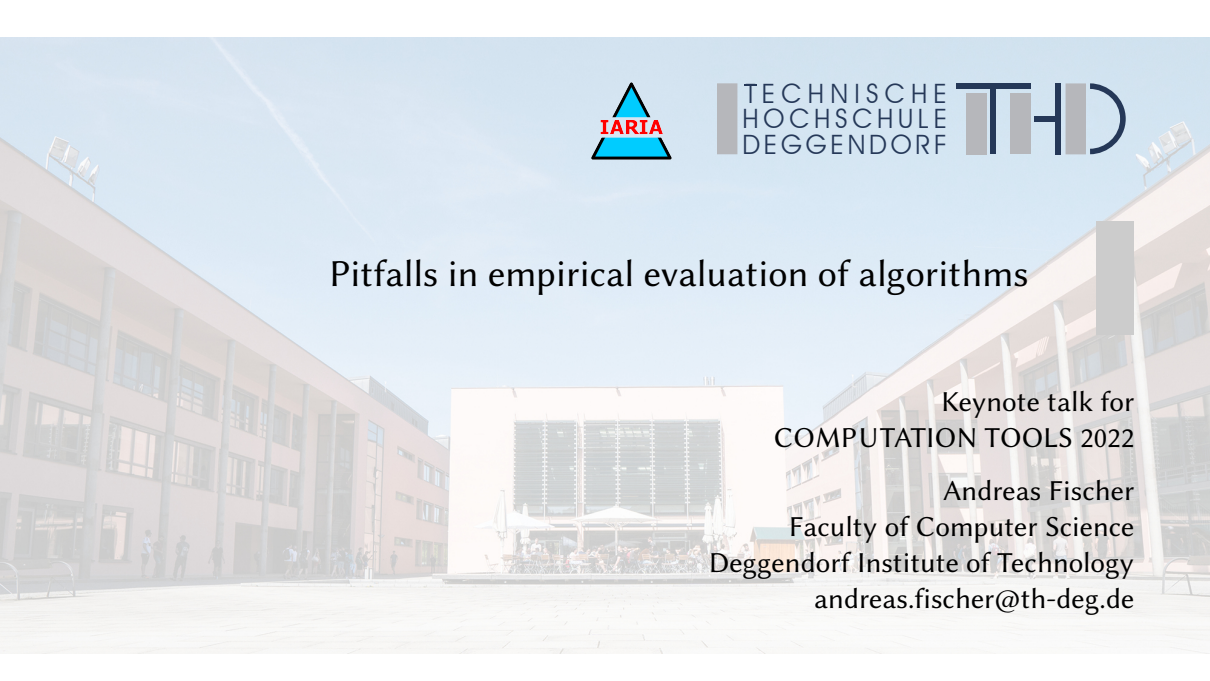


Pitfalls in empirical evaluation of algorithms

Keynote talk for
COMPUTATION TOOLS 2022

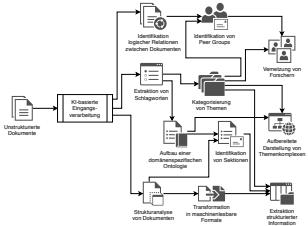
Andreas Fischer
Faculty of Computer Science
Deggendorf Institute of Technology
andreas.fischer@th-deg.de

About the presenter

Andreas Fischer

since 2020	Chief Information Officer, Deggendorf Institute of Technology
since 2017	Full Professor at Deggendorf Institute of Technology.
2017	Postdoctoral researcher, Karlstad University.
2017	PhD in computer science (Dr. rer. nat.), University of Passau.
2008–2017	Research Associate, University of Passau.

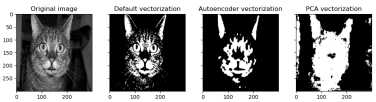
Current research interests



Applied natural language processing

Applied document retrieval

AI-based process analysis



AI-optimized image vectorization

Overview

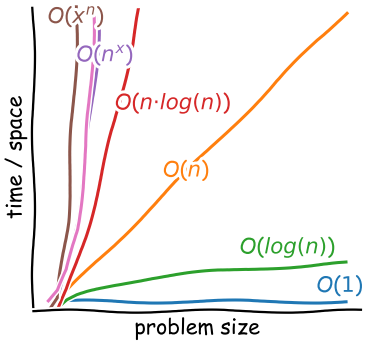
Why do we do empirical evaluation?

What can go wrong?

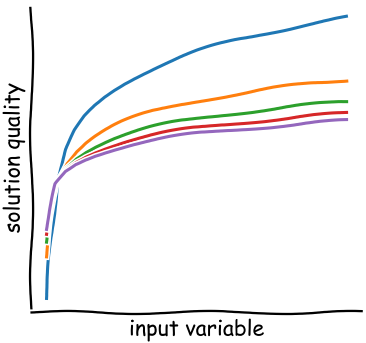
What can go wrong in AI?

Summary

How do we judge algorithms?



Complexity Analysis



Empirical Evaluation

Advantages of each approach

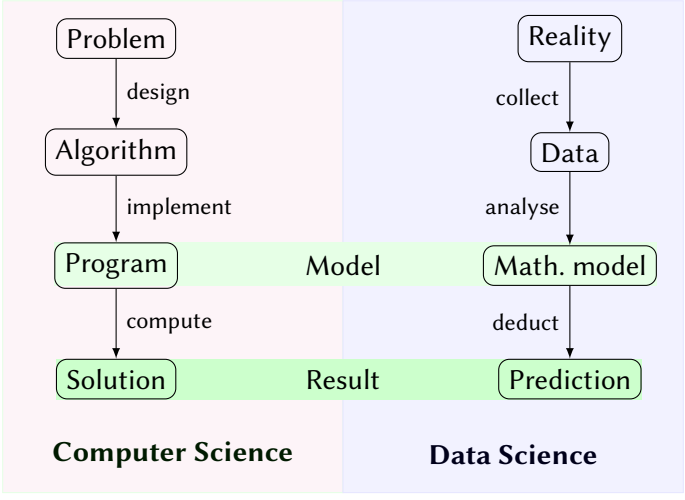
Complexity Analysis

- …gives theoretical / abstract results
- …focuses on time/space requirements
- …is difficult to perform and interpret in multivariate case
- …is hard work

Empirical Evaluation

- …gives concrete data
- …allows to explore impact of many variables
- …enables detailed comparison of solutions with similar complexity
- …can be set up rapidly
- …gives nice graphs

A word on Data Science



So ...

→ Doing experiments in computer science complements theoretical analysis

→ We need experiments in data science to understand the data

Overview

Why do we do empirical evaluation?

What can go wrong?

What can go wrong in AI?

Summary

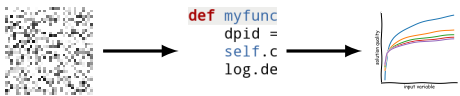
No data for experiments? Create your own!

The situation we are in

- Good real-world data may be hard to get
  - Either only small data sets
  - Or bad data quality
- Algorithm should not be overfit to a particular setting (e.g., “works only in this specific lab scenario”)
- Parameter space typically too large to explore all dimensions

The solution

- Focus on a few key parameters to investigate
- Generate random data for the rest
- Either out of the blue, or by manipulating real data randomly



No data for experiments? Create your own!

What could possibly go wrong?

Failure to create repeatable experiments

Problem

- Experiment is set up on some random machine
- Code and data change during experimentation
- Keeping metadata about the computing environment remains an afterthought

Effect

- Previous experiments cannot be repeated
- Environment cannot be reproduced later
- Comparability between results at different experimental stages is questionable

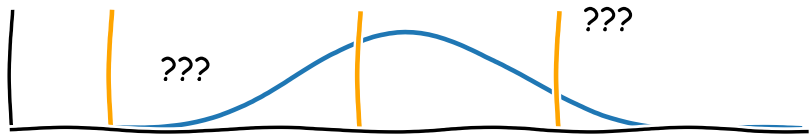
How to handle this?

- Record parameter settings for each experimental run
- Use and record predefined seeds for random number generation
- Use versioning for code and data
  - E.g., with GIT
  - Make separate repositories for code and experimental data

→ This is actually good practice in terms of research data management

→ Similar issues in CI/CD: We can learn from DevOps here

Failure to generate enough data



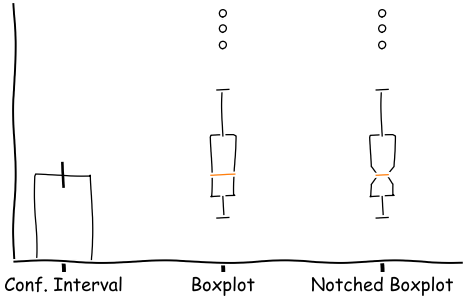
Problem

- Not enough data is collected for each parameter set
- Random influences are not accounted for

Effect

- Interpretation of data is unclear
- Comparability among different experimental runs suffers

How to handle this?

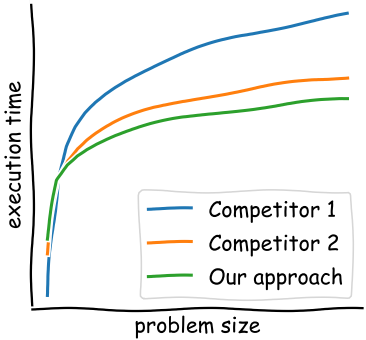


- Perform multiple runs for each algorithm/parameter combination
- Use appropriate plot types to show aggregated data instead of individual data points

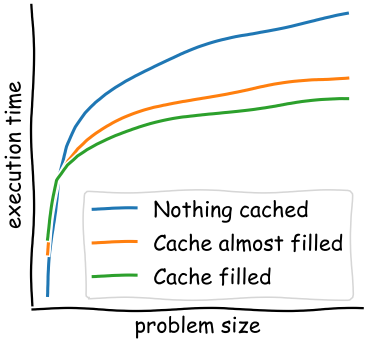
How much data is enough?

- Depends on variance → try to make the confidence interval small
- Rule of thumb: 30+ iterations (Law of large numbers)

Failure to take run-time effects into account



What we would like to see



What the data may actually show

How to handle this?

Things to look out for

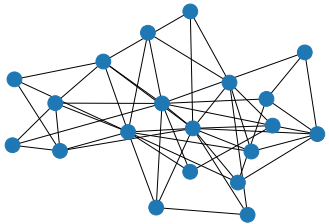
- Caching
- Garbage collection
- Varying system load

Good rules-of-thumb

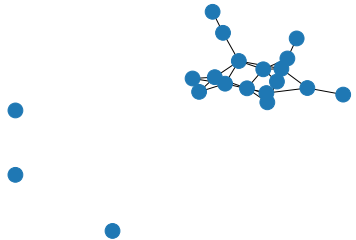
- Add 2–3 experimental runs at the beginning to stabilize the environment. Ignore those results.
- Do not interleave experimental runs with different parameters or algorithms. Order by experiment parameters.

Failure to appreciate complex structures

- Random generation of complex structures can have unintended effects
- E.g., trying to naively generating a random graph



Expected random structure



Obtained random structure

How to handle this?

- Know the characteristics of your data points very well.
- Check generated structures for plausibility (e.g., unconnected graphs). Either fix or eliminate implausible structures.
- Generating good random data may be a task of its own.

Overview

Problems with AI experiments

https://xkcd.com/2451/

Problems with AI experiments

https://xkcd.com/2451/

- Machine Learning (in particular Deep Learning) needs huge amounts of data
- But: Data is often not easily available for experiments
  - Insufficient amounts
  - Unavailable due to legal reasons
  - Unaccessible

→ We may have to produce our own data

Generating data with an AI

Idea

- Some AI approaches can not only classify existing data, but also generate new data
- E.g., via GANs or Transformers
- Current approaches: StyleGAN, GPT-3, DALL·E

→ Can we use that to generate more data for training?

Problems with AI-generated data

Problems

- Good examples are very convincing, but there are also bad examples
- Data is not representative of reality
- Data may be mislabeled

- Trained AI does not model the real world, but models another model
- May increase the bias of an already biased data set
- Dangerously close to circular reasoning ↻

How to handle this?

- Augmented data should complement, but not replace real-world data
- Unfortunately, no easy rule-of-thumb here: Good data augmentation is still an active research field

Overview

Why do we do empirical evaluation?

What can go wrong?

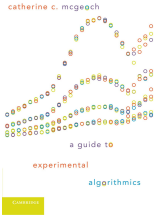
What can go wrong in AI?

Summary

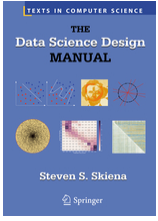
What did we learn?

- Empirical evaluation complements complexity analysis
- Even necessary in data science, where the focus is on the data, not so much on the algorithms
- Naive set up of experiments can lead to serious pitfalls
- Planning experiments carefully helps to avoid the most common pitfalls

Literature



C. C. McGeoch
A Guide to
Experimental Algorithmics
Cambridge Univ. Press, 2012



S. Skiena
The Data Science
Design Manual
Springer, 2017

Key take-aways

1. Plan experiments carefully and be prepared to retrace your steps several times.
2. Take close control of where randomness influences your results.
3. Do not look only at results—check generated data for plausibility.