Widespread Flaws in Offline Evaluation of Recommender Systems

Balázs Hidasi
balazs.h@taboola.com
@balazshidasi

Ádám Czapp
adam-tibor.c@taboola.com
@adam_czapp

a Taboola company
GRAVITY
Research & Development

Steps of Offline Evaluation

1. **Task definition:** the task defines the evaluation setup, not vica versa
2. **Decide on evaluation methodology & metrics:** *behavior prediction* or *interaction prediction;* Recall, MRR, AUC, etc.
3. **Choose dataset(s):** not every dataset is appropriate for every task
4. **Preprocess the data:** reduce noise & tailor it towards the task
5. **Train / test split**: methodology should be suitable for the task, but must not have information leaks between train & test or from future data
6. **Measure metrics:** rank all items and compute metrics

Evaluation Flaws

Dataset–task mismatch (3.)

Common causes

- Not enough public datasets are available for a certain task
- Reusing evaluation setups of other papers without validation

Examples

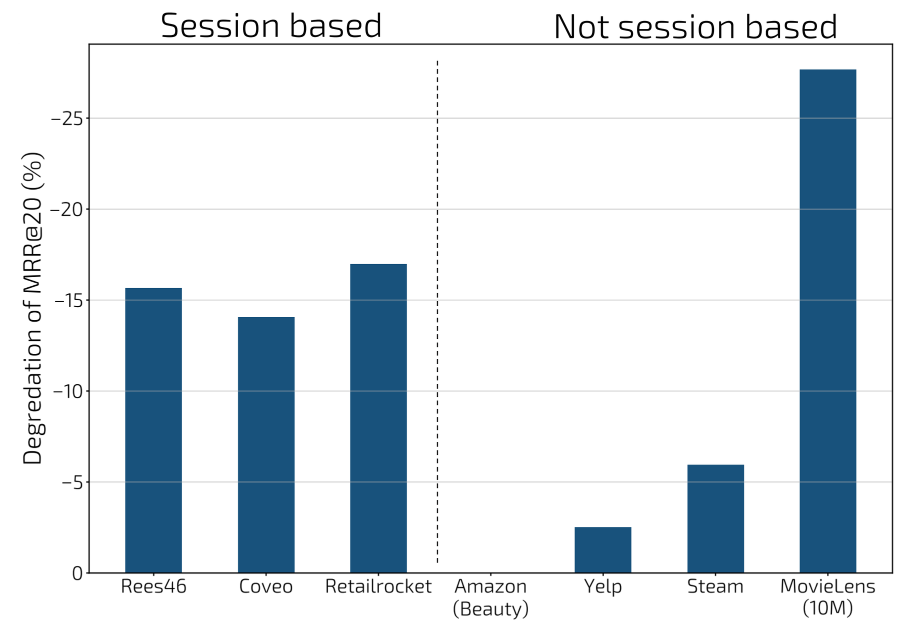
- Sequential recommendation on non-sequential data
- CTR prediction on organic behavior data

Sequential recommendation using non-sequential data

1. *Ratings obscure the sequentiality of user behavior*
   - Rating is disjoint from the time of consumption
2. *Long term behavior with low resolution (infrequent events) hides sequentiality*
   - E. g.: Buy sequence: TV → Cheese → Shoe
     - Is it useful for sequential recomendation?
3. *Low timestamp resolution may cause the loss of ordering*
   - E.g. daily timestamp resolution
   - The order of events can not be determined, causes event collosion

Experiment

How does removing sequence modeling from GRU4Rec degrade performance on session-based and rating datasets?



Artificial sequences in MovieLens

- High event collision rate (27.3% of events)
  - → loss of original ordering
- Presorted by user and item ID
  - → new ordering via increasing item ID

Overzealous preprocessing (4.)

Preprocessing is essential

1. *Noise reduction*
   - Data collection errors, unusual user behavior, bot traffic, etc.
2. *Tailoring data towards the task*
   - Some preprocessing might be required by the task
   - E.g. testing cold/warm-start algorithms

Preprocessing affects

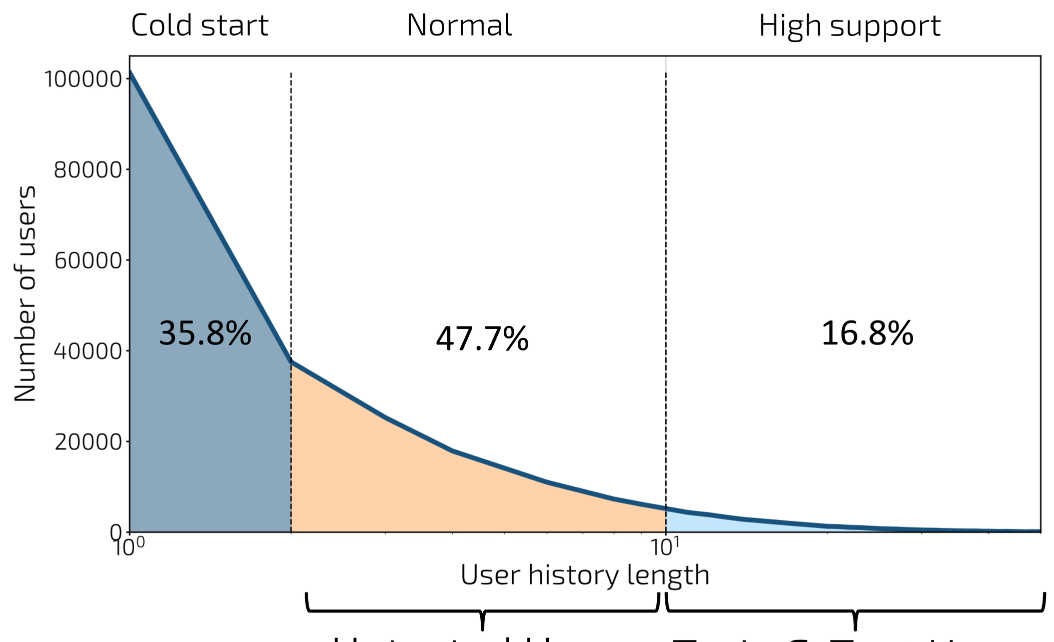
- Interpretation of results
- Comparability with results in previous work
- Generality of claims

Unnecessary preprocessing

- Ignores performance on a (potentially) important subset of the data

User Type	History Length	Percent of Data
Cold start	0 <= HL < 2	35.8%
Normal	2 <= HL < 10	47.7%
High supp.	10 <= HL	16.8%

Model	Metric on Test	Metric on Untested
s.o.t.a.	0.5	0.4
Proposed	0.6	0.35



Testing only on high support users gives misleading results for the full user-base

- High support users might be better for an algorithm
- If it is not specified that the goal is to improve on this small subset, claiming it is better than the s.o.t.a. is misleading

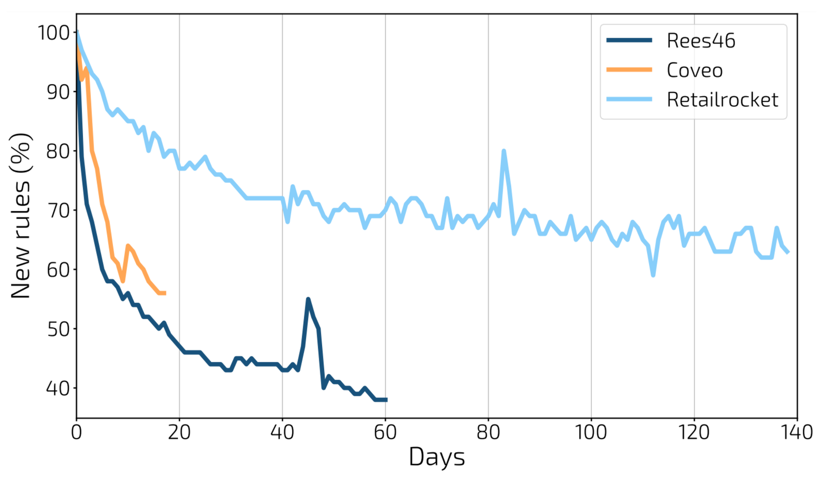
Solution

- Use only necessary preprocessing steps
- State clear claims that are in line with your tests

Information leaking through time (5.)

Information leaks

1. *Train → test*
   - Evaluating on training examples overestimates performance
2. *Overlapping time intervals*
   - Patterns may be specific to a period
   - Overestimating performance of algorithms memorizing these patterns



User behavior data changes constantly

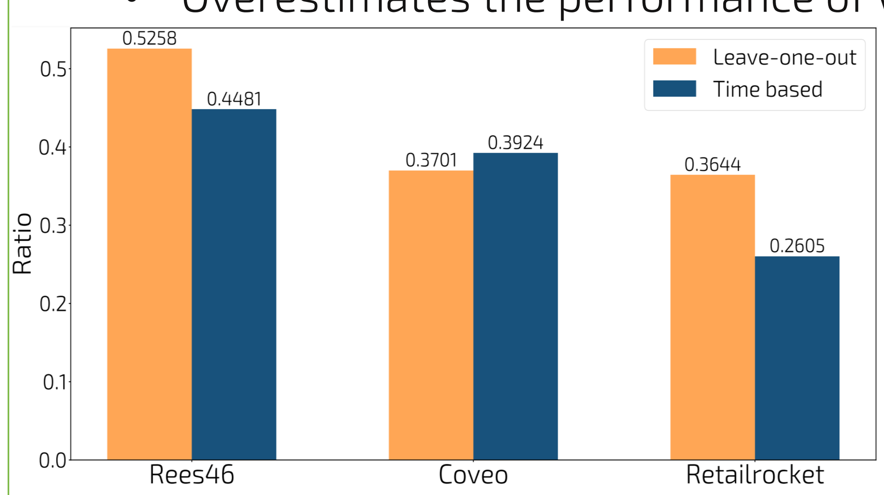
- Proportion of previously unseen A→B sequences remain high even after many days of data collection
- Concept drift

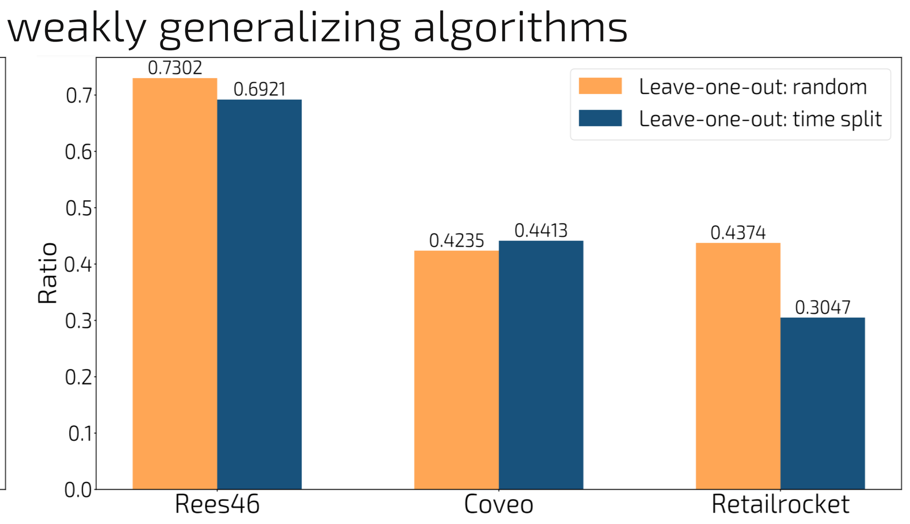
Non-time-based splits

- Cause information leaking through time
- Examples
  - Random split
  - Leave-one-out (certain versions)

Overestimating the performance of memorization algorithms

- Less concept drift between train and test
- Worse approximation of online performance
- Overestimates the performance of weakly generalizing algorithms





Test sets of non-time-based splits have a higher proportion of previously seen A→B sequences

- Reduced concept drift → easier setup
- Memorizing training sequences yields better results → generalization is less important

Separate your test and train sets in time!

Negative sampling during testing (6.)

Weak negative samples

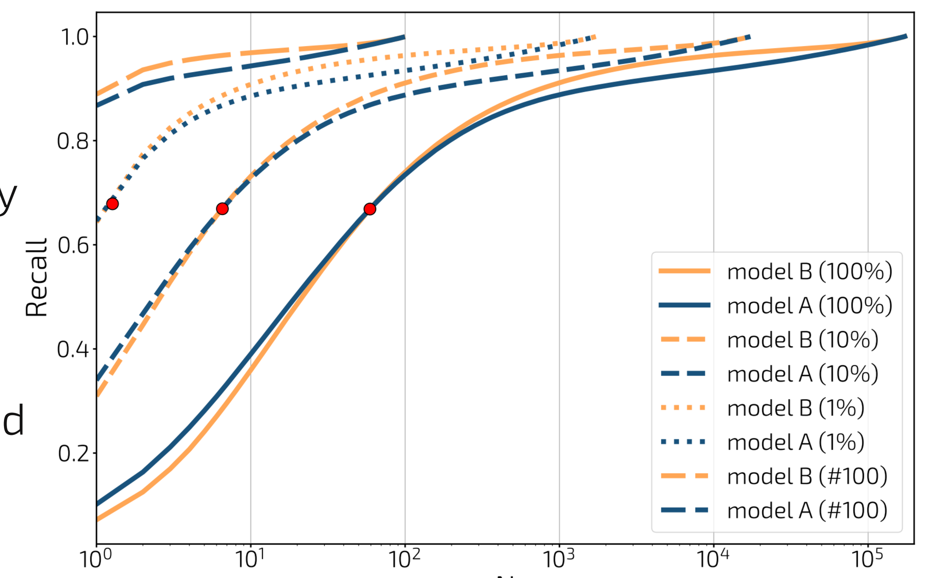
- Easy to rank the target before the sample
- Random samples are most likely weak

Using weak negative samples

- Overestimates performance
- Changes the performance-based ordering of models

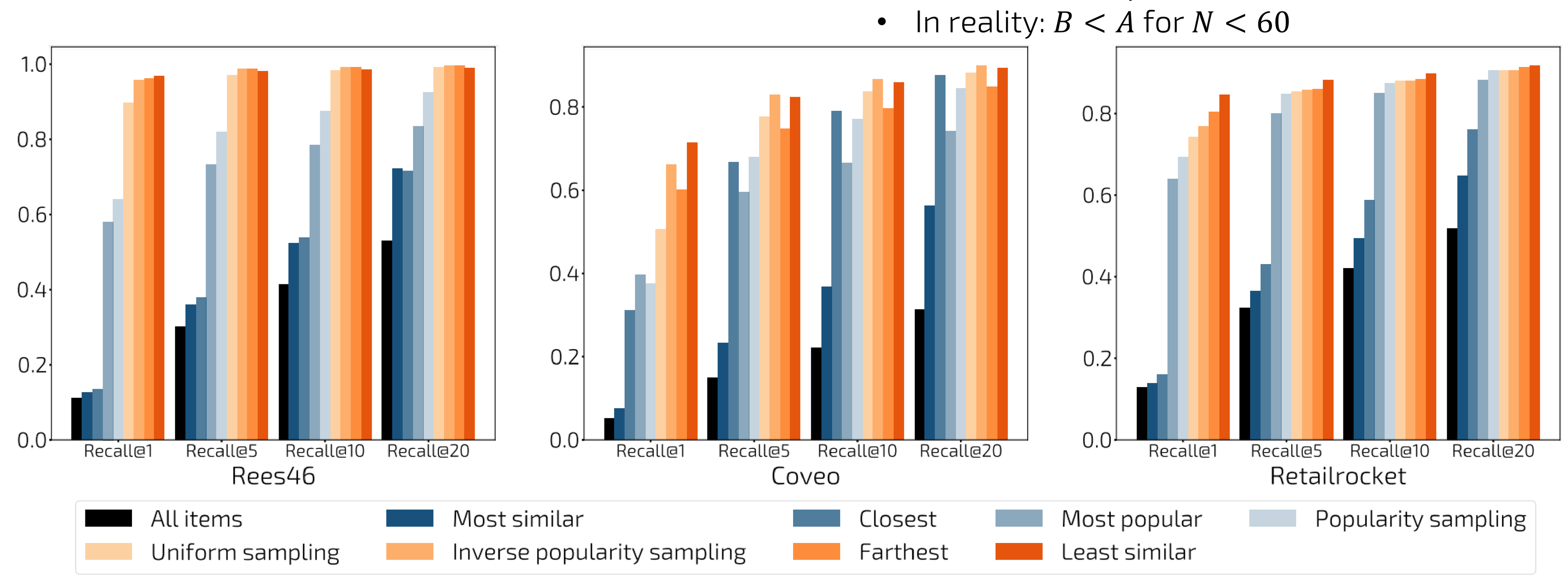
Unnecessary

- Full ranking takes too much time = the model/code is not scalable
- Too large test set (>1M rankings) → sample rankings (e.g. users) instead



Sampling changes the performance-based ordering of models

- Order of models based on Recall@N depends on N
- Switch happens around $N = 60$
- We care about $5 \le N \le 20$
- Sampling shifts the changing point to the left
- It is shifted into the interval we care about
- With 100 samples it is at 0
  - → $B > A$ for any N
  - In reality: $B < A$ for $N < 60$



Even non-random samples are probably weak

- No simple sampling strategy is consistent with the full evaluation strategy.

Impact

a) Any of these flaws can severly impact the results of evaluation
b) Papers might claim s.o.t.a. performance based on incorrect experiments
c) Evaluation setups are often copied without questioning their validity, these flaws spread
d) At least one flaw is present in ~50% of the examined papers and ~25% contains all four

Read the paper