Unbiased Learning to Rank Meets Reality: Lessons from Baidu's Large-Scale Search Dataset

Philipp Hager*, Romain Deffayet*, Jean-Michel Renders, Onno Zoeter, Maarten de Rijke

Submitted to SIGIR Reproducibility 2024

Soos Talk

Philipp Hager - 19th March, 2024

University of Amsterdam
NAVER LABS Europe
Mercury machine learning lab

Motivation

- Top-ranked documents in web search gather more attention and clicks
- **Biases:** Position bias, trust bias, outlier bias, surrounding items, …
- **Unbiased learning to rank** learns ranking models from biased clicks

Eye tracking study in web search[1]

[1] Granka, Laura, Matthew Feusner, and Lori Lorigo. Eyetracking in online search. In Passive eye monitoring 2008

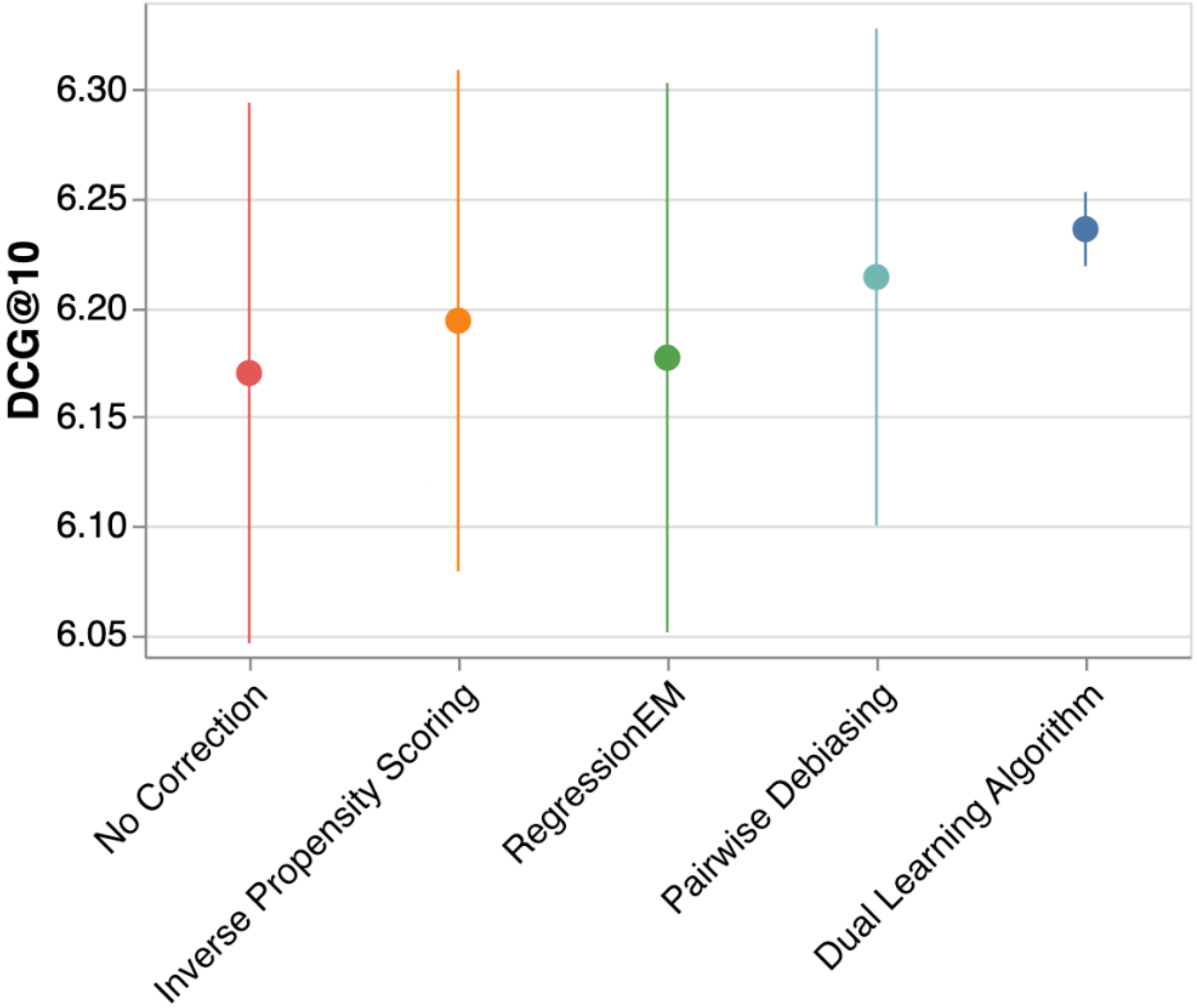
Motivation

- Most (academic) work in unbiased learning to rank is evaluated in **semi-synthetic simulation:**[1]
  - Real queries/documents, synthetic clicks
  - But does ULTR work in reality?
- **Baidu ULTR** is the first large-scale web search dataset with real clicks for offline evaluation (≈380M queries, 1.2B user sessions)[2]

[1] Ai, Qingyao, et al. Unbiased Learning to Rank: Online or Offline? In TOIS 2021.

[2] Zou, Lixin, et al. A Large Scale Search Dataset for Unbiased Learning to Rank. In NeurIPS 2022.

A reality check for ULTR at NeurIPS 2022



Four ULTR methods using cross-encoders trained on the Baidu-ULTR dataset[1]
Models were trained on user clicks and evaluated expert annotations

[1] Zou, Lixin, et al. A Large Scale Search Dataset for Unbiased Learning to Rank. In NeurIPS 2022.

Why reproduce this work?

- The finding that **ULTR does not outperform a naive baseline** warrants more scrutiny
- **WSDM Cup participants reported much higher ranking performance**[2], in fact, all rankers are **outperformed by random shuffling** (DCG@10≈6.25 vs DCG@10≈6.69)
- The authors did **not properly estimate position bias** and **20% of the dataset consists of two documents**
- Zou et al. [1] focused on **pointwise methods**

[1] Zou, Lixin, et al. A Large Scale Search Dataset for Unbiased Learning to Rank. In NeurIPS 2022.
[2] Chen, Xiaoshu, et al. Multi-feature integration for perception-dependent examination-bias estimation. In WSDM Cup 2023.

Motivation

RQ1: Does unbiased learning-to-rank improve performance on the Baidu ULTR dataset over naive, non-debiasing models?

RQ2: How do ranking losses and input features affect ranking performance on Baidu ULTR?

RQ3: Can ULTR methods be applied during language model training?

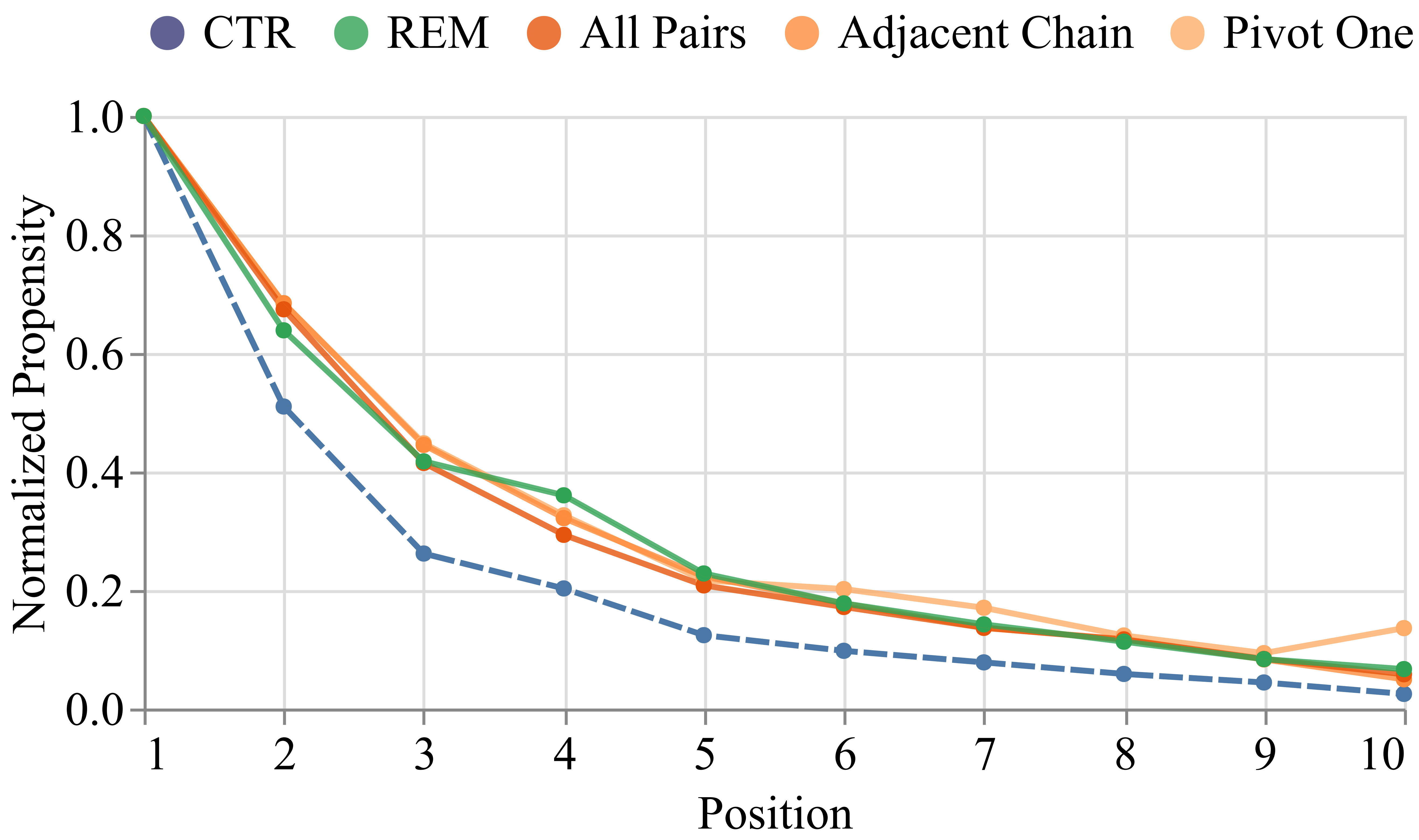
The Baidu ULTR dataset

Overview

- **Training:** 1.2B user sessions randomly sampled from Baidu in April 2022 (usually top 10 docs per session)
- **Testing:** 7K annotated queries (≈400K query-document pairs, up to top 1,000 docs)
- **Content features:** Query, title, abstract tokenized with a **private vocabulary -> no pretrained LLMs**
- **User feedback:** clicks, dwell time, skipping, bouncing, …
- **Presentation features:** item type, height, position, …

Baidu search engine in March 2024

Position bias on Baidu ULTR



Four different position bias estimation methods arrive at a similar bias estimation, hinting at a noticeable position bias in the dataset.

Therefore, we expect ULTR methods to be helpful on this dataset.

Experimental Setup

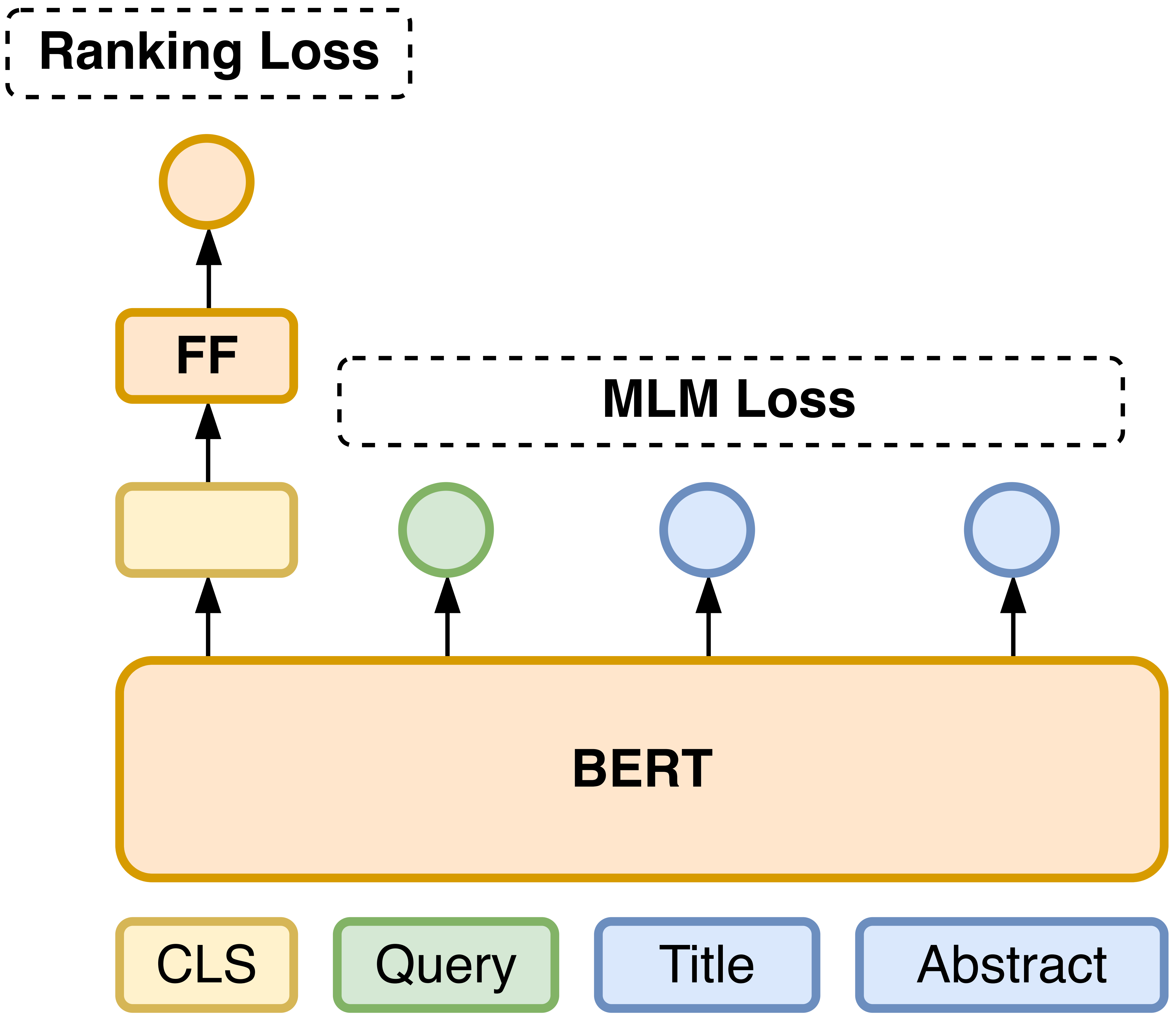
Cross Encoder Setup

MonoBERT cross encoder

- BERT base: 12 layers, 12 heads, 768 dims
- 2M steps x 256 batch size
- HuggingFace FlaxBERT
  (≈50% faster than PyTorch in our setup)

Losses

- **Ranking loss:** Binary cross-entropy on clicks
- **MLM loss:** Masking rate of 0.3



Reranking Dataset

We use CLS token of the pretrained MonoBERT models as embeddings for a downstream reranking model:

- Three partitions for training, one partition for validation/testing on clicks
- Pre-computed **query-document embeddings**
  - **Original Baidu MonoBERT** CLS token (pre-trained on click prediction)
  - **Our MonoBERT** CLS token (pre-trained on click prediction)
  - **Our LTR features** (TF-IDF, BM25, QL Jelinek Mercer, QL Dirichlet)

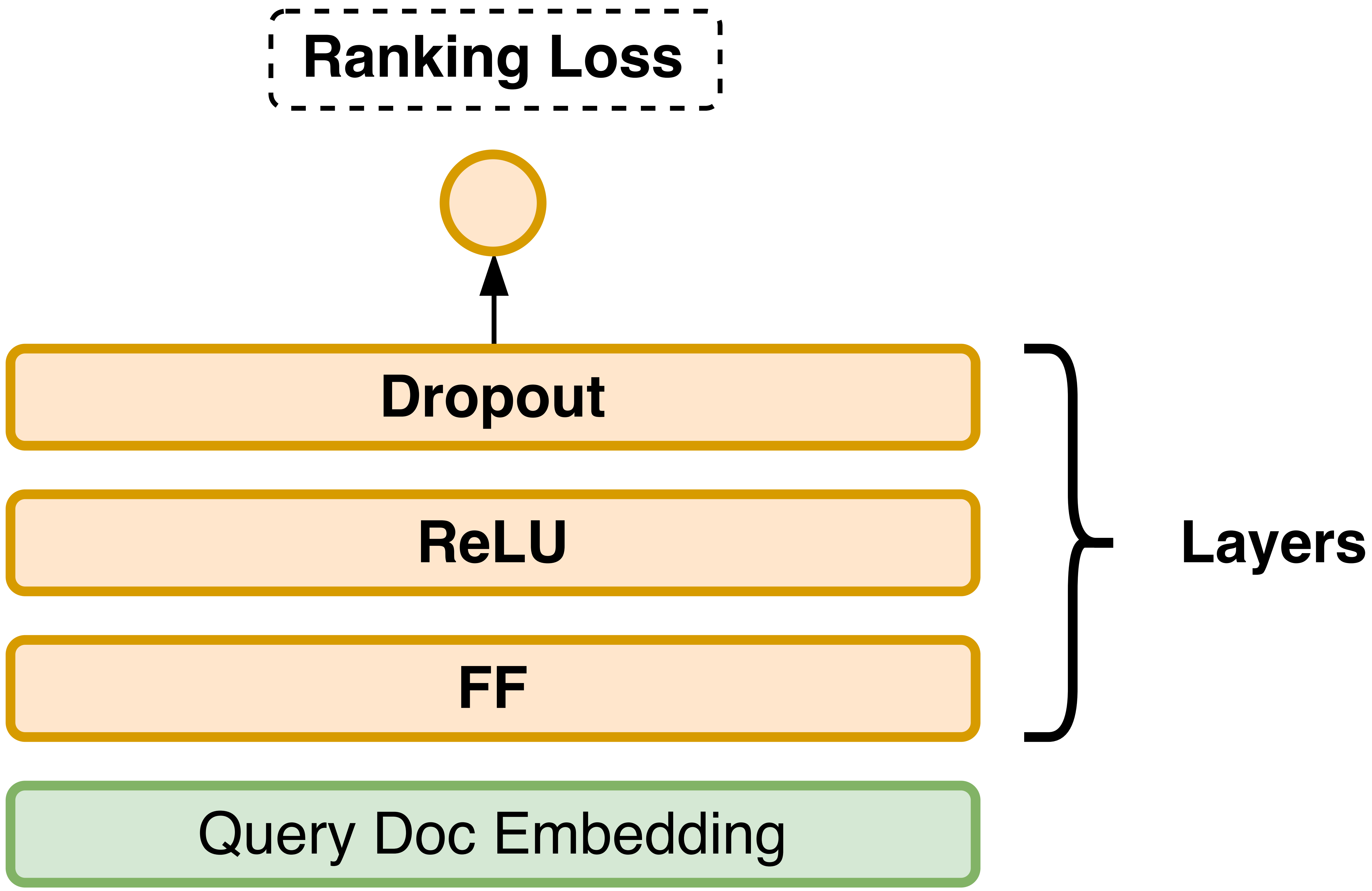
Reranking Model

Feed forward ReLU network

- 64 - 1024 hidden dims
- 2 - 5 layers
- Optional dropout
- Log1p normalization for LTR

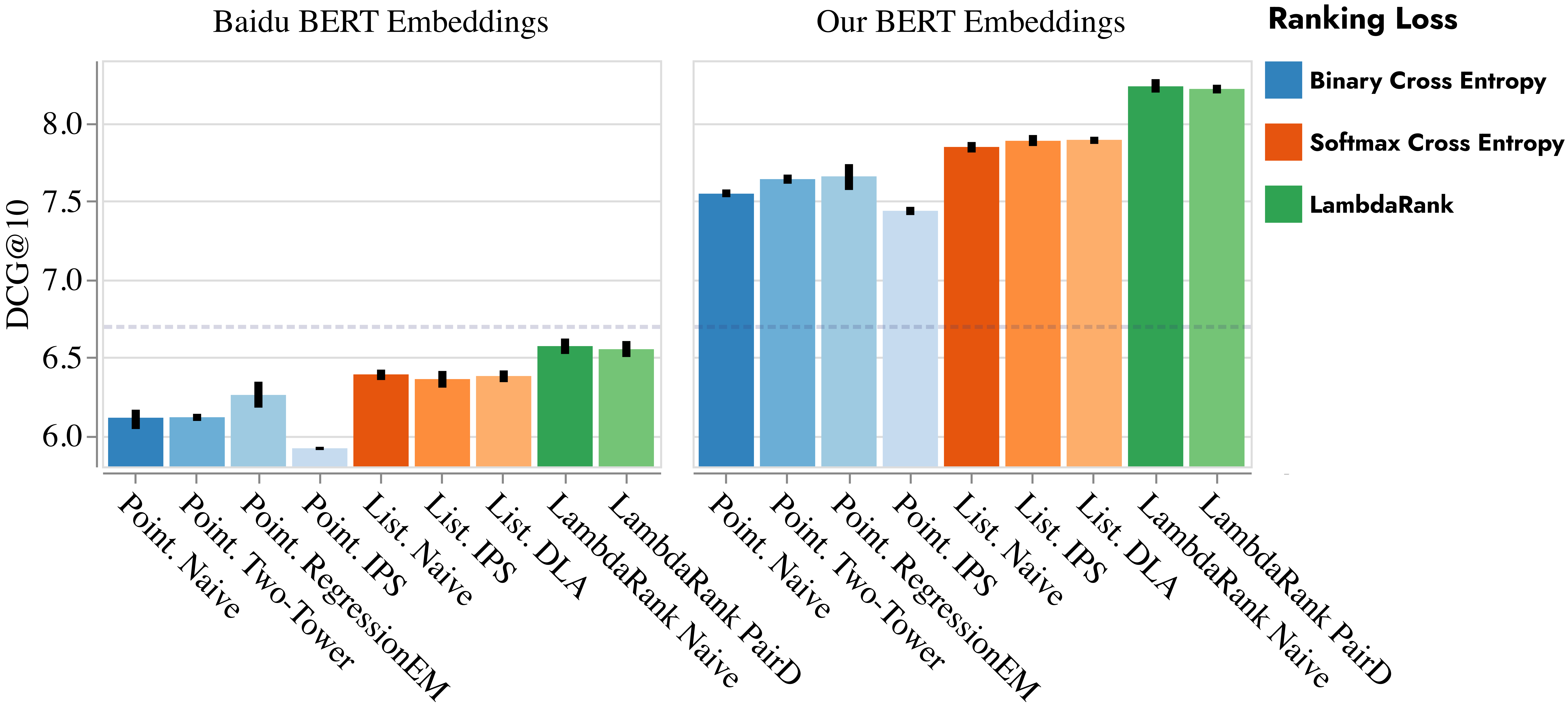
Ranking Loss

- **Pointwise:** Binary cross-entropy
- **Listwise:** Softmax cross-entropy
- **Listwise:** LambdaRank

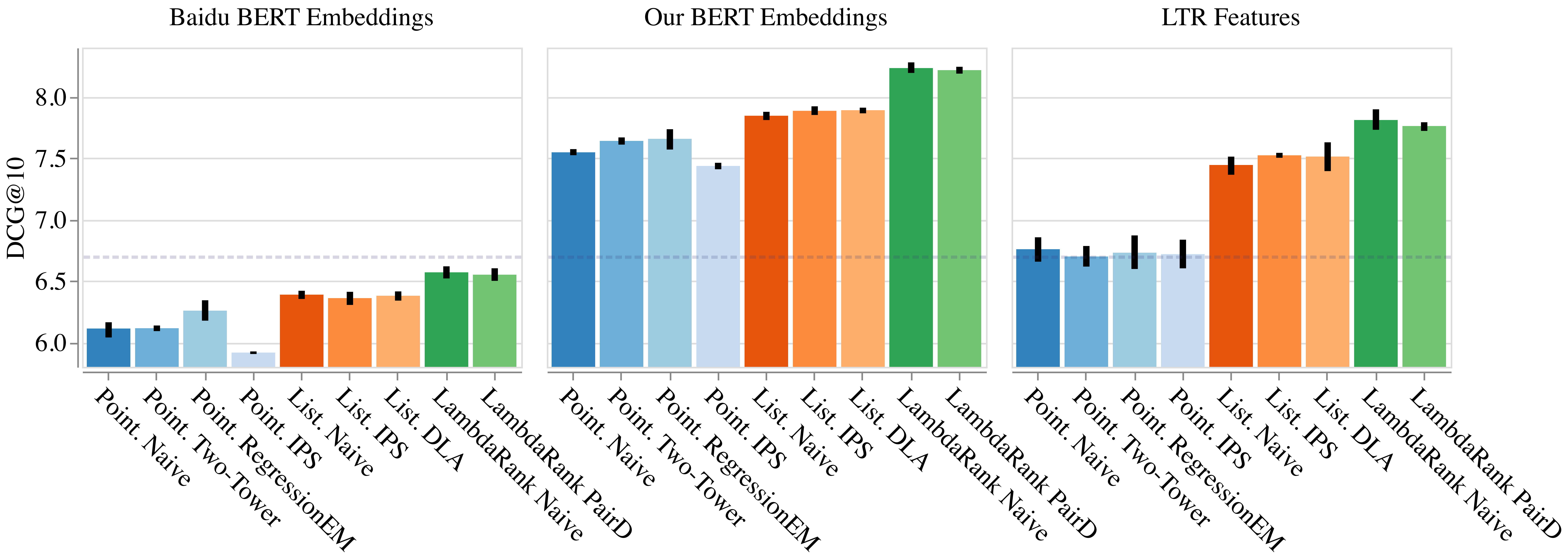


Results

RQ 1: Does ULTR improve performance?



RQ 2: How do results compare across features and losses?



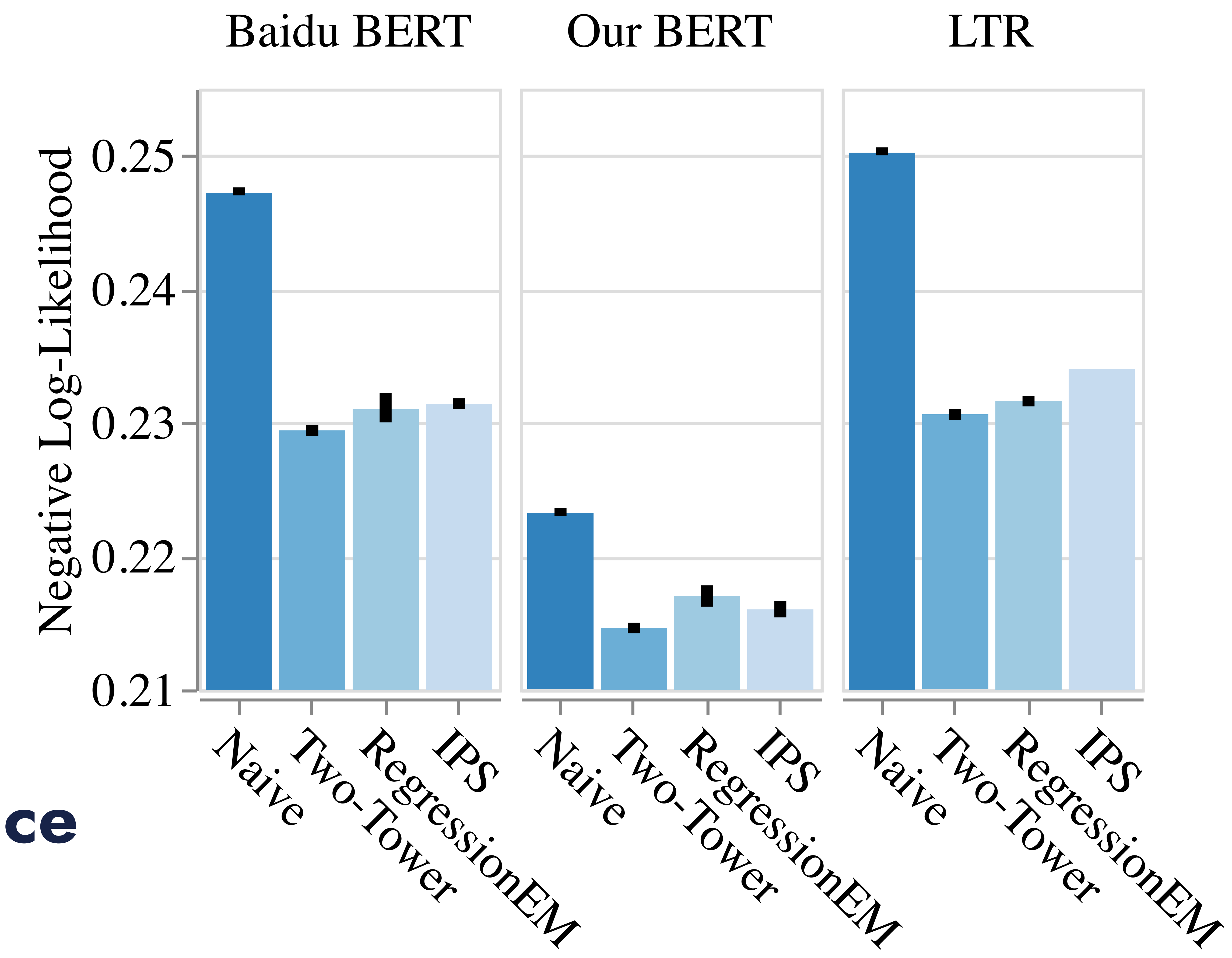
RQ3: Can ULTR methods be applied during language model pre-training?

- We train three pointwise and three listwise MonoBERTs from scratch
- Pointwise < Listwise
- IPS-based methods degrade performance
- Results need further investigation

Model	DCG@10 ↑	NLL ↓
Pointwise Naive	7.251	0.227
Pointwise Two-Tower	7.456	0.217
Pointwise IPS	6.296	0.317
Listwise Naive	8.478	-
Listwise IPS	7.450	-
Listwise DLA	7.802	-

Ranking vs. click prediction

- ULTR consistently leads to better click prediction
- **Click prediction does NOT translate to better ranking performance**
- BM25 alone achieves a DCG@10≈9.54, better than any trained model
- **Click prediction and ranking performance on annotations are diverging objectives**



Why might ULTR not help?

- No position bias
- **User behavior more complex**
- Lack of variability leads to a lack of identifiability
- Strong logging policy
- **Distribution shift between training and testing (top-10 vs top-1000)**
- **User-annotator disagreement**

Implications for the field

- **Our results confirm the original authors,** ULTR leads only to (at best) marginal improvements on the largest ULTR public dataset
- **Interaction between ULTR and transformers** needs further exploration
- **Measuring success in ULTR** (clicks vs. annotations) is non-trivial

Lastly, we only challenge the **validity of ULTR on this particular dataset**

Contributions

- We publish three smaller, cleaned, and pre-processed Baidu ULTR datasets with BERT embeddings and LTR features
- We publish Jax implementations of five standard ULTR methods
- We train six MonoBERT models from scratch, releasing their weights
- We publish code for four position bias estimation methods

Lessons

- Always evaluate a random baseline
- Understand your parameter space using random/grid search before using more advanced Bayesian tuning methods (sensible defaults work well)
- Connecting with the original authors (w. Maarten) was very helpful
- Jax can be incredibly fast but is hard to debug, is not as mature as PyTorch, and has subtle API differences (NumPy vs PyTorch)

Backup

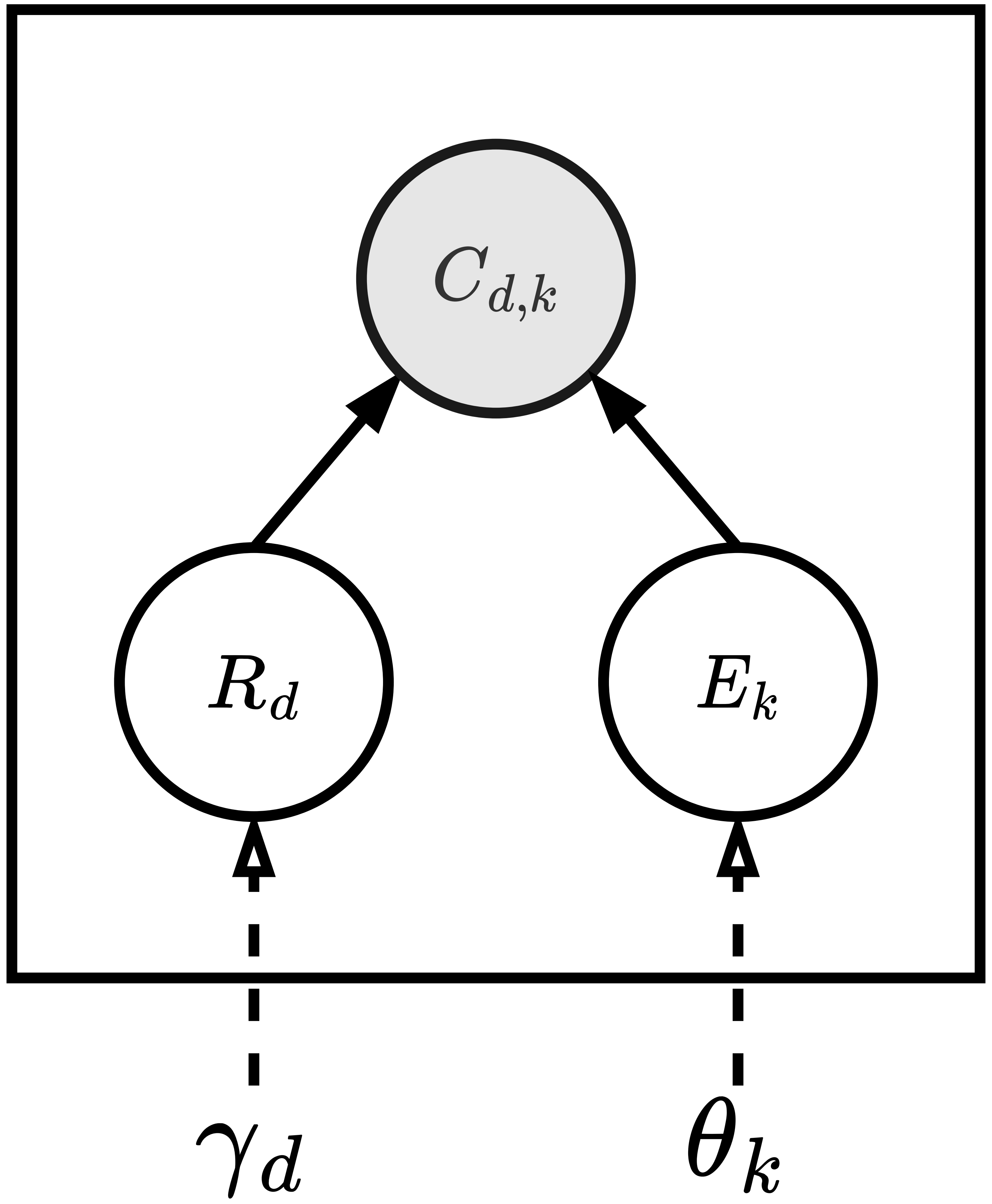
Unbiased Learning to Rank

Position Based Model

PBM

Users click on examined and relevant items:

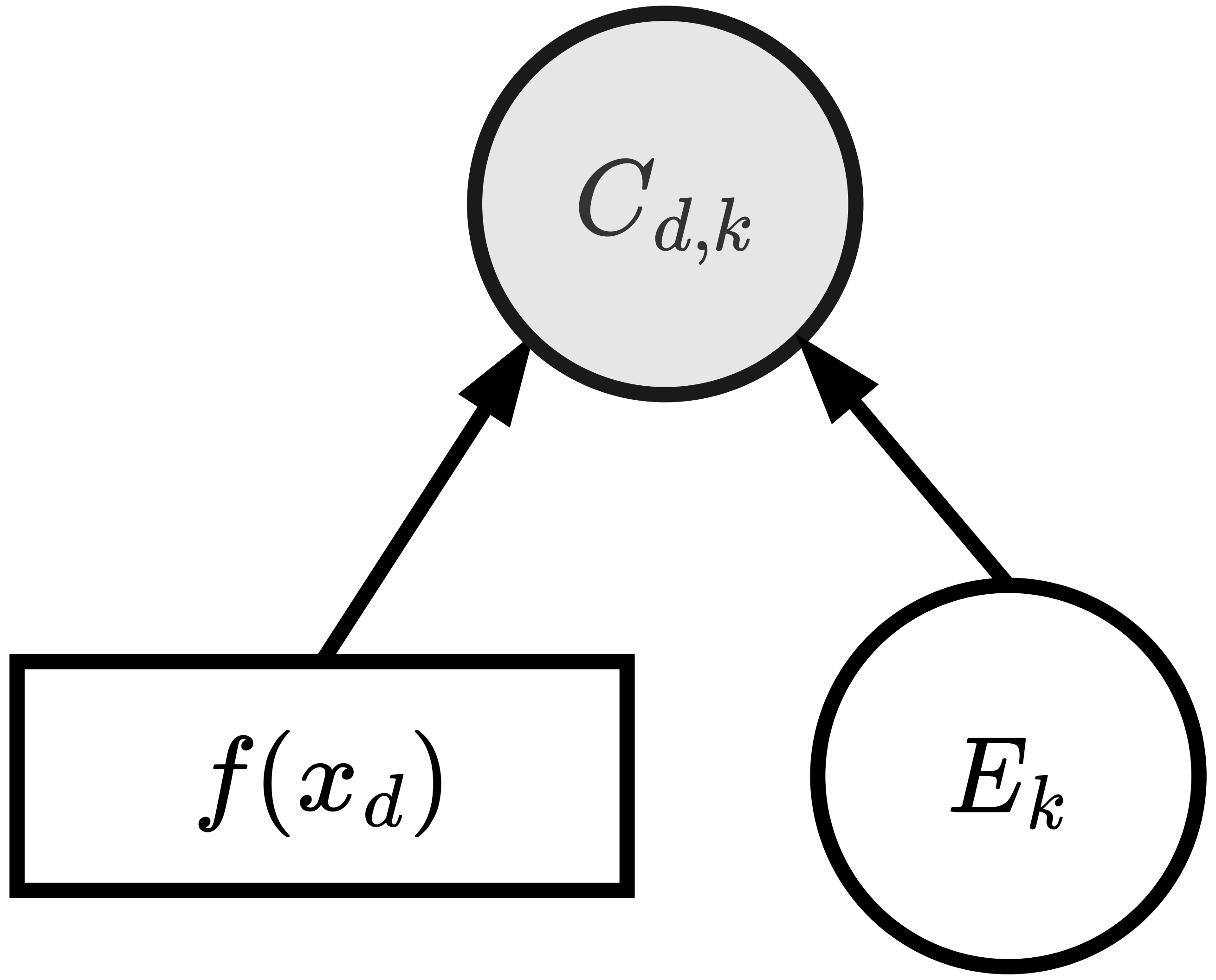
$$P(C = 1 \mid d, k) = \underbrace{P(E = 1 \mid k)}_{\text{Prob. of examining rank k}} \cdot \underbrace{P(R = 1 \mid d)}_{\text{Prob. of doc d being relevant}}$$



Jointly modeling bias & relevance

Two Towers: Mirrors the PBM in a neural network setup, optimizes parameters using BCE.

RegressionEM: Explicitly computes posterior distributions of bias and relevance in loss.



Prob. of doc d being relevant

Prob. of examining rank k

[1] Agarwal, Aman, et al. Estimating position bias without intrusive interventions. In WSDM 2019.

Inverse Propensity Scoring

Reweight clicks by position bias to estimate unbiased relevance:

$$P(R = 1 \mid d, k) = \frac{P(C = 1 \mid d)}{P(E = 1 \mid k)}$$

For example, if an item has a 25% chance of being viewed, each click is weighted 4x

Requires estimate of position bias (intervention harvesting, RegressionEM)

[1] Joachims, Thorsten, Adith Swaminathan, and Tobias Schnabel. Unbiased learning-to-rank with biased feedback. In WSDM 2017.

Dual Learning Algorithm

Uses IPS to learn position bias

1. Estimate relevance given the current position bias estimate (same as IPS):

$$P(R = 1 \mid d, k) = \frac{P(C = 1 \mid d)}{P(E = 1 \mid k)}$$

2. Estimate position bias given the current relevance estimate:

$$P(E = 1 \mid k) = \frac{P(C = 1 \mid d)}{P(R = 1 \mid d, k)}$$

[1] Ai, Qingyao, et al. Unbiased learning to rank with unbiased propensity estimation. In SIGIR 2018.

Pairwise Debiasing / Unbiased LambdaMART

Estimates propensity ratios for clicked and non-clicked documents:

$$\frac{\mathscr{L}(\tilde{r}(q,d);c)}{\tilde{e}^{+}(k)\cdot\tilde{e}^{-}(k)} + \left\| \tilde{e}^{+}(k) \right\| + \left\| \tilde{e}^{-}(k) \right\|$$

$\tilde{e}^{-}(k)$ is the reciprocal of the probability of an unclicked document being irrelevant at position k

Assumptions challenged in Oosterhuis [2]

[1] Hu, Ziniu, et al. Unbiased lambdamart: an unbiased pairwise learning-to-rank algorithm. In WWW 2019.
[2] Oosterhuis, Harrie. Reaching the end of unbiasedness: Uncovering implicit limitations of click-based learning to rank. In ICTIR 2022.