifth ACM conference on Recommender systems*, RecSys '11, pages 333–336, New York, NY, USA, 2011. ACM.

[2] P. Campos, F. Díez, and I. Cantador. Time-aware recommender systems: a comprehensive survey and analysis of existing evaluation protocols. *User Modeling and User-Adapted Interaction*, pages 1–53, 2013.

[3] Ò. Celma. *Music Recommendation and Discovery in the Long Tail*. PhD thesis, Universitat Pompeu Fabra, Barcelona, 2008.

[4] P. Cremonesi, F. Garzotto, S. Negro, A. Papadopoulos, and R. Turrin. Comparative evaluation of recommender system quality. In *Proceedings of the 2011 annual conference extended abstracts on Human factors in computing systems*, CHI EA '11, pages 1927–1932, New York, NY, USA, 2011. ACM.

[5] P. Cremonesi, Y. Koren, and R. Turrin. Performance of recommender algorithms on top-n recommendation tasks. In *Proceedings of the fourth ACM conference on Recommender systems*, RecSys '10, pages 39–46, New York, NY, USA, 2010. ACM.

[6] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Trans. Inf. Syst.*, 22(1):5–53, Jan. 2004.

[7] W. Hill, L. Stead, M. Rosenstein, and G. Furnas. Recommending and evaluating choices in a virtual community of use. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '95, pages 194–201, New York, NY, USA, 1995. ACM Press/Addison-Wesley Publishing Co.

[8] G. Karypis. Evaluation of item-based top-n recommendation algorithms. In *Proceedings of the tenth international conference on Information and knowledge management*, CIKM '01, pages 247–254, New York, NY, USA, 2001. ACM.

[9] R. Kohavi. Online controlled experiments: introduction, learnings, and humbling statistics. In *Proceedings of the sixth ACM conference on Recommender systems*, RecSys '12, pages 1–2, New York, NY, USA, 2012. ACM.

[10] C. D. Manning, P. Raghavan, and H. Schütze. Evaluation in information retrieval. In *Introduction to Information Retrieval*, pages 151–175. Cambridge University Press, 2008.

[11] S. M. McNee, J. Riedl, and J. A. Konstan. 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, pages 1097–1101, New York, NY, USA, 2006. ACM.

[12] J. Picault, M. Ribière, D. Bonnefoy, and K. Mercer. How to get the recommender out of the lab? In F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, editors, *Recommender Systems Handbook*, pages 333–365. Springer US, 2011.

[13] P. Pu, L. Chen, and R. Hu. A user-centric evaluation framework for recommender systems. In *Proceedings of the fifth ACM conference on Recommender systems*, RecSys '11, pages 157–164, New York, NY, USA, 2011. ACM.

[14] A. Said, S. Berkovsky, and E. W. De Luca. Putting things in context: Challenge on context-aware movie recommendation. In *Proceedings of the Workshop on Context-Aware Movie Recommendation*, CAMRa '10, pages 2–6, New York, NY, USA, 2010. ACM.

[15] A. Said, B. Fields, B. J. Jain, and S. Albayrak. User-centric evaluation of a k-furthest neighbor collaborative filtering recommender algorithm. In *Proceedings of the 2013 conference on Computer supported cooperative work*, CSCW '13, pages 1399–1408, New York, NY, USA, 2013. ACM.

[16] G. Shani and A. Gunawardana. Evaluating recommendation systems. In F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, editors, *Recommender Systems Handbook*. Springer US, 2011.

[17] H. Steck. Item popularity and recommendation accuracy. In *Proceedings of the fifth ACM conference on Recommender systems*, RecSys '11, pages 125–132, New York, NY, USA, 2011. ACM.

[18] T. Vanderbilt. The science behind the netflix algorithms that decide what you'll watch next. `http://www.wired.com/underwire/2013/08/qq_netflix-algorithm/`, 08 2013.