Replication of Recommender Systems Research

Alejandro Bellogín
Universidad Autónoma de Madrid
@abellogin

Alan Said
University of Skövde
@alansaid
Who are we?

Alejandro Bellogín
- Lecturer (~Asst. Prof) @ Universidad Autónoma de Madrid, Spain
- PhD @ UAM, 2012
- Research on
  - Evaluation
  - Similarity metrics
  - Replication & reproducibility

Alan Said
- Lecturer (~Asst. Prof) @ University of Skövde, Sweden
- PhD @ TU Berlin, 2013
- Research on
  - Evaluation
  - Replication & reproducibility
  - Health
Outline

• Motivation
• Replication and reproducibility
• Replication in Recommender Systems
• Demo
• Conclusions and Wrap-up
• Questions
Motivation

In RecSys, we find inconsistent results, for the “same”

- Dataset
- Algorithm
- Evaluation metric
Motivation

In RecSys, we find inconsistent results, for the “same”
  - Dataset
  - Algorithm
  - Evaluation metric

So what?

<table>
<thead>
<tr>
<th>Metric</th>
<th>Algorithm</th>
<th>$k$-Item</th>
<th>$k$-User</th>
<th>PureSVD</th>
<th>Pop-item</th>
<th>IMM</th>
</tr>
</thead>
<tbody>
<tr>
<td>P@5</td>
<td>0.00135</td>
<td>0.006</td>
<td>0.067</td>
<td>0.227</td>
<td>0.267</td>
<td></td>
</tr>
<tr>
<td>NDCG@5</td>
<td>0.0036</td>
<td>0.0091</td>
<td>0.0566</td>
<td>0.216</td>
<td>0.245</td>
<td></td>
</tr>
<tr>
<td>MAP</td>
<td>0.013</td>
<td>0.041</td>
<td>0.081</td>
<td>0.119</td>
<td>0.156</td>
<td></td>
</tr>
</tbody>
</table>

Movielens 100k
[Cremonesi et al, 2010]

MovieLens 100k, SVD
[Jambor & Wang, 2010]
Motivation

A proper evaluation culture allows the field to advance

Improvements That Don’t Add Up: Ad-Hoc Retrieval Results Since 1998

Timothy G. Armstrong, Alistair Moffat, William Webber, Justin Zobel
Computer Science and Software Engineering
The University of Melbourne
Victoria 3010, Australia
{tgar,aalistair,wew,jz}@csse.unimelb.edu.au

... or at least, identify when there is a problem!
Motivation

In RecSys, we find inconsistent results, for the “same”
- Dataset
- Algorithm
- Evaluation metric

<table>
<thead>
<tr>
<th>Metric</th>
<th>Algorithm</th>
<th>k-Item</th>
<th>k-User</th>
<th>PureSVD</th>
<th>Pop-item</th>
<th>IMM</th>
</tr>
</thead>
<tbody>
<tr>
<td>P@5</td>
<td>0.00135</td>
<td>0.006</td>
<td>0.067</td>
<td>0.227</td>
<td>0.267</td>
<td></td>
</tr>
<tr>
<td>NDCG@5</td>
<td>0.0036</td>
<td>0.0091</td>
<td>0.0566</td>
<td>0.216</td>
<td>0.245</td>
<td></td>
</tr>
<tr>
<td>MAP</td>
<td>0.013</td>
<td>0.041</td>
<td>0.061</td>
<td>0.119</td>
<td>0.156</td>
<td></td>
</tr>
</tbody>
</table>

Movielens 100k
[Cremonesi et al, 2010]

Movielens 100k, SVD
[Jambor & Wang, 2010]
Motivation

In RecSys, we find inconsistent results, for the “same”

- Dataset
- Algorithm
- Evaluation metric

We need to understand why this happens
Goal of this tutorial

• Identify the steps that can act as hurdles when replicating experimental results
  – Focusing on the specific details inherent to the recommender systems

• We will analyze this problem using the following representation of a generic recommender system process
Train
Test
Validation
Splitting
Recommender
generates
ranking (for a user)
prediction for a given item (& user)
Evaluator
Collector
Stats

Ranking Prediction Coverage Diversity
In this tutorial

• We will focus on replication and reproducibility
  – Define the context
  – Present typical setting and problems
  – Propose some guidelines
  – Exhibit the most typical scenarios where experimental results in recommendation may hinder replication
NOT in this tutorial

• Definition of evaluation in recommendation:
  – In-depth analysis of evaluation metrics
  – Novel evaluation dimensions
  – User evaluation
  ➢ Wednesday’s lectures on evaluation
Outline

• Motivation
• Replication and reproducibility
• Replication in Recommender Systems
• Demo
• Conclusions and Wrap-up
• Questions
Reproducible Experimental Design

• We need to distinguish
  – Replicability
  – Reproducibility

• Different aspects:
  – Algorithmic
  – Published results
  – Experimental design

• Goal:
  – to have an environment for reproducible experiments
Definition: Replicability

To *copy* something
- The results
- The data
- The approach

Being able to evaluate in the same setting and obtain the same results
Definition: Reproducibility

To *recreate* something

- The (complete) set of experiments
- The (complete) set of results
- The (complete) experimental setup

To (re)launch it in production with the same results
Comparing against the state-of-the-art

Your settings are not exactly like those in paper X, but it is a relevant paper

Do results match the original paper?

Yes!
Congrats, you’re done!

No!

Do results agree with original paper?

They agree

They do not agree

Congrats! You have shown that paper X behaves different in the new setting

There is something wrong/incomplete in the experimental design. Try again!
Comparing against the state-of-the-art

Your settings are not exactly like those in paper X, but it is a relevant paper

Reproduce results of paper X

Do results match the original paper?

Yes!

Congrats, you’re done!

No!

Replicate results of paper X

They agree

Do results agree with original paper?

Yes!

They agree

No!

They do not agree

There is something wrong/incomplete in the experimental design. Try again!

Congrats! You have shown that paper X behaves different in the new setting
What about Reviewer 3?

• “It would be interesting to see this done on a different dataset…”
  – Repeatability
  – The same person doing the whole pipeline over again

• “How does your approach compare to [Reviewer 3 et al. 2003]?”
  – Reproducibility or replicability (depending on how similar the two papers are)
Repeat vs. replicate vs. reproduce vs. reuse

- **repeat**: same experiment, same lab
- **replicate**: same experiment, different lab
- **reproduce**: same experiment, different set up
- **reuse**: different experiment, some of same

Figure by Carole Goble adapted from Drummond C, Replicability is not Reproducibility: Nor is it Good Science, online and Peng RD, Reproducible Research in Computational Science Science 2 Dec 2011: 1226-1227.
Motivation for reproducibility

In order to ensure that our experiments, settings, and results are:

– Valid
– Generalizable
– Comparable
– Of use for others
– etc.

we must make sure that others can reproduce our experiments in their setting
Making reproducibility easier

• Description, description, description
• No magic numbers
• Specify values for all parameters
• Motivate!
• Keep a detailed **protocol** of everything
• Describe process **clearly**
• Use **standards**
• Publish code (nobody expects you to be an awesome developer, you’re a researcher)
• Publish data
• Publish supplemental material
Replicability, reproducibility, and progress

• Can there be *actual progress* if no valid comparison can be done?
• What is the point of comparing two approaches if the comparison is flawed?
• How do replicability and reproducibility facilitate actual progress in the field?
Summary

• Important issues when running experiments
  – Validity of results (replicability)
  – Comparability of results (reproducibility)
  – Validity of experimental setup (repeatability)

• We need to incorporate reproducibility and replication to facilitate progress in the field

• If your research is reproducible for others, it has more value
Outline

• Motivation
• Replication and reproducibility
• Replication in Recommender Systems
  – Dataset collection
  – Splitting
  – Recommender algorithms
  – Candidate items
  – Evaluation metrics
  – Statistical testing
• Demo
• Conclusions and Wrap-up
• Questions
Replication in Recommender Systems

• Replicability/reproducibility/repeatability: useful and desirable in any field
  – How can they be addressed when dealing with recommender systems?
• Proposal: analyze the recommendation process and identify each stage that may affect the final results
Train

Test

Validation

Splitting

Recommender generates ranking (for a user)

Evaluator

prediction for a given item (& user)

Stats

Ranking Prediction Coverage Diversity

Collector

ACM RecSys Summer School 2017
Train
Test
Validation
Splitting

Recommender generates ranking (for a user)
prediction for a given item (& user)

Ranking
Prediction
Coverage
Diversity

25-Aug-17
ACM RecSys Summer School 2017
DATA CREATION AND COLLECTION
What is a dataset?
Public datasets

- Movielens 20M
  - “Users were selected at random for inclusion. All selected users had rated at least 20 movies.”
- Netflix Prize
  - Details withheld
- Xing (RecSys Challenge 2016/2017)
  - Details withheld
- Last.fm (360k, MSD)
  - Undocumented cleaning applied
- MovieTweetings
  - All IMDb ratings… from Twitter
  - 2nd hand information
Creating your own datasets

• Ask yourself:
  – What are we collecting?
  – How are we collecting it?
    • How should we be collecting it?
  – Are we collecting all (vital) interactions?
    • dwell time vs. clicks vs. comments vs. swipes vs. likes vs. etc.
  – Are we documenting the process in sufficient detail?
  – Are we sharing the dataset in a format understood by others (and supported by software)?
The user-item matrix

<table>
<thead>
<tr>
<th>User</th>
<th>Item</th>
<th>Interaction</th>
<th>Timestamp</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2017-…</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td>…</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>2</td>
<td>…</td>
</tr>
</tbody>
</table>
Releasing the dataset

• Make the dataset publicly available
  – Otherwise your work is not reproducible

• Provide an in-depth overview
  – Website, paper, etc.

• Communicate it
  – Mailing lists, RecSysWiki, website, etc.
Releasing the dataset

• Consider releasing official training, test, validation splits.
• Present baseline algorithm results for released splits.
• Have code examples of how to work with the data (splits, evaluations, etc.)
Recommender generates ranking (for a user)
prediction for a given item (& user)

Evaluator

Stats

Splitting
Train
Validation
Test

Collector

Ranking Prediction Coverage Diversity
Train

Test

Validation

Splitting

Recommender generates ranking (for a user)

Prediction

Ranking

Coverage

Diversity

Evaluator

Collector

Stats
Train
Test
Validation

Splitting

Recommender generates ranking (for a user) prediction for a given item (user).

Evaluator
Collector
Stats

Ranking Prediction Coverage Diversity
DATA SPLITTING AND PREPARATION
Splitting
- Train
- Validation
- Test

25-Aug-17
ACM RecSys Summer School 2017
• Sizes?
• How to split?
• Filtering?
• How to document?
• Sizes?
• How to split?
• Filtering?
• How to document?

• What’s the task?
  – Rating prediction
  – Top-n

• What’s important for the algorithm?
  – Time
  – Relevance
• Which are the candidate items that we will be recommending?
• Who are the candidate users we will be recommending (and evaluating) for?
• Do we have any limitations on numbers?
  – Cold start?
  – Temporal/trending recommendations?
  – Other?
Scenarios

• Random
• All users at once
• All items at once
• One user at once
• One item at once
• Temporal
• Temporal for one user
• Relevance thresholds
Random

- The split does not take into consideration
  - Whether users or items are left out of the training or test sets.
  - The relevance of items
  - The scenario of the recommendation
All users at once

• The split does not take into consideration
  – Whether items are left out of the training or test sets.

• Can take into consideration
  – The relevance of items (per user or in general)
All items at once

• The split does not take into consideration
  – Whether users are left out of the training or test sets.
One user at once

• The split takes into consideration
  – The interactions of all other users when creating the splits for one specific user

• Resulting training set contains all other user-item interactions
One item at once

• The split takes into consideration
  – The interactions of all other users when creating the splits for one specific item

• Resulting training set contains all other user-item interactions
Temporal

• The split takes into consideration
  – The timestamp of interactions
• All items newer than a certain timestamp are discarded part of the test set.
Filters

• What filters?

• Why filters?

• Removing items/users with few interactions creates a skewed dataset
  – Sometimes this is a good thing
  – Needs proper motivation

Movielens 20M
  – “Users were selected at random for inclusion. All selected users had rated at least 20 movies.”
Implementation

• Most recommender system frameworks implement some form of splitting

however

• Documenting what choices were selected for the splitting is crucial for the work to be reproducible. Even when using established frameworks
Data splitting - LensKit

Data Processing in the Evaluator

Additional Cross-Folding Options

Crossfolding (the crossfold command) is implemented by CrossfoldTask. It supports several additional directives to control its behavior:

- **source**: the input data
- **partitions**: the number of train-test splits to create.
- **holdout N**: hold out $N$ items per user.
- **retain N**: retain $N$ items per user (holding out all other items).
- **holdoutFraction f**: hold out a fraction $f$ of each user's items.
- **method**: specify the crossfold method.
- **sampleSize N**: For sampling-based crossfold methods, the size of each sample.
- **order**: specify an ordering for user items prior to holdout. Can be either RandomOrder for random splitting or TimestampOrder for time-based splitting.
- **name**: a name for the data source, used for referring to the task & the default output names. The string parameter to the crossfold directive, if provided, sets the name.
- **train**: a format string taking a single integer specifying the name of the training data output files, e.g. ml-100k.train.%d.csv. The default is name + ".train.%d.csv". The format string is applied to the number of the partition.
- **test**: same as train, but for the test set.

http://lenskit.org/documentation/evaluator/data/
2. Splitter

LibRec has several ways to split the data. First, data can be split to the train set, test set (and validation set) following a certain ratio. Second, leaving one sample as the validation set. Third, leaving several (N) samples as the validation set. Fourth, K-fold cross-validation. Specifically, users can apply the mentioned methods to split the data on users or items.

2.1 ratio

Split the data according to a ratio.

2.2 loocv

Randomly pick up one user or item, or select the last user or item as the test data, and the rest as the train data.

```
data.model.splitter=loocv
```

2.3 given

Keep N users or items as the test data, and the rest as the train data.

```
data.model.splitter=given
```

2.4 kcv

K-fold cross-validation, splits the data into K folds. Every time, it selects one fold as the test set and the rest as the train set. Evaluation would be applied on each fold. After K times, the final evaluation result would be the average of all the folds.

```
data.model.splitter=kcv
```

2.5 testset

When using preserved data as the test set, users need to set the 'data.testset.path' configuration to specify the path of preserved test data. The path of preserved data should be under the directory of the train set, which means when reading all the data, preserved data can also be read.

```
data.model.splitter=testset
data.testset.path=nameoftestfile/dir
```
Partitions
Partitions
The Recommender generates ranking (for a user) from the Splitting process, which includes Train, Validation, and Test sets. The Evaluator takes the prediction for a given item (and user) and calculates Ranking Prediction, Coverage, Diversity, etc. The Collector provides the dataset for the splitting process.
Splitting

Train

Validation

Test

Ranking
Prediction
Coverage
Diversity

ACM RecSys Summer School 2017
Train

Test

Validation

Splitting

Recommender generates ranking (for a user) prediction for a given item & user

Ranking Prediction Coverage Diversity

Evaluator Collector

Stats

25-Aug-17 ACM RecSys Summer School 2017
RECOMMENDATION
The recommender

LensKit

librec

RankSys
Defining the recommender

• Many versions of the same thing
• Various implementations/design choices for
  – Collaborative Filtering
  – Similarity/distance measures
  – Factorization techniques (MF/FM)
  – Probabilistic modeling
Design

• There are multiple ways of implementing the same algorithms, similarities, metrics.

• Irregularities arise even when using a known implementation (from an existing framework)
  – Look at and understand the source code
  – Report implementational variations (esp. when comparing to others)
    • Magic numbers
    • Rounding errors
    • Thresholds
    • Optimizations
Collaborative Filtering

\[
\tilde{r}(u, i) = \tilde{r}(u) + C \sum_{v \in N_k(u)} \text{sim}(u, v) (r(v, i) - \tilde{r}(v))
\]

\[
\tilde{r}(u, i) = C \sum_{v \in N_k(u)} \text{sim}(u, v) r(v, i)
\]
Collaborative Filtering

- Both equations are usually referred to using the same name, i.e. k-nearest neighbor, user-based, cf.

\[
\tilde{r}(u, i) = \bar{r}(u) + C \sum_{v \in N_k(u)} \text{sim}(u, v) (r(v, i) - \bar{r}(v))
\]

\[
\tilde{r}(u, i) = C \sum_{v \in N_k(u)} \text{sim}(u, v)r(v, i)
\]
Similarities

• Similarity metrics may have different design choices as well
  – Normalized (by user) ratings/values
  – Shrinking parameter

\[
sim_s(a, b) = \frac{n_{a,b}}{n_{a,b} + \lambda_s} \cdot \text{sim}(a, b)
\]
CF Common Exceptions

• Two users having one rating/interaction each (same item)
• Both have liked it/rated similarly

• What is the similarity of these two?
CF Common Exceptions

• Two users having rated 500 items each
• 5 item intersect and have the same ratings/values

• What is the similarity of these two?
CF Implementation

• LensKit
  – No matter the recommender chosen, there is always a backup recommender to your chosen one. (BaselineScorer)
  – If your chosen recommender cannot fill your list of recommender items, the backup recommender will do so instead.
CF Implementation

• RankSys
  – Allows setting a similarity exponent, making the similarity stronger/weaker than normal
  – Similarity score defaults to 0.0
CF Implementation

• LibRec
  – Defaults to global mean when it cannot predict a rating for a user
Matrix Factorization

• Ranking vs. Rating prediction
  – Implementations vary between various frameworks.
  – Some frameworks contain several implementations of the same algorithms to tender to ranking specifically or rating prediction specifically.
MF Implementation

• RankSys
  – Bundles probabilistic modeling (PLSA) with matrix factorization (ALS)
  – Has three parallel ALS implementations
    • Generic ALS
MF Implementation

• LibRec
  – Separate rating prediction and ranking models
  – RankALSRecommender - Ranking
  – MFALSRecommender – Rating Prediction
    • Zhou et al. Large-Scale Parallel Collaborative Filtering for the Netflix Prize. AAIM 2008
Probabilistic Modeling

• Various ways of implementing the same algorithm
  – LDA using Gibbs sampling
  – LDA using variational Bayes
Probabilistic Implementation

• RankSys
  – Uses Mallet’s LDA implementation
Probabilistic Implementation

• LibRec
  – LDA for implicit feedback
Algorithms

When users use the configuration and command line to run programs, the recommendation algorithm is specified by `rec.recommender.class`. The configuration is shown as follows.

The approach for userKNN and itemKNN is different in the case of ranking and prediction. For ranking, we rank items according to their summation of item similarities. For prediction, we adopt the weighted average method.

In the Java implementation, after making instances of the Configuration object, the DataModel object, and the Similarity matrix object, these three instances are passed in as constructor parameters to generate the RecommenderContext object. Users can make the corresponding instance of the recommendation algorithm, that is to say, no need to set the `rec.recommender.class` configuration. The example code is shown as follows.

```java
RecommenderContext context = new RecommenderContext(conf, dataModel, similarity);

conf.set("rec.neighbors.knn.number","50");
conf.set("rec.recommender.isranking","false");

Recommender recommender = new UserKNNRecommender();
recommender.recommend(context);
```

https://www.librec.net/dokuwiki/doku.php?id=Recommender
https://github.com/guoguibing/librec/issues/76
Recommending LensKit

Configuration Points
As with all LensKit algorithms, the user-user CF implementation is highly configurable to allow you to experiment with a wide variety of variants and configurations. This section describes the primary configuration points for customizing the default components that drive the user-user CF implementation.

Unlike most other algorithms, the user-user filter does not really have a model that is built (though some things such as the global mean rating used by baselines are computed at model build time).

Here are some of the additional configuration points (’@’ indicates a parameter to be set with set rather than bind):

- **UserVectorizer** — normalizes user rating vectors prior to similarity computation and prediction.
- **NeighborhoodFinder** — finds neighborhoods for scoring items. The default implementation is [SimpleNeighborhoodFinder](https://lenskit.org/). Since LensKit 2.1, you can use [SnapshotNeighborhoodFinder](https://lenskit.org/) to embed an optimized snapshot of the ratings data into the neighborhood finder to improve performance on medium-sized data sets.
- **UserSimilarity** — compute similarities between users. The default implementation, [UserVectorSimilarity](https://lenskit.org/), just compares the users’ vectors using a vector similarity function; the default vector similarity is [CosineVectorSimilarity](https://lenskit.org/).
Outline

• Motivation
• Replication and reproducibility
• **Replication in Recommender Systems**
  – Dataset collection
  – Splitting
  – Recommender algorithm
  – **Candidate items**
  – Evaluation metrics
  – Statistical testing
• Demo
• Conclusions and Wrap-up
• Questions
Train Test Validation Splitting Recommender generates ranking (for a user)

Evaluator prediction for a given item (& user)

Stats

Ranking Prediction Coverage Diversity
Recommender generates ranking (for a user) prediction for a given item & user.

Ranking Prediction Coverage Diversity
Train, test, validation splitting.

Recommender generates ranking (for a user).

Evaluator collector, stats.

Ranking prediction, coverage, diversity.

Generates prediction for a given item (& user).

84
Candidate item generation

Different ways to select candidate items to be ranked:

- TestRatings: rated items by u in test set
- TestItems: every item in test set
- TrainingItems: every item in training
- AllItems: all items in the system

Note: in CF, AllItems and TrainingItems produce the same results
Candidate item generation

Different ways to select candidate items to be ranked:

Solid triangle represents the target user. Boxed ratings denote test set.

[Bellogín et al, 2011]
Candidate item generation

Impact of different strategies for candidate item selection:

**RPN: RelPlusN**

a ranking with

1 relevant and

N non-relevant items

**UT: UserTest**

same as TestRatings

---

<table>
<thead>
<tr>
<th>Alg.</th>
<th>F.W.</th>
<th>Time (sec.)</th>
<th>RMSE</th>
<th>nDCG@10</th>
<th>User cov.(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>RPN</td>
<td>UT</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>UT</td>
<td></td>
</tr>
<tr>
<td>IBCos</td>
<td>AM</td>
<td>238</td>
<td>1.041</td>
<td>0.003</td>
<td>98.16</td>
</tr>
<tr>
<td></td>
<td>LK</td>
<td>44</td>
<td>0.953</td>
<td>0.199</td>
<td>98.16</td>
</tr>
<tr>
<td></td>
<td>MML</td>
<td>75</td>
<td>NA</td>
<td>0.488</td>
<td>98.16</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.521</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>99.71</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>99.67</td>
<td></td>
</tr>
<tr>
<td>IBPea</td>
<td>AM</td>
<td>237</td>
<td>1.073</td>
<td>0.022</td>
<td>97.88</td>
</tr>
<tr>
<td></td>
<td>LK</td>
<td>31</td>
<td>1.093</td>
<td>0.033</td>
<td>97.86</td>
</tr>
<tr>
<td></td>
<td>MML</td>
<td>1,346</td>
<td>0.857</td>
<td>0.882</td>
<td>98.16</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.654</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2.87</td>
<td>99.83</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>99.31</td>
<td></td>
</tr>
<tr>
<td>SVD50</td>
<td>AM</td>
<td>132</td>
<td>0.950</td>
<td>0.236</td>
<td>98.12</td>
</tr>
<tr>
<td></td>
<td>LK</td>
<td>7</td>
<td>1.004</td>
<td>0.280</td>
<td>98.16</td>
</tr>
<tr>
<td></td>
<td>MML</td>
<td>1,324</td>
<td>0.848</td>
<td>0.882</td>
<td>98.18</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.648</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2.87</td>
<td>99.83</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>99.67</td>
<td></td>
</tr>
<tr>
<td>UBCos50</td>
<td>AM</td>
<td>5</td>
<td>1.178</td>
<td>0.378</td>
<td>35.66</td>
</tr>
<tr>
<td></td>
<td>LK</td>
<td>25</td>
<td>1.026</td>
<td>0.223</td>
<td>98.16</td>
</tr>
<tr>
<td></td>
<td>MML</td>
<td>38</td>
<td>NA</td>
<td>0.519</td>
<td>98.16</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.551</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>99.88</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>99.67</td>
<td></td>
</tr>
<tr>
<td>UBPea50</td>
<td>AM</td>
<td>6</td>
<td>1.126</td>
<td>0.375</td>
<td>48.50</td>
</tr>
<tr>
<td></td>
<td>LK</td>
<td>25</td>
<td>1.026</td>
<td>0.223</td>
<td>98.16</td>
</tr>
<tr>
<td></td>
<td>MML</td>
<td>1,261</td>
<td>0.847</td>
<td>0.883</td>
<td>98.18</td>
</tr>
</tbody>
</table>

[Said & Bellogín, 2014]
Candidate item generation

Impact of test size for different candidate item selection strategies:

The actual value of the metric may be affected by the amount of known information.

[Bellogín et al, 2017]
Outline

• Motivation
• Replication and reproducibility
• **Focus on Recommender Systems**
  – Dataset collection
  – Splitting
  – Recommender algorithm
  – Candidate items
  – **Evaluation metrics**
  – Statistical testing
• Demo
• Conclusions and Wrap-up
• Questions
Train | Test | Validation
--- | --- | ---

Recommender generates ranking (for a user)

Evaluator

Prediction for a given item (& user)

Collector

Stats

Ranking Prediction Coverage Diversity

ACM RecSys Summer School 2017
Train
Test
Validation
Splitting

Recommender generates ranking (for a user) prediction for a given item (& user)

Evaluator

Ranking Prediction Coverage Diversity

25-Aug-17 ACM RecSys Summer School 2017
Evaluation metric computation

When coverage is not complete, how are the metrics computed?

– If a user receives 0 recommendations

**Option a:**

\[
\text{metric} = \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} \text{metric}(u) \quad \text{considering} \quad R(u) = \emptyset \Rightarrow \text{metric}(u) = 0
\]

**Option b:**

\[
\text{metric} = \frac{1}{|\{u \in \mathcal{U} : \text{rec}(u) \neq \emptyset\}|} \sum_{u \in \mathcal{U} \cap \text{rec}(u) \neq \emptyset} \text{metric}(u)
\]
Evaluation metric computation

When coverage is not complete, how are the metrics computed?

– If a user receives 0 recommendations

<table>
<thead>
<tr>
<th>User</th>
<th>rec1</th>
<th>rec2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>#recs</td>
<td>metric(u)</td>
</tr>
<tr>
<td>u₁</td>
<td>5</td>
<td>0.8</td>
</tr>
<tr>
<td>u₂</td>
<td>3</td>
<td>0.2</td>
</tr>
<tr>
<td>u₃</td>
<td>0</td>
<td>--</td>
</tr>
<tr>
<td>u₄</td>
<td>1</td>
<td>1.0</td>
</tr>
<tr>
<td>Option a</td>
<td>0.50</td>
<td>0.55</td>
</tr>
<tr>
<td>Option b</td>
<td>0.66</td>
<td>0.55</td>
</tr>
<tr>
<td>User coverage</td>
<td>3/4</td>
<td>4/4</td>
</tr>
</tbody>
</table>
Evaluation metric computation

When coverage is not complete, how are the metrics computed?

– If a user receives 0 recommendations
– If a value is not predicted (esp. for error-based metrics)

\[
MAE = \frac{1}{|Te|} \sum_{(u,i) \in Te} |\tilde{r}(u,i) - r(u,i)|
\]

\[
RMSE = \sqrt{\frac{1}{|Te|} \sum_{(u,i) \in Te} (\tilde{r}(u,i) - r(u,i))^2}
\]

MAE = Mean Absolute Error
RMSE = Root Mean Squared
When coverage is not complete, how are the metrics computed?

- If a user receives 0 recommendations
- If a value is not predicted (esp. for error-based metrics)

<table>
<thead>
<tr>
<th>User-item pairs</th>
<th>Real</th>
<th>Rec1</th>
<th>Rec2</th>
<th>Rec3</th>
</tr>
</thead>
<tbody>
<tr>
<td>(u₁, i₁)</td>
<td>5</td>
<td>4</td>
<td>NaN</td>
<td>4</td>
</tr>
<tr>
<td>(u₁, i₂)</td>
<td>3</td>
<td>2</td>
<td>4</td>
<td>NaN</td>
</tr>
<tr>
<td>(u₁, i₃)</td>
<td>1</td>
<td>1</td>
<td>NaN</td>
<td>1</td>
</tr>
<tr>
<td>(u₂, i₁)</td>
<td>3</td>
<td>2</td>
<td>4</td>
<td>NaN</td>
</tr>
</tbody>
</table>

MAE/RMSE, ignoring NaNs

<table>
<thead>
<tr>
<th></th>
<th>MAE/RMSE</th>
<th>MAE/RMSE</th>
<th>MAE/RMSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ignoring NaNs</td>
<td>0.75/0.87</td>
<td>2.00/2.00</td>
<td>0.50/0.70</td>
</tr>
<tr>
<td>NaNs as 0</td>
<td>0.75/0.87</td>
<td>2.00/2.65</td>
<td>1.75/2.18</td>
</tr>
<tr>
<td>NaNs as 3</td>
<td>0.75/0.87</td>
<td>1.50/1.58</td>
<td>0.25/0.50</td>
</tr>
</tbody>
</table>

MAE = Mean Absolute Error
RMSE = Root Mean Squared Error
Evaluation metric computation

Variations on metrics:

Error-based metrics can be normalized or averaged per user:

- Normalize RMSE or MAE by the range of the ratings
  (divide by $r_{\text{max}} - r_{\text{min}}$)
- Average RMSE or MAE to compensate for unbalanced distributions of items or users

\[
\text{uMAE} = \frac{1}{|U|} \sum_{u \in U} \frac{1}{|T_{eu}|} \sum_{i \in T_{eu}} |\tilde{r}(u, i) - r(u, i)|
\]
Evaluation metric computation

Variations on metrics:

**nDCG** has at least two discounting functions (linear and exponential decay)

\[
\text{nDCG} = \frac{1}{|U|} \sum_{u} \frac{1}{\text{IDCG}_u} \sum_{p=1}^{p_u} f_{\text{dis}}(\text{rel}(u, i_p), p)
\]

\[
f_{\text{dis}}(x, y) = (2^x - 1)/\log(1 + y)
\]

\[
f_{\text{dis}}(x, y) = x/\log y \text{ if } y > 1
\]
Evaluation metric computation

Variations on metrics:

**Ranking-based metrics** are usually computed up to a ranking position or cutoff $k$

$$P@k = \frac{1}{|U|} \sum_{u \in U} \frac{|\text{Rel}_u@k|}{k}$$

$$R@k = \frac{1}{|U|} \sum_{u \in U} \frac{|\text{Rel}_u@k|}{|\text{Rel}_u|}$$

$$\text{MAP} = \frac{1}{|U|} \sum_{u \in U} \frac{1}{|\text{Rel}_u|} \sum_{i \in \text{Rel}_u} P@\text{rank}(u, i)$$

Is the cutoff being reported? Are the metrics computed until the end of the list? Is that number the same across all the users?
Evaluation metric computation

If ties are present in the ranking scores, results may depend on the implementation.

Table VI. Average Ratio of Tied Items per User, at Different Cutoffs for the Evaluated Recommenders

<table>
<thead>
<tr>
<th>Recommender type</th>
<th>Tied items at 5</th>
<th>Tied items at 10</th>
<th>Tied items at 50</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Min</td>
<td>Avg</td>
<td>Max</td>
</tr>
<tr>
<td>UB</td>
<td>15.11</td>
<td>130.64</td>
<td>280.83</td>
</tr>
<tr>
<td>SimPop</td>
<td>4.50</td>
<td>235.31</td>
<td>736.50</td>
</tr>
<tr>
<td>SVD</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>PureSocial</td>
<td>2</td>
<td>50.75</td>
<td>172</td>
</tr>
<tr>
<td>FriendsPop</td>
<td>10</td>
<td>350.20</td>
<td>1057</td>
</tr>
<tr>
<td>Personal</td>
<td>0</td>
<td>1.80</td>
<td>65</td>
</tr>
<tr>
<td>Combined</td>
<td>2.80</td>
<td>62.20</td>
<td>205.10</td>
</tr>
</tbody>
</table>

[Bellogín et al, 2013]
Evaluation metric computation

Internal evaluation methods of different frameworks (Mahout (AM), LensKit (LK), MyMediaLite (MML)) present different implementations of these aspects.

(a) nDCG for AM and LK

<table>
<thead>
<tr>
<th>Alg.</th>
<th>F.W.</th>
<th>nDCG</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBCos</td>
<td>AM</td>
<td>0.000414780</td>
</tr>
<tr>
<td></td>
<td>LK</td>
<td>0.942192050</td>
</tr>
<tr>
<td>IBPea</td>
<td>AM</td>
<td>0.005169231</td>
</tr>
<tr>
<td></td>
<td>LK</td>
<td>0.924546132</td>
</tr>
<tr>
<td>SVD50</td>
<td>AM</td>
<td>0.105427298</td>
</tr>
<tr>
<td></td>
<td>LK</td>
<td>0.943464094</td>
</tr>
<tr>
<td>UBCos50</td>
<td>AM</td>
<td>0.169295451</td>
</tr>
<tr>
<td></td>
<td>LK</td>
<td>0.948413562</td>
</tr>
<tr>
<td>UBPea50</td>
<td>AM</td>
<td>0.169295451</td>
</tr>
<tr>
<td></td>
<td>LK</td>
<td>0.948413562</td>
</tr>
</tbody>
</table>

(b) RMSE values for LK and MML.

<table>
<thead>
<tr>
<th>Alg.</th>
<th>F.W.</th>
<th>RMSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBCos</td>
<td>LK</td>
<td>1.01390931</td>
</tr>
<tr>
<td></td>
<td>MML</td>
<td>0.92476162</td>
</tr>
<tr>
<td>IBPea</td>
<td>LK</td>
<td>1.05018614</td>
</tr>
<tr>
<td></td>
<td>MML</td>
<td>0.92933246</td>
</tr>
<tr>
<td>SVD50</td>
<td>LK</td>
<td>1.01209290</td>
</tr>
<tr>
<td></td>
<td>MML</td>
<td>0.93074012</td>
</tr>
<tr>
<td>UBCos50</td>
<td>LK</td>
<td>1.02545490</td>
</tr>
<tr>
<td></td>
<td>MML</td>
<td>0.95358984</td>
</tr>
<tr>
<td>UBPea50</td>
<td>LK</td>
<td>1.02545490</td>
</tr>
<tr>
<td></td>
<td>MML</td>
<td>0.93419026</td>
</tr>
</tbody>
</table>

[Said & Bellogín, 2014]
Evaluation metric computation

Decisions (implementations) found in some recommendation frameworks:

Regarding coverage:

- LK and MML use backup recommenders
- LR: not actual backup recommender, but default values (global mean) are provided when not enough neighbors (or all similarities are negative) are found in KNN
- RS allows to average metrics explicitly by option a or b (see different constructors of AverageRecommendationMetric)

LK: https://github.com/lenskit/lenskit
LR: https://github.com/guoguibing/librec
MML: https://github.com/zenogantner/MyMediaLite
RS: https://github.com/RankSys/RankSys
Evaluation metric computation

Decisions (implementations) found in some recommendation frameworks:

Regarding metric variations:

• LK, LR, MML use a logarithmic discount for nDCG
• RS also uses a logarithmic discount for nDCG, but relevance is normalized with respect to a threshold
• LK does not take into account predictions without scores for error metrics
• LR fails if coverage is not complete for error metrics
• RS does not compute error metrics

LK: https://github.com/lenskit/lenskit
LR: https://github.com/guoguibing/librec
MML: https://github.com/zenogantner/MyMediaLite
RS: https://github.com/RankSys/RankSys
Evaluation metric computation

Decisions (implementations) found in some recommendation frameworks:

Regarding candidate item generation:

• LK allows defining a candidate and exclude set
• LR: delegated to the Recommender class. AbstractRecommender defaults to TrainingItems
• MML allows different strategies: training, test, their overlap and union, or a explicitly provided candidate set
• RS defines different ways to call the recommender: without restriction, with a list size limit, with a filter, or with a candidate set
Evaluation metric computation

Decisions (implementations) found in some recommendation frameworks:

Regarding ties:

- MML: not deterministic (Recommender.Recommend sorts items by descending score)
- LK: depends on using `predict` package (not deterministic: LongUtils.keyValueComparator only compares scores) or `recommend` (same ranking as returned by algorithm)
- LR: not deterministic (Lists.sortItemEntryListTopK only compares the scores)
- RS: deterministic (IntDoubleTopN compares values and then keys)
Outline

• Motivation
• Replication and reproducibility
• **Focus on Recommender Systems**
  – Dataset collection
  – Splitting
  – Recommender algorithm
  – Candidate items
  – Evaluation metrics
  – **Statistical testing**
• Demo
• Conclusions and Wrap-up
• Questions
Train Test Validation Splitting Recommender generates ranking (for a user) prediction for a given item (& user)

Evaluator

Ranking Prediction Coverage Diversity

Stats
Train Test Validation Splitting

Recommender generates ranking (for a user) prediction for a given item (user)

Evaluator

Ranking Prediction Coverage Diversity

25-Aug-17 ACM RecSys Summer School 2017
Train
Test
Validation
Splitting
Recommender generates ranking (for a user) prediction for a given item (user)

Ranking Prediction Coverage Diversity

Stats

25-Aug-17
ACM RecSys Summer School 2017
Statistical testing

Make sure the statistical testing method is reported

– Paired/unpaired test, effect size, confidence interval
– Specify why this specific method is used
– Related statistics (such as mean, variance, population size) are useful to interpret the results

[Sakai, 2014]
Statistical testing

When doing cross-validation, there are several options to take the samples for the test:

- One point for each aggregated value of the metric
  - Very few points (one per fold)

- One point for each value of the metric, on a user basis
  - If we compute a test for each fold, we may find inconsistencies
  - If we compute a test with all the (concatenated) values, we may distort the test: many more points, not completely independent

[Bouckaert, 2003]
Replication and Reproducibility in RecSys: Summary

• Reproducible experimental results depend on acknowledging every step in the recommendation process
  – As black boxes so every setting is reported
  – Applies to data collection, data splitting, recommendation, candidate item generation, metric computation, and statistics

• There exist several details (in implementation) that might hide important effects in final results
Replication and Reproducibility in RecSys:
Key takeaways

• Every decision has an impact
  – We should log every step taken in the experimental part and report that log

• There are more things besides papers
  – Source code, web appendix, etc. are very useful to provide additional details not present in the paper

• You should not fool yourself
  – You have to be critical about what you measure and not trust intermediate “black boxes”
Replication and Reproducibility in RecSys: Next steps?

- We should agree on standard implementations, parameters, instantiations, ...
  - Example: trec_eval in IR
Replication and Reproducibility in RecSys: Next steps?

- We should agree on standard implementations, parameters, instantiations, …
- Replicable badges for journals / conferences
Replication and Reproducibility in RecSys: Next steps?

• We should agree on standard implementations, parameters, instantiations, …

• Replicable badges for journals / conferences

The aim of the Reproducibility Initiative is to identify and reward high quality reproducible research via independent validation of key experimental results

http://validation.scienceexchange.com
Replication and Reproducibility in RecSys: Next steps?

• We should agree on standard implementations, parameters, instantiations, …

• Replicable badges for journals / conferences

• Investigate how to improve reproducibility
Replication and Reproducibility in RecSys: Next steps?

• We should agree on standard implementations, parameters, instantiations, …
• Replicable badges for journals / conferences
• Investigate how to improve reproducibility
• Benchmark, report, and store results
Pointers

• Email and Twitter
  – Alejandro Bellogín
    • alejandro.bellogin@uam.es
    • @abellogin
  – Alan Said
    • alansaid@acm.org
    • @alansaid

• Slides:
  • https://github.com/recommenders/rsss2017
RiVal

Recommender System Evaluation Toolkit

http://rival.recommenders.net
http://github.com/recommenders/rival
Thank you!
References and Additional reading

- [Bellogín et al, 2013] An Empirical Comparison of Social, Collaborative Filtering, and Hybrid Recommenders. ACM TIST
- [Ben-Shimon et al, 2015] RecSys Challenge 2015 and the YOOCHOOSE Dataset. RecSys
- [Bouckaert, 2003] Choosing Between Two Learning Algorithms Based on Calibrated Tests. ICML
- [Filippone & Sanguinetti, 2010] Information Theoretic Novelty Detection. Pattern Recognition
- [Fleder & Hosanagar, 2009] Blockbuster Culture’s Next Rise or Fall: The Impact of Recommender Systems on Sales Diversity. Management Science
References and Additional reading

• [Jambor & Wang, 2010] Goal-Driven Collaborative Filtering. ECIR
• [Koren, 2008] Factorization Meets the Neighborhood: a Multifaceted Collaborative Filtering Model. KDD
• [Li et al, 2010] Improving One-Class Collaborative Filtering by Incorporating Rich User Information. CIKM
References and Additional reading

- [Schein et al, 2002] Methods and Metrics for Cold-Start Recommendations. SIGIR
- [Weng et al, 2007] Improving Recommendation Novelty Based on Topic Taxonomy. WI-IAT
- [Yin et al, 2012] Challenging the Long Tail Recommendation. VLDB
- [Ziegler et al, 2005] Improving Recommendation Lists Through Topic Diversification. WWW