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

	Algorithm				
Metric	k-Item	k-User	PureSVD	*Pop-item*	IMM
P@5	0.00135	0.006	0.067	0.227	0.267
NDCG@5	0.0036	0.0091	0.0566	0.216	0.245
MAP	0.013	0.041	0.061	0.119	0.156

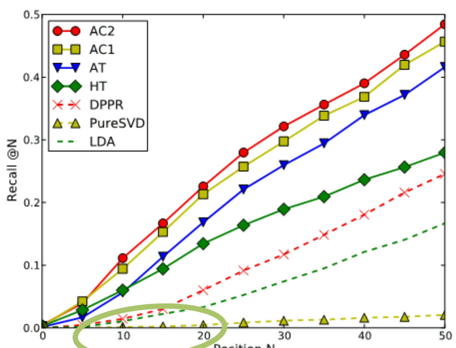
Movielens 100k

[Gorla et al, 2013]

	Baseline(Test)
MAP	0.447
MRR	0.889
NDCG@1(	0.720
NDCG@5	0.570
NDCG@3	0.447

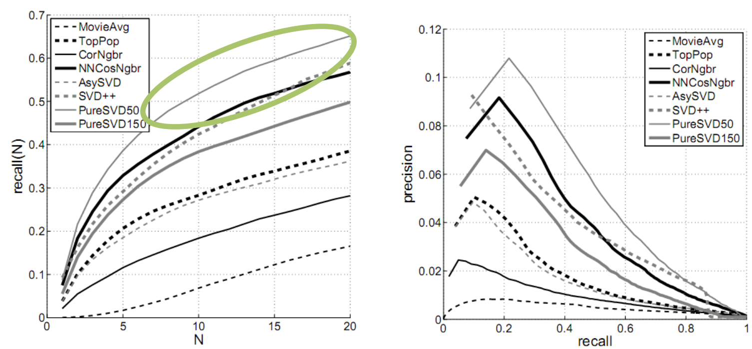
Movielens 100k, SVD

[Jambor & Wang, 2010]



Movielens 1M

[Yin et al, 2012]



(a) recall (b) precision vs recall

Movielens 1M

[Cremonesi et al, 2010]

Motivation

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

	Algorithm				
Metric	k-Item	k-User	PureSVD	*Pop-item*	IMM
P@5	0.00135	0.006	0.067	0.227	0.267
NDCG@5	0.0036	0.0091	0.0566	0.216	0.245
MAP	0.013	0.041	0.061	0.119	0.156

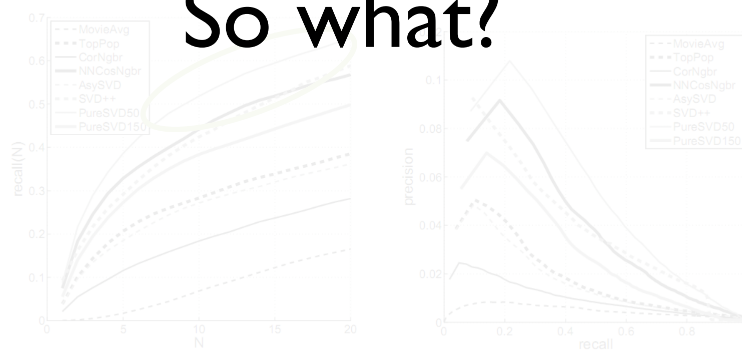
Movielens 100k
[Gorla et al, 2013]

	Baseline(Test)
MAP	0.447
MRR	0.889
NDCG@10	0.720
NDCG@5	0.570
NDCG@3	0.447

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

So what?

Movielens 1M
[Yin et al, 2012]



(a) recall (b) precision vs recall

Movielens 1M
[Cremonesi et al, 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,alistair,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

	Algorithm				
Metric	k-Item	k-User	PureSVD	*Pop-item*	IMM
P@5	0.00135	0.006	0.067	0.227	0.267
NDCG@5	0.0036	0.0091	0.0566	0.216	0.245
MAP	0.013	0.041	0.061	0.119	0.156

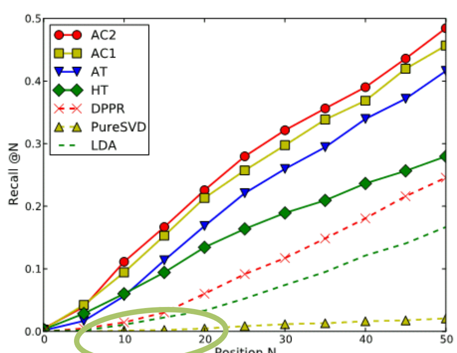
Movielens 100k

[Gorla et al, 2013]

	Baseline(Test)
MAP	0.447
MRR	0.889
NDCG@1(	0.720
NDCG@5	0.570
NDCG@3	0.447

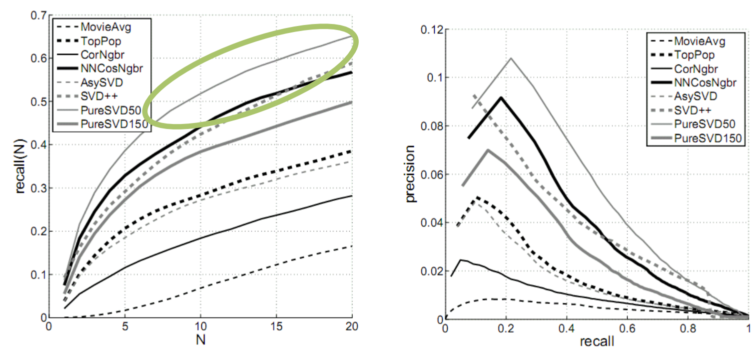
Movielens 100k, SVD

[Jambor & Wang, 2010]



Movielens 1M

[Yin et al, 2012]



(a) recall

(b) precision vs recall

Movielens 1M

[Cremonesi et al, 2010]

Motivation

In RecSys, we find inconsistent results, for the "same"

- Dataset
- Algorithm
- Evaluation metric

	Algorithm				
Metric	*k*-Item	*k*-User	PureSVD	*Pop-item*	IMM
P@5	0.00135	0.006	0.067	0.227	0.267
NDCG@5	0.0036	0.0091	0.0566	0.216	0.245
MAP	0.013	0.041	0.061	0.119	0.156

Movielens 100k

[Gorla et al, 2013]

	Baseline(Test)
MAP	0.447
MRR	0.889
NDCG@1(	0.720
NDCG@5	0.570
[illegible]	[illegible]

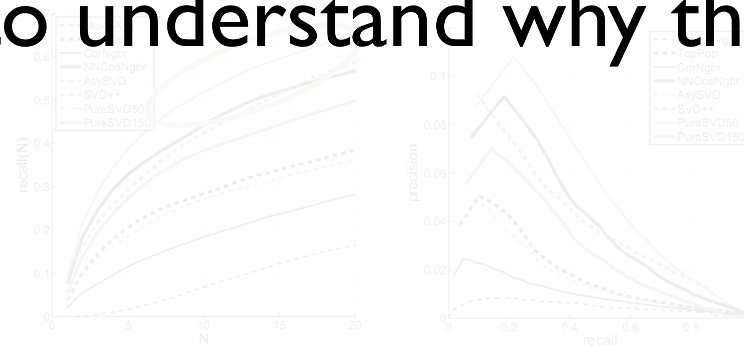
Movielens 100k, SVD

[Jambor & Wang, 2010]

We need to understand why this happens

Movielens 1M

[Yin et al, 2012]



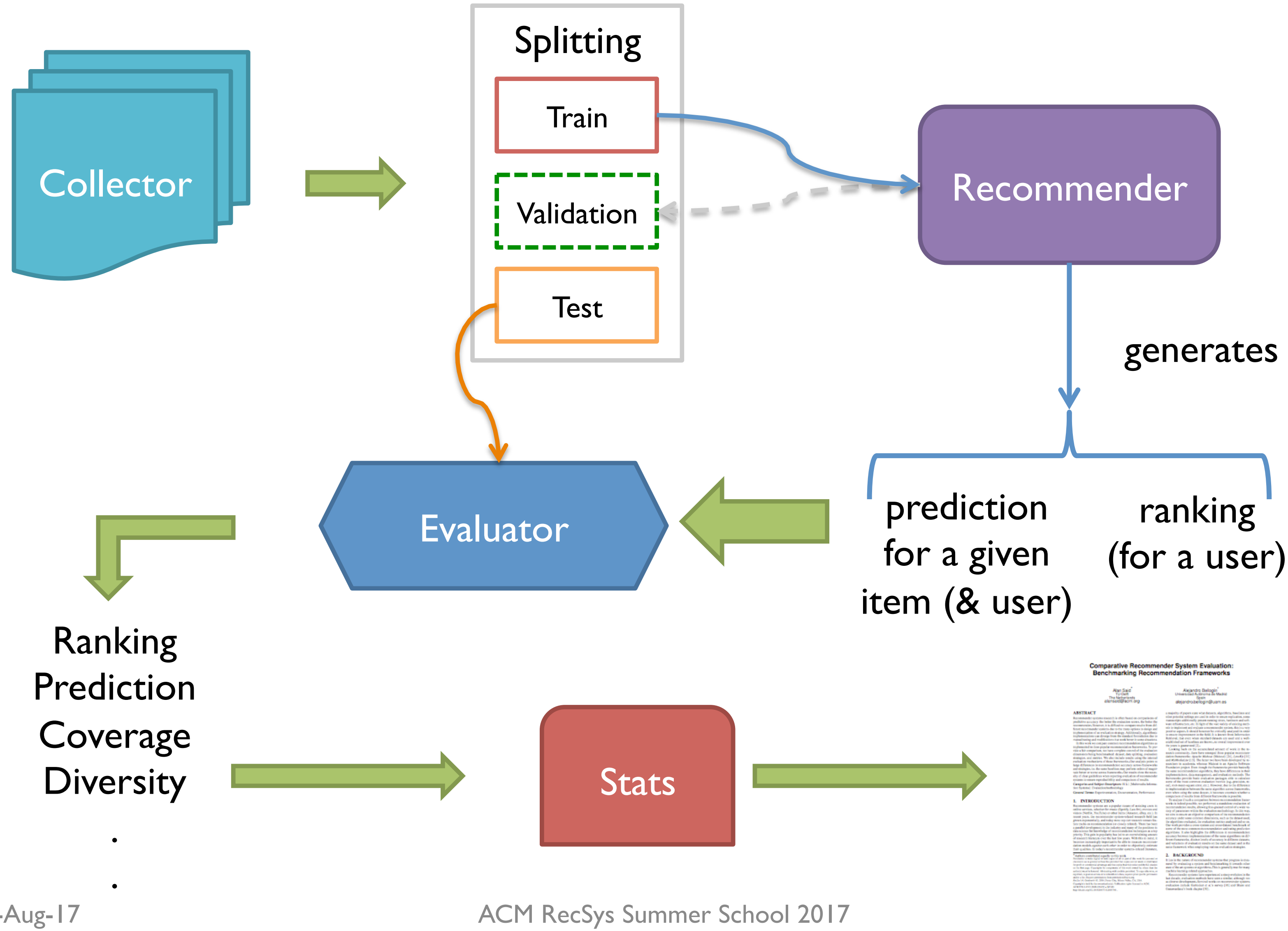
(a) recall (b) precision vs recall

Movielens 1M

[Cremonesi et al, 2010]

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



Collector
Splitting
Train
Validation
Test
Recommender
generates
prediction for a given item (& user)
ranking (for a user)
Evaluator
Ranking
Prediction
Coverage
Diversity
.
.
.
Stats
Comparative Recommender System Evaluation: Benchmarking Recommendation Frameworks

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 <u>reproducible experiments</u>

Definition: Replicability

To *copy* something

- The results
- The data
- The approach

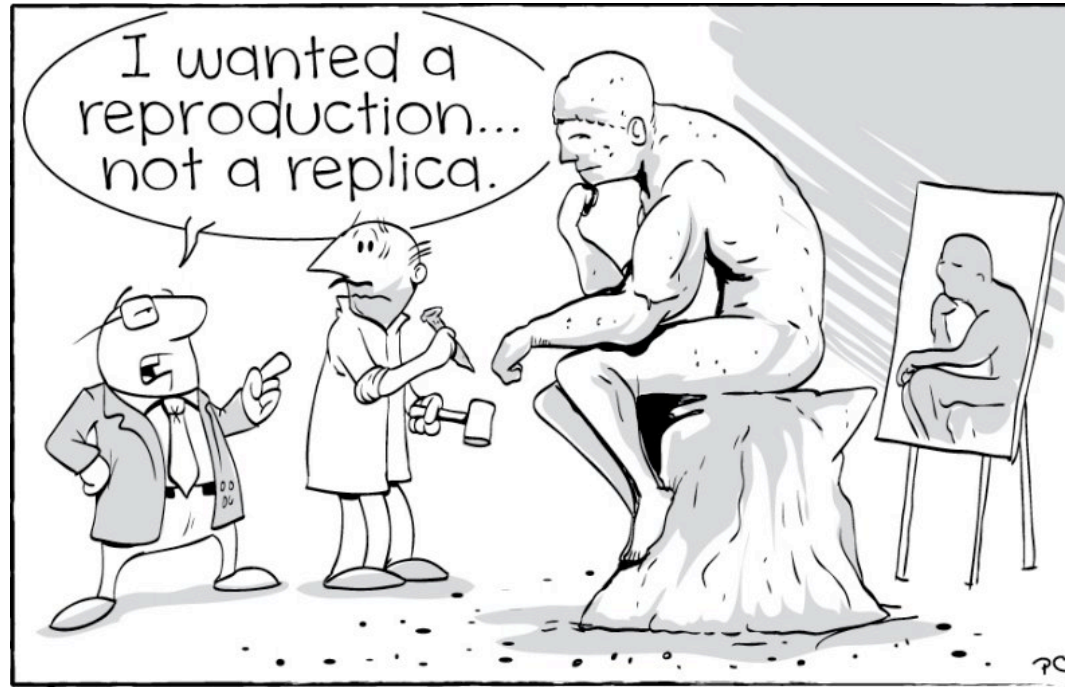
Being able to evaluate in the same setting and obtain the same results

Definition: Reproducibility

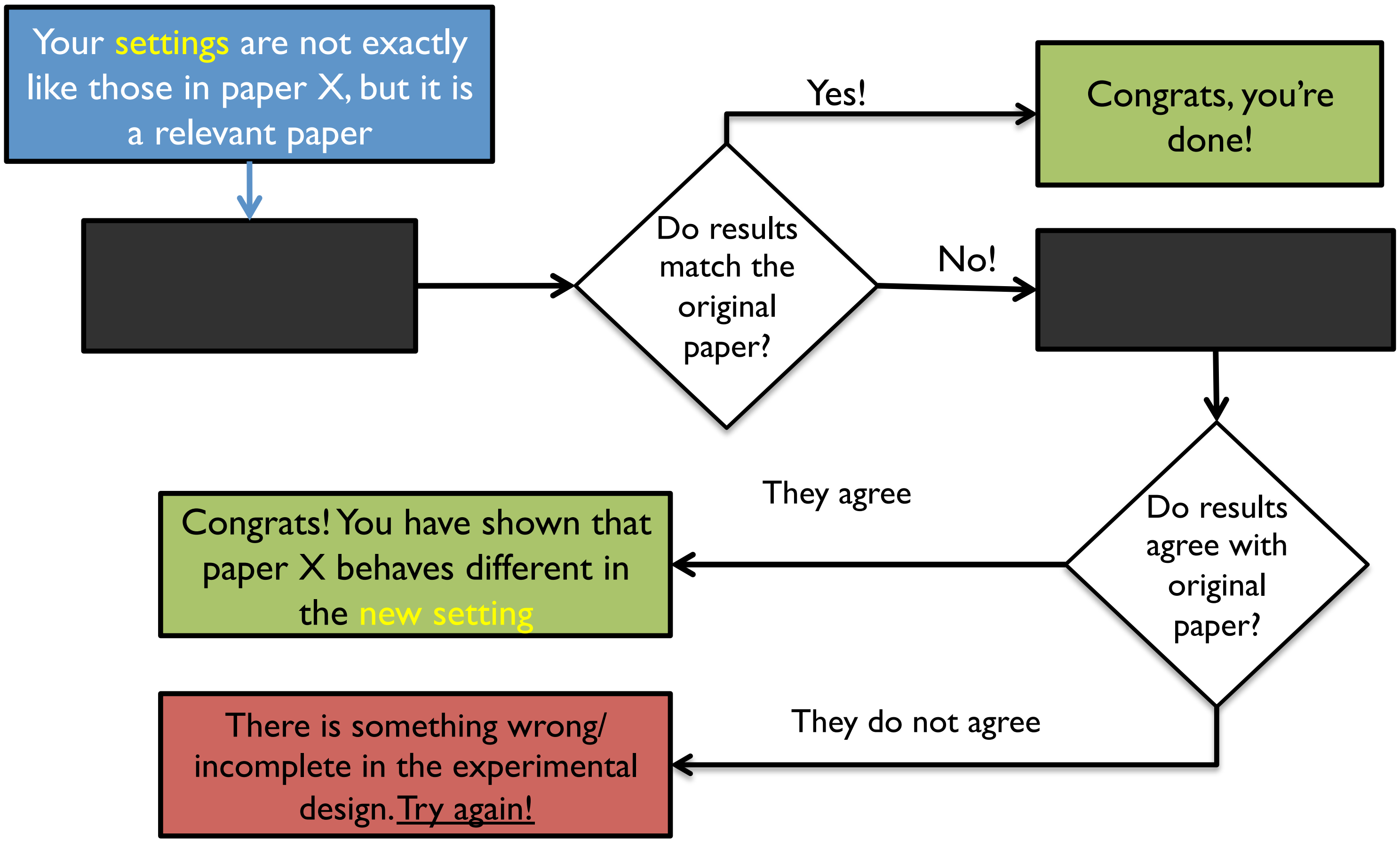
To <u>*recreate*</u> something

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

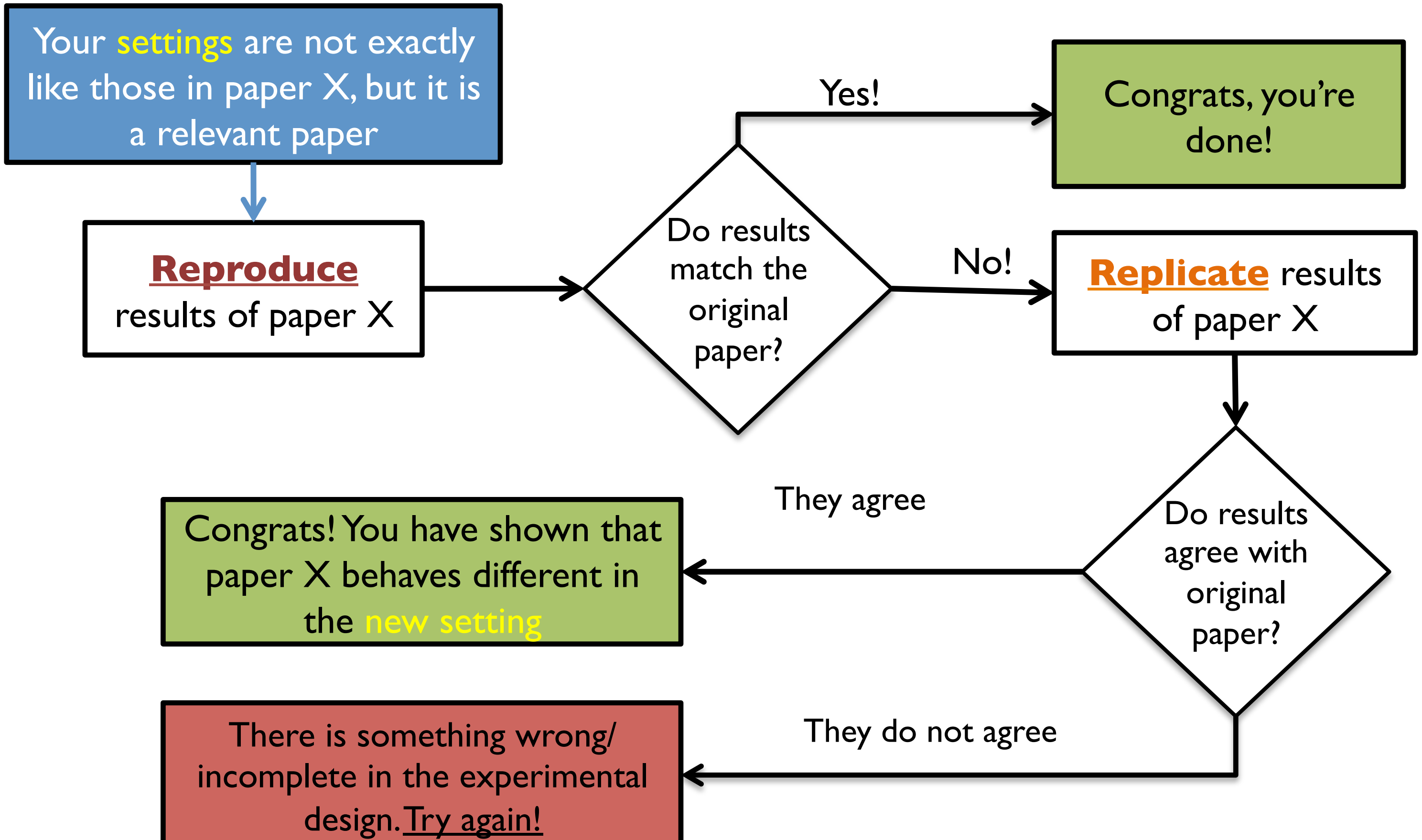
To (re)launch it in production with the same results



Comparing against the state-of-the-art



Comparing against the state-of-the-art



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

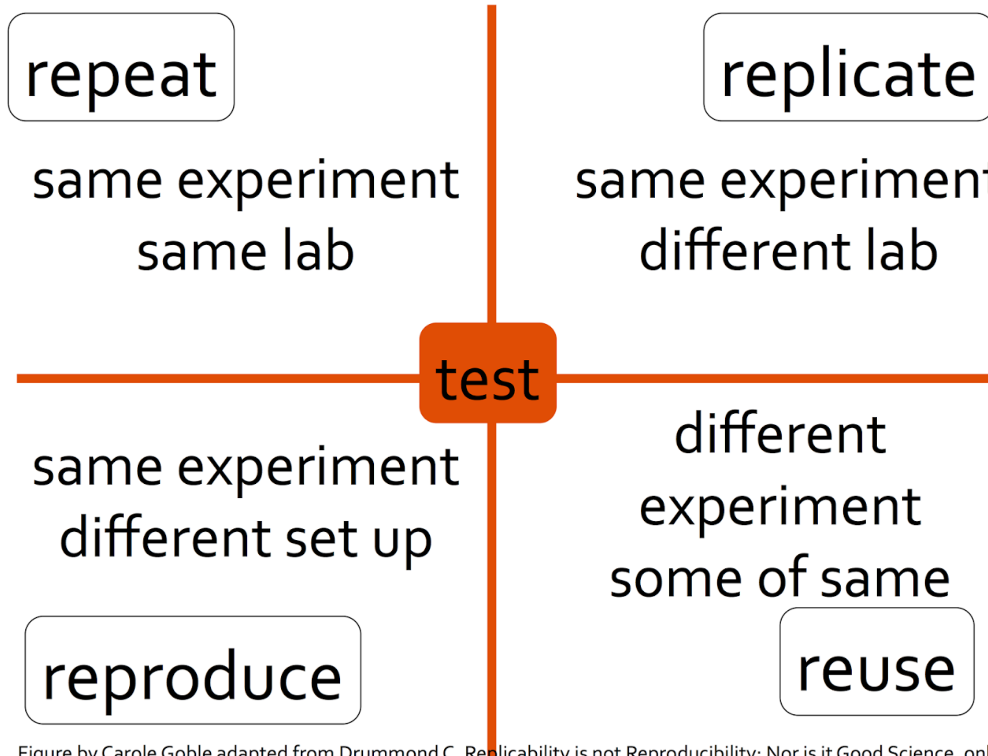


Figure by Carole Goble adapted from Drummond C, Replicability is not Reproducibility: Nor is it Good Science, online and Peng RD, Reproducible Research in Computational Science *Science 2 Dec 2011: 1226-1227.*

Motivation for reproducibility

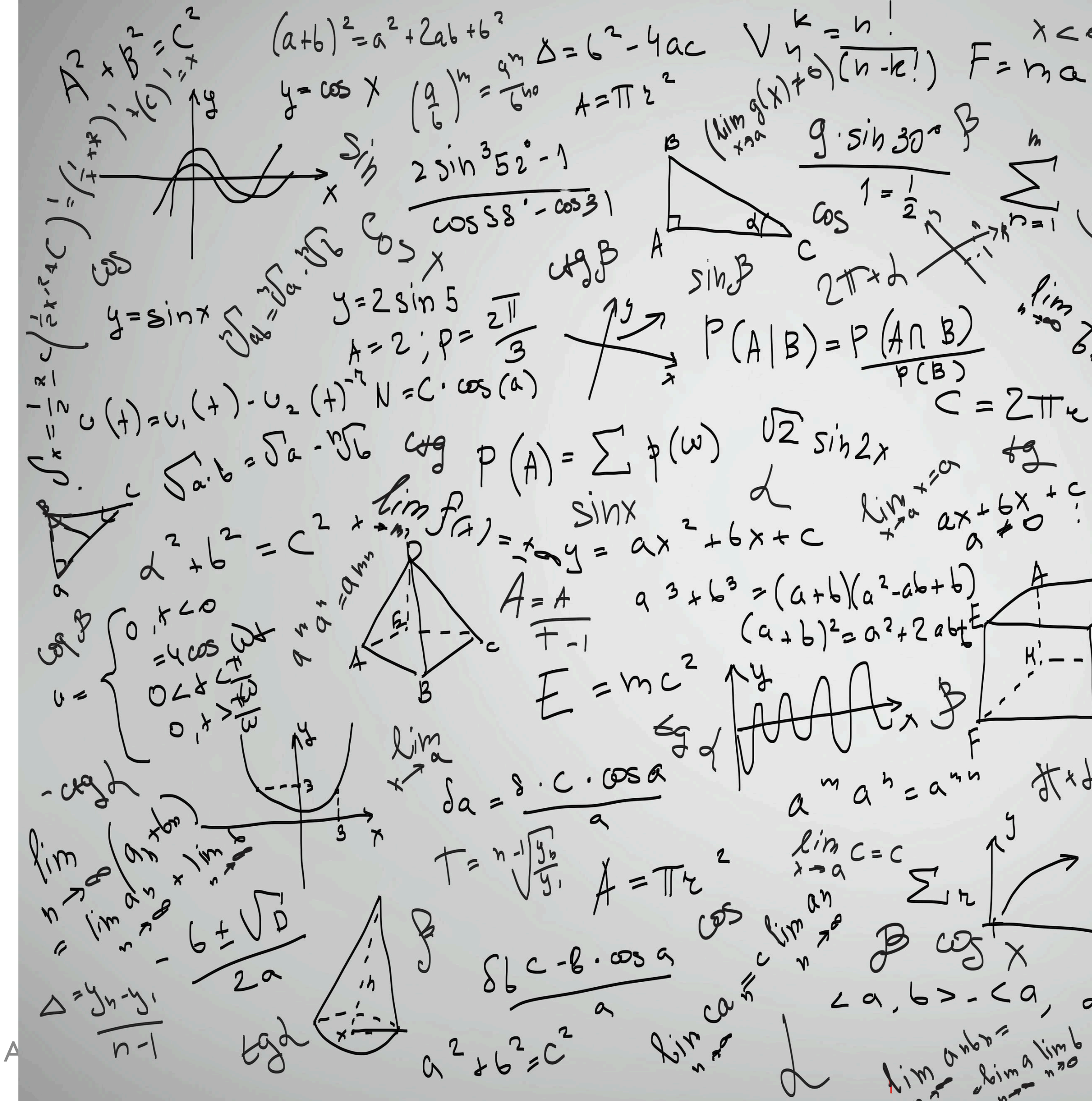
In order to ensure that our experiments, settings, and results are:

- 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)

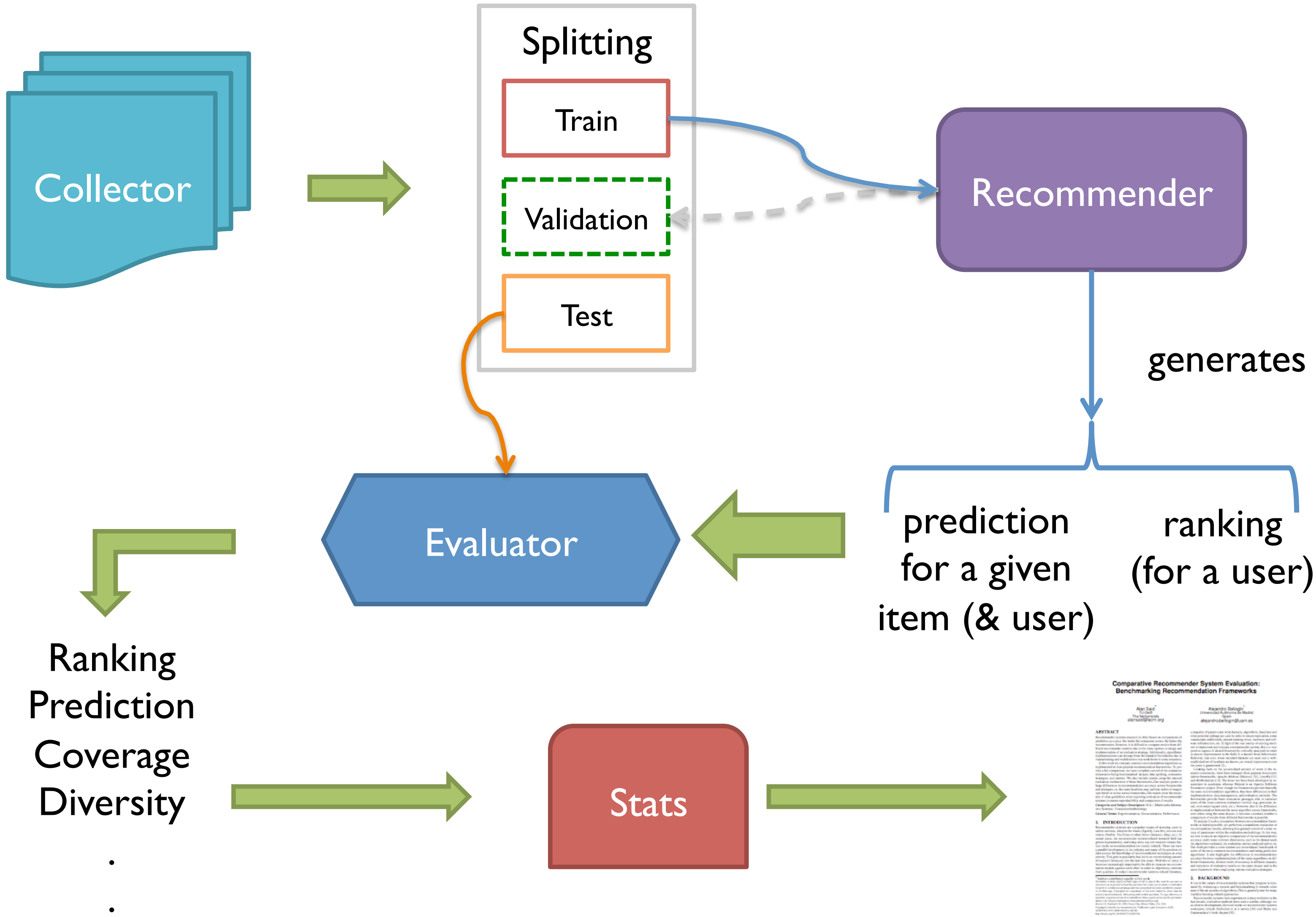
- We need to incorporate reproducibility and replication to facilitate progress in the field
- If your research is reproducible for others, it has more value

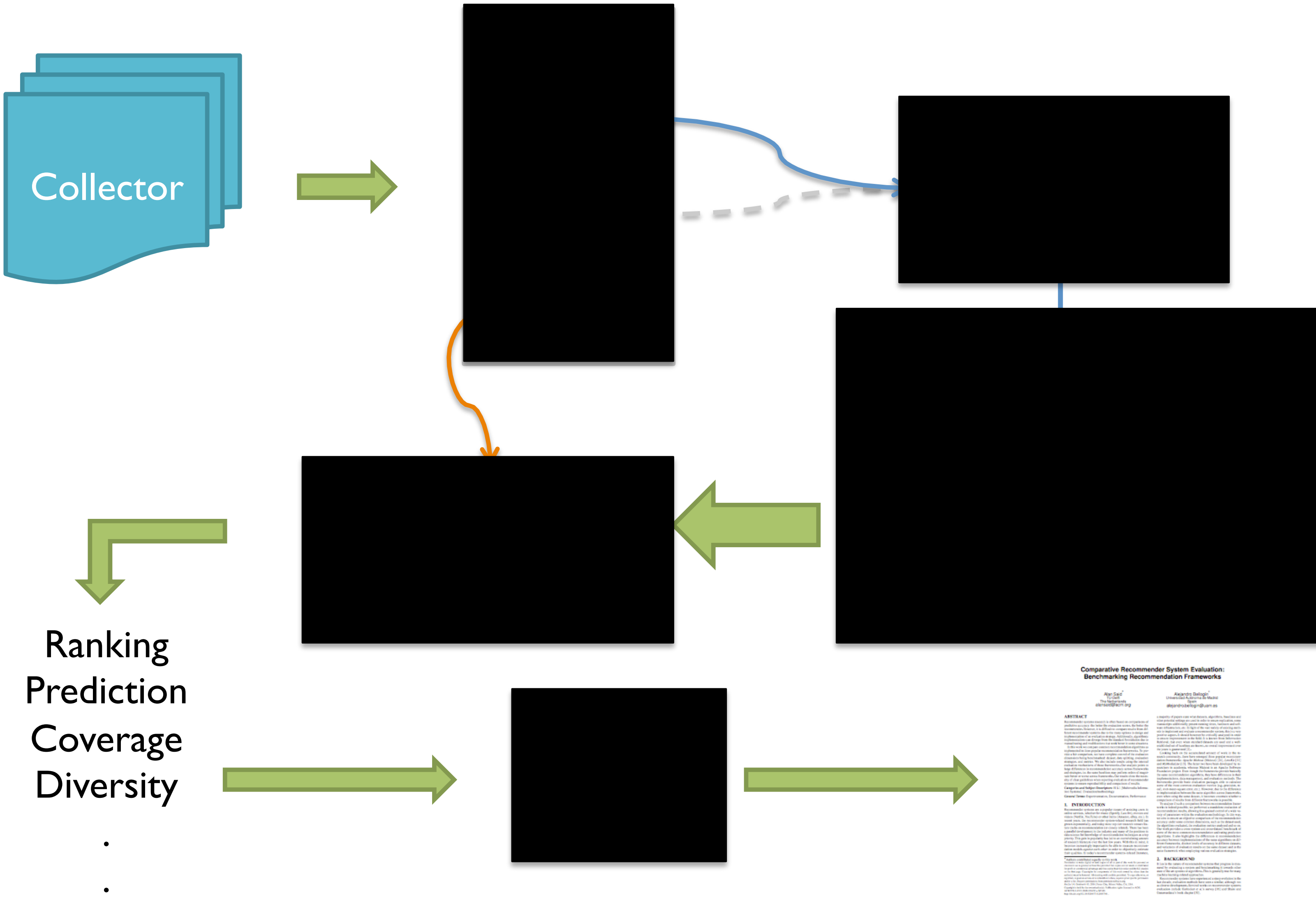
Outline

- 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

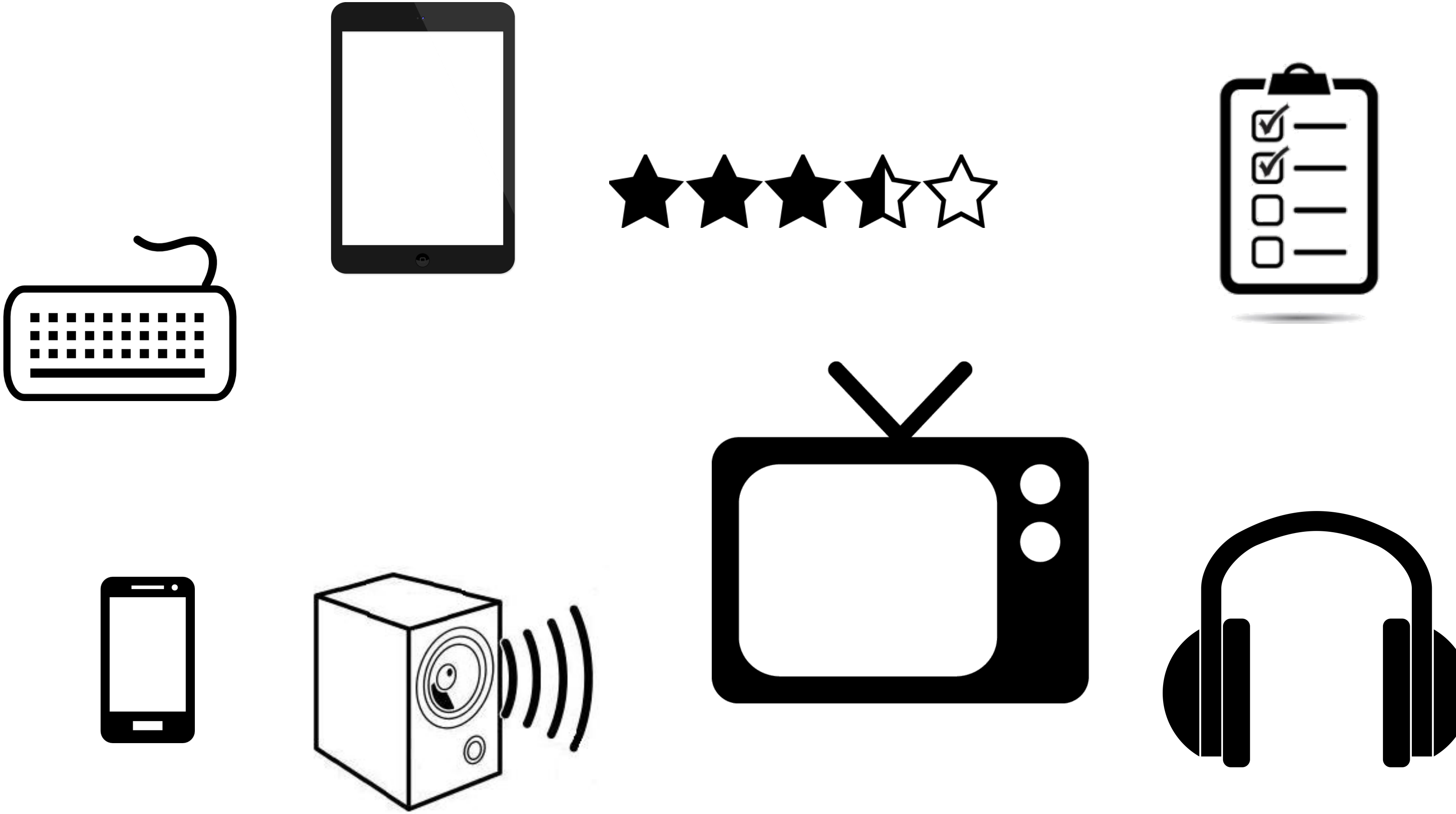
- Replicability/reproducibility/repeatability: useful and desirable in any field
  - How can they be addressed when dealing with recommender systems?
- Proposal: analyze the recommendation process and identify each stage that may affect the final results


Collector
Splitting
Train
Validation
Test
Recommender
generates
prediction for a given item (& user)
ranking (for a user)
Evaluator
Ranking
Prediction
Coverage
Diversity
.
.
.
Stats
Comparative Recommender System Evaluation: Benchmarking Recommendation Frameworks


Collector
Ranking
Prediction
Coverage
Diversity
Comparative Recommender System Evaluation:
Benchmarking Recommendation Frameworks

DATA CREATION AND COLLECTION

What is a dataset?



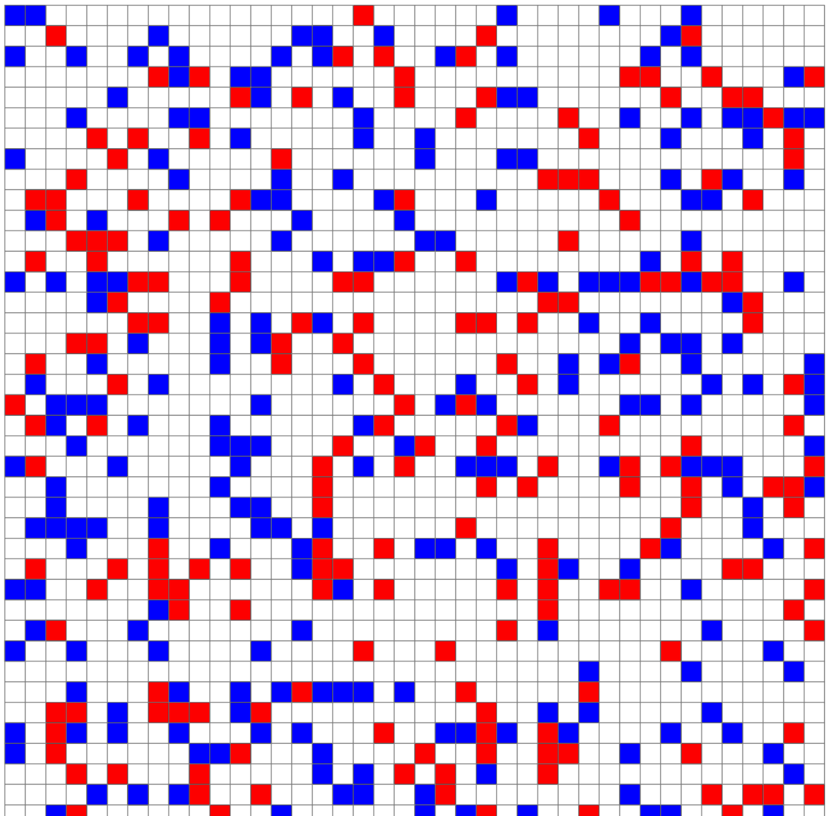
Public datasets

- 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



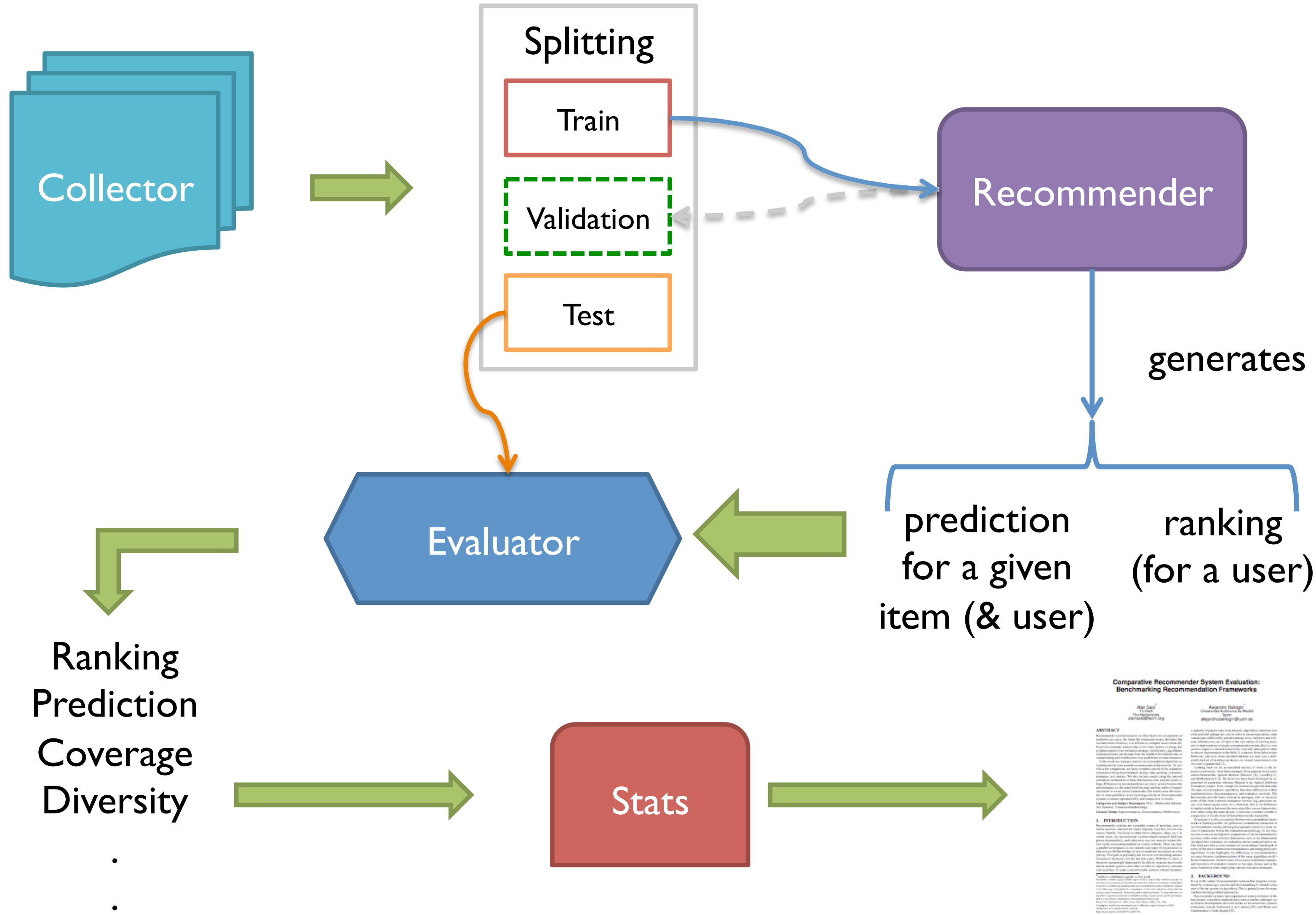
User	Item	Interaction	Timestamp
1	1	1	2017-…
1	2	1	…
2	3	2	…

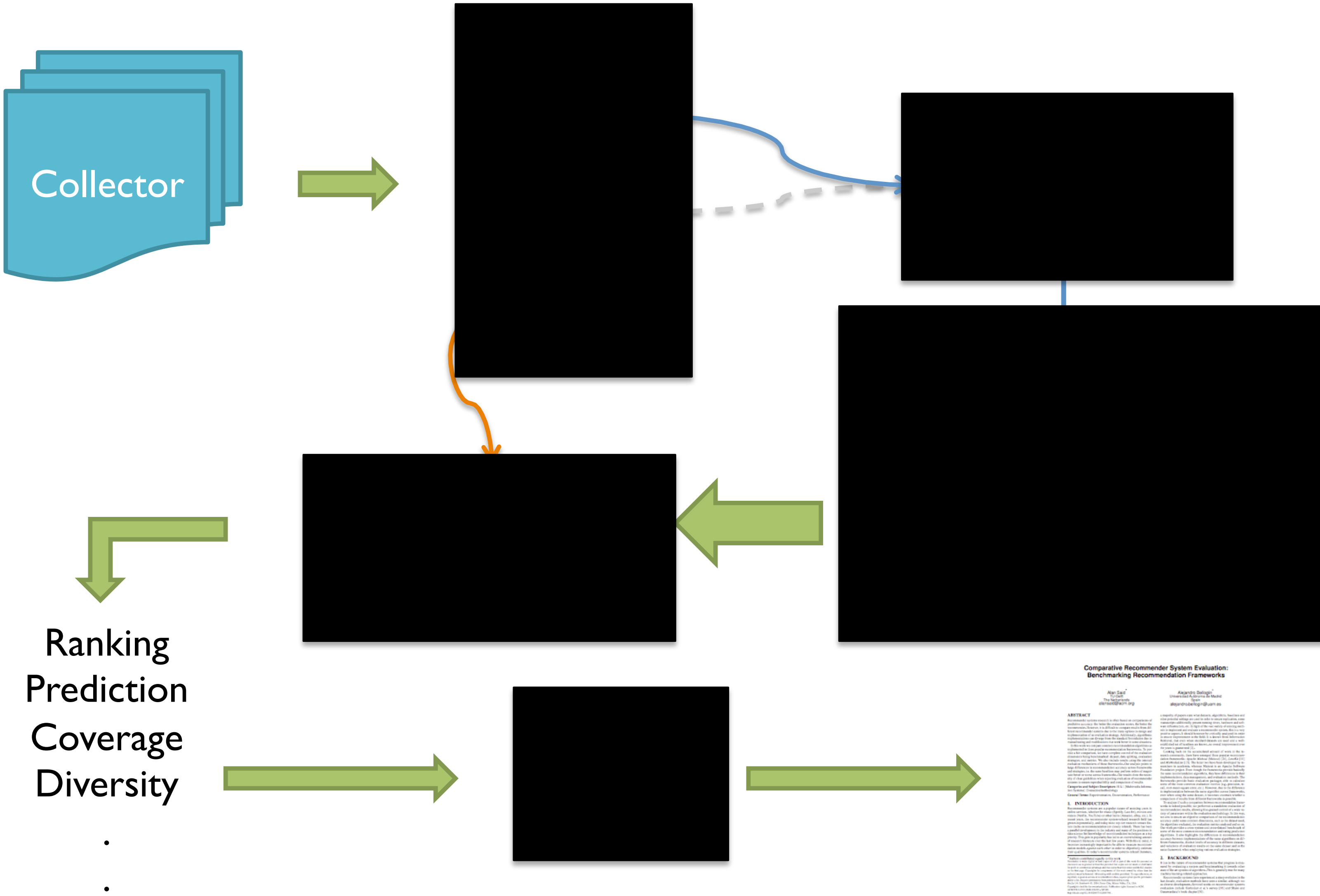
Releasing the dataset

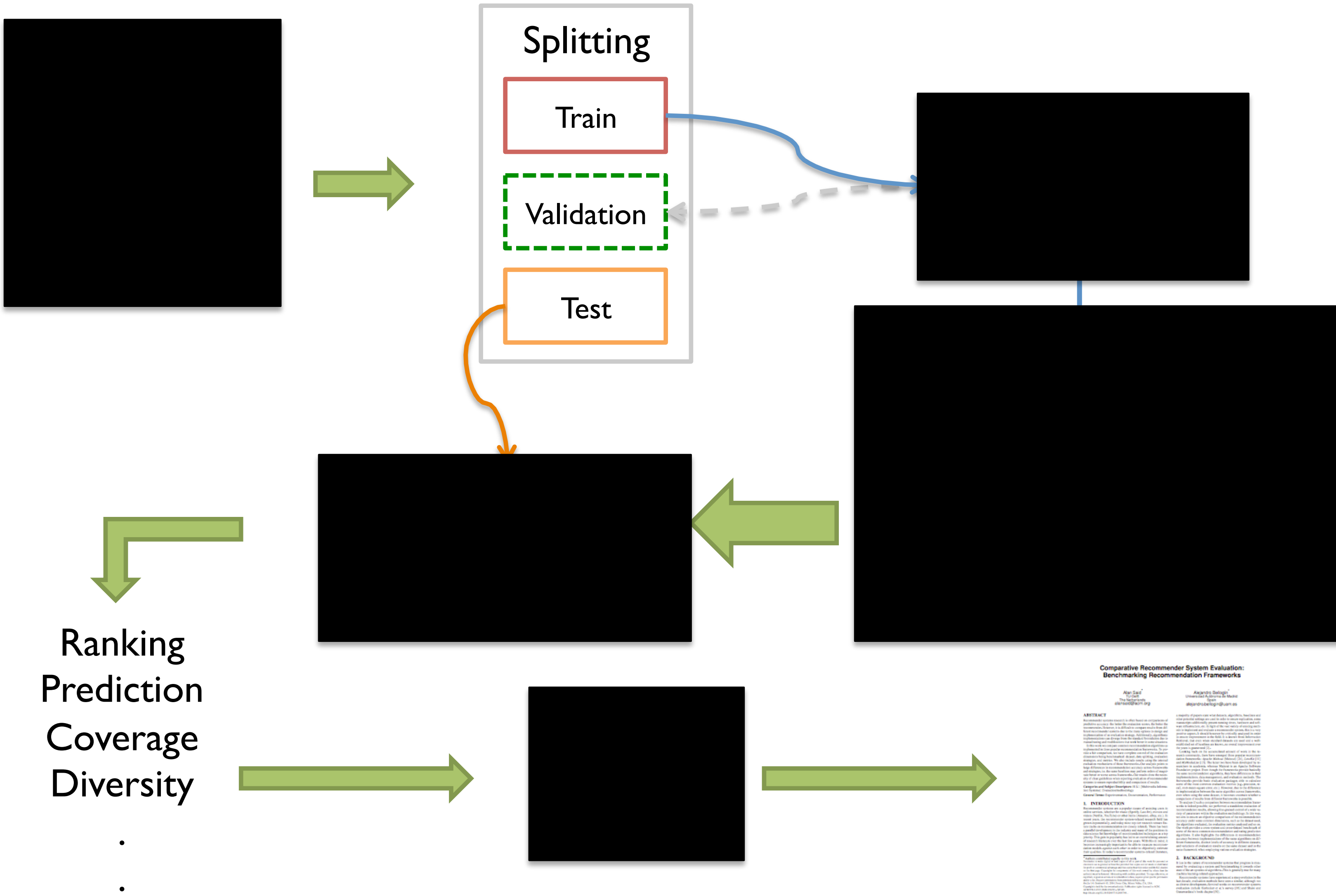
- Make the dataset publicly available
  - Otherwise your work is not reproducible
- Provide an in-depth overview
  - Website, paper, etc.
- Communicate it
  - Mailing lists, RecSysWiki, website, etc.

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.)

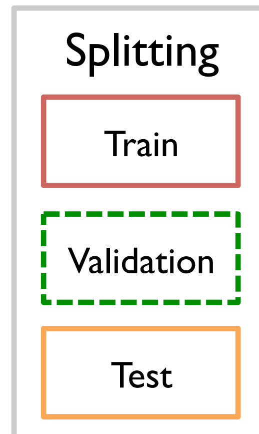

Collector
Splitting
Train
Validation
Test
Recommender
generates
prediction for a given item (& user)
ranking (for a user)
Evaluator
Ranking
Prediction
Coverage
Diversity
.
.
.
Stats
Comparative Recommender System Evaluation: Benchmarking Recommendation Frameworks


Collector
Ranking
Prediction
Coverage
Diversity
.
.
Comparative Recommender System Evaluation:
Benchmarking Recommendation Frameworks

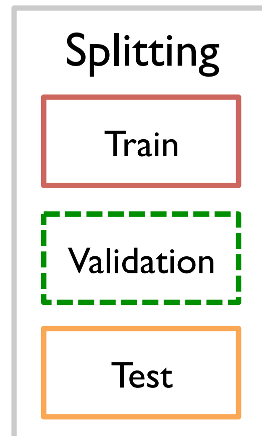


Splitting
Train
Validation
Test
Ranking
Prediction
Coverage
Diversity
.
.
.
Comparative Recommender System Evaluation: Benchmarking Recommendation Frameworks
Alan Said
Alejandro Bellogín

DATA SPLITTING AND PREPARATION


Splitting
Train
Validation
Test

- 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

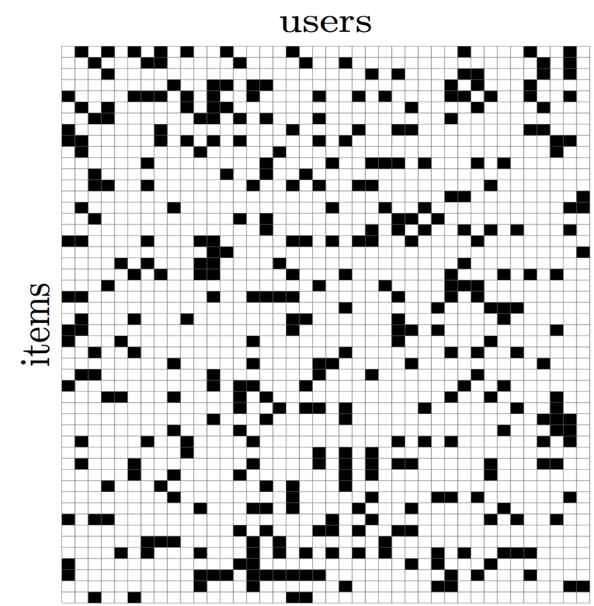
- Which are the candidate items that we will be recommending?
- Who are the candidate users we will be recommending (and evaluating) for?
- Do we have any limitations on numbers?
  - Cold start?
  - Temporal/trending recommendations?
  - Other?

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

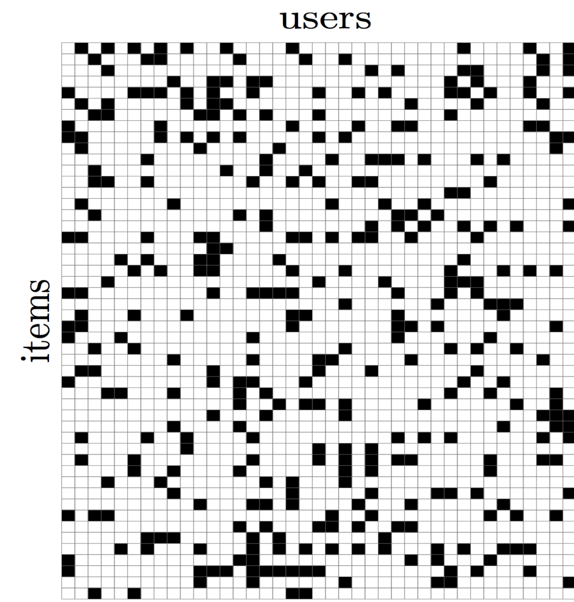
Random

- 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



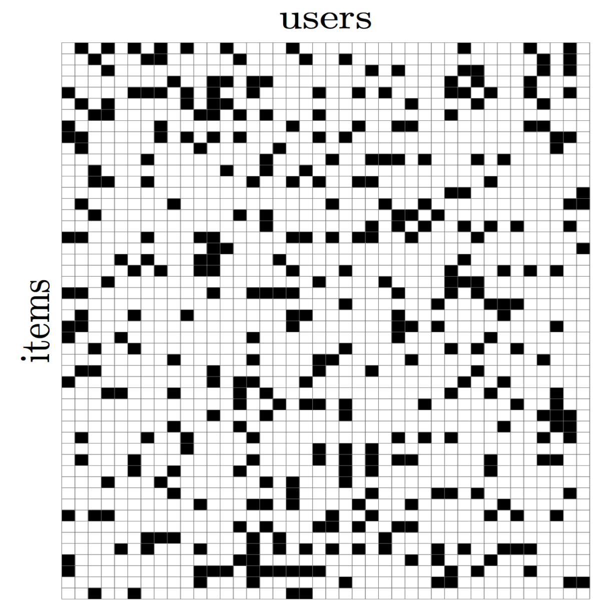
All users at once

- 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)



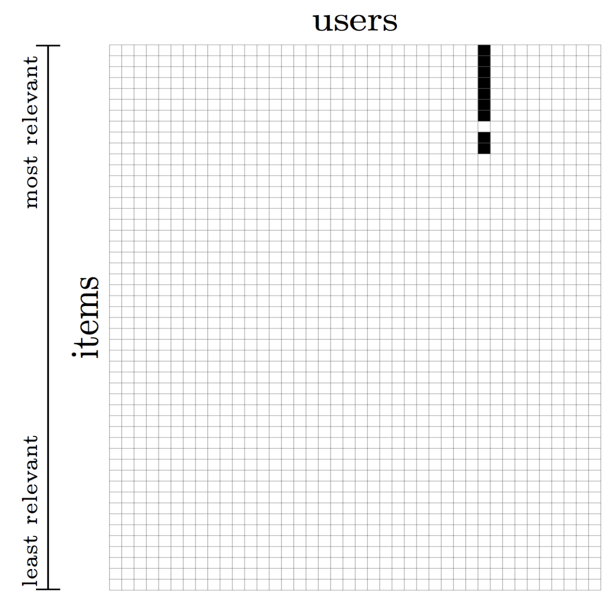
All items at once

- The split does not take into consideration
  - Whether users are left out of the training or test sets.



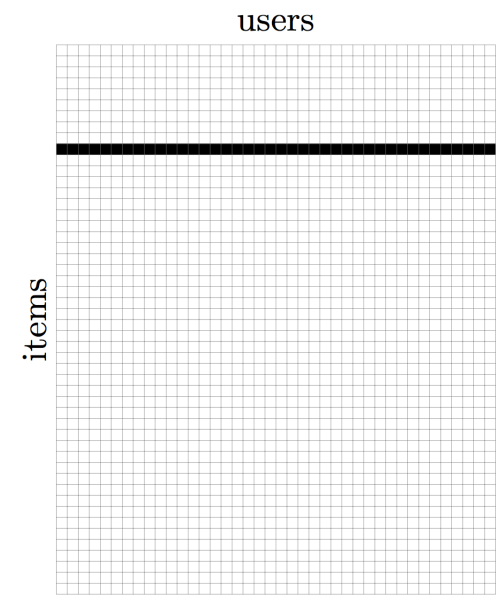
One user at once

- 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



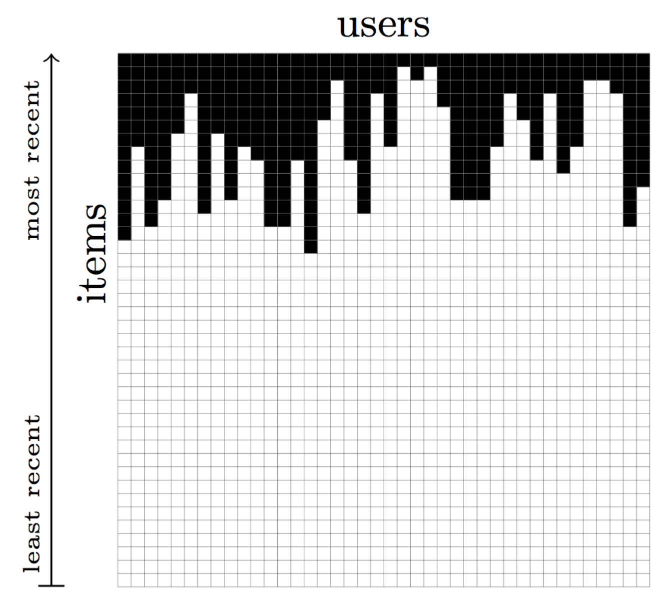
One item at once

- 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?

Movielens 20M

– "Users were selected at **random** for inclusion. All selected **users had rated at least 20 movies**."

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

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:

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

- `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.

Data splitting - LibRec

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 givenn

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

```
data.model.splitter=givenn
```

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 the average of all the folds.

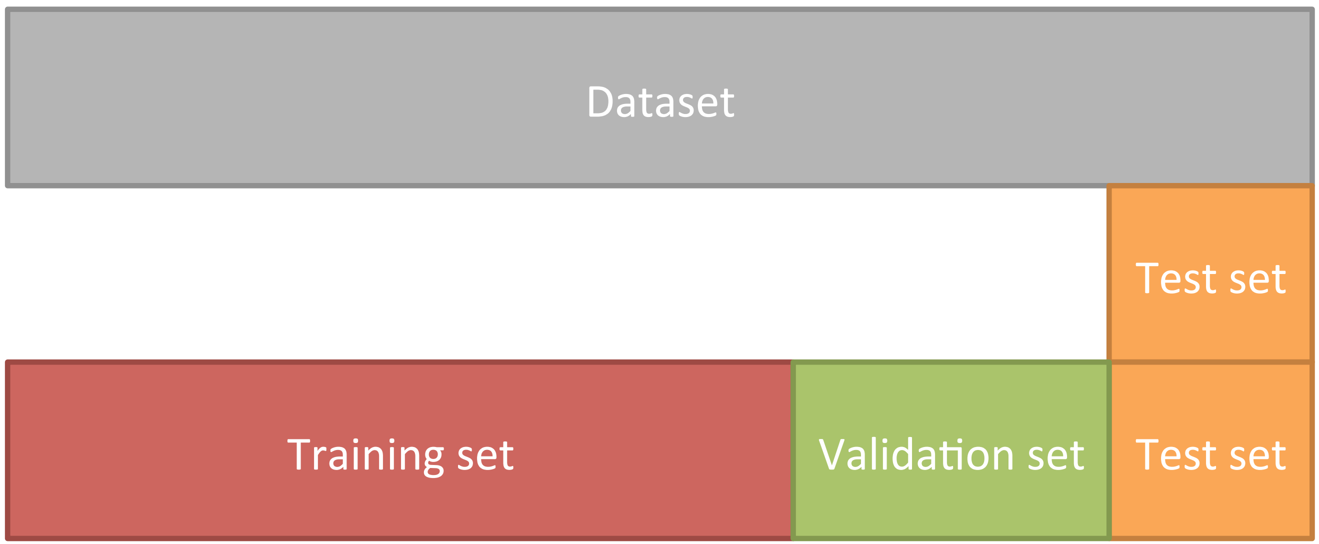
2.5 testset

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.

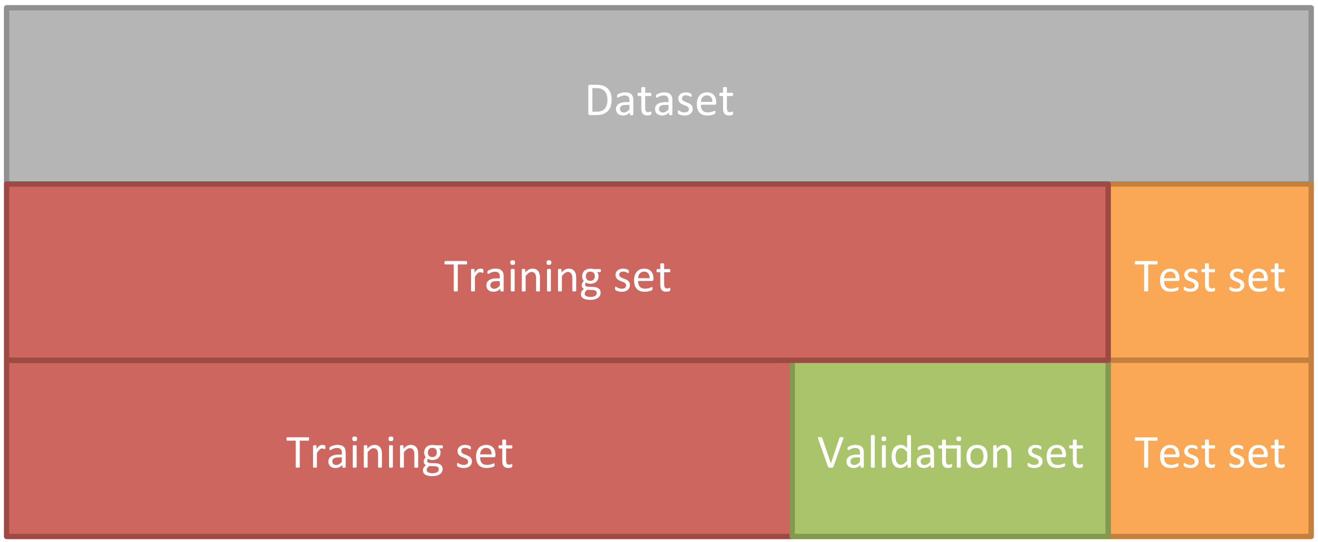
```
data.model.splitter=testset
data.testset.path=nameoftestfile/dir
```

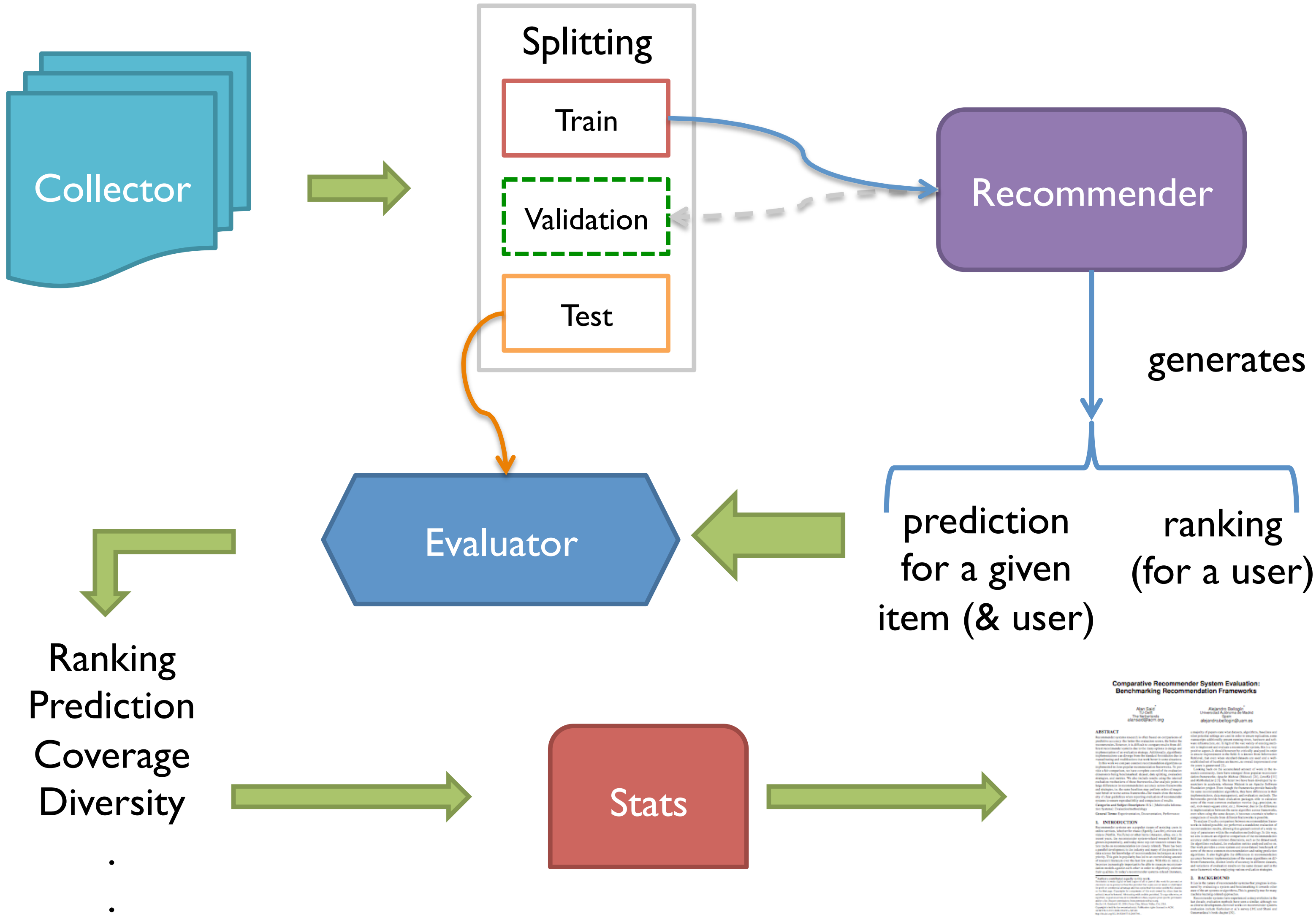
https://www.librec.net/dokuwiki/doku.php?id=DataModel

Partitions



Partitions


Dataset
Training set
Test set
Training set
Validation set
Test set


Collector
Splitting
Train
Validation
Test
Recommender
generates
prediction for a given item (& user)
ranking (for a user)
Evaluator
Ranking
Prediction
Coverage
Diversity
.
.
.
Stats
Comparative Recommender System Evaluation: Benchmarking Recommendation Frameworks

Splitting

Train

Validation

Test

Ranking

Prediction

Coverage

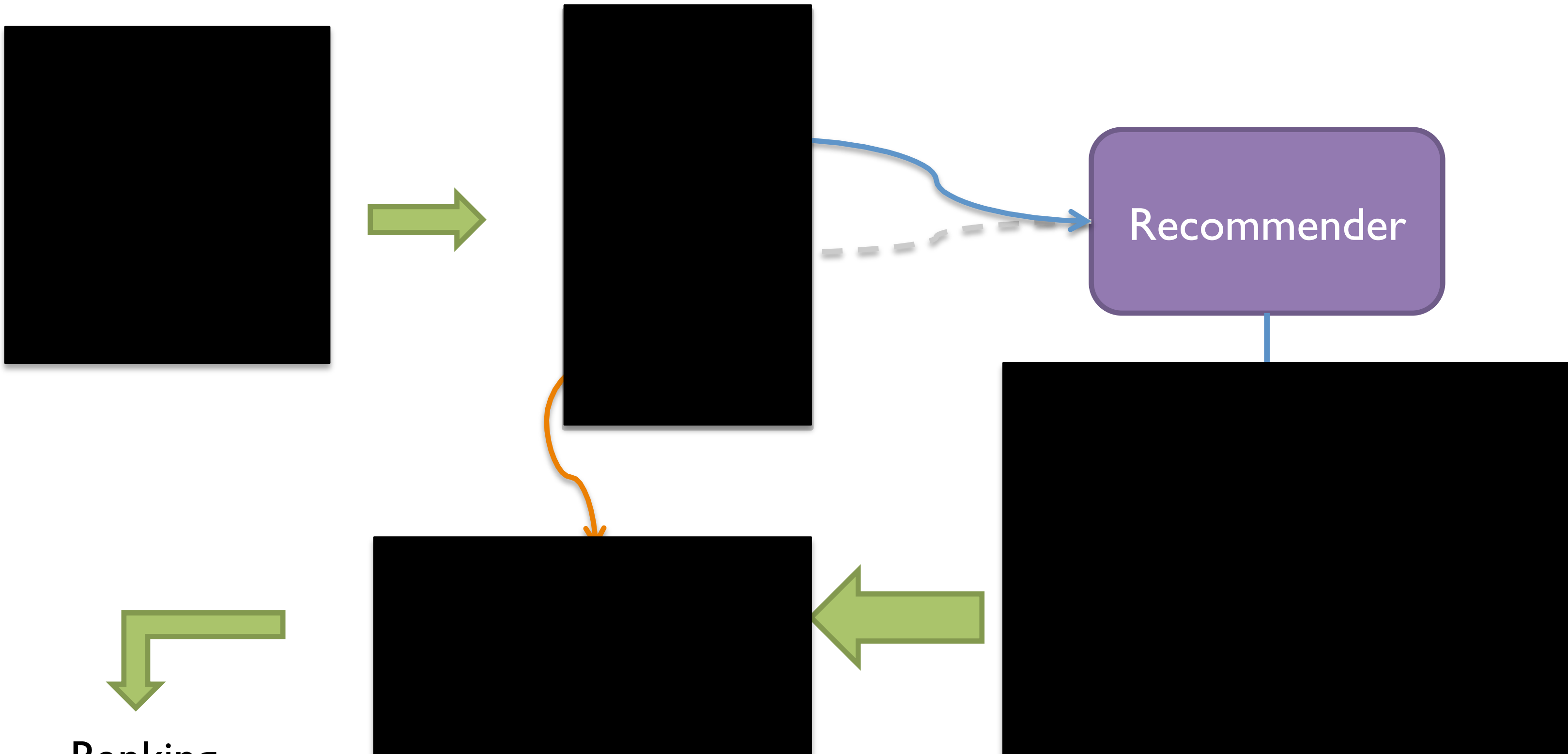
Diversity

.

.

.


Comparative Recommender System Evaluation:
Benchmarking Recommendation Frameworks



Ranking
Prediction
Coverage
Diversity
.
.
.


Comparative Recommender System Evaluation:
Benchmarking Recommendation Frameworks

Alan Said
TU-Delft
The Netherlands
alansaid@acm.org

Alejandro Bellogín
Universidad Autónoma de Madrid
Spain
alejandro.bellogin@uam.es

ABSTRACT

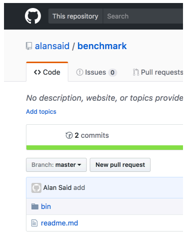
RECOMMENDATION

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

$$\tilde{r}(u,i) = \bar{r}(u) + C \sum_{v \in N_k(u)} \mathrm{sim}(u,v)\left(r(v,i) - \bar{r}(v)\right)$$

$$\tilde{r}(u,i) = C \sum_{v \in N_k(u)} \mathrm{sim}(u,v) r(v,i)$$

Collaborative Filtering

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

$$\tilde{r}(u,i) = \bar{r}(u) + C \sum_{v \in N_k(u)} \mathrm{sim}(u,v)\left(r(v,i) - \bar{r}(v)\right)$$

$$\tilde{r}(u,i) = C \sum_{v \in N_k(u)} \mathrm{sim}(u,v) r(v,i)$$

Similarities

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

$$\mathrm{sim}_s(a,b) = \frac{n_{a,b}}{n_{a,b} + \lambda_s} \mathrm{sim}(a,b)$$

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
    - *Y. Hu, Y. Koren, C. Volinsky. Collaborative filtering for implicit feedback datasets. ICDM 2008*
    - *I. Pilászy, D. Zibriczky and D. Tikk. Fast ALS-based Matrix Factorization for Explicit and Implicit Feedback Datasets. RecSys 2010.*

MF Implementation

- LibRec
  - Separate rating prediction and ranking models
  - RankALSRecommender - Ranking
    - *Takács and Tikk. Alternating Least Squares for Personalized Ranking. RecSys 2012.*
  - 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
  - Newman et al. 2009. *Distributed Algorithms for Topic Models*.

Probabilistic Implementation

- LibRec
  - LDA for implicit feedback
  - Griffiths. 2002. *Gibbs sampling in the generative model of Latent Dirichlet Allocation*

Recommending: LibRec

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.recommedner.class` configuration. The example code is shown as follows.

```
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.

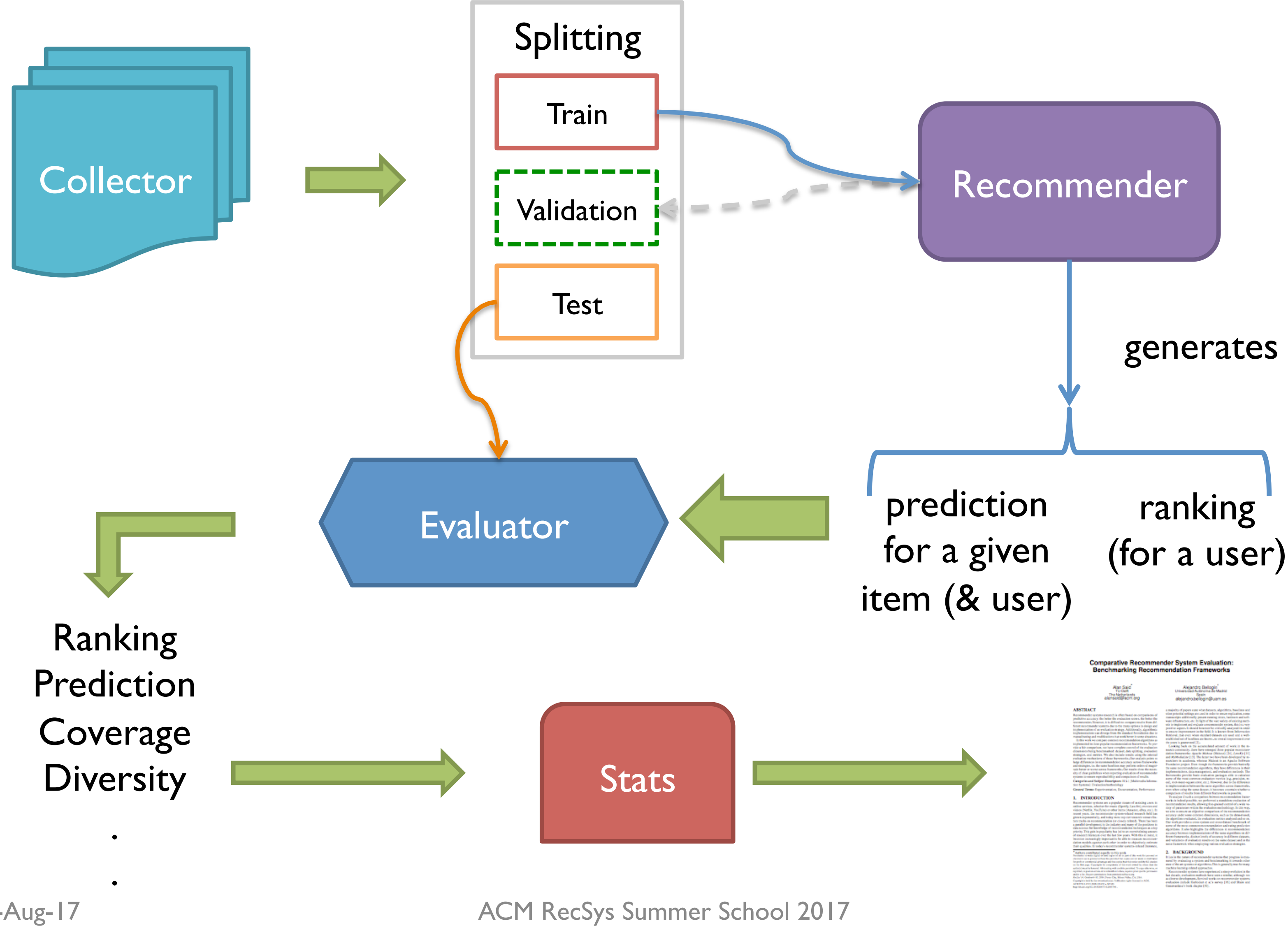
Unlike most other algorithms, the user-user filter does not really have a model that is built (though some things such as the global mean rating used by baselines are computed at model build time)

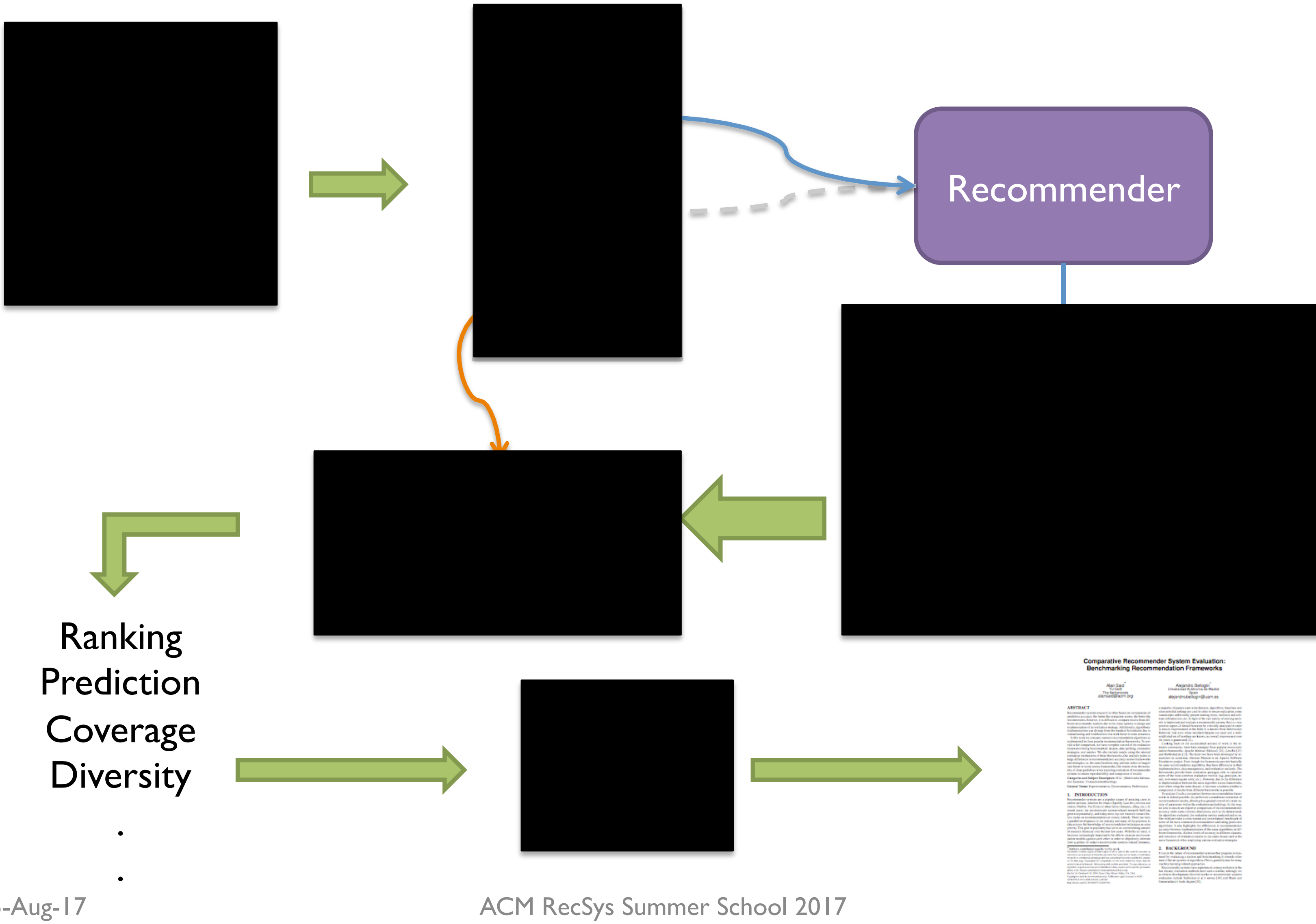
Here are some of the additional configuration points ('@' indicates a parameter to be set with `set` rather than `bind`):

- UserVectorNormalizer — normalizes user rating vectors prior to similarity computation and prediction.
- NeighborhoodFinder — finds neighborhoods for scoring items. The default implementation is SimpleNeighborhoodFinder. Since LensKit 2.1, you can use `SnapshotNeighborhoodFinder` 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][], just compares the users' vectors using a vector similarity function; the default vector similarity is CosineVectorSimilarity.

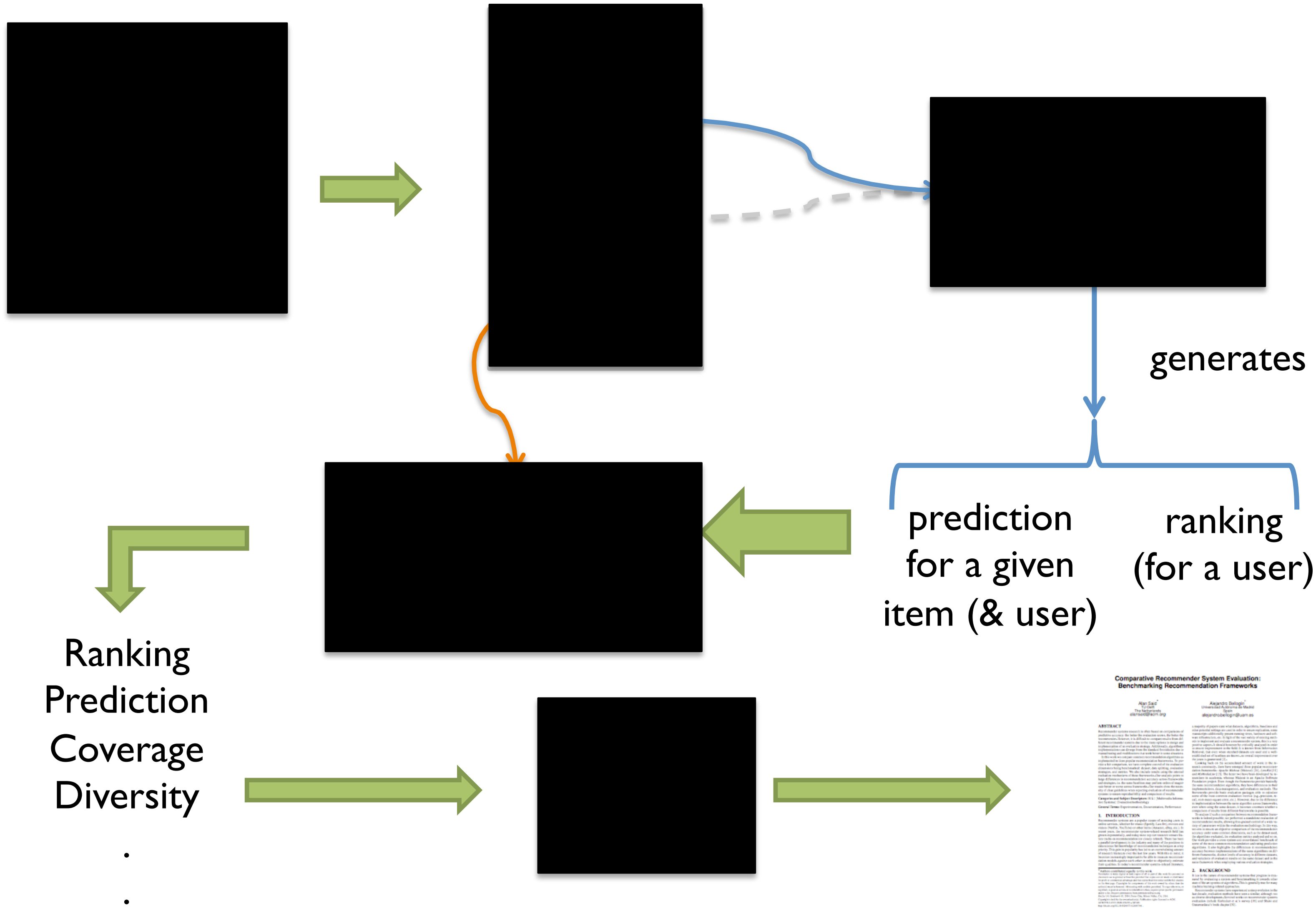
Outline

- 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


Collector
Splitting
Train
Validation
Test
Recommender
generates
prediction for a given item (& user)
ranking (for a user)
Evaluator
Ranking
Prediction
Coverage
Diversity
.
.
.
Stats
Comparative Recommender System Evaluation: Benchmarking Recommendation Frameworks



Recommender
Ranking
Prediction
Coverage
Diversity
.
.
.
Comparative Recommender System Evaluation: Benchmarking Recommendation Frameworks
Alan Said
Alejandro Bellogín


generates
prediction for a given item (& user)
ranking (for a user)
Ranking
Prediction
Coverage
Diversity
Comparative Recommender System Evaluation: Benchmarking Recommendation Frameworks

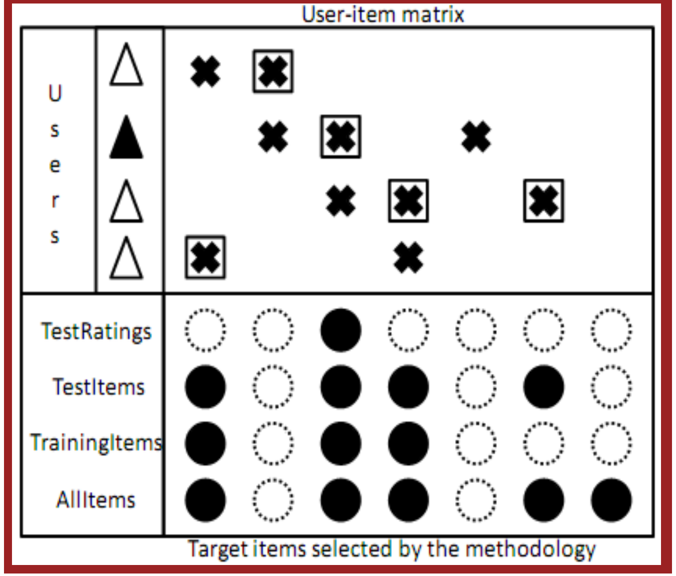
Candidate item generation

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

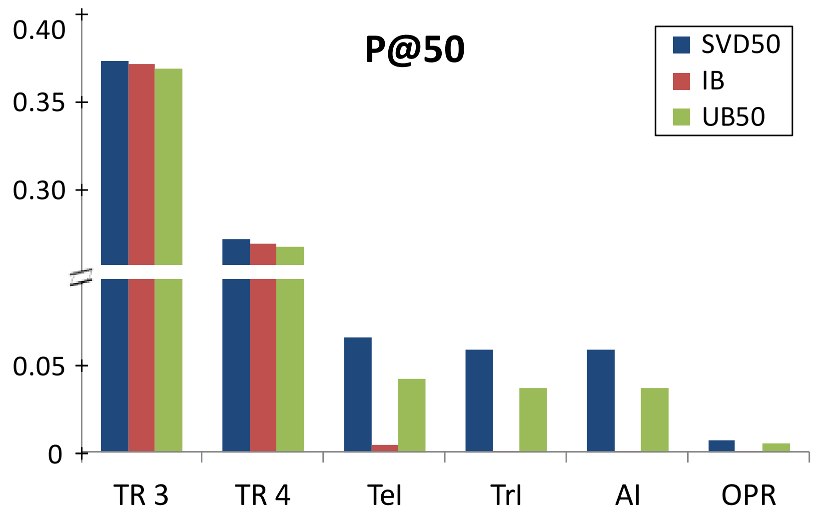
Note: in CF, AllItems and TrainingItems produce the same results

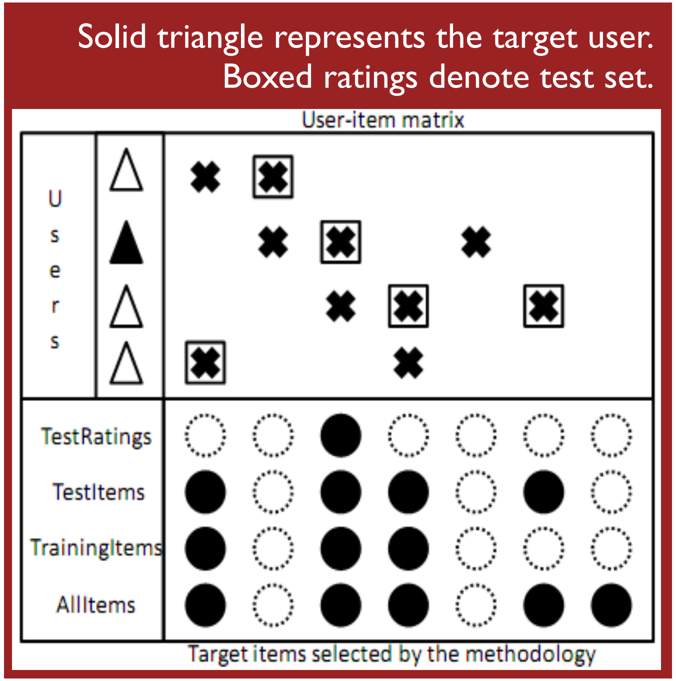
Solid triangle represents the target user.
Boxed ratings denote test set.



Candidate item generation

Different ways to select candidate items to be ranked:





[Bellogín et al, 2011]

Candidate item generation

Impact of <u>different strategies</u> for candidate item selection:

RPN: RelPlusN

a ranking with

1 relevant and

N non-relevant items

UT: UserTest

same as TestRatings

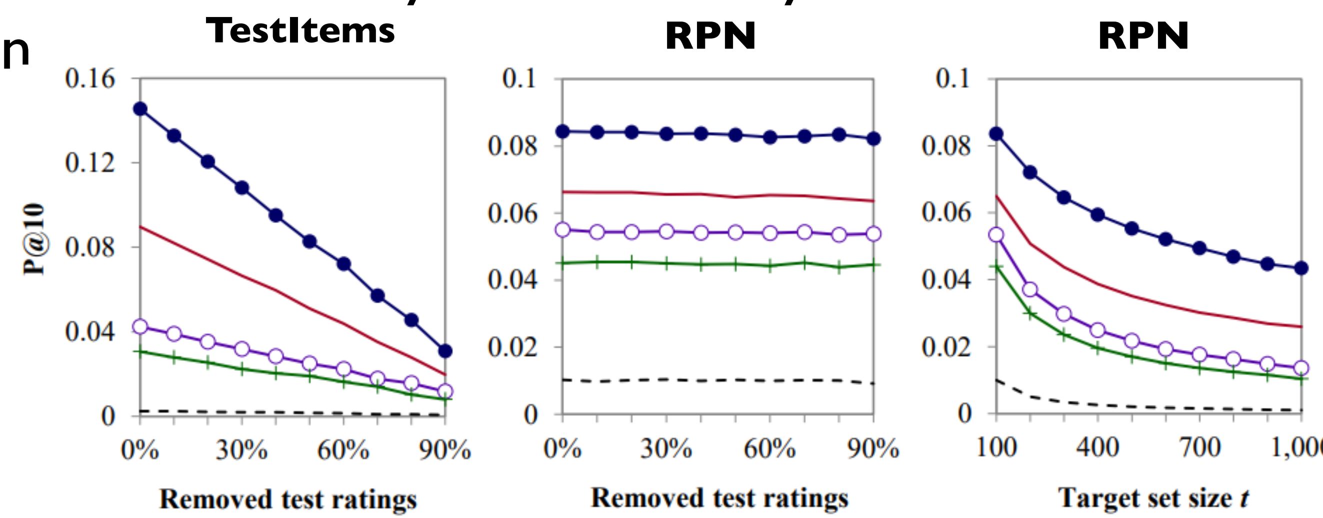
Alg.	F.W.	Time (sec.)	RMSE	nDCG@10		User cov.(%)		Cat. cov.(%)	
				RPN	UT	RPN	UT	RPN	UT
IBCos	AM	238	1.041	0.003	0.501	98.16	100	99.71	99.67
	LK	44	0.953	0.199	0.618	98.16	100	99.88	99.67
	MML	75	NA	0.488	0.521	98.16	100	100	99.67
IBPea	AM	237	1.073	0.022	0.527	97.88	100	86.66	99.31
	LK	31	1.093	0.033	0.527	97.86	100	86.68	99.31
	MML	1,346	0.857	0.882	0.654	98.16	100	2.87	99.83
SVD50	AM	132	0.950	0.286	0.657	98.12	100	99.88	99.67
	LK	7	1.004	0.280	0.621	98.16	100	100	99.67
	MML	1,324	0.848	0.882	0.648	98.18	100	2.87	99.83
UBCos50	AM	5	1.178	0.378	0.387	35.66	98.25	6.53	27.80
	LK	25	1.026	0.223	0.657	98.16	100	99.88	99.67
	MML	38	NA	0.519	0.551	98.16	100	100	99.67
UBPea50	AM	6	1.126	0.375	0.486	48.50	100	10.92	39.08
	LK	25	1.026	0.223	0.657	98.16	100	99.88	99.67
	MML	1,261	0.847	0.883	0.652	98.18	100	2.87	99.83

[Said & Bellogín, 2014]

Candidate item generation

Impact of <u>test size</u> for different candidate item selection strategies:

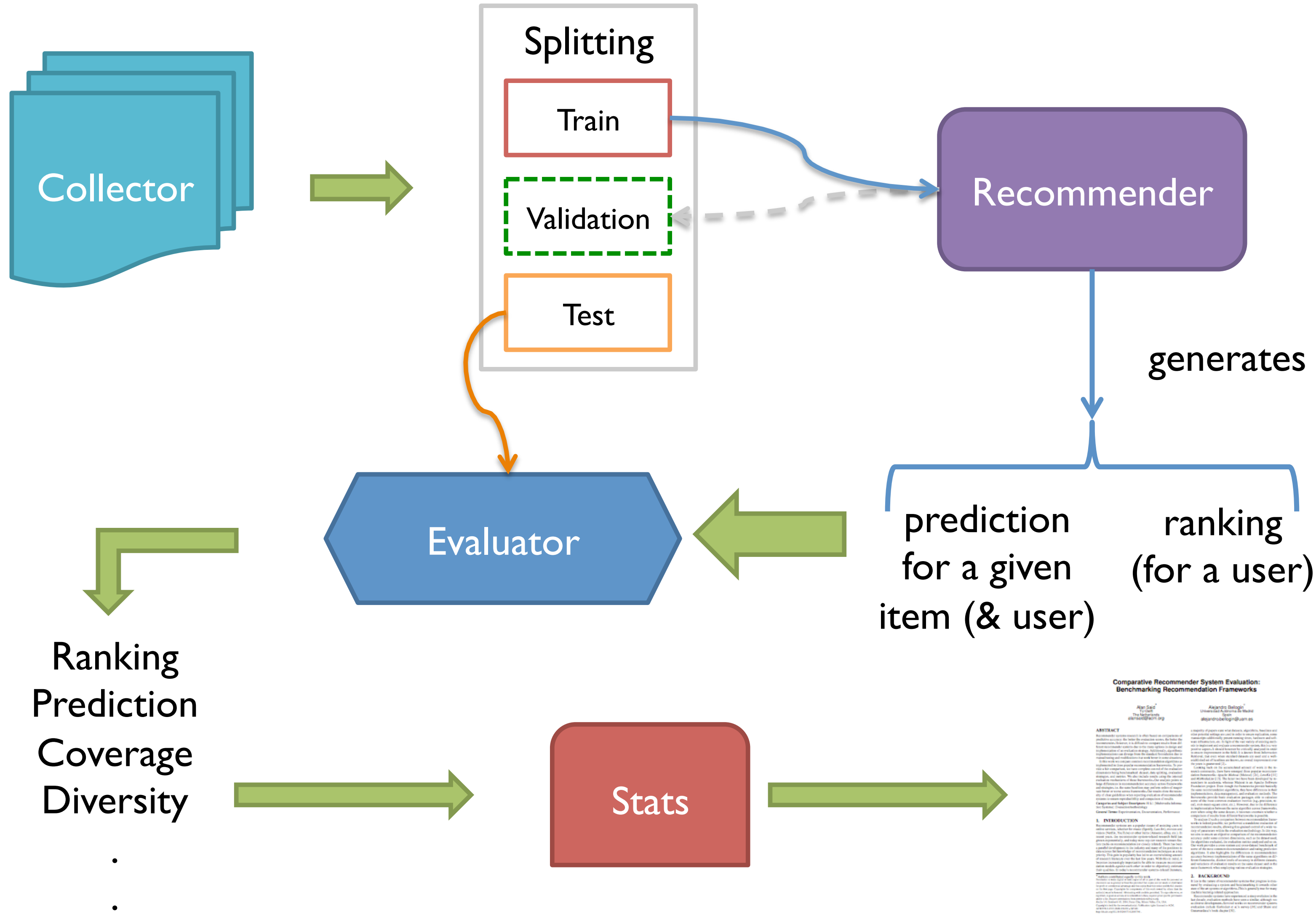
The actual value of the metric may be affected by the amount of known information



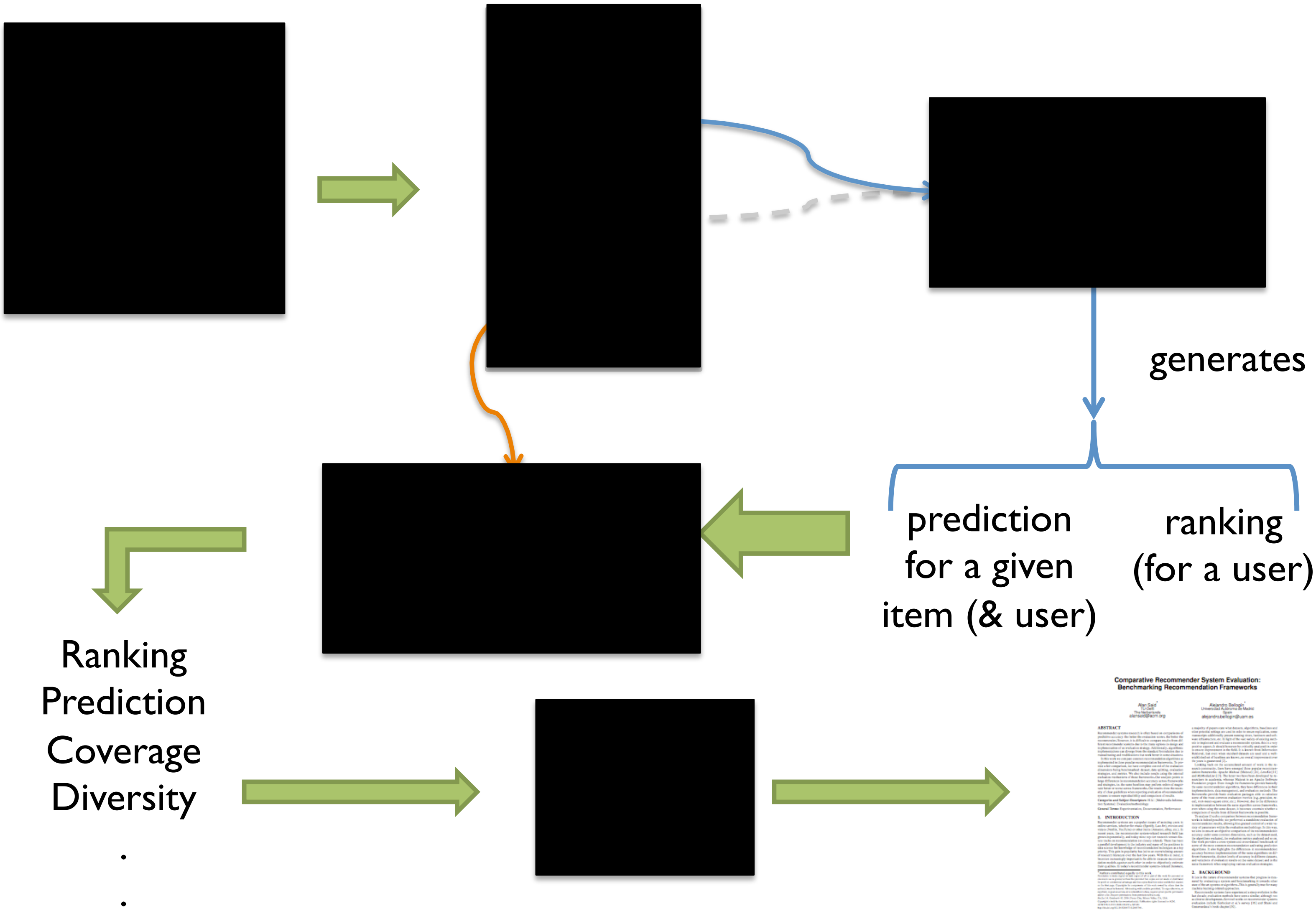
[Bellogín et al, 2017]

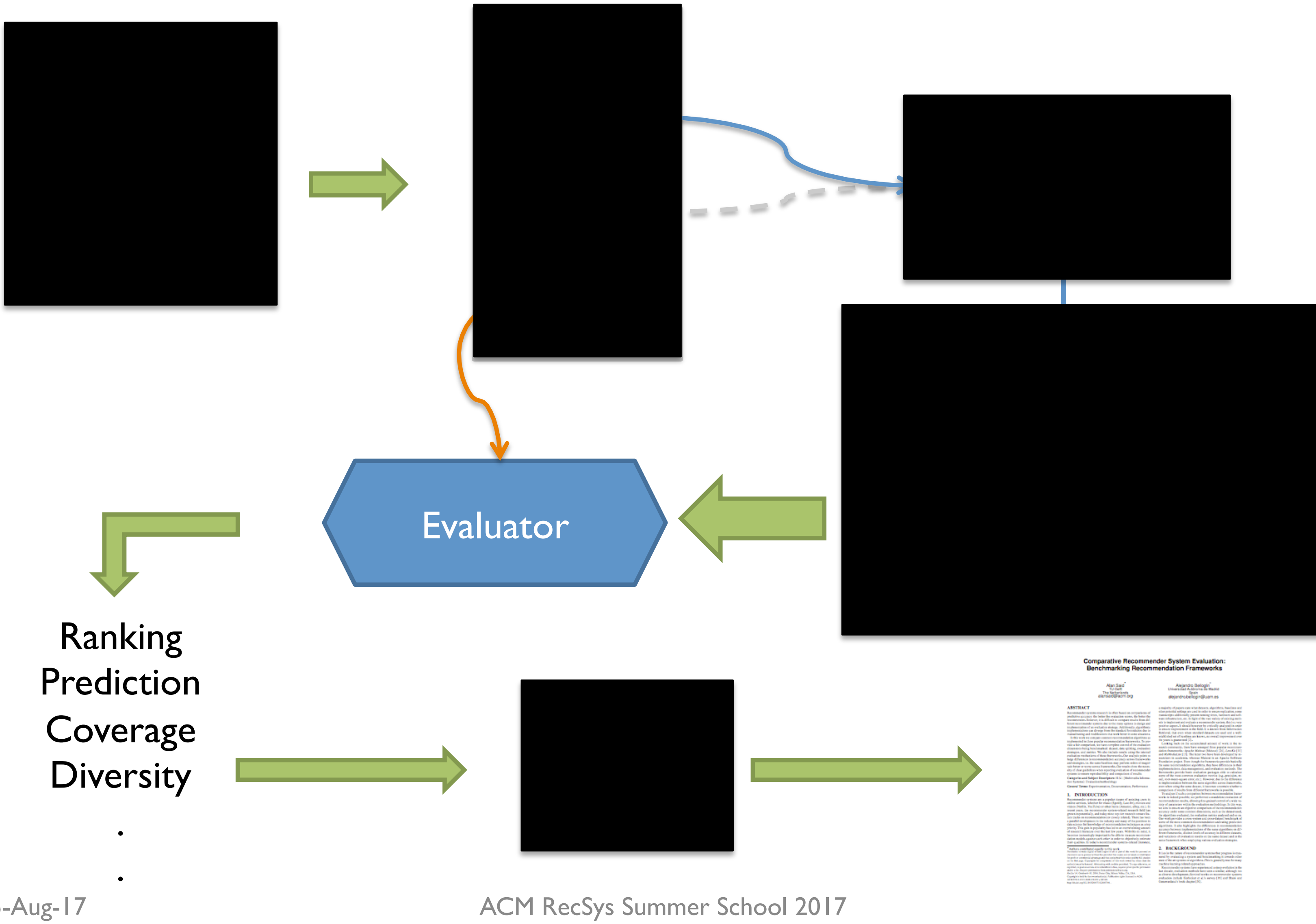
Outline

- 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



Collector
Splitting
Train
Validation
Test
Recommender
generates
prediction for a given item (& user)
ranking (for a user)
Evaluator
Ranking
Prediction
Coverage
Diversity
.
.
.
Stats
Comparative Recommender System Evaluation: Benchmarking Recommendation Frameworks


generates
prediction for a given item (& user)
ranking (for a user)
Ranking
Prediction
Coverage
Diversity
.
.
Comparative Recommender System Evaluation: Benchmarking Recommendation Frameworks
Alan Said
Alejandro Bellogín
ABSTRACT
1. INTRODUCTION
2. BACKGROUND


Evaluator
Ranking
Prediction
Coverage
Diversity
.
.
.
Comparative Recommender System Evaluation:
Benchmarking Recommendation Frameworks

Evaluation metric computation

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

– If a user receives 0 recommendations

<u>Option a</u>:

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

<u>Option b</u>:

$$\text{metric} = \frac{1}{|u \in \mathcal{U}: rec(u) \neq \emptyset|}\sum_{u\in\mathcal{U}\wedge rec(u)\neq\emptyset} \text{metric}(u)$$

Evaluation metric computation

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

– If a user receives 0 recommendations

User	rec1		rec2	
	#recs	metric(u)	#recs	metric(u)
u_1	5	0.8	5	0.7
u_2	3	0.2	5	0.5
u_3	0	--	5	0.3
u_4	1	1.0	5	0.7
Option a	0.50		0.55	
Option b	0.66		0.55	
User coverage	3/4		4/4	

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)

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

$$\mathrm{RMSE} = \sqrt{\frac{1}{|\mathrm{Te}|} \sum_{(u,i) \in \mathrm{Te}} \left(\tilde{r}(u,i) - r(u,i)\right)^2}$$

MAE = Mean Absolute Error
RMSE = Root Mean Squared

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)

User-item pairs	Real	Rec1	Rec2	Rec3
(u_1, i_1)	5	4	NaN	4
(u_1, i_2)	3	2	4	NaN
(u_1, i_3)	1	1	NaN	1
(u_2, i_1)	3	2	4	NaN
MAE/RMSE, ignoring NaNs		0.75/0.87	2.00/2.00	0.50/0.70
MAE/RMSE, NaNs as 0		0.75/0.87	2.00/2.65	1.75/2.18
MAE/RMSE, NaNs as 3		0.75/0.87	1.50/1.58	0.25/0.50

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 $r_{max} - r_{min}$)
- Average RMSE or MAE to compensate for unbalanced distributions of items or users

$$\text{uMAE} = \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} \frac{1}{|\text{Te}_u|} \sum_{i \in \text{Te}_u} |\tilde{r}(u,i) - r(u,i)|$$

Evaluation metric computation

Variations on metrics:

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

$$\text{nDCG} = \frac{1}{|\mathcal{U}|}\sum_{u}\frac{1}{\text{IDCG}_u^{p_u}}\sum_{p=1}^{p_u}\text{f}_{\text{dis}}\left(\text{rel}\left(u, i_p\right), p\right)$$

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

$$f_{\text{dis}}(x, y) = x/\log y \text{ if } y > 1$$

Evaluation metric computation

Variations on metrics:

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

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

P = Precision (Precision at k)
R = Recall (Recall at k)
MAP = Mean Average Precision

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

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

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

	Tied items at 5			Tied items at 10			Tied items at 50		
Recommender type	Min	Avg	Max	Min	Avg	Max	Min	Avg	Max
UB	15.11	130.64	280.83	14.33	130.52	280.83	7.83	128.20	281.39
SimPop	4.50	235.31	736.50	4.50	234.58	736.5	0	224.02	736.50
SVD	0	0	0	0	0	0	0	0.01	0.67
PureSocial	2	50.75	172	0	50.61	172	0	46.51	172
FriendsPop	10	350.20	1057	9	349.91	1052	0	343.13	1012
Personal	0	1.80	65	0	2.53	65	0	8.86	122.50
Combined	2.80	62.20	205.10	2	61.99	205.1	0.10	57.81	205.10

[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

Alg.	F.W.	nDCG
IBCos	AM LK	0.000414780 0.942192050
IBPea	AM LK	0.005169231 0.924546132
SVD50	AM LK	0.105427298 0.943464094
UBCos50	AM LK	0.169295451 0.948413562
UBPea50	AM LK	0.169295451 0.948413562

(b) RMSE values for LK and MML.

Alg.	F.W.	RMSE
IBCos	LK MML	1.01390931 0.92476162
IBPea	LK MML	1.05018614 0.92933246
SVD50	LK MML	1.01209290 0.93074012
UBCos50	LK MML	1.02545490 0.95358984
UBPea50	LK MML	1.02545490 0.93419026

[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 neigbors (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

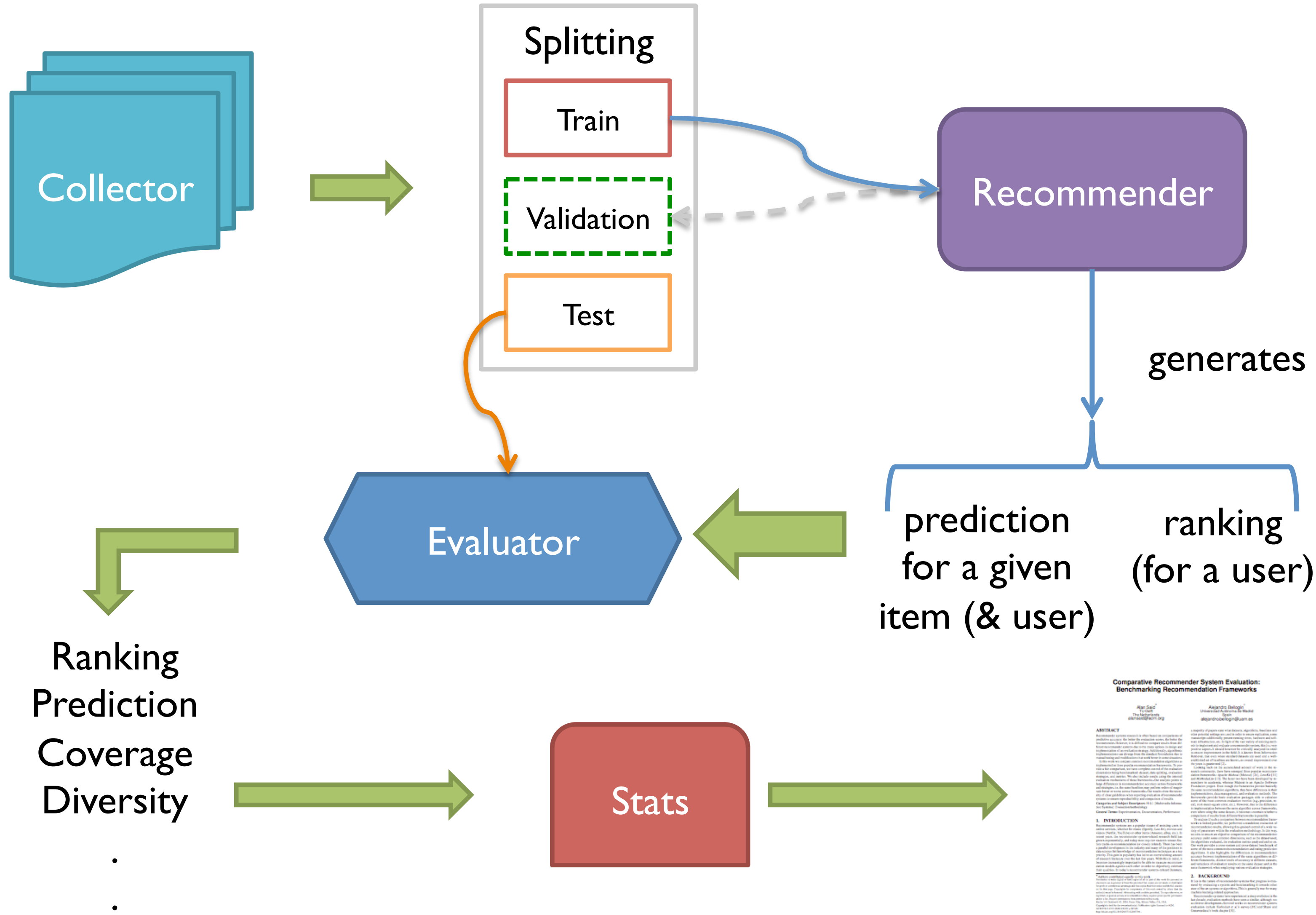
Decisions (implementations) found in some recommendation frameworks:

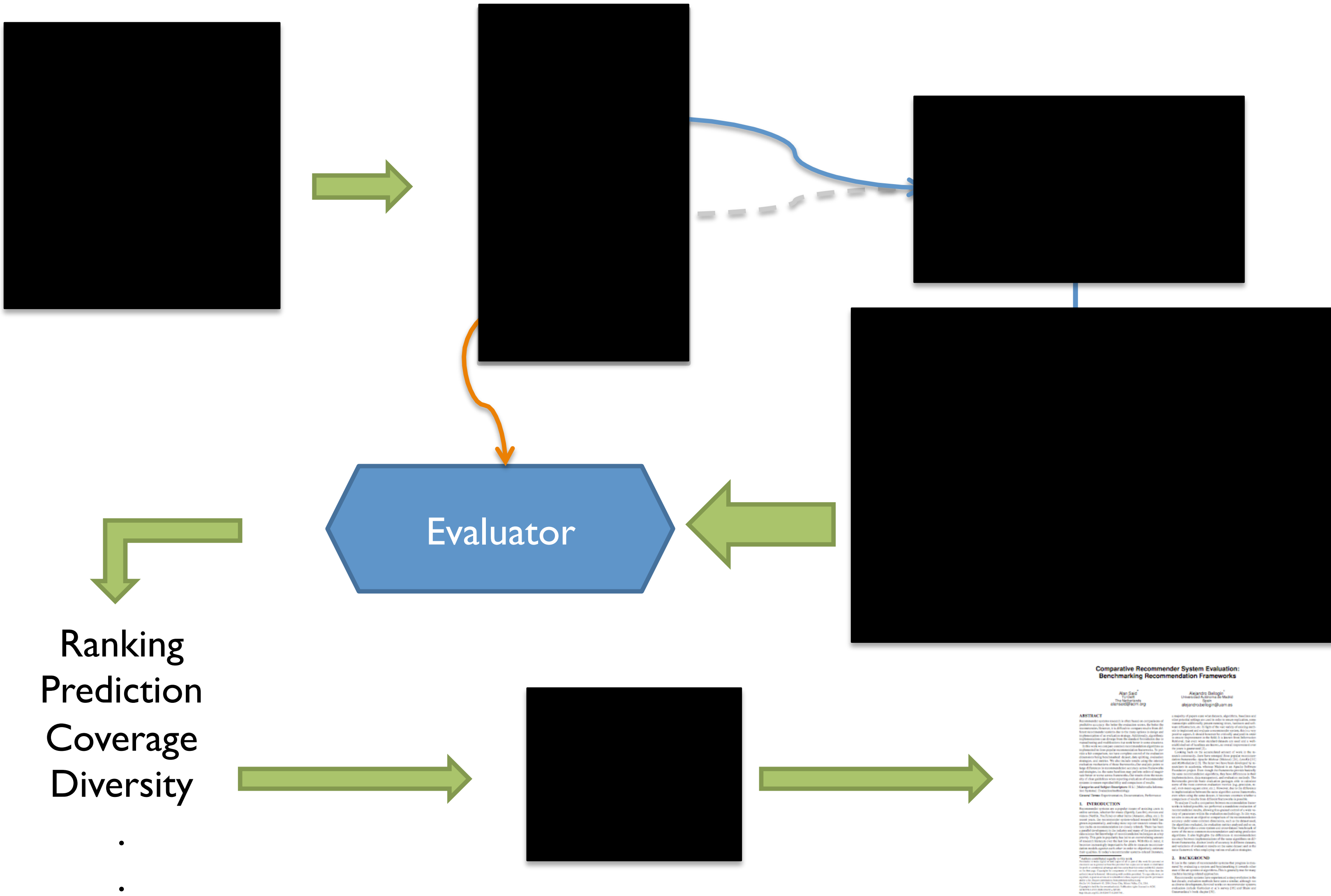
Regarding ties:

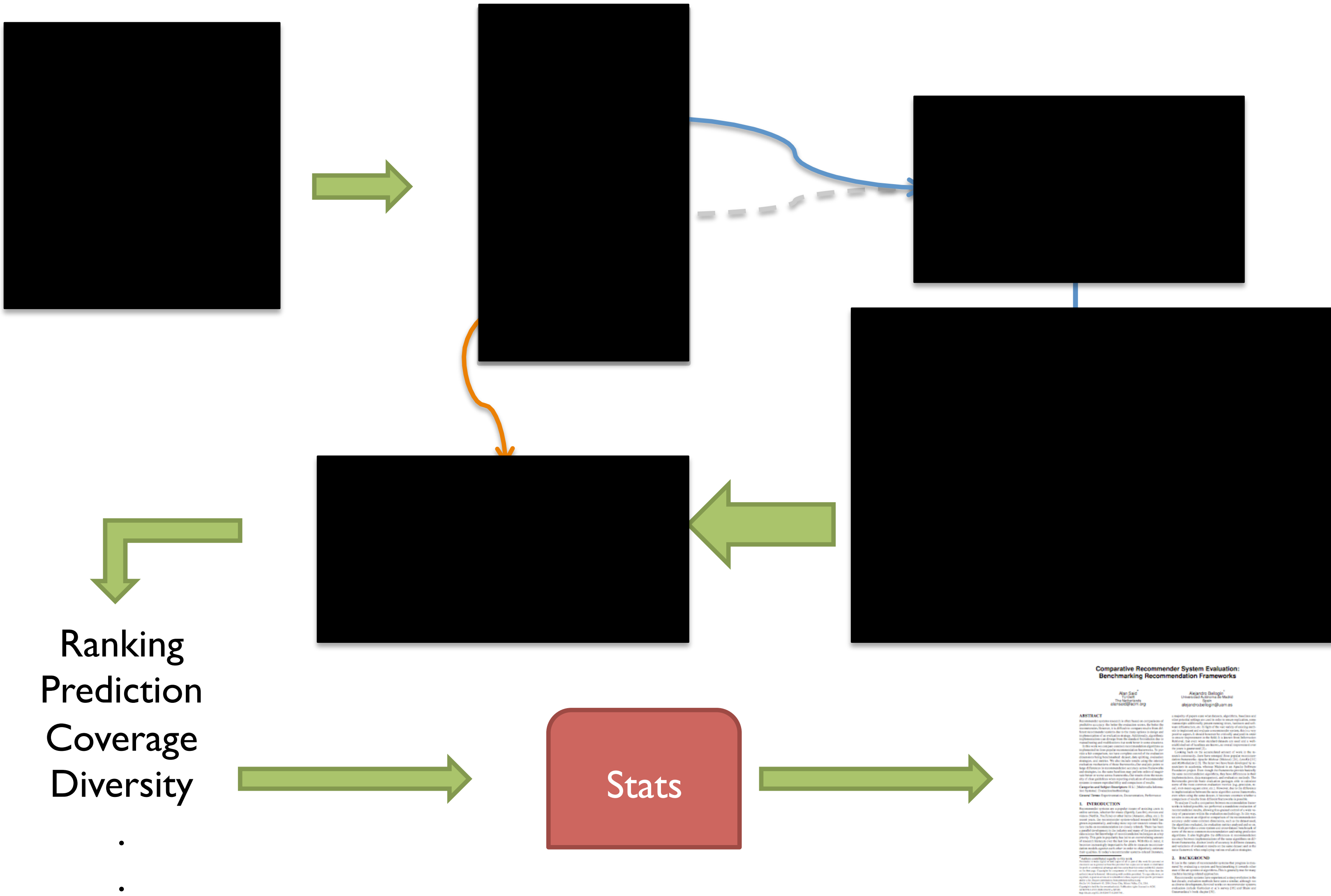
- MML: not deterministic (Recommender.Recommend sorts items by descending score)
- LK: depends on using *predict* package (not deterministic: LongUtils. keyValueComparator only compares scores) or *recommend* (same ranking as returned by algorithm)
- LR: not deterministic (Lists.sortItemEntryListTopK 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


Collector
Splitting
Train
Validation
Test
Recommender
generates
prediction for a given item (& user)
ranking (for a user)
Evaluator
Ranking
Prediction
Coverage
Diversity
Stats
Comparative Recommender System Evaluation: Benchmarking Recommendation Frameworks


Evaluator
Ranking
Prediction
Coverage
Diversity
.
.
.
Comparative Recommender System Evaluation:
Benchmarking Recommendation Frameworks


Ranking
Prediction
Coverage
Diversity
Stats
Comparative Recommender System Evaluation:
Benchmarking Recommendation Frameworks

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

Editorial: ACM TOMS Replicated Computational Results Initiative

MICHAEL A. HEROUX, Sandia National Laboratories

The scientific community relies on the peer review process for assuring the quality of published material, the goal of which is to build a body of work we can trust. Computational journals such as the *ACM Transactions on Mathematical Software* (TOMS) use this process for rigorously promoting the clarity and completeness of content, and citation of prior work. At the same time, it is unusual to independently confirm computational results.

ACM TOMS has established a *Replicated Computational Results* (RCR) review process as part of the manuscript peer review process. The purpose is to provide independent confirmation that results contained in a manuscript are replicable. Successful completion of the RCR process awards a manuscript with the Replicated Computational Results Designation.

This issue of ACM TOMS contains the first [Van Zee and van de Geijn 2015] of what we anticipate to be a growing number of articles to receive the RCR designation, and the related RCR reviewer report [Willenbring 2015]. We hope that the TOMS RCR process will serve as a model for other publications and increase the confidence in and value of computational results in TOMS articles.

Replicated Computational Results (RCR) Report for "BLIS: A Framework for Rapidly Instantiating BLAS Functionality"

JAMES M. WILLENBRING, Sandia National Laboratories

"BLIS: A Framework for Rapidly Instantiating BLAS Functionality" by Field G. Van Zee and Robert A. van de Geijn (see: http://dx.doi.org/10.1145/2764454) includes single-platform BLIS performance results for both level-2 and level-3 operations that is competitive with OpenBLAS, ATLAS, and Intel MKL. A detailed description of the configuration used to generate the performance results was provided to the reviewer by the authors. All the software components used in the comparison were reinstalled and new performance results were generated and compared to the original results. After completing this process, the published results are deemed replicable by the reviewer.

1. INTRODUCTION

The results replication effort for BLIS: A Framework for Rapidly Instantiating BLAS Functionality was focused on Section 7 of the manuscript, which provides performance comparisons for a number of level-2 and level-3 BLIS operations against BLAS operations in the MKL, ATLAS, and OpenBLAS libraries. The authors granted the reviewer access to the machine (described in Section 7.1) on which the results were generated. This machine was also used to generate all of the replicated results.

2. REPLICATING THE RESULTS

The RCR process consisted of installing the same four libraries used to produce the original performance results:

—MKL 11.0 Update 4,
—ATLAS 3.10.1,
—OpenBLAS 0.2.6,
—BLIS 0.1.0-20.

Replication and Reproducibility in RecSys: Next steps?

- We should agree on standard implementations, parameters, instantiations, …
- Replicable badges for journals / conferences

Reproducible IR

We are happy to announce a **Reproducible IR Research Track** for ECIR 2015. F
research to be reliable, referenceable and extensible for the future. Experimental
results can be tested and generalized by peers. This track specifically invites sub

REPRODUCIBILITY INITIATIVE

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

Comparative Recommender System Evaluation: Benchmarking Recommendation Frameworks

Alan Said[*]
TU-Delft
The Netherlands
alansaid@acm.org

Alejandro Bellogín[*]
Universidad Autónoma de Madrid
Spain
alejandro.bellogin@uam.es

Unfolding Off-the-shelf IR Systems for Reproducibility

Emanuele Di Buccio
Dept. Information Engineering
University of Padua, Italy
dibuccio@dei.unipd.it

Giorgio Maria Di Nunzio
Dept. Information Engineering
University of Padua, Italy
dinunzio@dei.unipd.it

Nicola Ferro
Dept. Information Engineering
University of Padua, Italy
ferro@dei.unipd.it

Donna Harman
National Institute of Standards and Technology (NIST), USA
donna.harman@nist.gov

Maria Maistro
Dept. Information Engineering
University of Padua, Italy
maistro@dei.unipd.it

Gianmaria Silvello
Dept. Information Engineering
University of Padua, Italy
silvello@dei.unipd.it

Using Simulation to Analyze the Potential for Reproducibility

Ben Carterette and Karankumar Sabhnani
{carteret,karans}@udel.edu
Department of Computer and Information Sciences
University of Delaware
Newark, DE 19716

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

ACM RecSys Wiki

mloss.org machine learning open source software

MLcomp

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

- [Armstrong et al, 2009] Improvements That Don't Add Up: Ad-Hoc Retrieval Results Since 1998. CIKM
- [Bellogín et al, 2010] A Study of Heterogeneity in Recommendations for a Social Music Service. HetRec
- [Bellogín et al, 2011] Precision-Oriented Evaluation of Recommender Systems: an Algorithm Comparison. RecSys
- [Bellogín et al, 2013] An Empirical Comparison of Social, Collaborative Filtering, and Hybrid Recommenders. ACM TIST
- [Bellogín et al, 2017] Statistical biases in Information Retrieval metrics for recommender systems. Information Retrieval Journal
- [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
- [Cremonesi et al, 2010] Performance of Recommender Algorithms on Top-N Recommendation Tasks. RecSys
- [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

- [Ge et al, 2010] Beyond accuracy: evaluating recommender systems by coverage and serendipity. RecSys
- [Gorla et al, 2013] Probabilistic Group Recommendation via Information Matching. WWW
- [Herlocker et al, 2004] Evaluating Collaborative Filtering Recommender Systems. ACM Transactions on Information Systems
- [Jambor & Wang, 2010] Goal-Driven Collaborative Filtering. ECIR
- [Knijnenburg et al, 2011] A Pragmatic Procedure to Support the User-Centric Evaluation of Recommender Systems. RecSys
- [Koren, 2008] Factorization Meets the Neighborhood: a Multifaceted Collaborative Filtering Model. KDD
- [Lathia et al, 2010] Temporal Diversity in Recommender Systems. SIGIR
- [Li et al, 2010] Improving One-Class Collaborative Filtering by Incorporating Rich User Information. CIKM
- [Pu et al, 2011] A User-Centric Evaluation Framework for Recommender Systems. RecSys
- [Said & Bellogín, 2014] Comparative Recommender System Evaluation: Benchmarking Recommendation Frameworks. RecSys
- [Sakai, 2014] Statistical reform in Information Retrieval? SIGIR Forum

References and Additional reading

- [Schein et al, 2002] Methods and Metrics for Cold-Start Recommendations. SIGIR
- [Shani & Gunawardana, 2011] Evaluating Recommendation Systems. Recommender Systems Handbook
- [Steck & Xin, 2010] A Generalized Probabilistic Framework and its Variants for Training Top-k Recommender Systems. PRSAT
- [Tikk et al, 2014] Comparative Evaluation of Recommender Systems for Digital Media. IBC
- [Vargas & Castells, 2011] Rank and Relevance in Novelty and Diversity Metrics for Recommender Systems. RecSys
- [Weng et al, 2007] Improving Recommendation Novelty Based on Topic Taxonomy. WI-IAT
- [Yin et al, 2012] Challenging the Long Tail Recommendation. VLDB
- [Zhang & Hurley, 2008] Avoiding Monotony: Improving the Diversity of Recommendation Lists. RecSys
- [Zhang & Hurley, 2009] Statistical Modeling of Diversity in Top-N Recommender Systems. WI-IAT
- [Zhou et al, 2010] Solving the Apparent Diversity-Accuracy Dilemma of Recommender Systems. PNAS
- [Ziegler et al, 2005] Improving Recommendation Lists Through Topic Diversification. WWW