rrecsys: an R-package for prototyping recommendation algorithms

Ludovik Çoba
Universiteti i Shkodrës "Luigj Gurakuqi"
Sheshi 2 Prilli
Shkodër, Albania
lcoba@unishk.edu.al

Markus Zanker
Free University of Bozen-Bolzano
piazza Domenicani, 3
39100 Bolzano, Italy
mzanker@unibz.it

ABSTRACT

We introduce rrecsys, an open source extension package in R for rapid prototyping and intuitive assessment of recommender system algorithms. As the only currently available R package for recommender algorithms (recommenderlab) did not include popular algorithm implementations such as matrix factorization or One-class Collaborative Filtering algorithms we developed rrecsys as an easily accessible tool that can, for instance, be employed for interactive demonstrations when teaching. This package replicates state-of-the-art Collaborative Filtering algorithms for rating and binary data and we compare results with the Java-based LensKit implementation and recommederlab for the purpose of benchmarking the implementation. Therefore this work can also be seen as a contribution in the context of replication of algorithm implementations and reproduction of evaluation results.

1. INTRODUCTION

R represents a popular choice in Data Analytics and Machine Learning. The software has low setup cost and contains a large selection of packages and functionalities to enhance and prototype algorithms with compact code and good visualization tools. Thus R represents a suitable environment for exploring the field of recommender systems. We present and contribute a novel R package, rrecsys[1], that replicates several state-of-the-art recommender algorithms for Likert scaled as well as binary rating values. Up to now there is only one package addressing recommender systems, recommenderlab [2], which lacks implementation of popular algorithms and we benchmark results in Section 3.
This work can be seen as a contribution towards the reproducibility of algorithms and results. Although this concept of reproducibility of experimental results is a fundamental prerequisite for scientific research it is many times not given for granted in the recommender systems field. For instance Said et al. [3] pointed out that the major recommendation frameworks such as *MyMediaLite*, *LensKit* and *Apache Mahout* show major differences in the implementation of the same algorithm variants and in their evaluation methodology. Differences which are according to Said et al. many times much larger than the typically reported performance improvements of a new algorithm over the selected baseline technique. Prototyping helps to shape recommender algorithms and evaluation methodologies as a strategy to tackle directly the issue of reproducibility. Furthermore, teaching recommendation concepts and evaluation methodology in hands-on sessions is highly relevant to understand ideas and algorithms from a didactics perspective and to make the learning experience more student-centered.

[1]https://cran.r-project.org/package=rrecsys
[2]https://cran.r-project.org/package=recommenderlab

RecSys 2016 Poster Proceedings, September 15-19, 2016, Boston, MA, USA.

Table 1: Benchmark in terms of RMSE between Lenskit, rrecsys and recommenderlab.

Algorithm	Lenskit	rrecsys	recommenderlab
globalMean	1.1278	1.1257	NA
itemAverage	1.0428	1.0246	NA
userAverage	1.0509	1.0416	NA
SVD(10 feat.)	0.9287	0.9277	3.7023
SVD (50 feat.)	0.9224	0.9207	3.7023
SVD (100 feat.)	0.9273	0.9191	3.7020
SVD (150 feat.)	0.9262	0.9188	3.7009
IB (20 neigh.)	0.9455	0.9851	1.1641
IB (50 neigh.)	0.9503	0.9477	1.1798
IB (100 neigh.)	0.9551	0.9416	1.2371

2. THE PACKAGE

rrecsys has a modular structure as well as includes expansion capabilities. The core of the package includes the implementation of several popular algorithms such as: *Most Popular*, *Global Average*, *Item Average*, *User Average*, *Item Based K-Nearest Neighbors*, *Simon Funk's SVD*, *Weighted Alternated Least Squares* and *Bayesian Personalized Ranking*. The package's evaluation module is based on *k-fold cross-validation* method. A stratified random selection procedure is applied when dividing the rated items of each user into k folds such that each user is uniformly represented in each fold. Based on the task (rating prediction or recommendation) the following metrics are computed: mean absolute error(MAE), root mean square error(RMSE), Precision, Recall, F1, True and False Positives, True and False Negatives, normalized discounted cumulative gain (NDCG), rank score, area under the ROC curve (AUC) and catalog coverage. RMSE and MAE metrics are computed according to their two variants, user-based vs. global.

3. RRECSYS IN ACTION

In this section we introduce an executable script in R for running some of the functionalities of rrecsys in order to demonstrate its intuitive use.

```r
# Install and load:
install.packages("rrecsys")
library(rrecsys)
# ML Latest is loaded on the package.
data("mlLatest100k")
# Define a rating matrix and explore it.
mlLatest <- defineData(mlLatest100k,
  minimum = .5, maximum = 5, halfStar = TRUE)
sparsity(mlLatest);  numRatings(mlLatest)
rowRatings(mlLatest);  colRatings(mlLatest)
smallMlLatest <- mlLatest[rowRatings(mlLatest)
>= 200, colRatings(mlLatest) > 10]
# Setting up the number of iterations for FunkSVD.
setStoppingCriteria(nrLoops = 50)
# Training a model using FunkSVD.
svd10 <- rrecsys(smallMlLatest, "FunkSVD", k = 10,
lambda = 0.001, gamma = 0.0015)
# Using the trained model to predict and recommend.
p <- predict(svd10)
r <- recommend(svd10, topN = 10)
# Instantiate an evaluation model.
model <- evalModel(smallMlLatest, folds = 5)
# Using the above model to evaluate predictions.
evalPred(model, "IBKNN", neigh = 10)
# Using the same model to evaluate recommendations.
evalRec(model, "globalAverage", topN = 10,
goodRating = 3)
```

4. BENCHMARK RESULTS

In Table 1 we report results from benchmarking the rrecsys implementation with the popular Lenskit [1] Java library and the recommenderlab R package. The reported results demonstrate the ability to clearly reproduce the results of Lenskit being the most well-known Java-based recommendation library and in contrast to recommenderlab. Evaluation is made using 5-fold cross validation on the MovieLens100K dataset. Lenskit and rrecsys were configured identically. In the case of recommenderlab we selected parameters such that its configuration was as close as possible to our and Lenskit's evaluation methodology. Reported error metrics were computed as a global average over the whole ratings in the test set. The SVD algorithm implementation in recommenderlab is based on an approximation estimated by the EM algorithm, resulting in bad prediction performance but enables the developer to vectorize, providing good computation performance. In the case of the item based k-nearest neighbor algorithm, rrecsys replicates recommenderlab implementation. Yet results of recommenderlab differ quite clearly proving that disparity in the implementation of the evaluation methodology significantly influences the reported results. We deployed a second set of experiments using MovieLens Latest[3][2] dataset cropped to a smaller chunk containing 620 users, 851 items, 58801 ratings, where each users has rated at least 20 items and each item was rated at least 25 times. In Figure 1 we report results of evaluation on this dataset with 5 folds. We ran these examples on a 2012's laptop computer with an Intel i5 at 2.60GHz and 8 GB RAM.We report execution times for a single prediction task and the full evaluation steps in Table 2 and 3.

[3]Authors express their gratitude to GroupLens for allowing redistribution of the MovieLens Latest data.

Table 2: Single prediction and evaluation times for the cropped MovieLens dataset.

Algorithm	Pred.	Eval. (5 folds)
IBKNN	1.51 ms	3539.4 s
Baseline Alg.	0.14 μs	0.4 s
BPR(20 features, 20 iteration)	0.95 μs	2.2 s
BPR(40 features, 20 iteration)	1.5 μs	3.5 s
wALS(20 features, 20 iteration)	1.1 ms	2583.3 s
wALS(40 features, 20 iteration)	1.7 ms	4098.6 s

Table 3: Single prediction and evaluation time on FunkSVD for the cropped MovieLens dataset.

# of features	40	80	100	120	140	180
Pred.(in μs)	2.6	8.7	11.3	14.1	16.6	24.7
Eval.(in s)	6.2	20.4	26.6	33.0	38.8	57.9

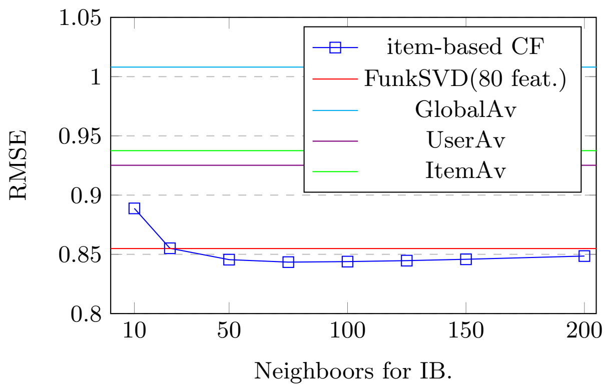

Figure 1: Evaluation on cropped MovieLens Latest.

5. CONCLUSIONS

This poster contributed a recently released package for prototyping and interactively demonstrating recommendation algorithms in R. It comes with a nice range of implemented standard CF algorithms. Reported results demonstrate that it reproduces results of the Java-based Lenskit toolkit. Thus it remains to hope that this effort will be of use for the field of recommender systems and the large R user community.

6. REFERENCES

[1] M. D. Ekstrand, M. Ludwig, J. A. Konstan, and J. T. Riedl. Rethinking the recommender research ecosystem: Reproducibility, openness, and lenskit. RecSys '11, pages 133–140, New York, NY, USA, 2011. ACM.

[2] F. M. Harper and J. A. Konstan. The movielens datasets: History and context. *ACM Trans. Interact. Intell. Syst.*, 5(4):19:1–19:19, Dec. 2015.

[3] A. Said and A. Bellogín. Comparative Recommender System Evaluation: Benchmarking Recommendation Frameworks. *RecSys*, pages 129–136, 2014.