Mercury

machine learning lab

ICAI: The Labs - Machine Learning in the service industry

Philipp Hager - 28th September, 2023

About the Lab

- Idea in 2018
- Start in late 2020
- 5 year runtime
- 6 PhD Students
- 2 Postdocs

Scientific Directors

Joris Mooij

UNIVERSITY OF AMSTERDAM

Frans Oliehoek

TUDelft

Matthijs Spaan

TUDelft

Onno Zoeter

Booking.com

Mission I

Learning from controlled sources

Developing a common toolkit for decision making and prediction based on data collected by previous production systems.

Examples

Evaluating and training new systems using biased data,
long-term decision making under uncertainty,
dealing with feedback loops, …

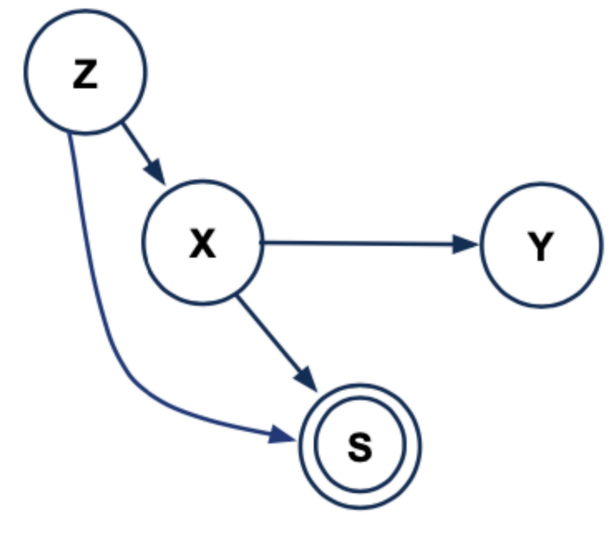
Mission II

Natural Language Processing

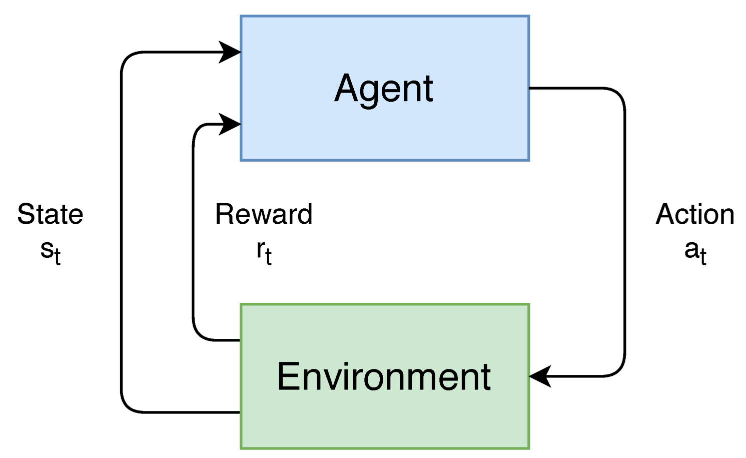
Developing explainable and robust lanuage models.

Examples
Explainable text classification.

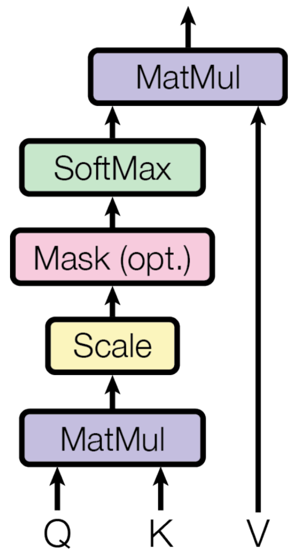
Research Areas



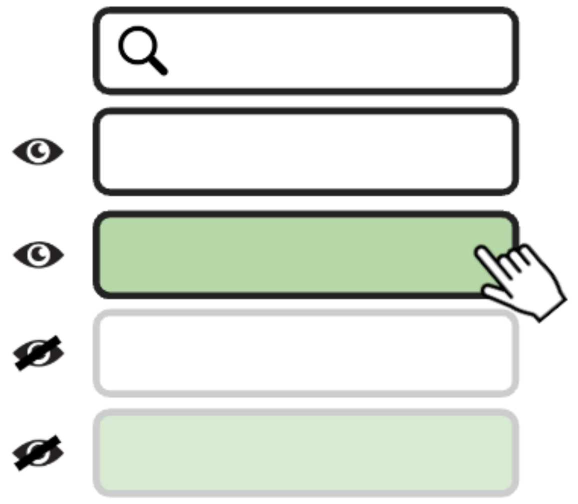
Causal Inference



Reinforcement Learning



Natural Language Processing

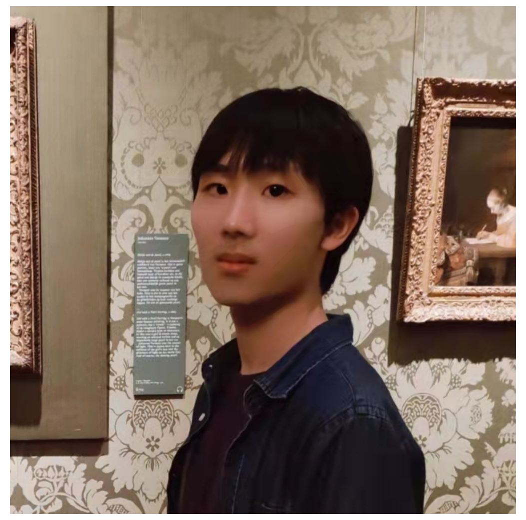


Search & Recommendation

PhDs, Postdocs, Management

Philip Boeken

Causal Inference
UvA



Leihao Chen

Causal Inference
UvA

Pedro Ferreira

Natural Language
Processing, UvA

Philipp Hager

Information Retrieval
UvA

Davide Mambelli

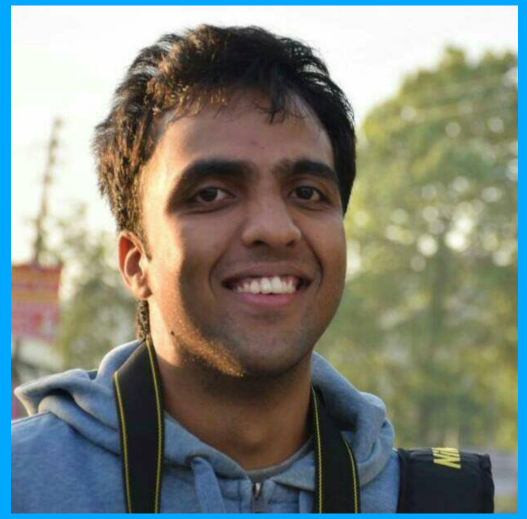
Reinforcement
Learning, TUDelft

Oussama Azizi

Reinforcement
Learning, TUDelft

Stephan Bongers

Causal Inference &
RL, TUDelft



Sourbh Bhadane

Causal Inference
UvA

Maryam Hashemi Shabestari

Project Manager
UvA

Find us online

https://icai.ai/mercury-machine-learning-lab/

Webinars

ADS Events

Publications

When Metrics Break Down
On Evaluating User Models from Clicks

Based on: An Offline Metric for the Debiasedness of Click Models

Romain Deffayet*, Philipp Hager*, Jean-Michel Renders, Maarten de Rijke - SIGIR 2023

ICAI: The Labs - Machine Learning in the service industry

Philipp Hager - 28th September, 2023

University of Amsterdam

NAVER LABS Europe

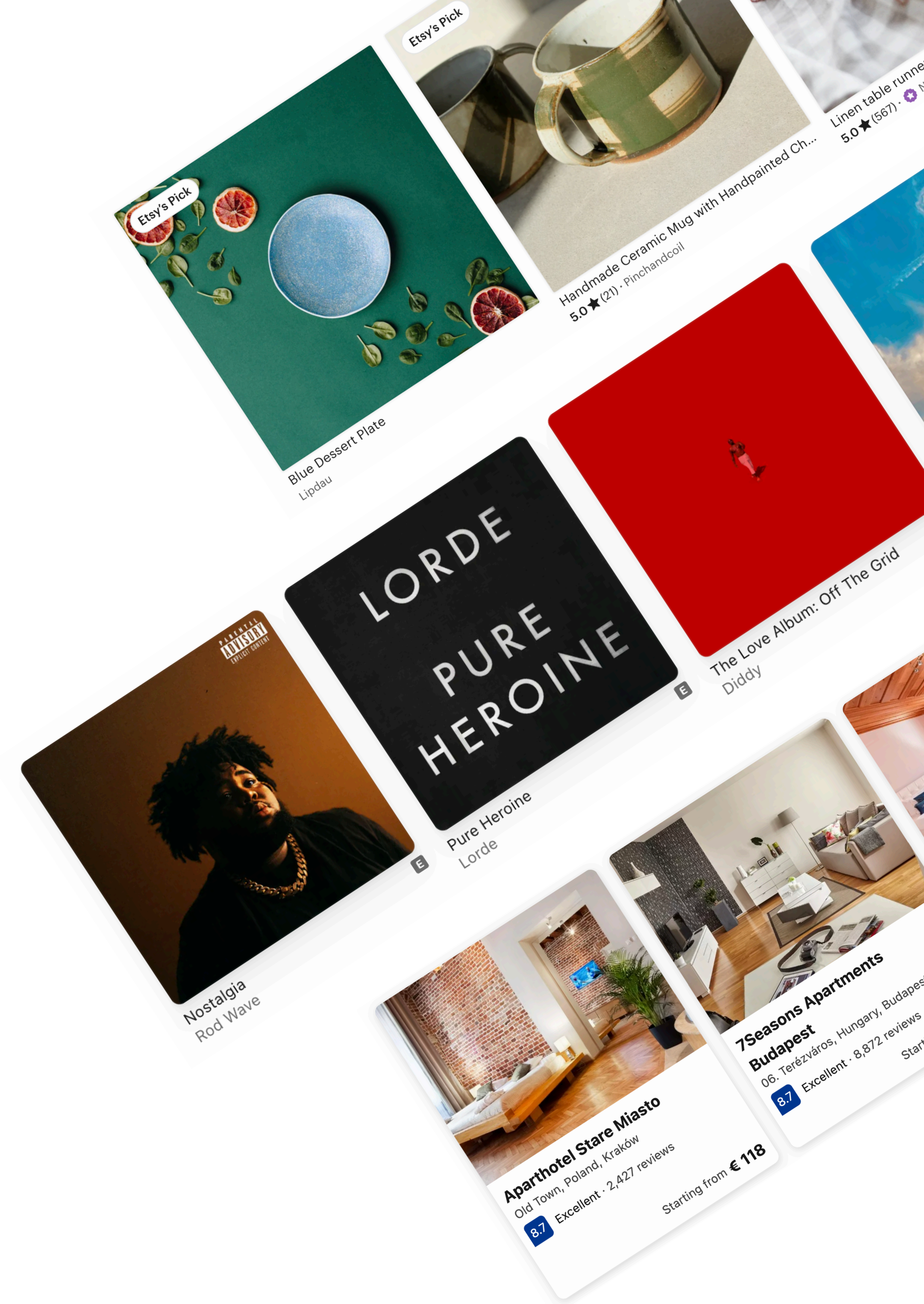
Mercury machine learning lab

Motivation

We interact with algorithms on a daily basis:
searching the web, listening to songs,
scrolling through photos, etc.

Most of our interactions are implicit:
we click, view, skip, or keep watching.

What happens if we use implicit feedback
to optimize search and recommender systems?



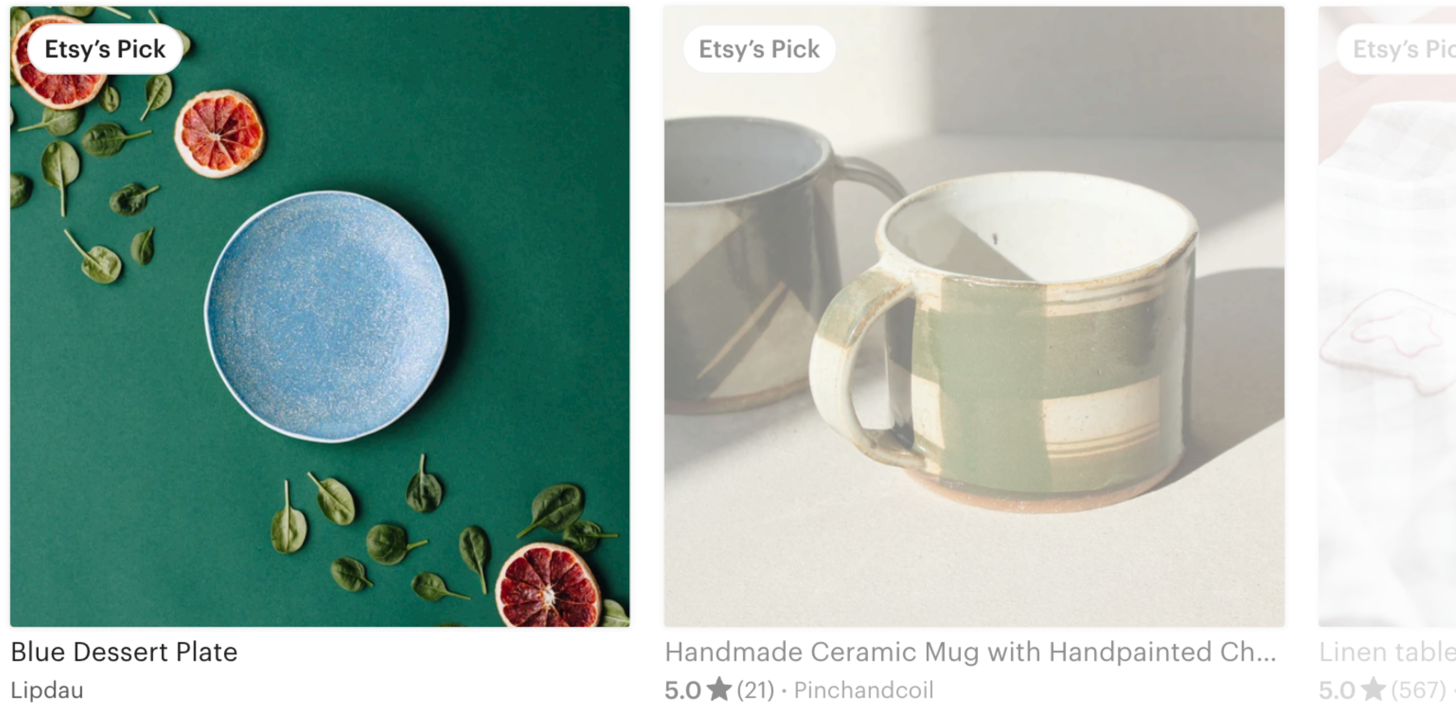
Motivation

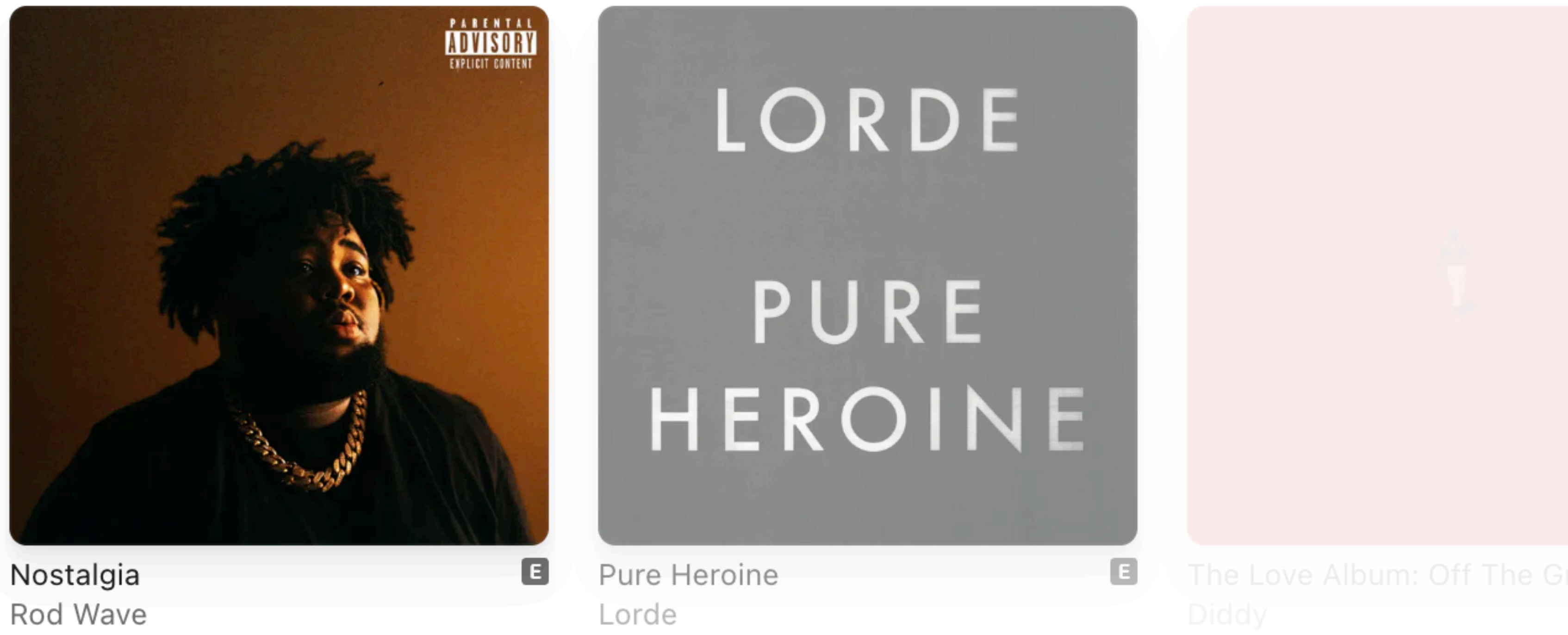
Implicit feedback is often a biased and leads to biased algorithms if used naively.

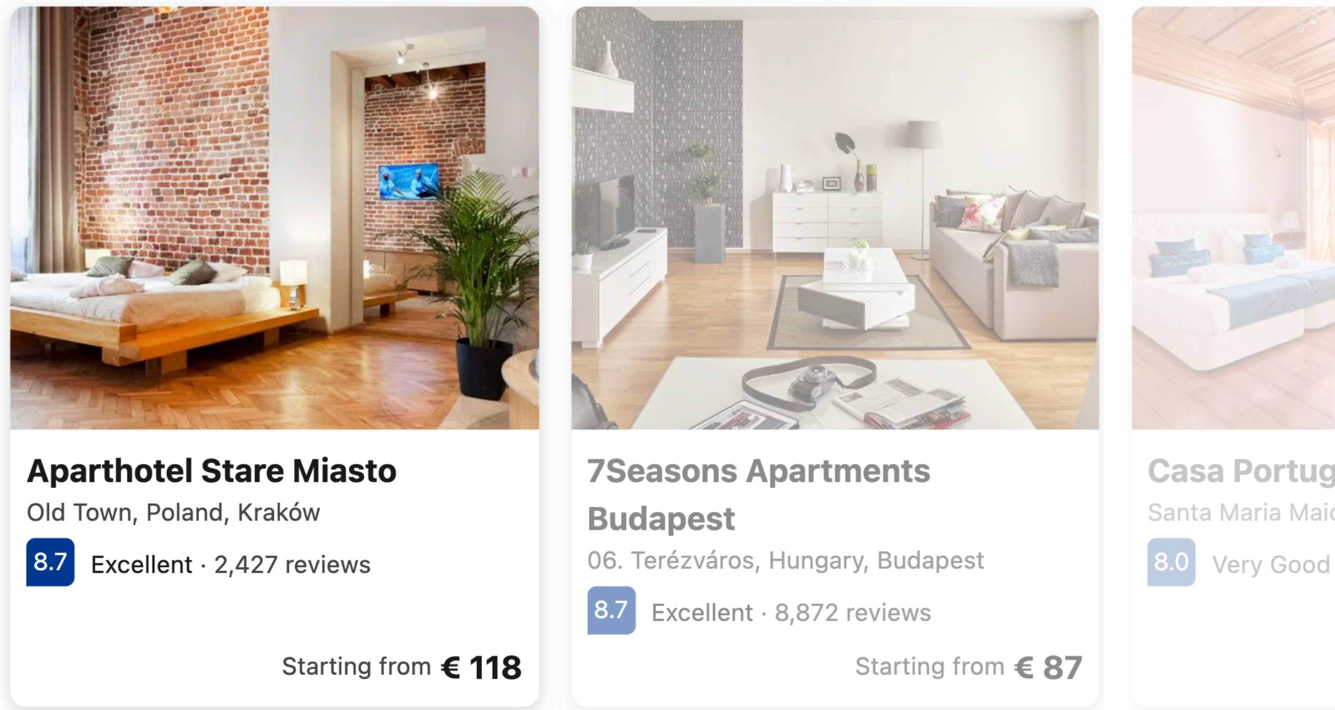
Selection bias: Users can only click on what is displayed.

Position bias: Users tend to look and click more on items at the beginning of a list.

Trust bias, presentation bias, contextual bias, ...





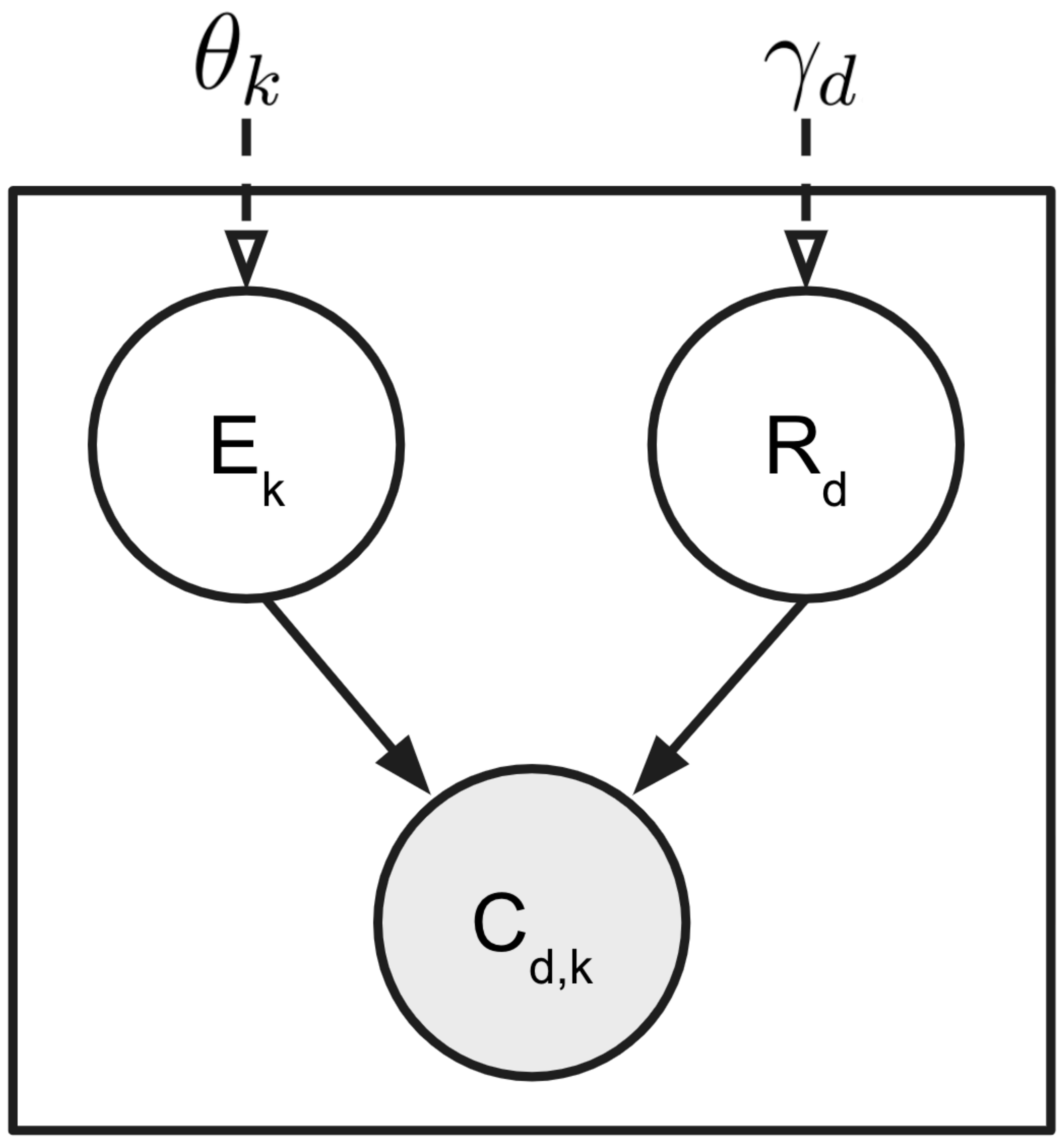


Click Models

How can we extract useful information about **biases** but also **user preferences** from clicks?

Click models **explicitly model effects** that impact a user's click, e.g.: position, trust, or item relevance.

Click models are useful for:
understanding users, evaluation metrics, estimating biases, simulating users, and predicting ad clicks.



Bayesian network of the position-based model

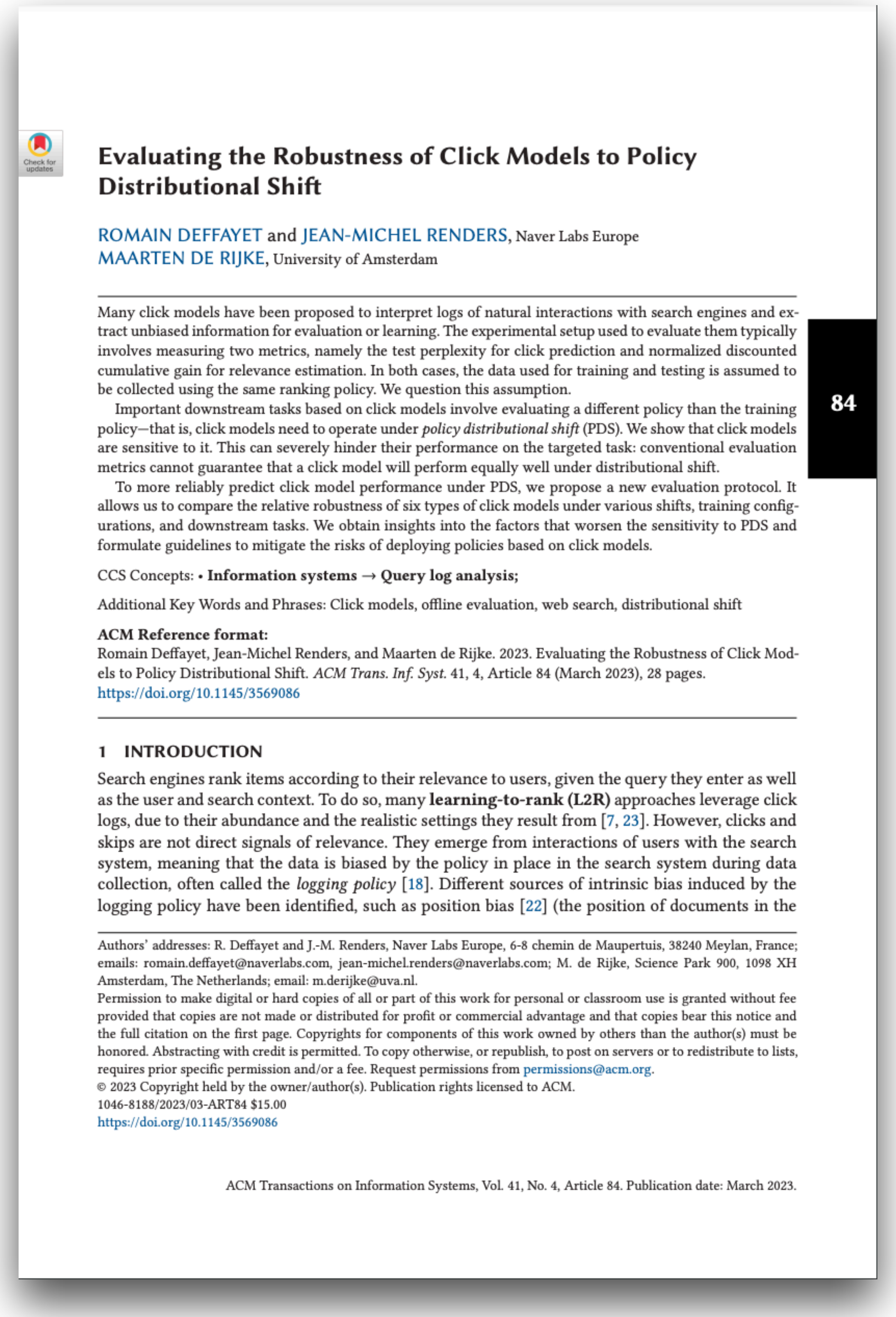
Evaluating Click Models

How do we evaluate click models?

Click prediction: Evaluating click prediction performance on an unseen test dataset (perplexity).

Ranking: Assessing predicted item relevance against expert annotations (e.g., nDCG).

Deffayet et al. show that these metrics **do not guarantee** that high-scoring **models generalize well**.



84

Evaluating the Robustness of Click Models to Policy Distributional Shift

ROMAIN DEFFAYET and JEAN-MICHEL RENDERS, Naver Labs Europe
MAARTEN DE RIJKE, University of Amsterdam

Many click models have been proposed to interpret logs of natural interactions with search engines and extract unbiased information for evaluation or learning. The experimental setup used to evaluate them typically involves measuring two metrics, namely the test perplexity for click prediction and normalized discounted cumulative gain for relevance estimation. In both cases, the data used for training and testing is assumed to be collected using the same ranking policy. We question this assumption.

Important downstream tasks based on click models involve evaluating a different policy than the training policy—that is, click models need to operate under *policy distributional shift* (PDS). We show that click models are sensitive to it. This can severely hinder their performance on the targeted task: conventional evaluation metrics cannot guarantee that a click model will perform equally well under distributional shift.

To more reliably predict click model performance under PDS, we propose a new evaluation protocol. It allows us to compare the relative robustness of six types of click models under various shifts, training configurations, and downstream tasks. We obtain insights into the factors that worsen the sensitivity to PDS and formulate guidelines to mitigate the risks of deploying policies based on click models.

CCS Concepts: • **Information systems → Query log analysis;**

Additional Key Words and Phrases: Click models, offline evaluation, web search, distributional shift

ACM Reference format:
Romain Deffayet, Jean-Michel Renders, and Maarten de Rijke. 2023. Evaluating the Robustness of Click Models to Policy Distributional Shift. *ACM Trans. Inf. Syst.* 41, 4, Article 84 (March 2023), 28 pages.
https://doi.org/10.1145/3569086

1 INTRODUCTION

Search engines rank items according to their relevance to users, given the query they enter as well as the user and search context. To do so, many **learning-to-rank (L2R)** approaches leverage click logs, due to their abundance and the realistic settings they result from [7, 23]. However, clicks and skips are not direct signals of relevance. They emerge from interactions of users with the search system, meaning that the data is biased by the policy in place in the search system during data collection, often called the *logging policy* [18]. Different sources of intrinsic bias induced by the logging policy have been identified, such as position bias [22] (the position of documents in the

Authors' addresses: R. Deffayet and J.-M. Renders, Naver Labs Europe, 6-8 chemin de Maupertuis, 38240 Meylan, France; emails: romain.deffayet@naverlabs.com, jean-michel.renders@naverlabs.com; M. de Rijke, Science Park 900, 1098 XH Amsterdam, The Netherlands; email: m.derijke@uva.nl.
Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2023 Copyright held by the owner/author(s). Publication rights licensed to ACM.
1046-8188/2023/03-ART84 $15.00
https://doi.org/10.1145/3569086

Deffayet et al.
TOIS 2023

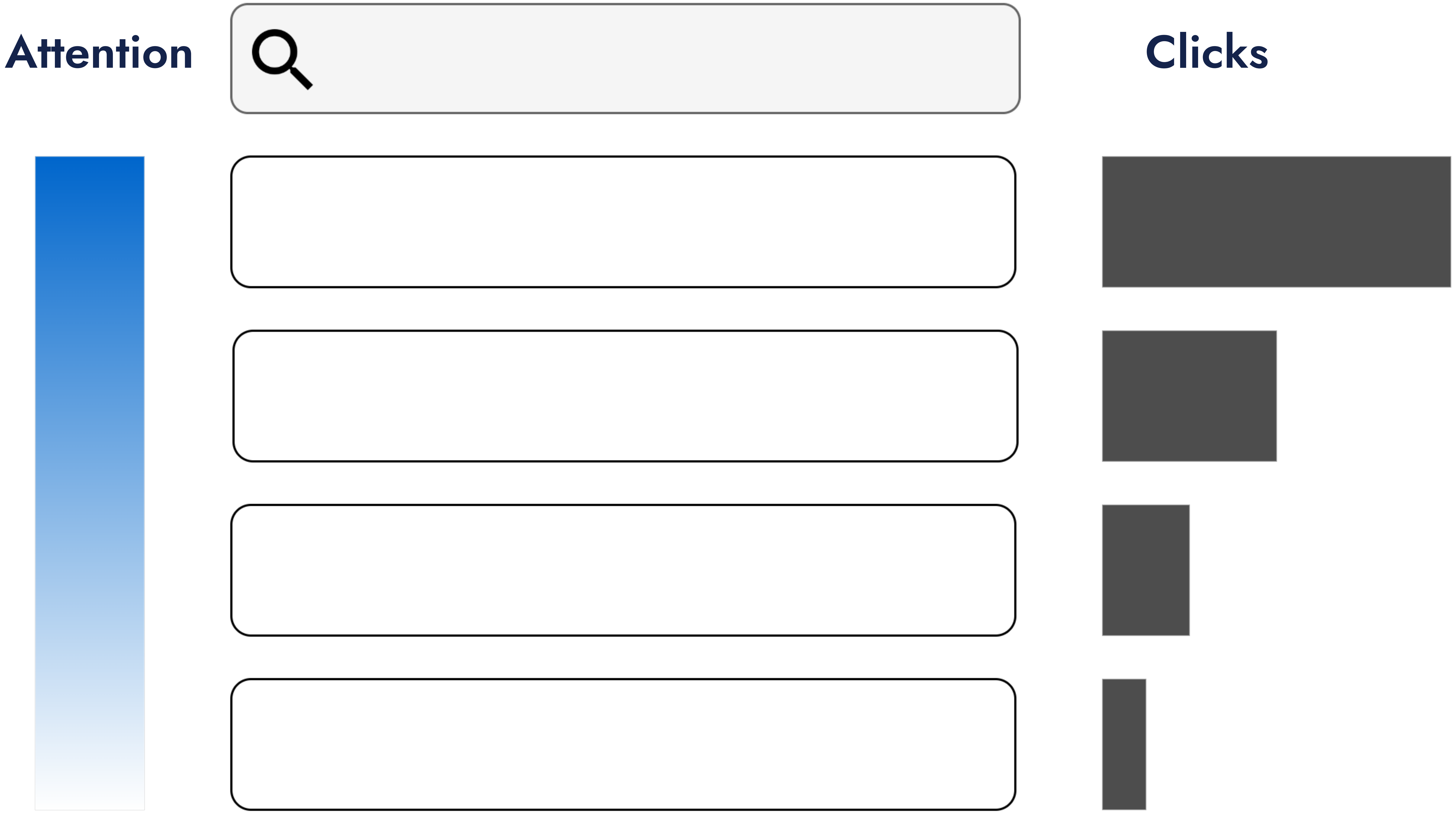
When metrics break down

Scenario I: Naive and biased click models
can score high in ranking metrics, especially when:

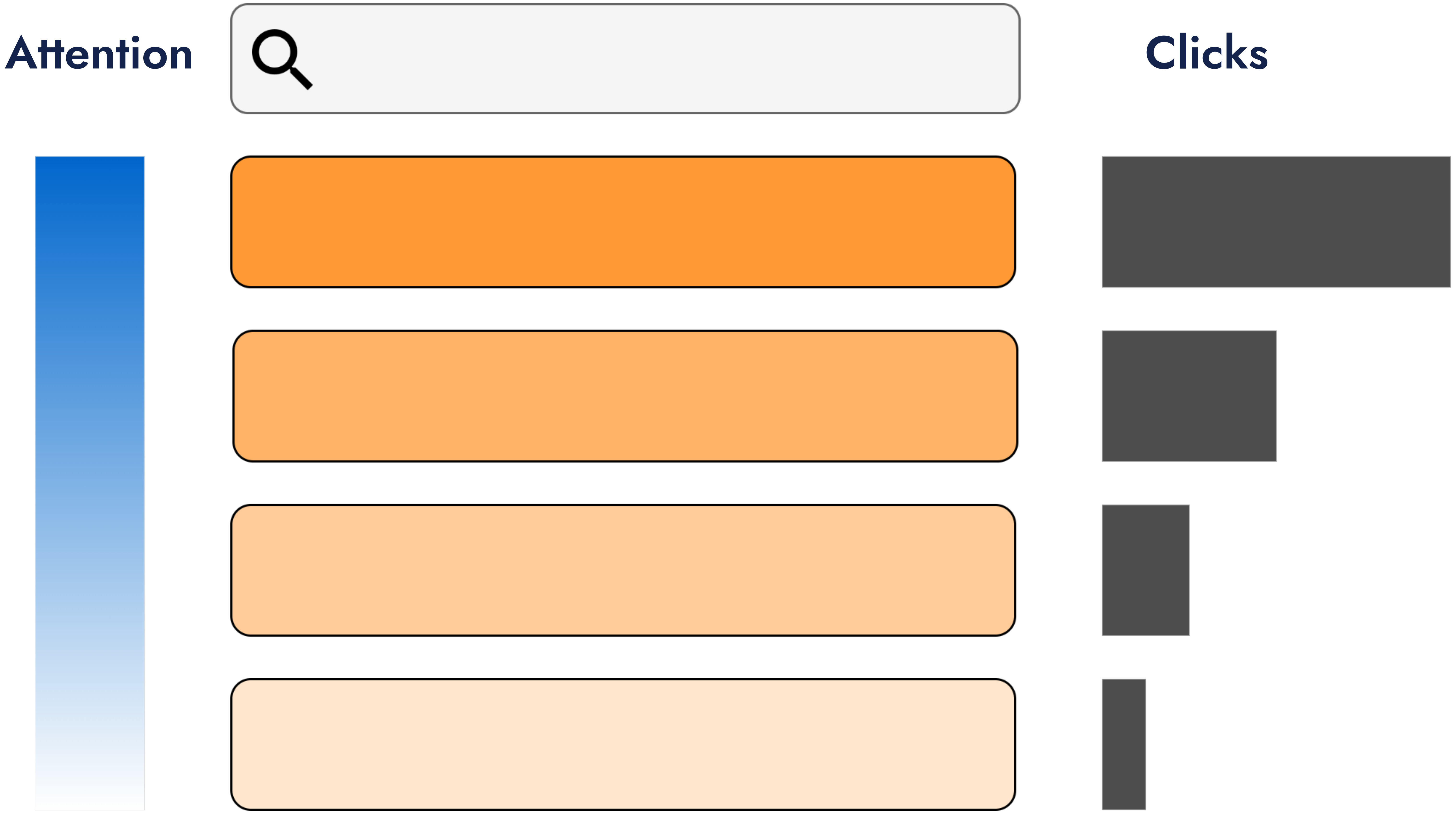
a.) The system collecting the data is **already very good.**

b.) The system tends to **display similar rankings.**

When the production system is already very good

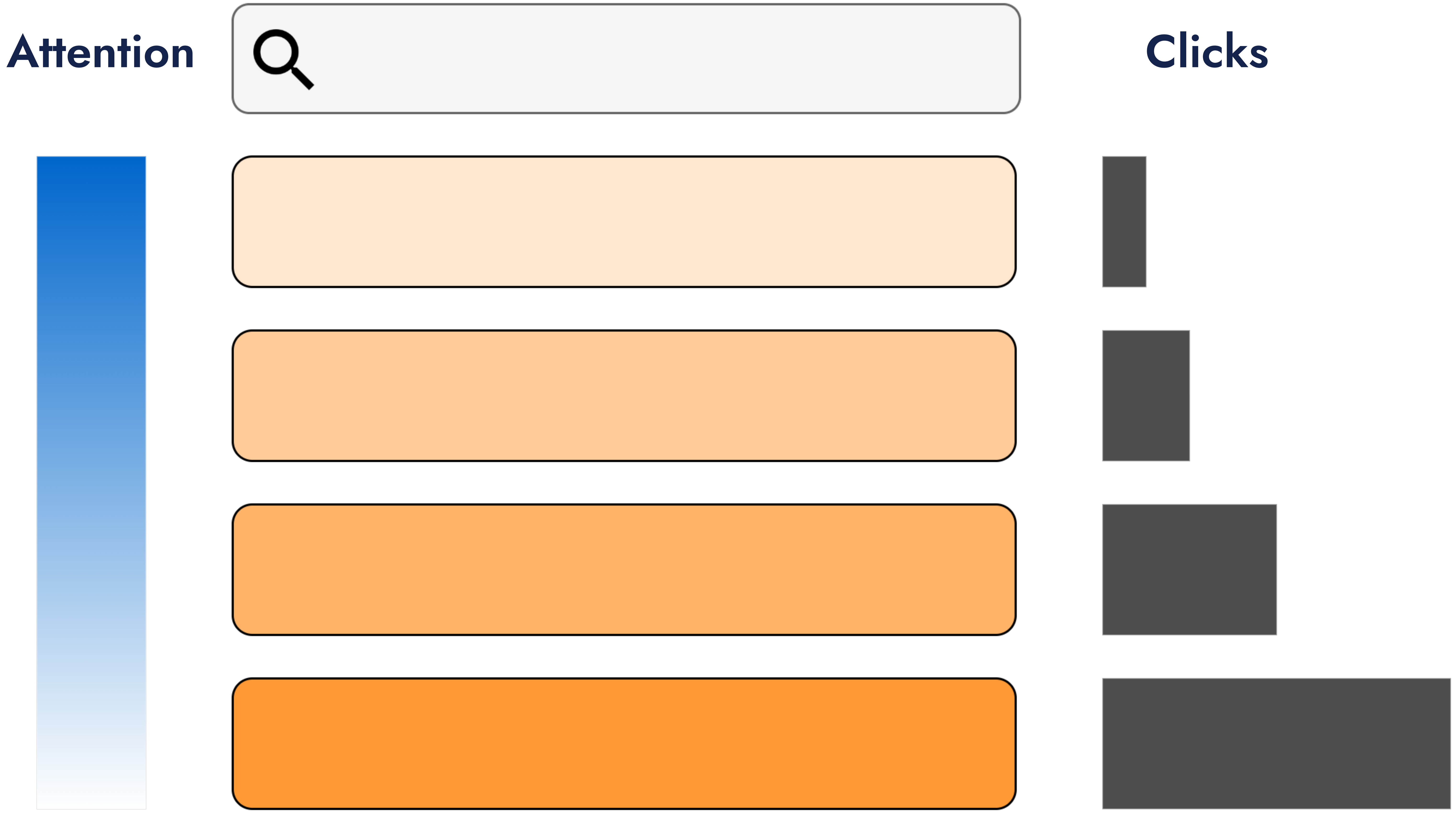


When the production system is already very good



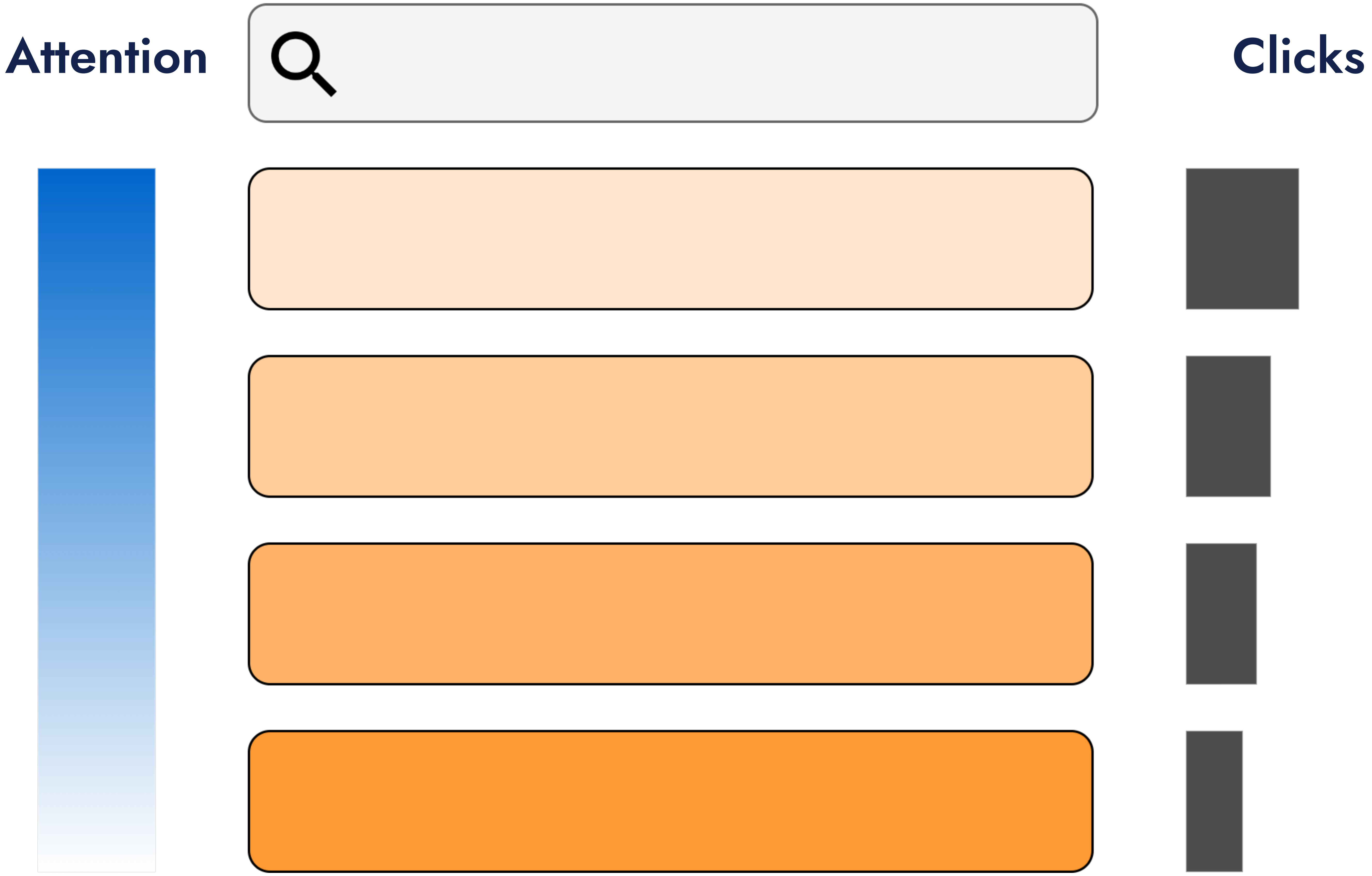
When the current ranking is near-optimal,
just replicating the current system achieves high ranking performance.

When the production system is already very good



But what if we predict clicks for the inverted ranking?

When the production system is already very good



The actual click distribution would look more like this…
the naive model does not generalize to unseen data.

When metrics break down

Scenario II: Deffayet et al. show in simulation that **perpelxity is less reliable when no models fits the observed user behavior.**

Perplexity quantifies how well we can predict clicks on **the current dataset,** there are **little guarantees for completely unseen rankings**.

More diverse test sets

Can't we avoid these problems by evaluating on **more diverse test sets?**

Having more diverse test sets helps.

However, it might be **costly or impractical to introduce a lot of variability** into real-world production systems.

More generally, ranking operates in factorial complexity O(N!), most datasets can only cover a fraction of all possible rankings.

Other ways to detect this problem?

In all of these settings, a main problem is that replicating (without understanding) the current produciton system is very effective.

How would you detect a cheater in school?

Comparing grades does not work,
students who cheat can score high grades just by copying.

Other ways to detect this problem?

In all of these settings, a main problem is that replicating (without understanding) the current produciton system is very effective.

How would you detect a cheater in school?

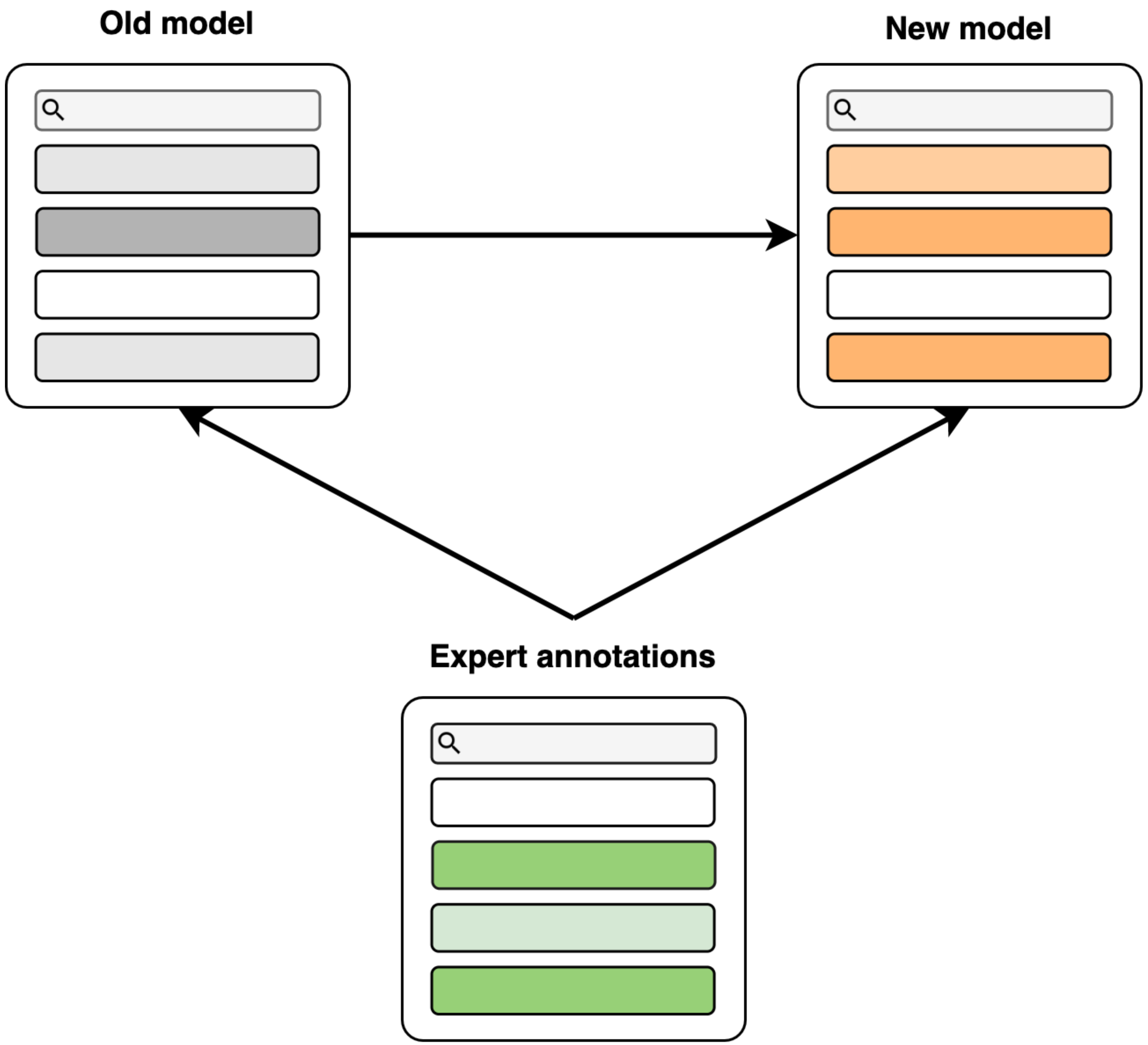
Comparing grades does not work,
students who cheat can score high grades just by copying.

We compare their mistakes!

CMIP

Using a small **set of expert annotations**, we can quantify if a new model makes similar mistakes to the previous model.

We leverage **conditional mutual information** estimation.



CMIP

Note that CMIP is a **necessary condition and not sufficient.**

Predicting random clicks scores well in CMIP, but is a bad click model.

CMIP extends the existing evaluation protocol.

Evaluation

We find in large-scale simulation experiments that
CMIP in conjunction with existing metrics:

1.) Significantly improves **predicting the downstream performance** of click models.

2.) Helps to **pick models that predict clicks well** on unseen rankings.

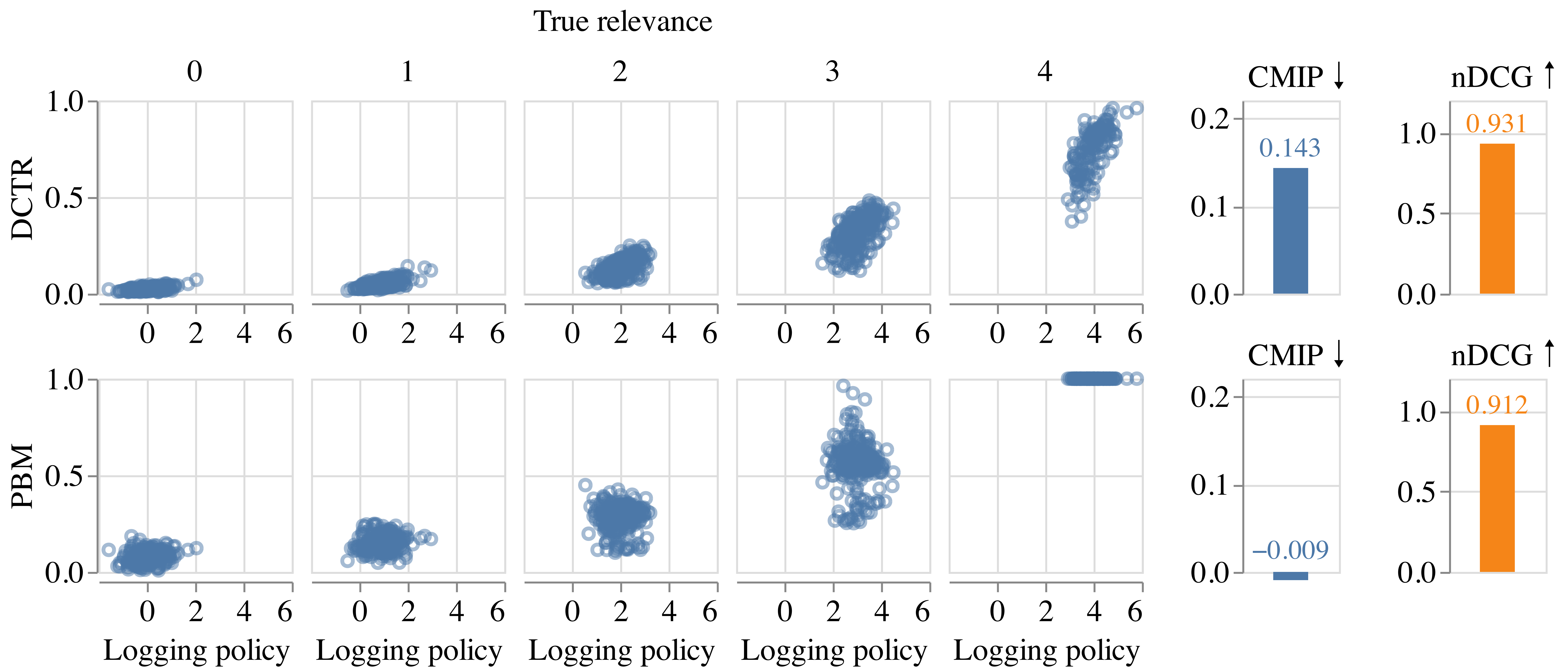
Limitations

Our work relies on:

- The **availability of expert annotations** / a ground truth.
- The assumption that there is **no systematic disagreement between experts and user clicks.**
- Simulation experiments (so far).

Paper **Code**

CMIP



A naive model (DCTR) outperforms an unbiased model (PBM) in terms of nDCG, but our CMIP metric catches the replication behavior.