# Replicable Evaluation of Recommender Systems

Alejandro Bellogín (Universidad Autónoma de Madrid, Spain)
Alan Said (Recorded Future, Sweden)

Tutorial at ACM RecSys 2015

UNIVERSIDAD AUTONOMA DE MADRID

Recorded Future
CREATING AN INSIGHTFUL WORLD

# Stephansdom

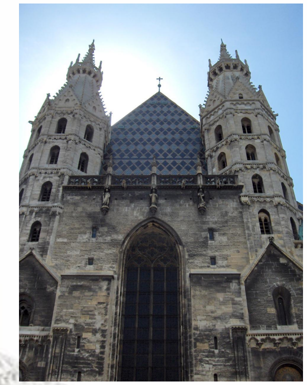
# Stephansdom

# Stephansdom

# Stephansdom

# Stephansdom

# Stephansdom



# #EVALTUT

# Outline

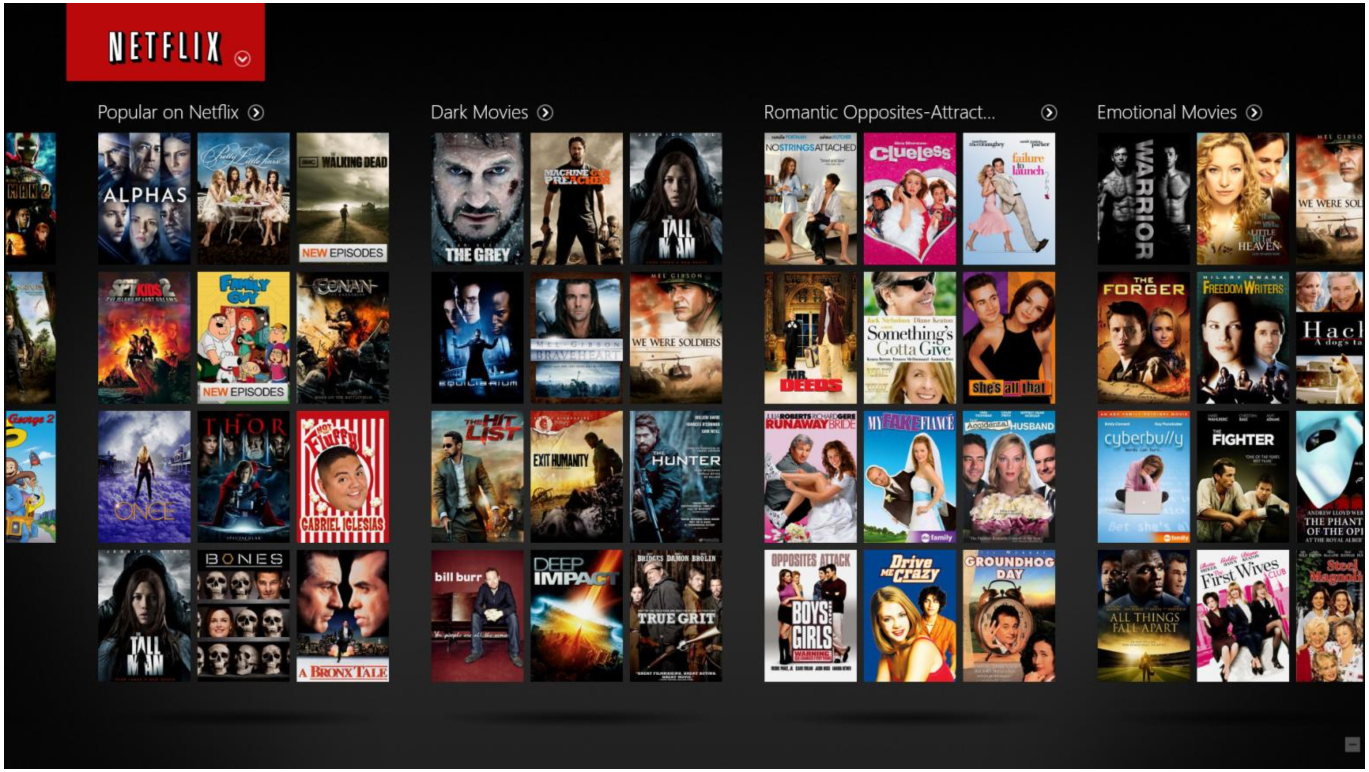
- Background and Motivation [10 minutes]
- Evaluating Recommender Systems [20 minutes]
- Replicating Evaluation Results [20 minutes]
- Replication by Example [20 minutes]
- Conclusions and Wrap-up [10 minutes]
- Questions [10 minutes]

# Outline

- **Background and Motivation**
- Evaluating Recommender Systems
- Replicating Evaluation Results
- Replication by Example
- Conclusions and Wrap-up
- Questions

# Background

- A <u>recommender system</u> aims to find and suggest items of **likely interest** based on the **users' preferences**



# Background

- A <u>recommender system</u> aims to find and suggest items of **likely interest** based on the **users' preferences**


# Background

- A recommender system aims to find and suggest items of **likely interest** based on the **users' preferences**

- Examples:
  - **Netflix**: TV shows and movies
  - **Amazon**: products
  - **LinkedIn**: jobs and colleagues
  - **Last.fm**: music artists and tracks
  - **Facebook**: friends

# Background

- Typically, the interactions between user and system are recorded in the form of ratings
  - But also: clicks (implicit feedback)
- This is represented as a user-item matrix:

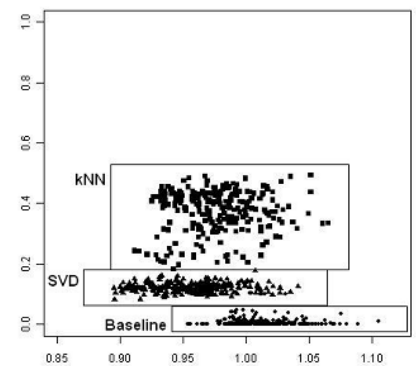
| | $i_1$ | … | $i_k$ | … | $i_m$ |
|---|---|---|---|---|---|
| $u_1$ | ★★★★★ | | ★★★★★ | | ★☆☆☆☆ |
| ⋮ | | | | | |
| $u_j$ | ★★★☆☆ | | ? | | ★★☆☆☆ |
| ⋮ | | | | | |
| $u_n$ | ★★☆☆☆ | | ★★★★☆ | | ★★☆☆☆ |

# Motivation

- Evaluation is an integral part of any experimental research area
- It allows us to compare methods…

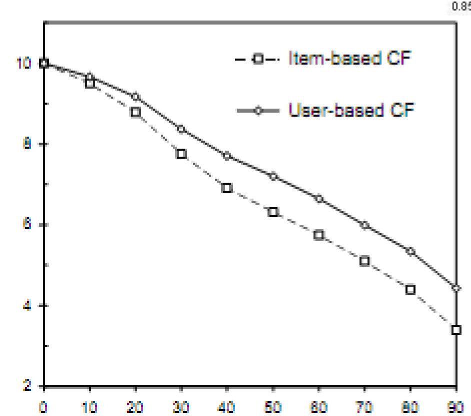
| Methods | MAP | Gain in UCF | MPR % | Gain in UCF % |
|---|---|---|---|---|
| UCF | 0.0513 | - | 28.5 | - |
| UCFWithCT | 0.0856 | 0.0343 | 18.1 | 10.4 |
| UCFWithCT +SchKW | 0.1022 | 0.0509 | 15.4 | 13.1 |
| UCFWithCT +SchKW+CT | 0.1037 | **0.0524** | 15.0 | **13.5** |

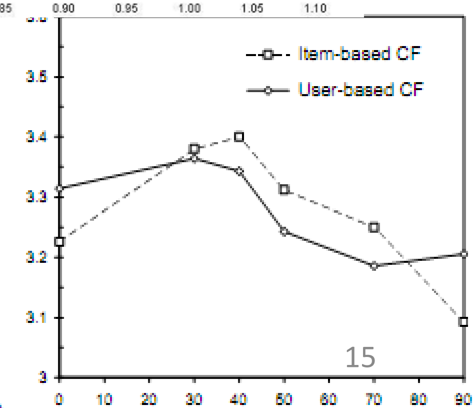
| | MovieLens | | | LastFM | | |
|---|---|---|---|---|---|---|
| K | r@5 | r@10 | r@20 | r@5 | r@10 | r@20 |
| 1 | 0.529 | 0.691 | 0.84 | 0.541 | 0.643 | 0.737 |
| 2 | 0.539 | 0.699 | 0.846 | 0.543 | 0.657 | 0.752 |
| 5 | 0.531 | 0.690 | 0.841 | 0.544 | 0.658 | 0.749 |
| 10 | 0.525 | 0.691 | 0.839 | 0.530 | 0.639 | 0.736 |
| 25 | 0.525 | 0.689 | 0.838 | 0.537 | 0.642 | 0.737 |




| Model | 50 factors | 100 factors | 200 factors |
|---|---|---|---|
| SVD | 0.9046 | 0.9025 | 0.9009 |
| Asymmetric-SVD | 0.9037 | 0.9013 | 0.9000 |
| SVD++ | 0.8952 | 0.8924 | 0.8911 |

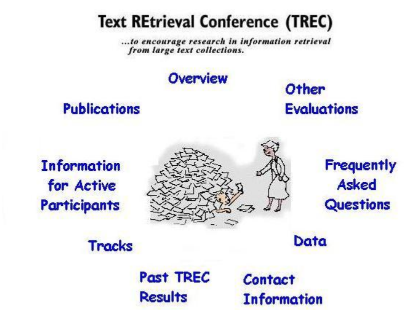
| | Lastfm | | YahooMusic | | BookCrossing | |
|---|---|---|---|---|---|---|
| SVDR | 0.113 | 0.083 | 0.237 | 0.207 | 0.078 | 0.063 |
| ASVDR | 0.114 | 0.087 | 0.237 | 0.210 | 0.078 | 0.062 |
| NMFR | 0.114 | 0.090 | 0.218 | 0.189 | 0.073 | 0.054 |
| ANMFR | 0.110 | 0.089 | 0.211 | 0.190 | 0.071 | 0.056 |
| SVDN | 0.002 | 0.003 | 0.001 | 0.001 | 0.005 | 0.003 |
| ASVDN | 0.003 | 0.002 | 0.007 | 0.005 | 0.005 | 0.003 |
| HSVD | **0.180** | **0.158** | **0.258** | **0.227** | 0.075 | 0.065 |



# Motivation

- Evaluation is an integral part of any experimental research area
- It allows us to compare methods…
- … and identify winners (in competitions)



# Motivation

A proper evaluation culture allows advance the field

CIKM 2009
Hong Kong, China November 2–6, 2009

**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!

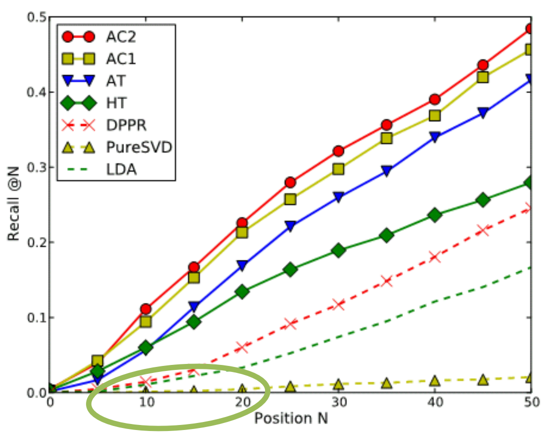
# Motivation

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

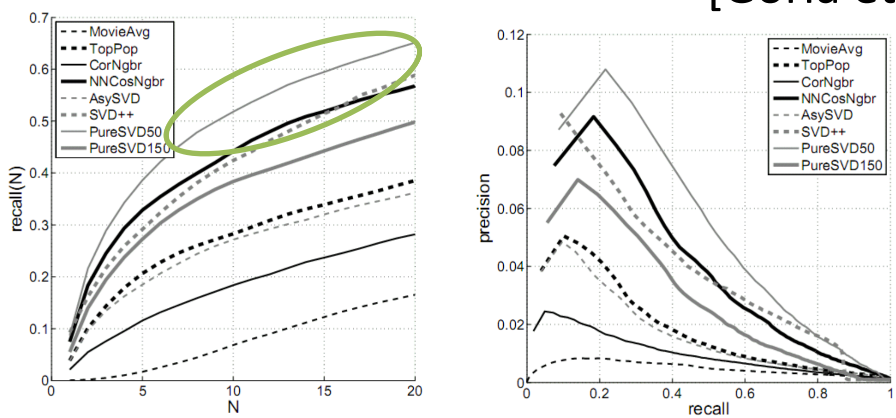
- Dataset
- Algorithm
- Evaluation metric

| | 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]




Movielens 1M
[Yin et al, 2012]




(a) recall (b) precision vs recall

Movielens 1M
[Cremonesi et al, 2010]

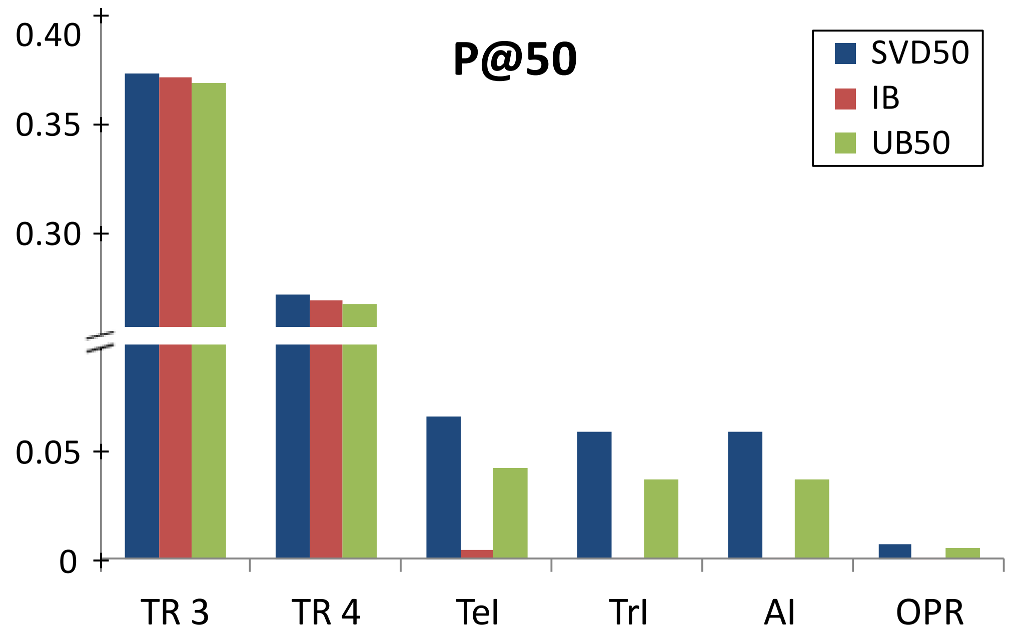
| | Baseline(Test) |
|---|---|
| MAP | 0.447 |
| MRR | 0.889 |
| NDCG@1( | 0.720 |
| NDCG@5 | 0.570 |
| NDCG@3 | 0.447 |

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

# Motivation

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

- Dataset
- Algorithm
- Evaluation metric




[Bellogín et al, 2011]

# Motivation

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

– Dataset
– Algorithm
– Evaluation metric


We need to understand why this happens

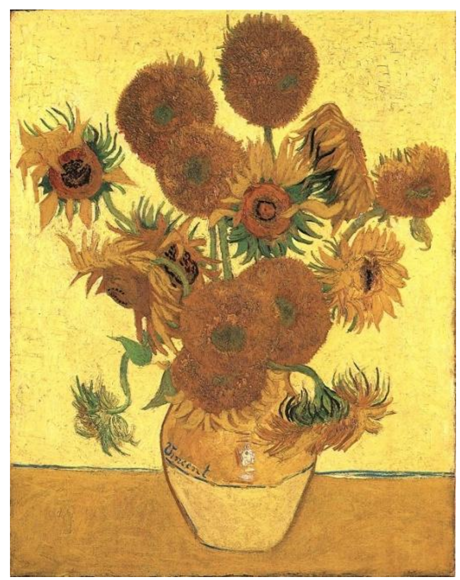
# In this tutorial

- We will present the basics of evaluation
  - Accuracy metrics: error-based, ranking-based
  - Also coverage, diversity, and novelty

- We will focus on <u>replication</u> and <u>reproducibility</u>
  - Define the context
  - Present typical problems
  - Propose some guidelines

# Replicability

- Why do we need to replicate?



# Reproducibility

Why do we need to reproduce?

Because these two are not the same

# NOT in this tutorial

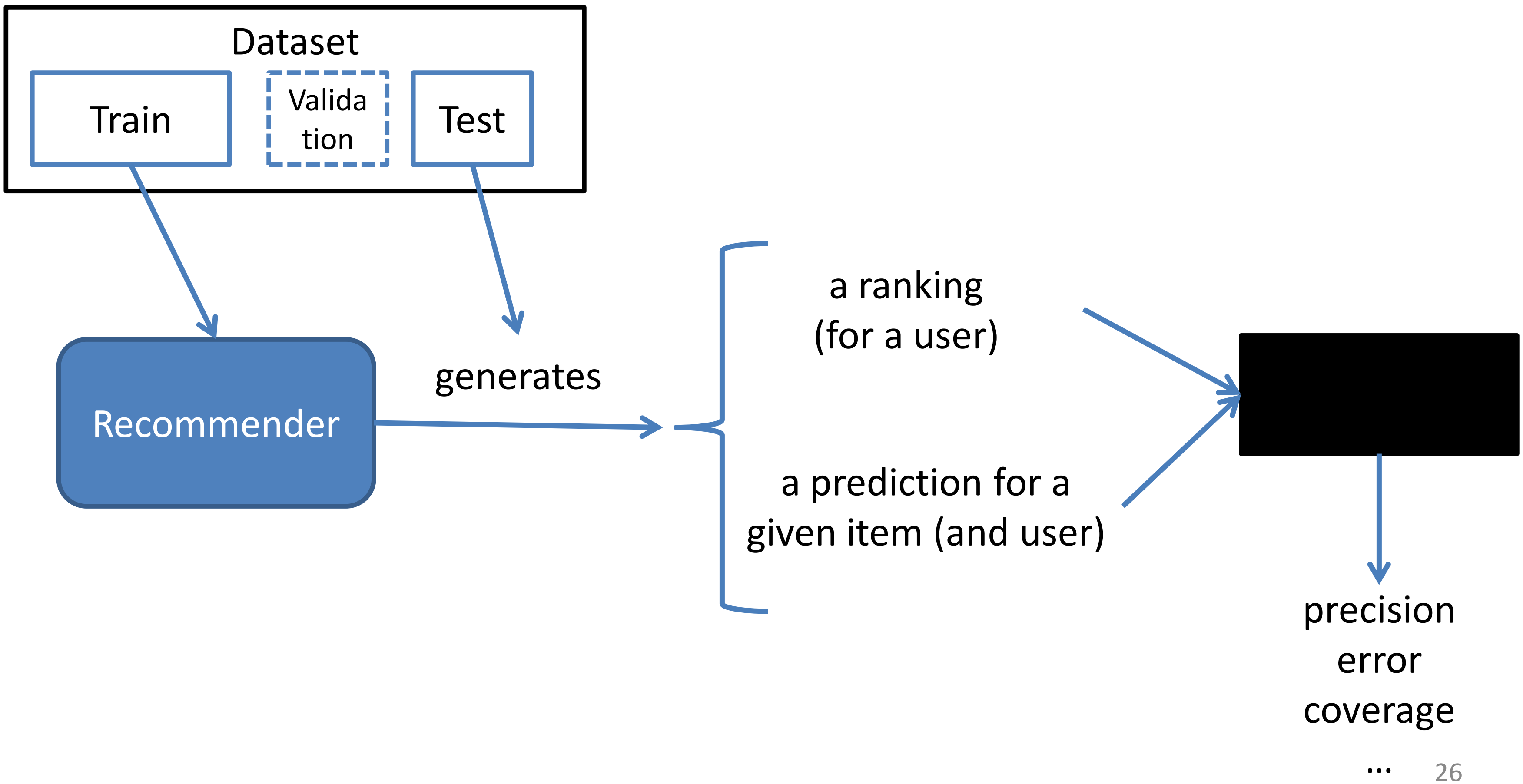
- In-depth analysis of evaluation metrics
  - See chapter 9 on handbook [Shani & Gunawardana, 2011]
- Novel evaluation dimensions
  - See tutorials at WSDM ’14 and SIGIR ‘13 on diversity and novelty
- User evaluation
  - See tutorial at RecSys 2012
- Comparison of evaluation results in research
  - See RepSys workshop at RecSys 2013
  - See [Said & Bellogín 2014]

# Outline

- Background and Motivation
- **Evaluating Recommender Systems**
- Replicating Evaluation Results
- Replication by Example
- Conclusions and Wrap-up
- Questions

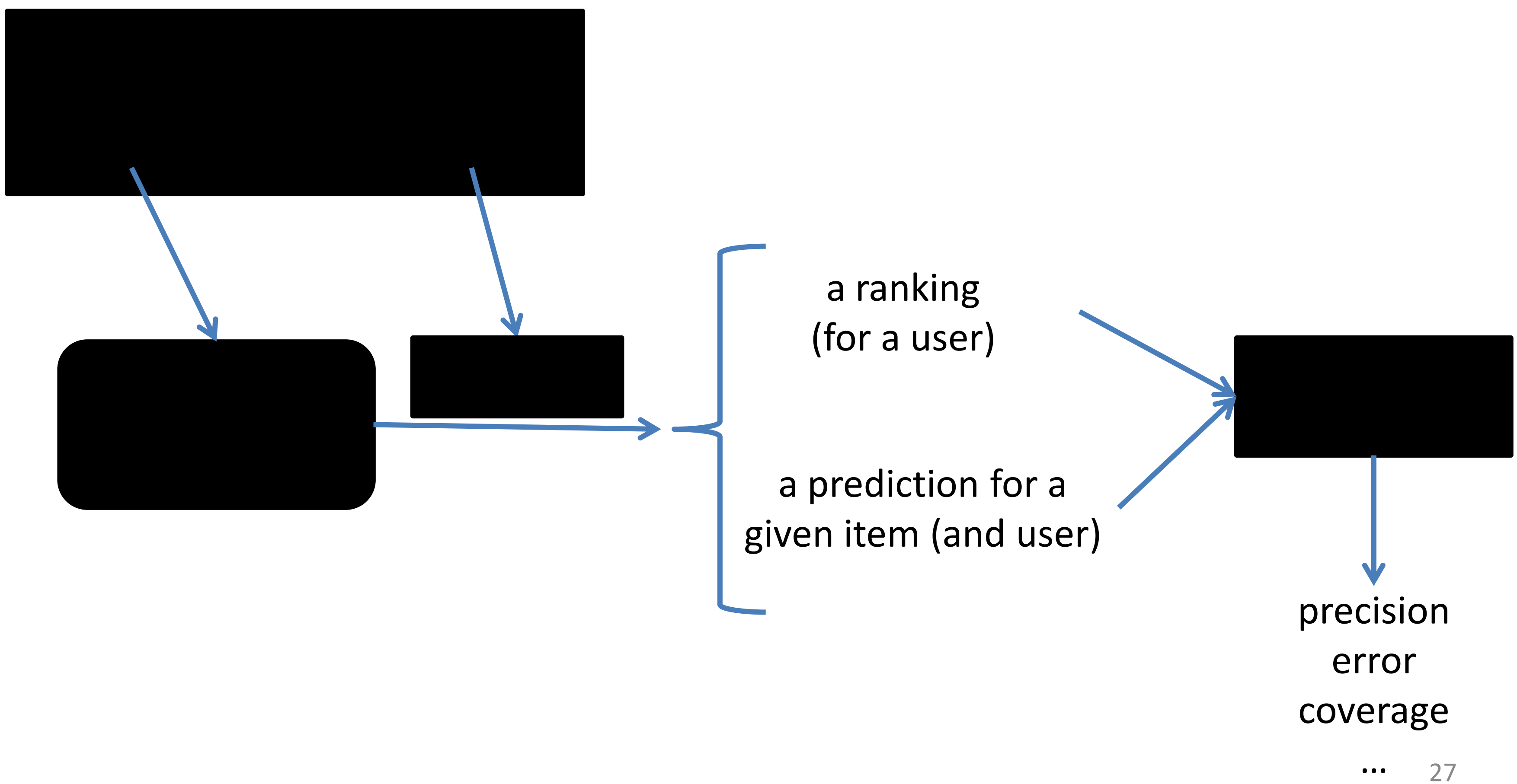
# Recommender Systems Evaluation

## Typically: as a black box

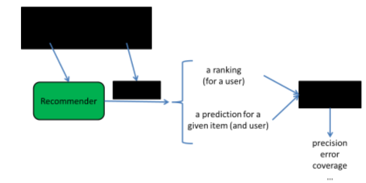


# Recommender Systems Evaluation

## The reproducible way: as black boxes



# Recommender as a black box



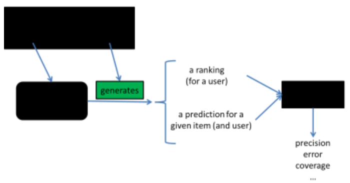

## What do you do when a recommender cannot predict a score?

### This has an impact on coverage

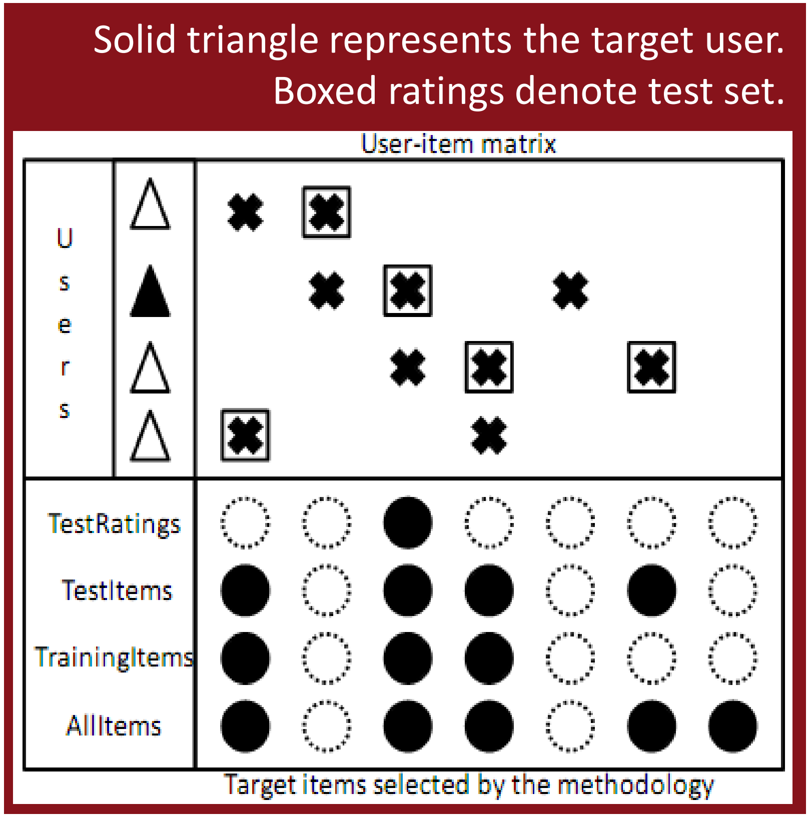
| Alg. | F.W. | Time (sec.) | RMSE | nDCG@10 RPN | nDCG@10 UT | User cov.(%) RPN | User cov.(%) UT | Cat. cov.(%) RPN | Cat. cov.(%) UT |
|---|---|---|---|---|---|---|---|---|---|
| IBCos | AM | 238 | 1.041 | 0.003 | 0.501 | 98.16 | 100 | 99.71 | 99.67 |
| | LK | 44 | 0.953 | 0.199 | 0.618 | 98.16 | 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 |

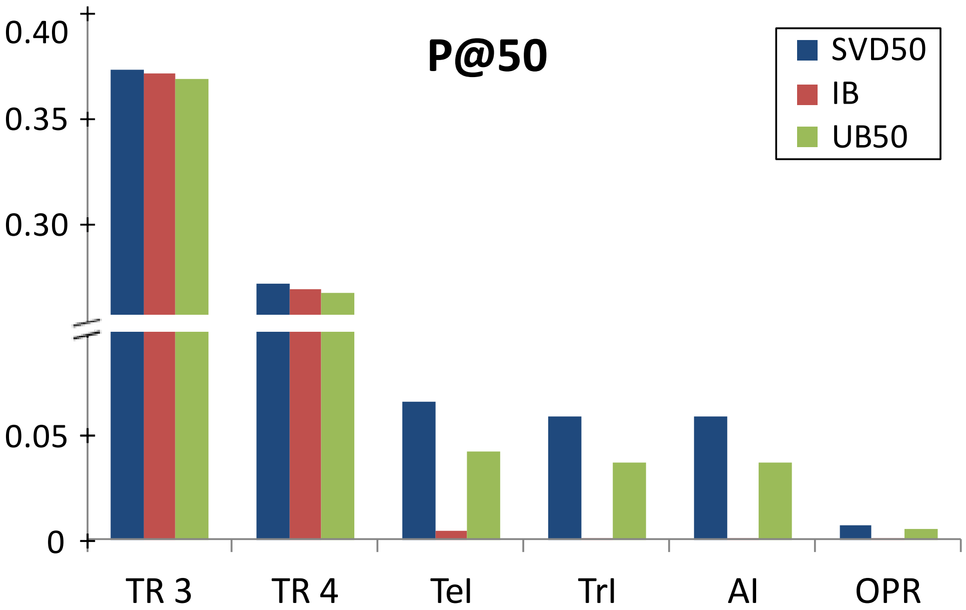
[Said & Bellogín, 2014]

# Candidate item generation as a black box

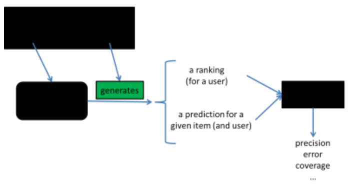



How do you select the candidate items to be ranked?



# Candidate item generation as a black box

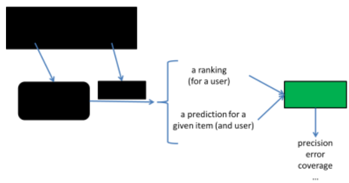



## How do you select the candidate items to be ranked?

| Alg. | F.W. | Time (sec.) | RMSE | nDCG@10 RPN | nDCG@10 UT | User cov.(%) RPN | User cov.(%) UT | Cat. cov.(%) RPN | Cat. cov.(%) 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]

# Evaluation metric computation as a black box




What do you do when a recommender cannot predict a score?

- This has an impact on coverage
- It can also affect **error-based metrics**

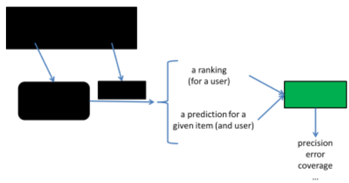
$$\mathrm{MAE} = \frac{1}{|\mathrm{Te}|} \sum_{(u,i) \in \mathrm{Te}} |\tilde{r}(u,i) - r(u,i)|$$

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

MAE = Mean Absolute Error

RMSE = Root Mean Squared Error

# Evaluation metric computation as a black box

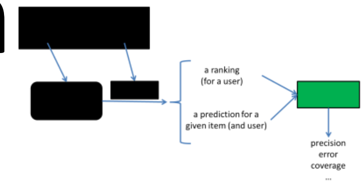



What do you do when a recommender cannot predict a score?

- This has an impact on coverage
- It can also affect 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 |

# Evaluation metric computation as a black box




## Using internal evaluation methods in Mahout (AM), LensKit (LK), and MyMediaLite (MML)

(a) nDCG for AM and LK

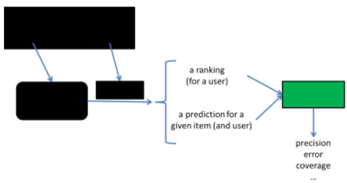
| Alg. | F.W. | nDCG |
|---|---|---|
| IBCos | AM | 0.000414780 |
| | LK | 0.942192050 |
| IBPea | AM | 0.005169231 |
| | LK | 0.924546132 |
| SVD50 | AM | 0.105427298 |
| | LK | 0.943464094 |
| UBCos50 | AM | 0.169295451 |
| | LK | 0.948413562 |
| UBPea50 | AM | 0.169295451 |
| | LK | 0.948413562 |

(b) RMSE values for LK and MML.

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

[Said & Bellogín, 2014]

# Evaluation metric computation as a black box



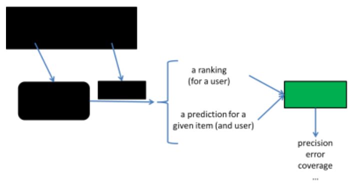

Variations on metrics:

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

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

# Evaluation metric computation as a black box




## Variations on metrics:

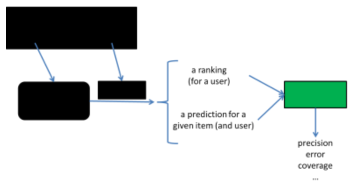
**nDCG** has at least two discounting functions (<u>linear</u> and <u>exponential</u> decay)

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

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

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

# Evaluation metric computation as a black box




## Variations on metrics:

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

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

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

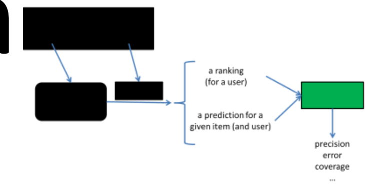
$$\mathrm{MAP} = \frac{1}{|\mathcal{U}|} \sum_{u} \frac{1}{|\mathrm{Rel}_u|} \sum_{i \in \mathrm{Rel}_u} \mathrm{P@rank}(u, i)$$

P = Precision (Precision at k)

R = Recall (Recall at k)

MAP = Mean Average Precision

# Evaluation metric computation as a black box



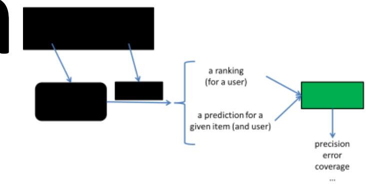

If ties are present in the ranking scores, results may depend on the implementation

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 as a black box




Not clear how to measure diversity/novelty in offline experiments (directly measured in online experiments):

- Using a taxonomy (items about novel topics) [Weng et al, 2007]
- New items over time [Lathia et al, 2010]
- Based on entropy, self-information and Kullback-Leibler divergence [Bellogín et al, 2010; Zhou et al, 2010; Filippone & Sanguinetti, 2010]

# Recommender Systems Evaluation: Summary

- Usually, evaluation seen as a black box
- The evaluation process involves everything: splitting, recommendation, candidate item generation, and metric computation

- We should agree on standard implementations, parameters, instantiations, …
  - Example: trec_eval in IR

# Outline

- Background and Motivation
- Evaluating Recommender Systems
- **Replicating Evaluation Results**
- Replication by Example
- Conclusions and Wrap-up
- Questions

# Reproducible Experimental Design

- We need to distinguish
  - Replicability
  - Reproducibility
- Different aspects:
  - Algorithmic
  - Published results
  - Experimental design
- Goal: have a <u>reproducible experimental environment</u>

# Definition: Replicability

To *copy* something

- The results
- The data
- The approach

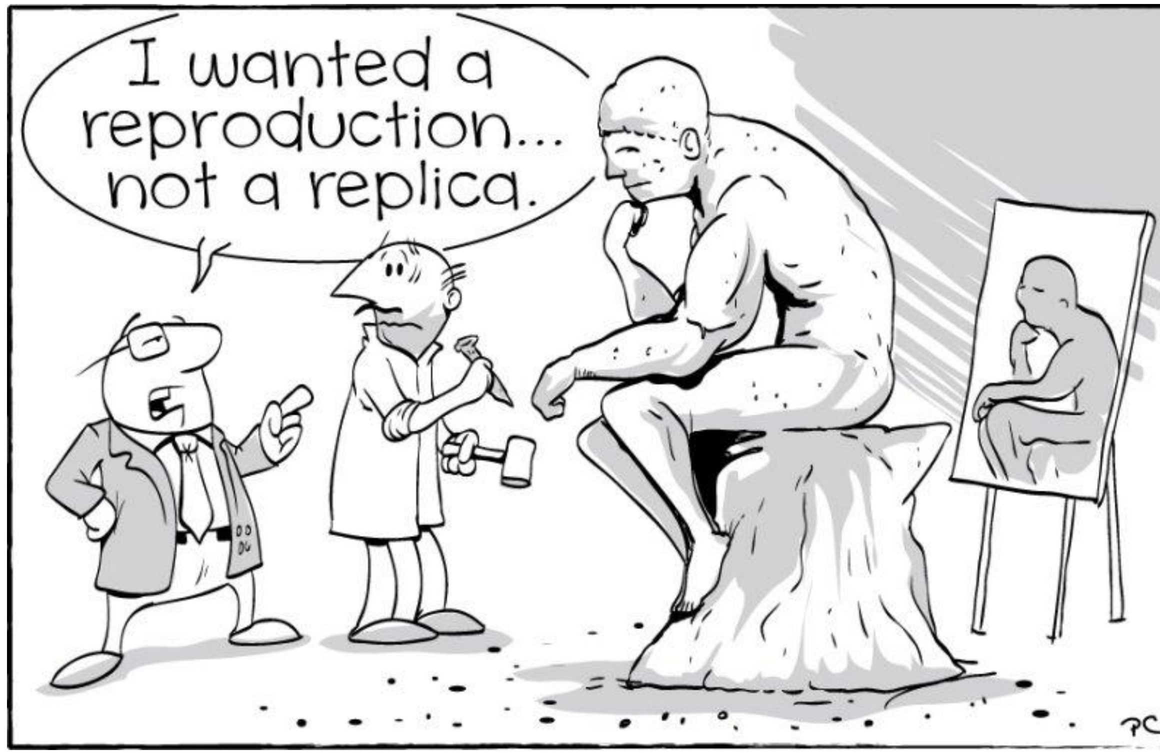
Being able to evaluate in the same setting and obtain the same results

# Definition: Reproducibility

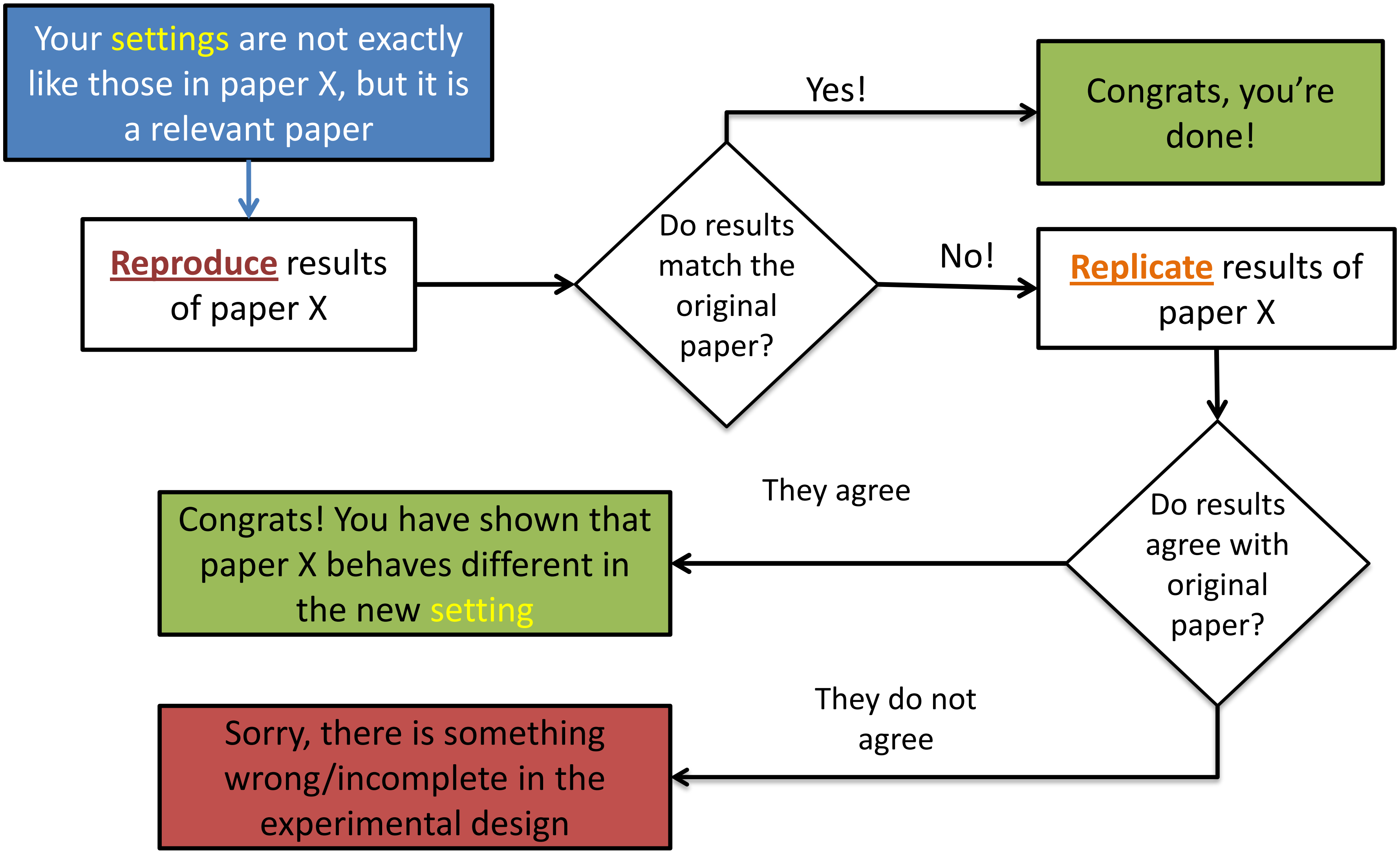
To <u>recreate</u> something

- 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



# 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

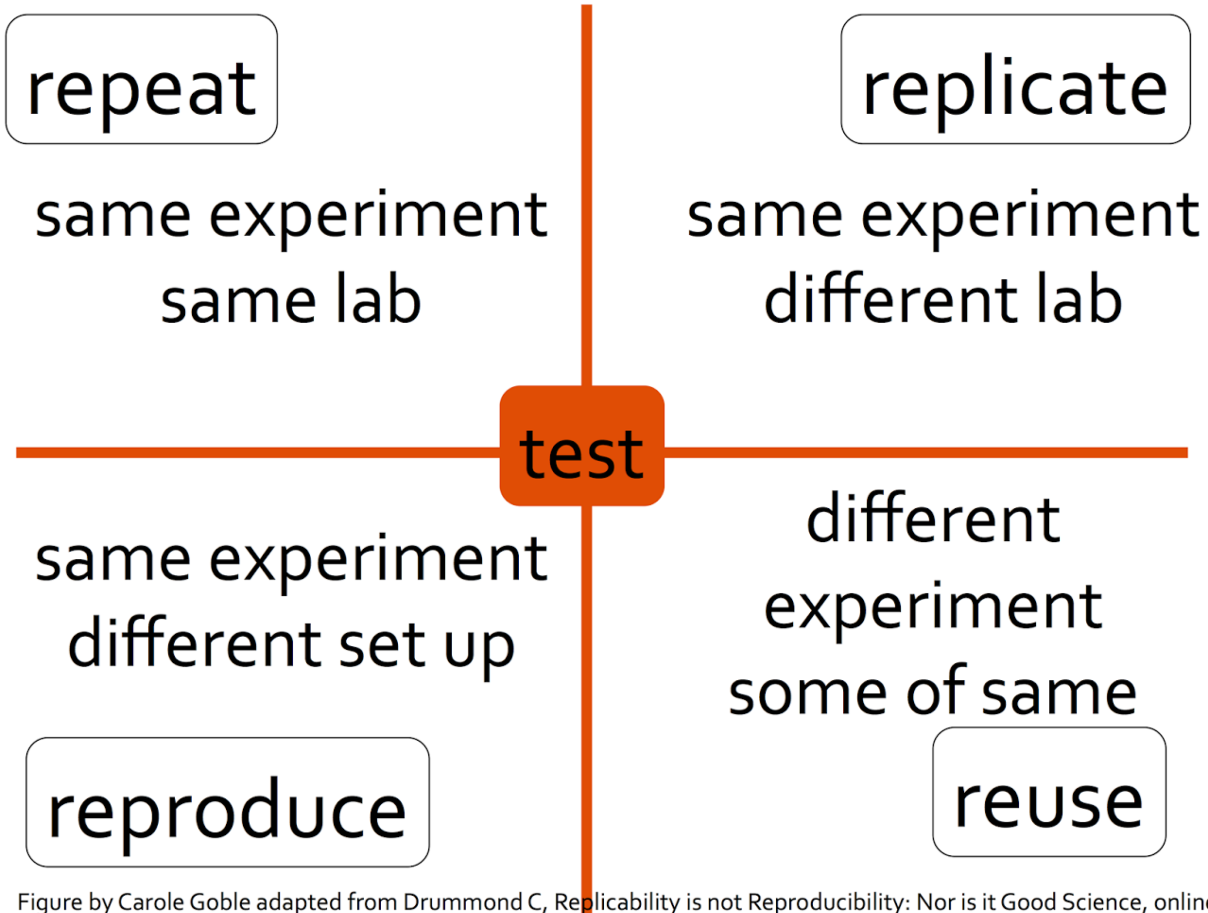



Figure by Carole Goble adapted from Drummond C, Replicability is not Reproducibility: Nor is it Good Science, online and Peng RD, Reproducible Research in Computational Science *Science 2 Dec 2011: 1226-1227.*

# Motivation for reproducibility

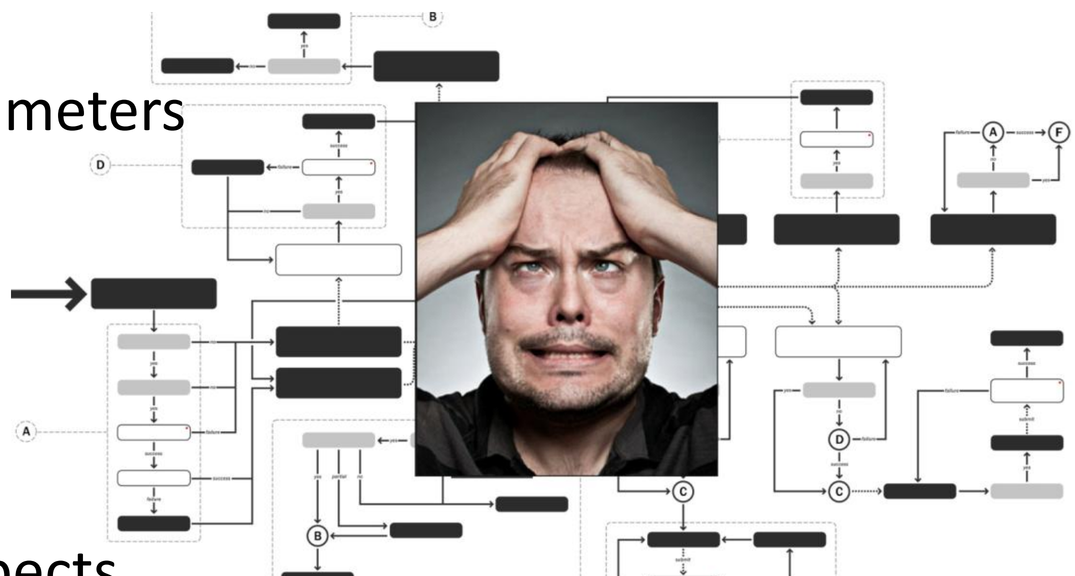
In order to ensure that our experiments, settings, and results are:

- Valid
- Generalizable
- 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**
- Describe process **clearly**
- Use **standards**
- Publish code (nobody expects you to be an awesome developer, you're a researcher)



# 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 in recommendation
  - Validity of results (replicability)
  - Comparability of results (reproducibility)
  - Validity of experimental setup (repeatability)

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

# Outline

- Background and Motivation
- Evaluating Recommender Systems
- Replicating Evaluation Results
- **Replication by Example**
- Conclusions and Wrap-up
- Questions

# Replication by Example

- Demo time!
- Check
  - http://www.recommenders.net/tutorial
- Checkout
  - https://github.com/recommenders/tutorial.git

# The things we write

mvn exec:java -Dexec.mainClass="net.recommenders.tutorial.CrossValidation”

```
NDCG@10: 0.0292752140037415
RMSE: 1.108653420946922
P@10: 0.039915164369035125
```

# The things we forget to write

mvn exec:java -Dexec.mainClass="net.recommenders.tutorial.CrossValidation”

```
NDCG@10: 0.0292752140037415
RMSE: 1.108653420946922
P@10: 0.039915164369035125
```

mvn -o exec:java -Dexec.mainClass="net.recommenders.tutorial.CrossValidation"
-Dexec.args=”-u false"

```
NDCG@10: 0.02921891771562769
RMSE: 1.104452226664006
P@10: 0.04091198303287395
```

# The things we forget to write

mvn exec:java -Dexec.mainClass="net.recommenders.tutorial.CrossValidation”

```
NDCG@10: 0.0292752140037415
RMSE: 1.108653420946922
P@10: 0.039915164369035125
```

mvn -o exec:java -Dexec.mainClass="net.recommenders.tutorial.CrossValidation"
-Dexec.args=”-u false"

```
NDCG@10: 0.02921891771562769
RMSE: 1.104452226664006
P@10: 0.04091198303287395
```

mvn -o exec:java -Dexec.mainClass="net.recommenders.tutorial.CrossValidation"
-Dexec.args="-t 4.0"

```
NDCG@10: 0.0292752140037415
RMSE: 1.108653420946922
P@10: 0.033149522799575906
```

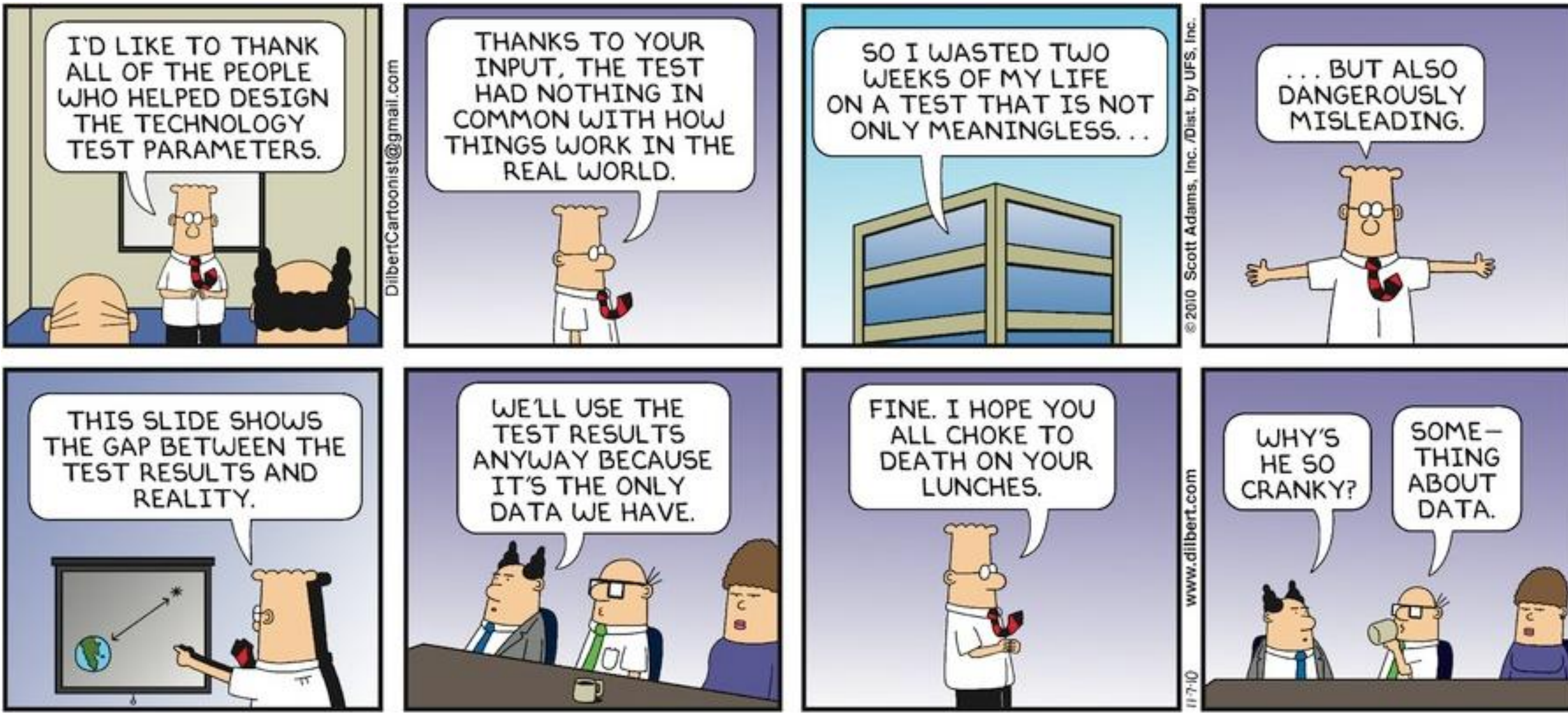
# Outline

- Background and Motivation
- Evaluating Recommender Systems
- Replicating Evaluation Results
- Replication by Example
- **Conclusions and Wrap-up**
- Questions

# 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”

# We must avoid this




From http://dilbert.com/strips/comic/2010-11-07/

# Next steps?

- Agree on standard implementations
- 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.

# Next steps?

- Agree on standard implementations
- 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

# Next steps?

- Agree on standard implementations
- 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

# Next steps?

- Agree on standard implementations
- 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:
    - In Slideshare... soon!

# 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
- [Ben-Shimon et al, 2015] RecSys Challenge 2015 and the YOOCHOOSE Dataset. RecSys
- [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
- [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

# References and Additional reading


- [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
- [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

# References and Additional reading


- [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

# Rank-score (Half-Life Utility)

Using a different discount function, the **rank score** or **half-life utility** metric (Breese et al., 1998; Herlocker et al., 2004; Huang et al., 2006) can be obtained as follows:

$$\mathrm{HL} = 100\left(\sum_u \mathrm{HL}_u^{\max}\right)^{-1} \sum_u \mathrm{HL}_u \,; \quad \mathrm{HL}_u = \sum_{p=1}^{p_u} \frac{\max(\tilde{r}(u, i_p) - d, 0)}{2^{(p-1)/(\alpha-1)}}$$

where $d$ is the default ranking, and $\alpha$ is the half-life utility that represents the rank of the item on the list such that there is a 50% chance that the user will view that item. In (Breese et al., 1998) the authors use a value of $5$ in their experiments, and note that they did not obtain different results with a half-life of $10$.

# Mean Reciprocal Rank

**Mean reciprocal rank** (MRR) favours rankings whose first correct result occurs near the top ranking results (Baeza-Yates and Ribeiro-Neto, 2011). It is defined as follows:

$$\mathrm{MRR} = \sum_{u} \frac{1}{s_r(u)}$$

where $s_r(u)$ is a function that returns the position of the first relevant item obtained for user $u$. This metric is similar to the **average rank of correct recommendation** (ARC) proposed in (Burke, 2004) and to the **average reciprocal hit-rank** (ARHR) defined in (Deshpande and Karypis, 2004).

# Mean Percentage Ranking

Mean Percentage Ranking, which is used in [11] and [4], to measure the user satisfaction of items in an ordered list. Let $rank_{ui}$ be the percentile-ranking of item $i$ within the ordered list of all items for user $u$. $rank_{ui} = 0\%$ means that item $i$ is most preferred by user $u$. The higher ranking (until $rank_{ui} = 100\%$ is reached) indicates that $i$ is predicted to be less desirable for user $u$. The way of calculating the MPR in our experiment setup is as the following: for each actual pair of a user and the purchased item, we randomly select 1000 other items, and produce an ordered list of these items. Then, we keep track of where the actual purchased item is ranked, and calculate the expected percentage ranking for all users and items:

$$MPR = \frac{\sum_{u,i} r_{ui} \times rank_{ui}}{\sum_{u,i} r_{ui}}$$

[Li et al, 2010]

Where $r_{ui}$ is a binary variable indicating whether user $u$ purchases item $i$. It is expected that a randomly produced list would have a MPR of around 50%.

# Global ROC

We use a global ROC (GROC) curve to measure performance when we are allowed to recommend more often to some users than others. GROC curves are constructed in the following manner:

1. Order the predictions $\mathbf{pred}(p_i, m_j)$ in a list by magnitude, imposing an ordering: $(p, m)_k$.
2. Pick $n$, calculate hit/miss rates caused by predicting the top $n$ $(p, m)_k$ by magnitude, and plot the point.

By selecting different $n$ (e.g. incrementing $n$ by a fixed amount) we draw a curve on the graph.

[Schein et al, 2002]

# Customer ROC

Customer ROC (CROC) curves measure performance of a recommender system when we are constrained to recommend the same number of items to each user. Unlike the GROC curve, the CROC curve is not a special case of the ROC curve, though it is constructed in an analogous manner:

```
1.  For each person p_i, order the predictions
pred(p_i, m_j) in a list by magnitude imposing an
ordering:  (m)_k.
2.  Pick n, calculate global hit/miss rates
caused by recommending the top predicted n
movies to each person and plot the point.
```

We vary $n$ as in the GROC case.

[Schein et al, 2002]

In a GROC curve, the perfect recommender will generate a curve with area one, but for the CROC curve this is not the case. To see why, imagine using an omniscient recommender on a data set with three people: person $a$ sees four movies, person $b$ sees two movies, and person $c$ sees six movies. When we recommend four movies to each person, we end up with two false-positives from person $b$, lowering the area of the curve. However, for any particular data set, we can plot the curve and calculate the area of the omniscient recommender in order to facilitate comparison.

# Popularity-stratified recall

Assuming that there are no additional (possibly hidden) factors underlying the missing data mechanism concerning the relevant ratings, we obviously obtain an unbiased estimate for recall on the (unknown) complete data by calculating the *popularity-stratified recall* (for user $u$),

$$\text{recall}_u(k) = \frac{\sum_{i \in S_u^{+,k}} s_i}{\sum_{i \in S_u^{+}} s_i} \tag{12}$$

on the available MNAR data; $S_u^+$ denotes the set of *relevant* items of user $u$; $S_u^{+,k}$ is the subset of relevant items ranked in the top $k$ items based on the predictions of the recommender system; the popularity-stratification weight for each item $i$ is

$$s_i = \frac{1}{p_{\text{obs}}(i)}.$$

$$s_i \propto 1/(N_{\text{obs},i}^{+})^{\gamma/(\gamma+1)}.$$

We consider the case that the probability of observing a relevant rating depends on the popularity of items. We define the popularity of an item by the number $N_{\text{complete},i}^{+}$ of *relevant* ratings it obtained in the (unknown) complete data. Let $N_{\text{obs},i}^{+}$ be the number of relevant ratings observed in the available data; then the probability of observing a relevant rating regarding item $i$ is

$$p_{\text{obs}}(i) = \frac{N_{\text{obs},i}^{+}}{N_{\text{complete},i}^{+}}. \tag{11}$$

Finally, we define $\text{recall}(k) = \sum_u w^u \text{recall}_u(k)$ as the average recall over all users, with normalized weights, $\sum_u w^u = 1$, like in [17]. In our experiments, we choose $w^u \propto \sum_{i \in S_u^+} s_i$, as a generalization of the definition in [7, 17].

[Steck & Xin, 2010]