The Effect of Third Party Implementations on Reproducibility

Balázs Hidasi | Gravity R&D, a Taboola Company | @balazshidasi

Ádám Czapp | Gravity R&D, a Taboola Company | @adam_czapp

a Taboola company GRAVITY Research & Development

Challenges of comparing the performance of algorithms

- Why does reproducibility matter?
  - Science: core of the process
  - Application: which papers should I try?

Challenges of comparing the performance of algorithms

- Why does reproducibility matter?
  - Science: core of the process
  - Application: which papers should I try?

Offline evaluation
- Imperfect proxy
- Offline metrics

Challenges of comparing the performance of algorithms

- Why does reproducibility matter?
  - Science: core of the process
  - Application: which papers should I try?

Offline evaluation
- Imperfect proxy
- Offline metrics

↔

Online A/B test
- On business KPIs
- Not reproducible

Challenges of comparing the performance of algorithms

- Why does reproducibility matter?
  - Science: core of the process
  - Application: which papers should I try?

Chain of approximation

Offline evaluation
- Imperfect proxy
- Offline metrics

↔

Online A/B test
- On business KPIs
- Not reproducible

↔

Business goals & metrics

↔

Goal of the recommender

Challenges of comparing the performance of algorithms

- Why does reproducibility matter?
  - Science: core of the process
  - Application: which papers should I try?

Chain of approximation

Offline evaluation
- Imperfect proxy
- Offline metrics

Online A/B test
- On business KPIs
- Not reproducible

Business goals & metrics

Goal of the recommender

Evaluation setup
- Best setup & metric for task
- Dataset
- Preprocessing
- Splitting
- Measuring

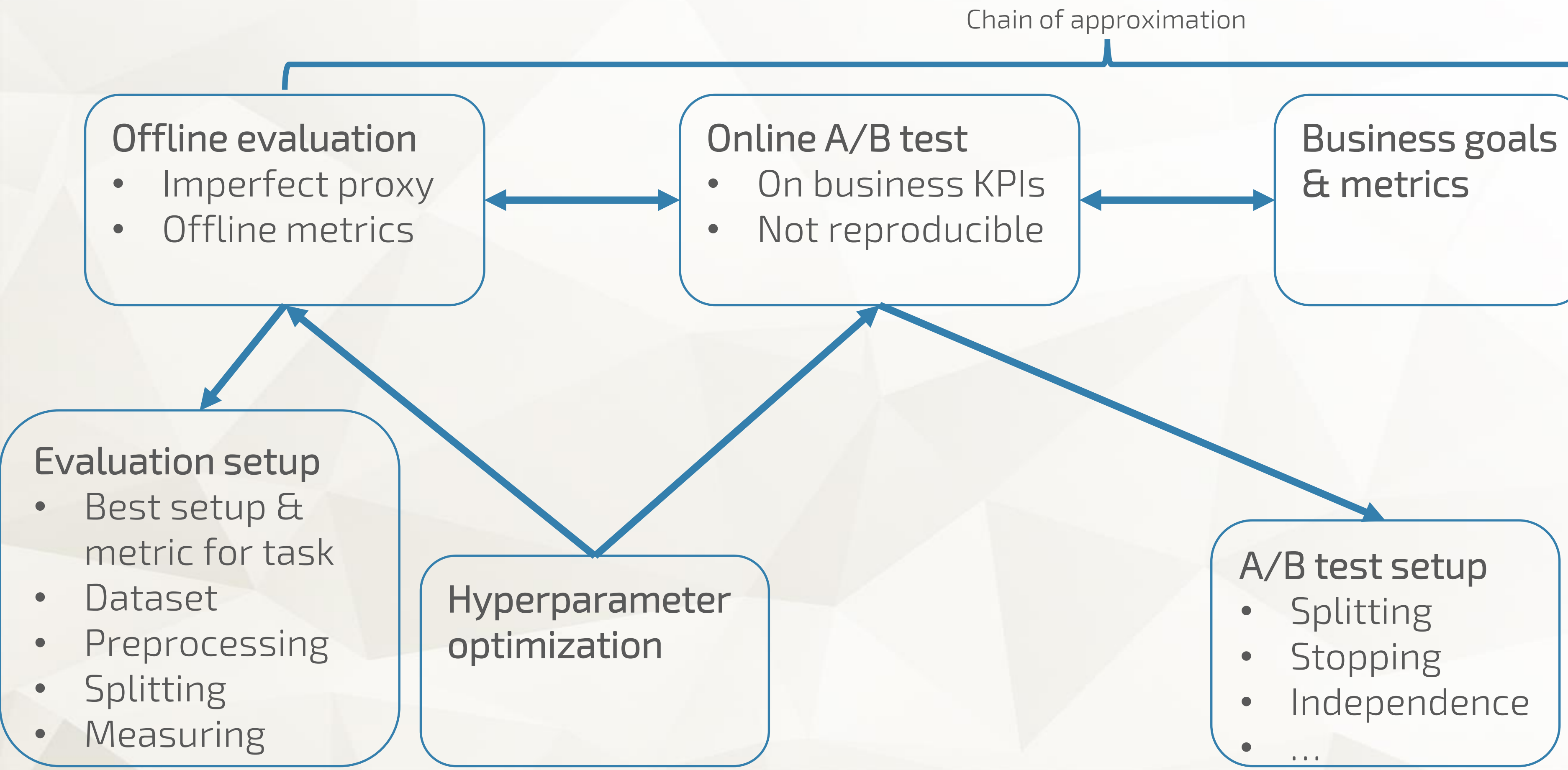
A/B test setup
- Splitting
- Stopping
- Independence
- …

Check out our poster on evaluation flaws tomorrow!

a Taboola company
GRAVITY
Research & Development

Challenges of comparing the performance of algorithms

- Why does reproducibility matter?
  - Science: core of the process
  - Application: which papers should I try?



Check out our poster on evaluation flaws tomorrow!

a Taboola company GRAVITY Research & Development

Challenges of comparing the performance of algorithms

- Why does reproducibility matter?
  - Science: core of the process
  - Application: which papers should I try?

Chain of approximation

Offline evaluation
- Imperfect proxy
- Offline metrics

Online A/B test
- On business KPIs
- Not reproducible

Business goals & metrics

Goal of the recommender

Evaluation setup
- Best setup & metric for task
- Dataset
- Preprocessing
- Splitting
- Measuring

Hyperparameter optimization

Algorithm (re)implementation

A/B test setup
- Splitting
- Stopping
- Independence
- …

Check out our poster on evaluation flaws tomorrow!

a Taboola company
GRAVITY
Research & Development

Why are algorithms reimplemented?

Why are algorithms reimplemented?

Personal reimplementation

- Use with custom evaluator
- Efficiency (time of experiments)
- Official code not available

Why are algorithms reimplemented?

Personal reimplementation

- Use with custom evaluator
- Efficiency (time of experiments)
- Official code not available

Production reimplementation

- Efficiency requirements
- Language/framework requirements

Why are algorithms reimplemented?

Personal reimplementation

- Use with custom evaluator
- Efficiency (time of experiments)
- Official code not available

Production reimplementation

- Efficiency requirements
- Language/framework requirements

Public reimplementation

- Accessibility
- Contributing
- Official code not available

Why are algorithms reimplemented?

Personal reimplementation
- Use with custom evaluator
- Efficiency (time of experiments)
- Official code not available

Production reimplementation
- Efficiency requirements
- Language/framework requirements

Public reimplementation
- Accessibility
- Contributing
- Official code not available

Benchmarking frameworks
- Use with unified evaluator
- Standardization/benchmarking
- Accessibility

Why are algorithms reimplemented?

Personal reimplementation
- Use with custom evaluator
- Efficiency (time of experiments)
- Official code not available

Production reimplementation
- Efficiency requirements
- Language/framework requirements

Public reimplementation
- Accessibility
- Contributing
- Official code not available

Benchmarking frameworks
- Use with unified evaluator
- Standardization/benchmarking
- Accessibility

Are reimplementations correct representations of the original?

Comparing reimplementations of an algorithm to the original

- We chose GRU4Rec, because...

Comparing reimplementations of an algorithm to the original

- We chose GRU4Rec, because…

Seminal work of its field

- Started the line of deep learning methods for session-based/sequential recommendations
- Often used as baseline

TITLE	CITED BY	YEAR
Session-based recommendations with recurrent neural networks B Hidasi, A Karatzoglou, L Baltrunas, D Tikk arXiv preprint arXiv:1511.06939	2589	2015
Recurrent neural networks with top-k gains for session-based recommendations B Hidasi, A Karatzoglou Proceedings of the 27th ACM international conference on information and …	693	2018

Comparing reimplementations of an algorithm to the original

- We chose GRU4Rec, because…

Seminal work of its field

- Started the line of deep learning methods for session-based/sequential recommendations
- Often used as baseline

TITLE	CITED BY	YEAR
Session-based recommendations with recurrent neural networks B Hidasi, A Karatzoglou, L Baltrunas, D Tikk arXiv preprint arXiv:1511.06939	2589	2015
Recurrent neural networks with top-k gains for session-based recommendations B Hidasi, A Karatzoglou Proceedings of the 27th ACM international conference on information and …	693	2018

Official public implementation

https://github.com/hidasib/GRU4Rec

- Since spring 2016
  - (ICLR publication)
- Still supported today
- Well-known

About

GRU4Rec is the original Theano implementation of the algorithm in "Session-based Recommendations with Recurrent Neural Networks" paper, published at ICLR 2016 and its follow-up "Recurrent Neural Networks with Top-k Gains for Session-based Recommendations". The code is optimized for execution on the GPU.

- Readme
- View license
- Activity
- 719 stars
- 41 watching
- 222 forks

Comparing reimplementations of an algorithm to the original

- We chose GRU4Rec, because…

Seminal work of its field

- Started the line of deep learning methods for session-based/sequential recommendations
- Often used as baseline

TITLE	CITED BY	YEAR
Session-based recommendations with recurrent neural networks B Hidasi, A Karatzoglou, L Baltrunas, D Tikk arXiv preprint arXiv:1511.06939	2589	2015
Recurrent neural networks with top-k gains for session-based recommendations B Hidasi, A Karatzoglou Proceedings of the 27th ACM international conference on information and ...	693	2018

Official public implementation

https://github.com/hidasib/GRU4Rec

- Since spring 2016
  - (ICLR publication)
- Still supported today
- Well-known

About

GRU4Rec is the original Theano implementation of the algorithm in "Session-based Recommendations with Recurrent Neural Networks" paper, published at ICLR 2016 and its follow-up "Recurrent Neural Networks with Top-k Gains for Session-based Recommendations". The code is optimized for execution on the GPU.

Readme
View license
Activity
719 stars
41 watching
222 forks

Simple algorithm but highly adapted

- Simple architecture
- Custom adaptations to the recommender domain
- Described in detail in the corresponding papers

Comparing reimplementations of an algorithm to the original

- We chose GRU4Rec, because…

Seminal work of its field
- Started the line of deep learning methods for session-based/sequential recommendations
- Often used as baseline

TITLE	CITED BY	YEAR
Session-based recommendations with recurrent neural networks B Hidasi, A Karatzoglou, L Baltrunas, D Tikk arXiv preprint arXiv:1511.06939	2589	2015
Recurrent neural networks with top-k gains for session-based recommendations B Hidasi, A Karatzoglou Proceedings of the 27th ACM international conference on information and …	693	2018

Official public implementation
https://github.com/hidasib/GRU4Rec
- Since spring 2016
  - (ICLR publication)
- Still supported today
- Well-known

About

GRU4Rec is the original Theano implementation of the algorithm in "Session-based Recommendations with Recurrent Neural Networks" paper, published at ICLR 2016 and its follow-up "Recurrent Neural Networks with Top-k Gains for Session-based Recommendations". The code is optimized for execution on the GPU.

Readme
View license
Activity
719 stars
41 watching
222 forks

Simple algorithm but highly adapted
- Simple architecture
- Custom adaptations to the recommender domain
- Described in detail in the corresponding papers

Implemented in Theano
- Discontinued DL framework (2018)
- Motivation for recoding in more popular frameworks

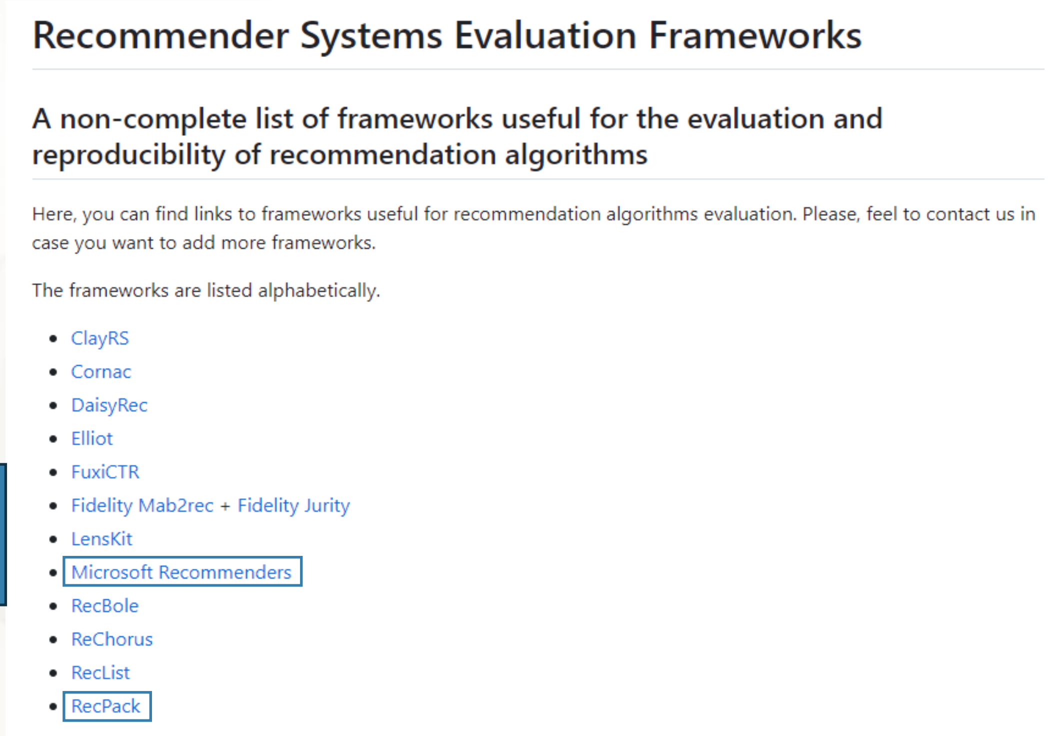
Reimplementations of GRU4Rec

- Checked
  - 2 PyTorch implementations
    - GRU4REC-pytorch
      - Popular reimplementation
      - Published in 2018
      - Last commit in 2021
    - Torch-GRU4Rec
      - Newer implementation from 2020
  - 2 Tensorflow/Keras implementations
    - GRU4Rec_Tensorflow
      - Popular reimplementation
      - Published in 2017
      - Last commit in 2019
    - KerasGRU4Rec
      - Published in 2018
      - Last meaningful update in 2020
  - 2 benchmarking framework implementations
    - Microsoft Recommenders
      - Large algorithm collection
    - Recpack
      - Recently released framework

Reimplementations of GRU4Rec

- Checked
  - 2 PyTorch implementations
    - GRU4REC-pytorch
      - Popular reimplementation
      - Published in 2018
      - Last commit in 2021
    - Torch-GRU4Rec
      - Newer implementation from 2020
  - 2 Tensorflow/Keras implementations
    - GRU4Rec_Tensorflow
      - Popular reimplementation
      - Published in 2017
      - Last commit in 2019
    - KerasGRU4Rec
      - Published in 2018
      - Last meaningful update in 2020
  - 2 benchmarking framework implementations
    - Microsoft Recommenders
      - Large algorithm collection
    - Recpack
      - Recently released framework

Recommended by RecSys 2023 CFP



Recommender Systems Evaluation Frameworks

A non-complete list of frameworks useful for the evaluation and reproducibility of recommendation algorithms

Here, you can find links to frameworks useful for recommendation algorithms evaluation. Please, feel to contact us in case you want to add more frameworks.

The frameworks are listed alphabetically.

- ClayRS
- Cornac
- DaisyRec
- Elliot
- FuxiCTR
- Fidelity Mab2rec + Fidelity Jurity
- LensKit
- Microsoft Recommenders
- RecBole
- ReChorus
- RecList
- RecPack

Reimplementations of GRU4Rec

- Checked
  - 2 PyTorch implementations
    - GRU4REC-pytorch
      - Popular reimplementation
      - Published in 2018
      - Last commit in 2021
    - Torch-GRU4Rec
      - Newer implementation from 2020
  - 2 Tensorflow/Keras implementations
    - GRU4Rec_Tensorflow
      - Popular reimplementation
      - Published in 2017
      - Last commit in 2019
    - KerasGRU4Rec
      - Published in 2018
      - Last meaningful update in 2020
  - 2 benchmarking framework implementations
    - Microsoft Recommenders
      - Large algorithm collection
    - Recpack
      - Recently released framework

- Others (we know of)
  - Discontinued PyTorch reimplementation
  - RecBole implementation
    - Doesn't even reference the right papers

Recommender Systems Evaluation Frameworks

A non-complete list of frameworks useful for the evaluation and reproducibility of recommendation algorithms

Here, you can find links to frameworks useful for recommendation algorithms evaluation. Please, feel to contact us in case you want to add more frameworks.

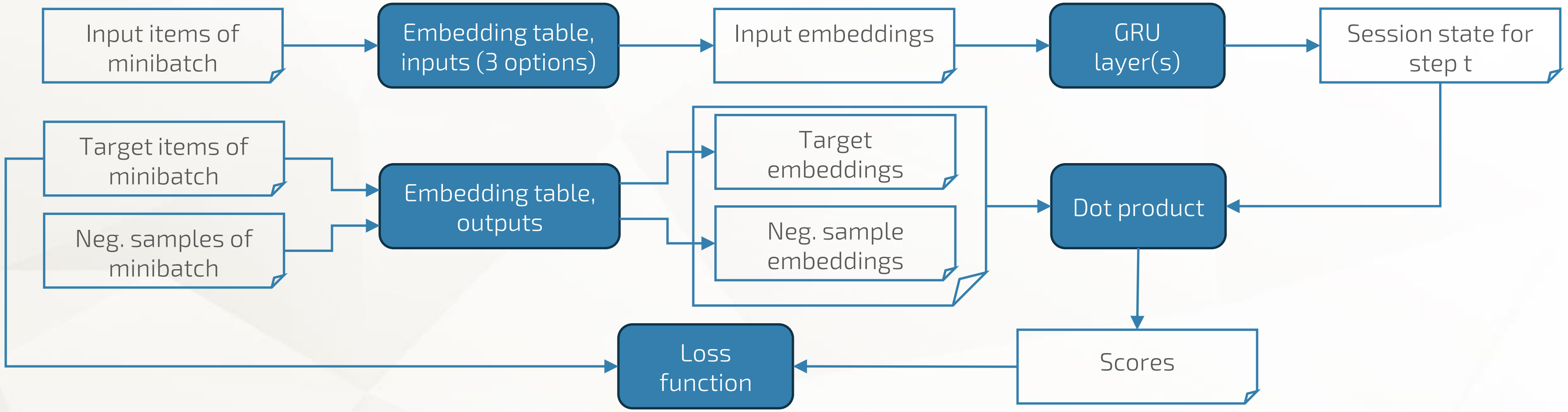
The frameworks are listed alphabetically.

- ClayRS
- Cornac
- DaisyRec
- Elliot
- FuxiCTR
- Fidelity Mab2rec + Fidelity Jurity
- LensKit
- Microsoft Recommenders
- RecBole
- ReChorus
- RecList
- RecPack

Recommended by RecSys 2023 CFP

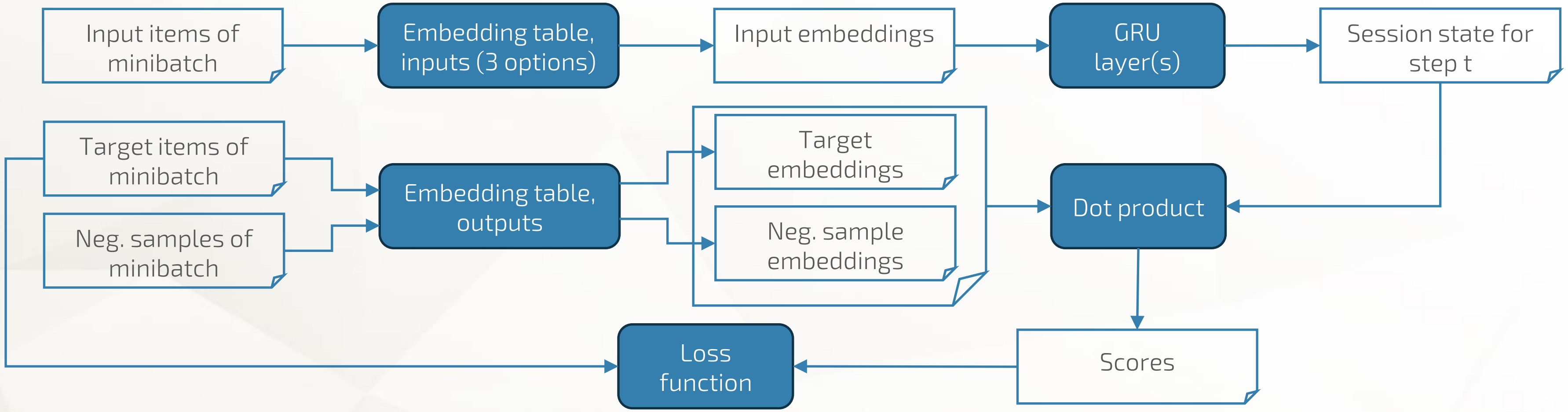
RQ1: Do they implement the same architecture as the original?

- Architecture of GRU4Rec



RQ1: Do they implement the same architecture as the original?

- Architecture of GRU4Rec

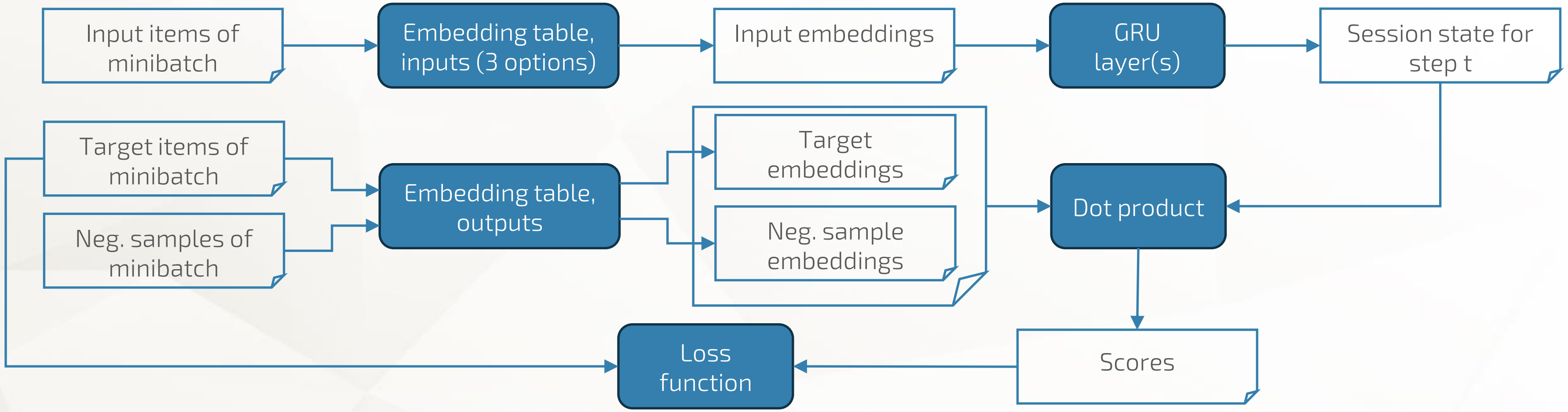


Those who could do it (5/6)

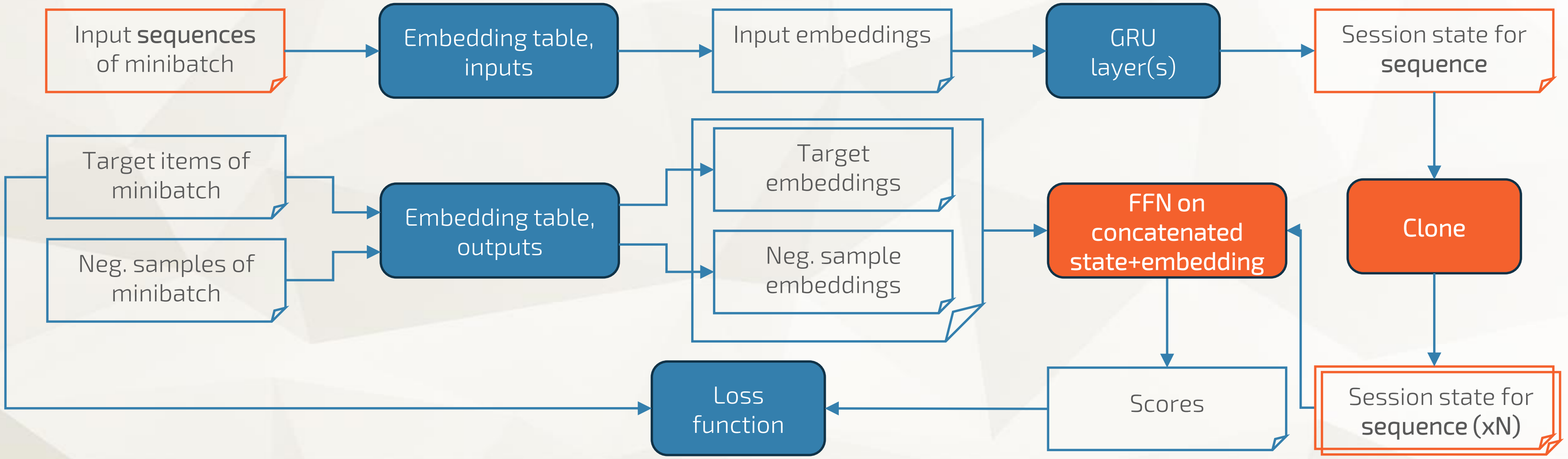
- GRU4REC-pytorch
- Torch-GRU4Rec
- GRU4Rec_Tensorflow
- KerasGRU4Rec
- Recpack

RQ1: Do they implement the same architecture as the original?

- Architecture of GRU4Rec



- Different architecture in MS recommenders

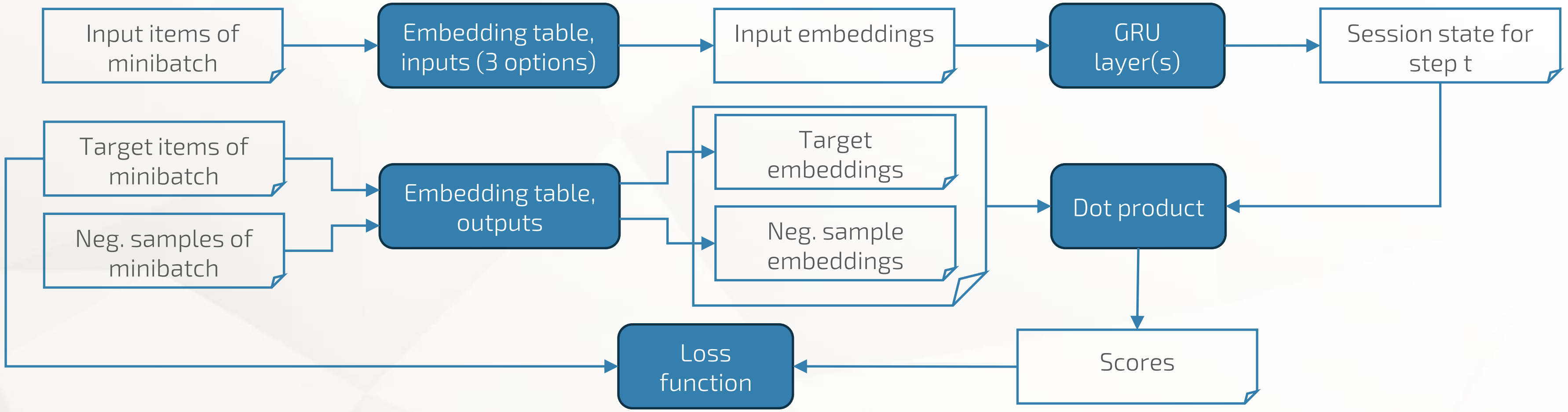


Those who could do it (5/6)

- GRU4REC-pytorch
- Torch-GRU4Rec
- GRU4Rec_Tensorflow
- KerasGRU4Rec
- Recpack

RQ1: Do they implement the same architecture as the original?

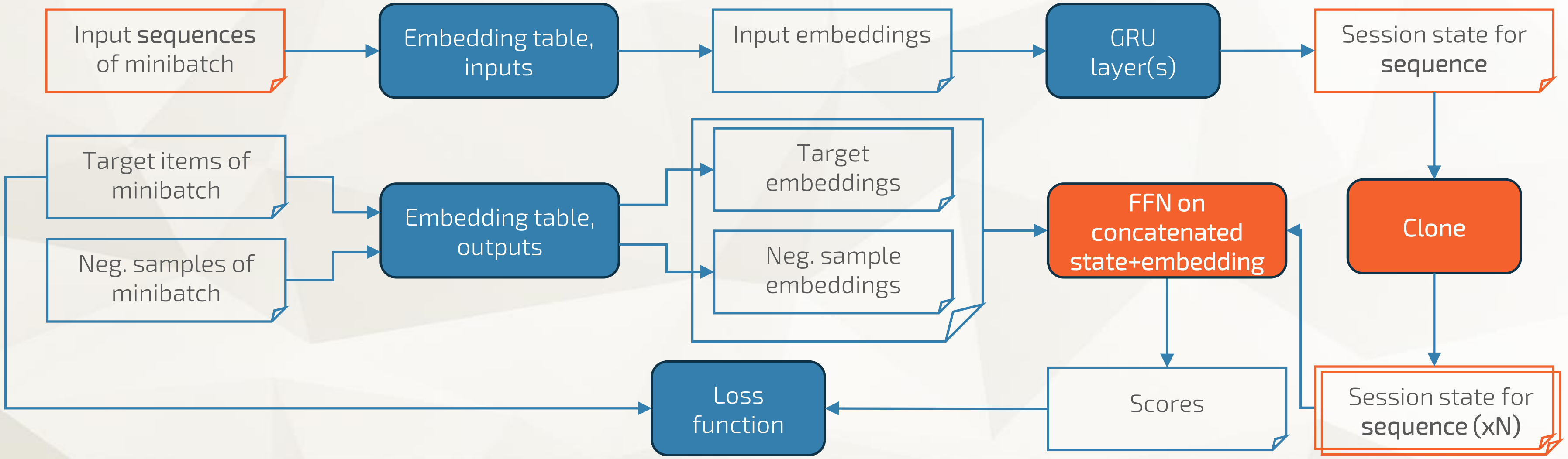
- Architecture of GRU4Rec



Those who could do it (5/6)

- GRU4REC-pytorch
- Torch-GRU4Rec
- GRU4Rec_Tensorflow
- KerasGRU4Rec
- Recpack

- Different architecture in MS recommenders



Severe scalability issue

- Number of negative samples is strictly limited during training
- Requires negative sampling during inference
  - Evaluation flaw
- Not able to rank all items during inference

a Taboola company
GRAVITY
Research & Development

RQ2: Do they have the same features as the original?

- GRU4Rec = GRU ***adapted*** to the recommendation problem
  - Missing features (see table)
  - Missing hyperparameters
    - All versions: momentum, logQ
    - Some versions: bpreg, embedding/hidden dropout, sample_alpha, …

✓ Included
✗ Missing
Partial or flawed

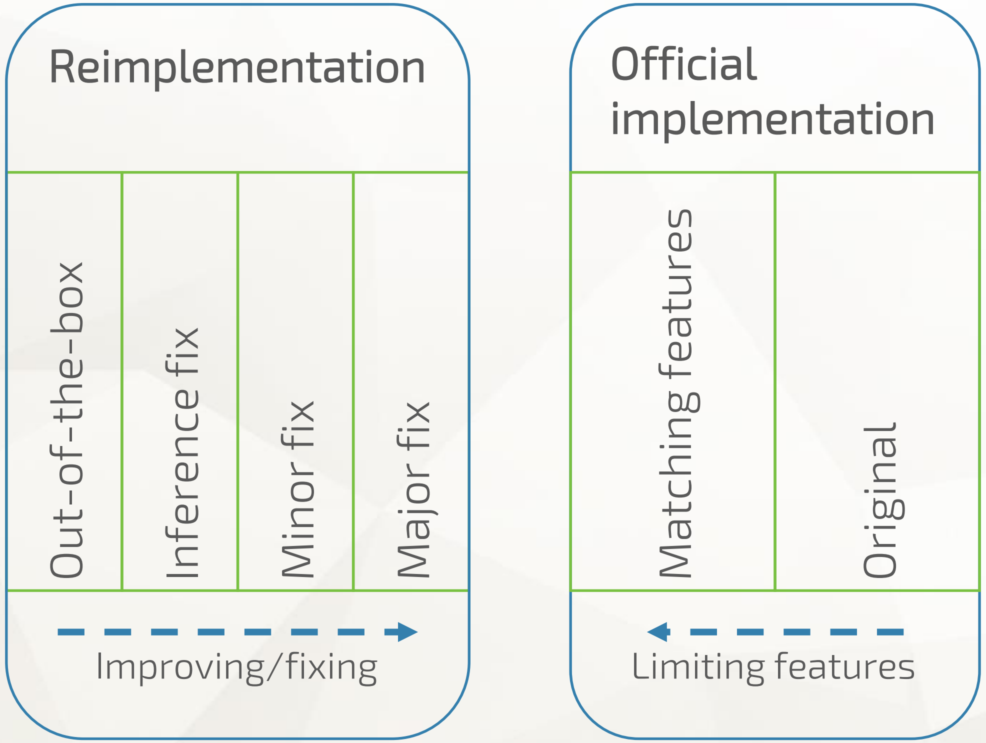
GRU4Rec feature		GRU4REC-pytorch	Torch-GRU4Rec	GRU4Rec_Tensorflow	KerasGRU4Rec	Recpack
Session parallel mini-batches		✓	✓	✓	✓	#
Negative sampling	Mini-batch	✓	✓	✓	✗	#
	Shared extra	✗	✓	✗	✗	#
Loss	Cross-entropy	#	✓	✓	✓	✓
	BPR-max	#	✓	✗	✗	✓
Embedding	No embedding	✓	✓	✗	✓	✗
	Separate	✓	✓	✓	✗	✓
	Shared	✗	✗	✗	✗	✗

RQ3: Do they suffer from implementation errors?

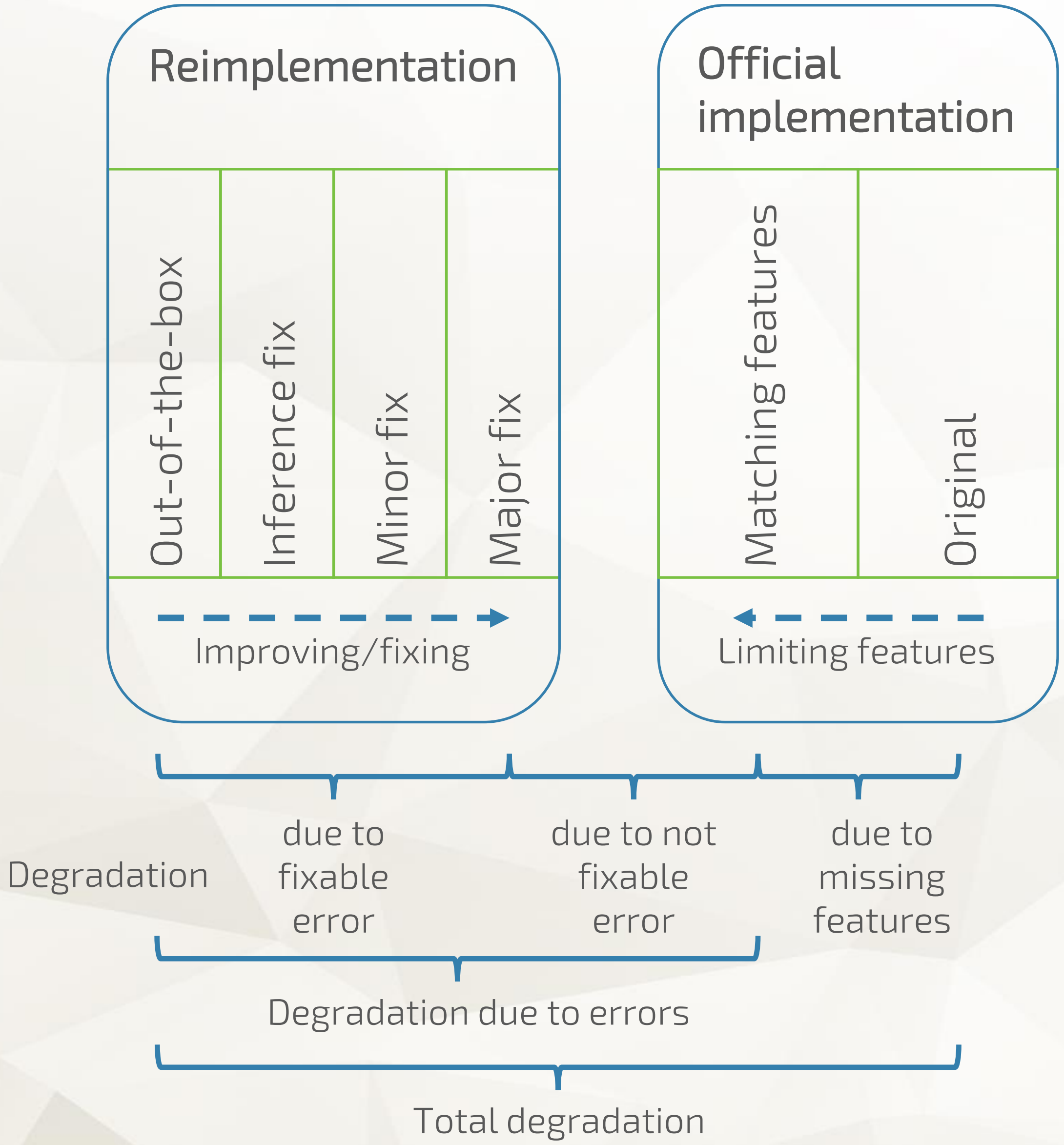
Nature of the error	Basic errors	Inference errors	Minor errors (easy to notice & fix)	Major errors (hard to notice or fix)	Core errors (full rewrite)
Effort to fix	Almost certainly fixed by any user	Potentially fixed by an involved user	Likely fixed by an experienced user	May be fixed by a very thorough user	Most likely NOT fixed by any user
Examples	- Typos/syntax errors - Variables on the incorrect device - P(dropout) is used as P(keep) - Code is not prepared for unseen test items	- Hidden states are not reset properly	- Large initial accumulator value prevents convergence - Differences to the original (learning rate decay, initialization, optimizer) - Hard-coded hyperparameters	- Sampling and softmax are in reverse order - Softmax applied twice - Hidden states are reset at incorrect times - Incorrect BPR-max loss - Dropout can be set, but not applied - Embedding and hidden dropout uses the same parameter by mistake	- Sampling and scoring are in reverse order
Number of occurrences					
GRU4REC-pytorch	1	1	0	5	1
Torch-GRU4Rec	1	0	0	0	1
GRU4Rec_Tensorflow	2	0	3	0	0
KerasGRU4Rec	0	0	2	2	0
Recpack	2	0	3	1	1

a Taboola company
GRAVITY
Research & Development

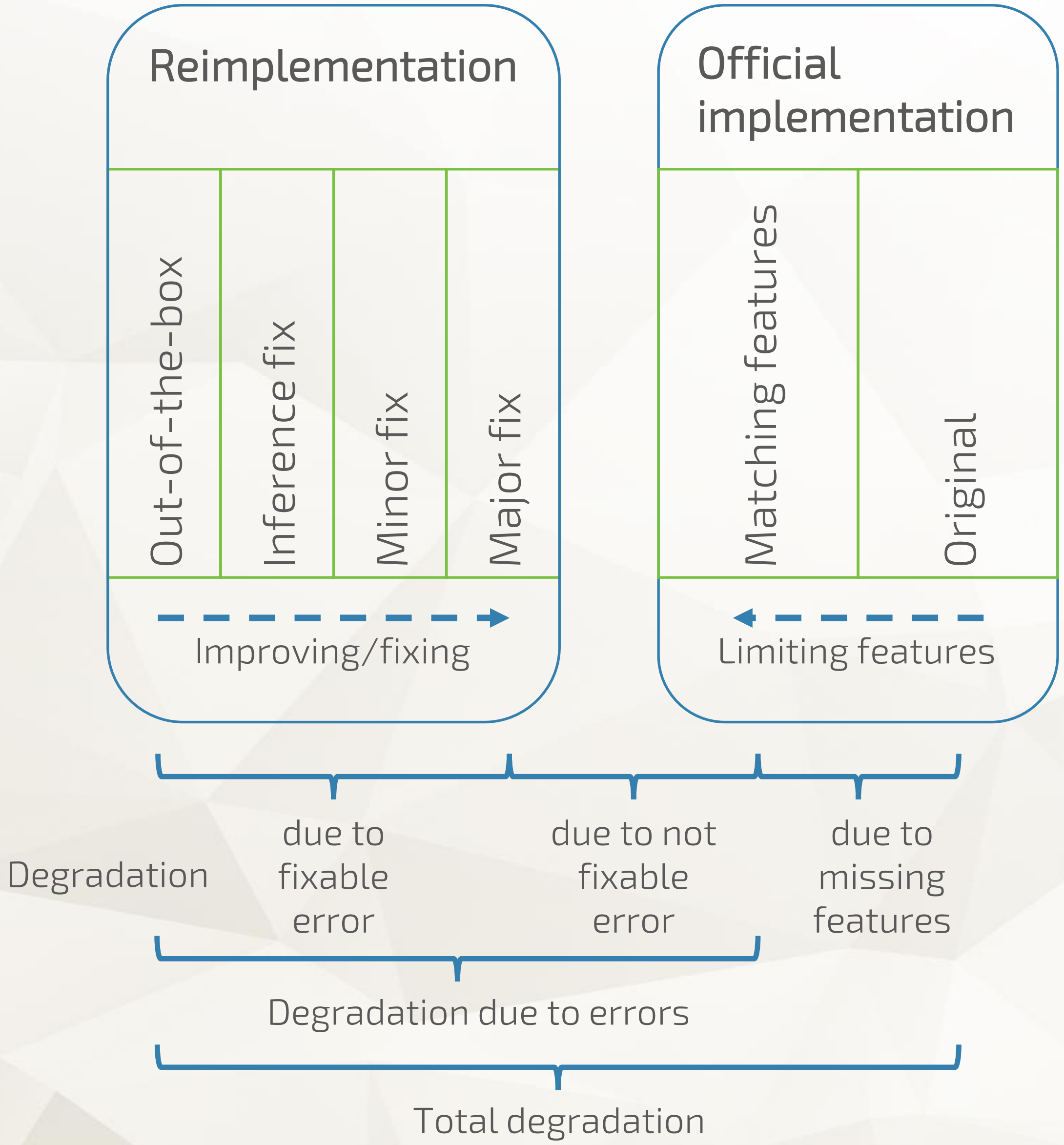
RQ4: How do missing features & errors affect offline results?



RQ4: How do missing features & errors affect offline results?



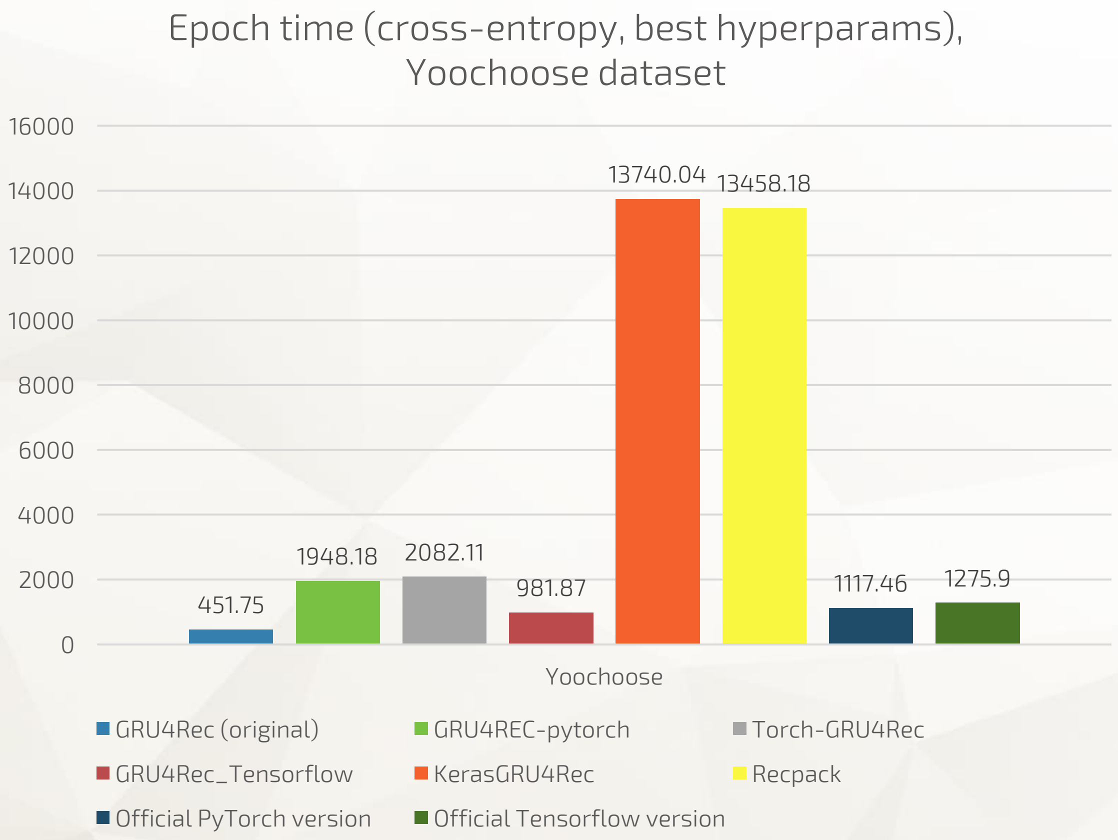
RQ4: How do missing features & errors affect offline results?

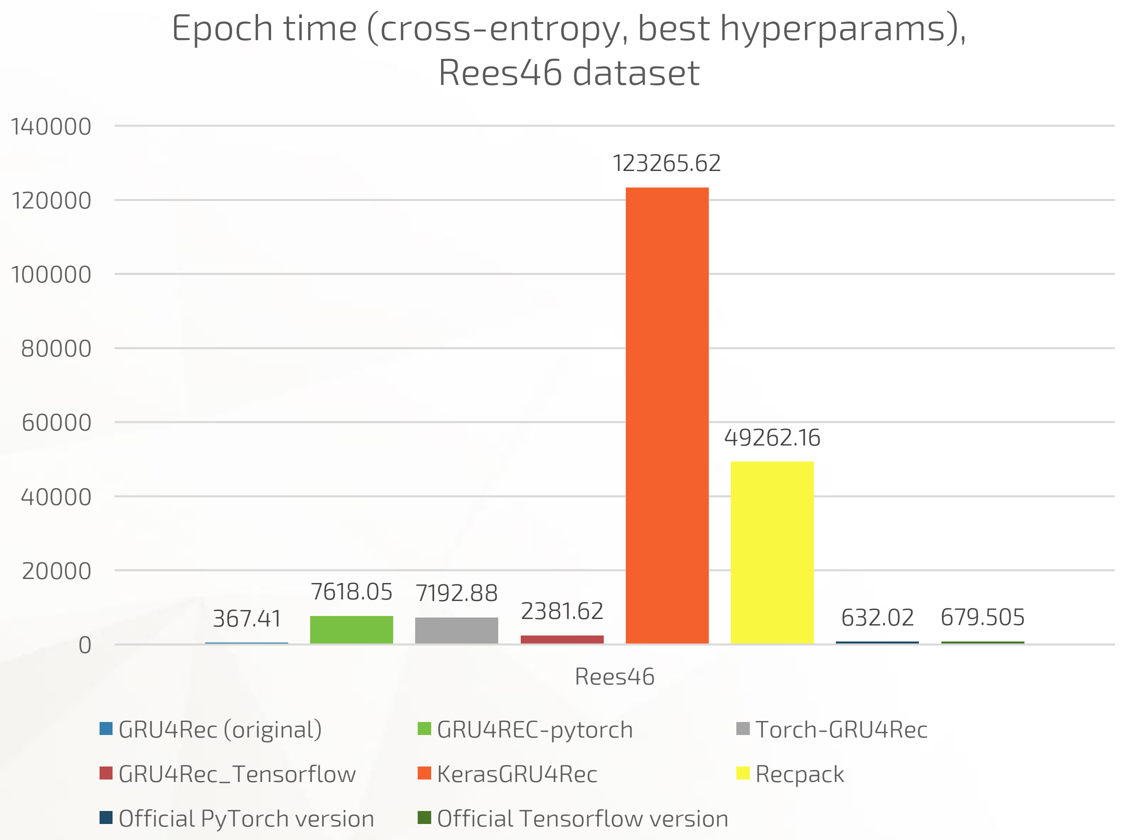


	Perf. loss via errors	Perf. loss via features	Total perf. loss
MEDIAN			
GRU4REC-pytorch	-56.34%	-46.14%	-75.73%
Torch-GRU4Rec	-1.29%	-5.90%	-7.55%
GRU4Rec_Tensorflow	-80.59%	-47.15%	-89.46%
KerasGRU4Rec	-9.54%	-11.94%	-21.32%
Recpack	-21.23%	-8.48%	-30.27%
MAX			
GRU4REC-pytorch	-99.38%	-63.88%	-99.62%
Torch-GRU4Rec	-10.46%	-18.92%	-27.24%
GRU4Rec_Tensorflow	-88.44%	-61.81%	-93.89%
KerasGRU4Rec	-26.69%	-15.26%	-37.87%
Recpack	-37.14%	-22.71%	-48.86%

- Measured on 5 public session-based datasets
  - Yoochoose, Rees46, Coveo, Retailrocket, Diginetica
- Next item prediction (strict)
- Recall & MRR

RQ5: Training time comparisons





- OOB versions vs. feature complete official versions
- Reimplementations are generally slow
- KerasGRU4Rec and Recpack versions scale badly (no sampling)
- Largest slowdown factor: 335.87x

a Taboola company GRAVITY Research & Development

What does this mean?

- Final tally
  - MS Recommender's version is GRU4Rec in name only and deeply flawed
  - Other versions miss at least one important feature of the original
  - All versions have performance decreasing bugs
  - Two implementations scale poorly

- Potentially a lot of research from the last 6-7 years used flawed baseline(s)
  - Hard to tell: no indication of the implementation used
  - Results might be invalidated

- Probably GRU4Rec is not the only algorithm affected
  - It has a public version to base reimplementations on, yet they are still flawed
  - Other well-known baselines should be checked

- Discussions
  - Responsibility
  - Trust in the tools we use
  - How to correct affected work?

a Taboola company
GRAVITY
Research & Development

What can you do?

If your research used a flawed version

- Rerun experiments with official code
- Extend your work with the results

What can you do?

If your research used a flawed version

- Rerun experiments with official code
- Extend your work with the results

If you want to help

- Check reimplementations of other popular baselines

What can you do?

If your research used a flawed version

- Rerun experiments with official code
- Extend your work with the results

If you want to help

- Check reimplementations of other popular baselines

As an author

- Always state the implementation you use for every baseline
    - Including link, optionally commit hash
- Use official code if possible

What can you do?

If your research used a flawed version

- Rerun experiments with official code
- Extend your work with the results

If you want to help

- Check reimplementations of other popular baselines

As an author

- Always state the implementation you use for every baseline
  - Including link, optionally commit hash
- Use official code if possible

If you reimplement an algorithm

- Validate your version against the original *before* using or releasing it
  - Compare metrics achieved on multiple datasets under multiple hyperparameter settings
  - Compare recommendation lists
  - Check if your version has every feature/setting
- Describe the validation process and its results in the README
- Consider if any future change to the original code (e.g. bugfix) should be added to your version as well
  - If implementations diverge due to the original changing, state it clearly

a Taboola company GRAVITY Research & Development

What can you do?

If your research used a flawed version

- Rerun experiments with official code
- Extend your work with the results

If you want to help

- Check reimplementations of other popular baselines

As an author

- Always state the implementation you use for every baseline
    - Including link, optionally commit hash
- Use official code if possible

If you reimplement an algorithm

- Validate your version against the original *before* using or releasing it
    - Compare metrics achieved on multiple datasets under multiple hyperparameter settings
    - Compare recommendation lists
    - Check if your version has every feature/setting
- Describe the validation process and its results in the README
- Consider if any future change to the original code (e.g. bugfix) should be added to your version as well
    - If implementations diverge due to the original changing, state it clearly

As maintainer of a benchmarking framework

- Same as reimplementing any algorithm
- + validate every reimplementation submitted by contributors

The wider picture (towards standardized benchmarking)

- State of RecSys benchmarking:
  - Little has changed in the last decade
  - Focus is on baseline reimplementations
  - Collection of algorithms
  - Evaluation is somewhat neglected
    - Incorrect assumptions:
      - One/few size fits all
      - Single correct evaluation setup

The wider picture (towards standardized benchmarking)

- State of RecSys benchmarking:
  - Little has changed in the last decade
  - Focus is on baseline reimplementations
  - Collection of algorithms
  - Evaluation is somewhat neglected
    - Incorrect assumptions:
      - One/few size fits all
      - Single correct evaluation setup

→

- Towards standardized benchmarking
  - Collect popular **recommendation tasks**
    - E.g. CTR prediction, session-based recommendation, user-based recommendation, warm/cold-start versions, reoccurrence prediction, etc.)
  - **Evaluation stems from the tasks**:
    - agree on offline **evaluation setups**
    - **datasets** (and their **preprocessing**)
    - for each task
  - **Focus on the evaluation** code of these setups
    - including dataset & preprocessing
  - **Provide** simple **interfaces for evaluating external algorithms**
    - Authors then can use the framework during research
  - Only once everything is ready, add some of the most well-known baselines

Thanks for your attention!

Read the paper!

Check out the project website!

We'd also like to help.

Official reimplementations of GRU4Rec

PyTorch

Tensorflow

a Taboola company
GRAVITY
Research & Development