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

Philipp Hager[1,3], **Romain Deffayet**[1,2], Jean-Michel Renders[2], Onno Zoeter[3], Maarten de Rijke[1]

[1]University of Amsterdam, [2]Naver Labs Europe, [3]Mercury ML Lab - Booking.com

SIGIR Reproducibility Track

Philipp Hager - July 16th, 2024

University of Amsterdam

NAVER LABS Europe

Mercury machine learning lab

Unbiased learning to rank (ULTR)

- **Position bias:** Top-ranked items gather more user attention and clicks
- We should not naively treat clicks/non-clicks as positive/negative feedback
- **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

Evaluation in simulation

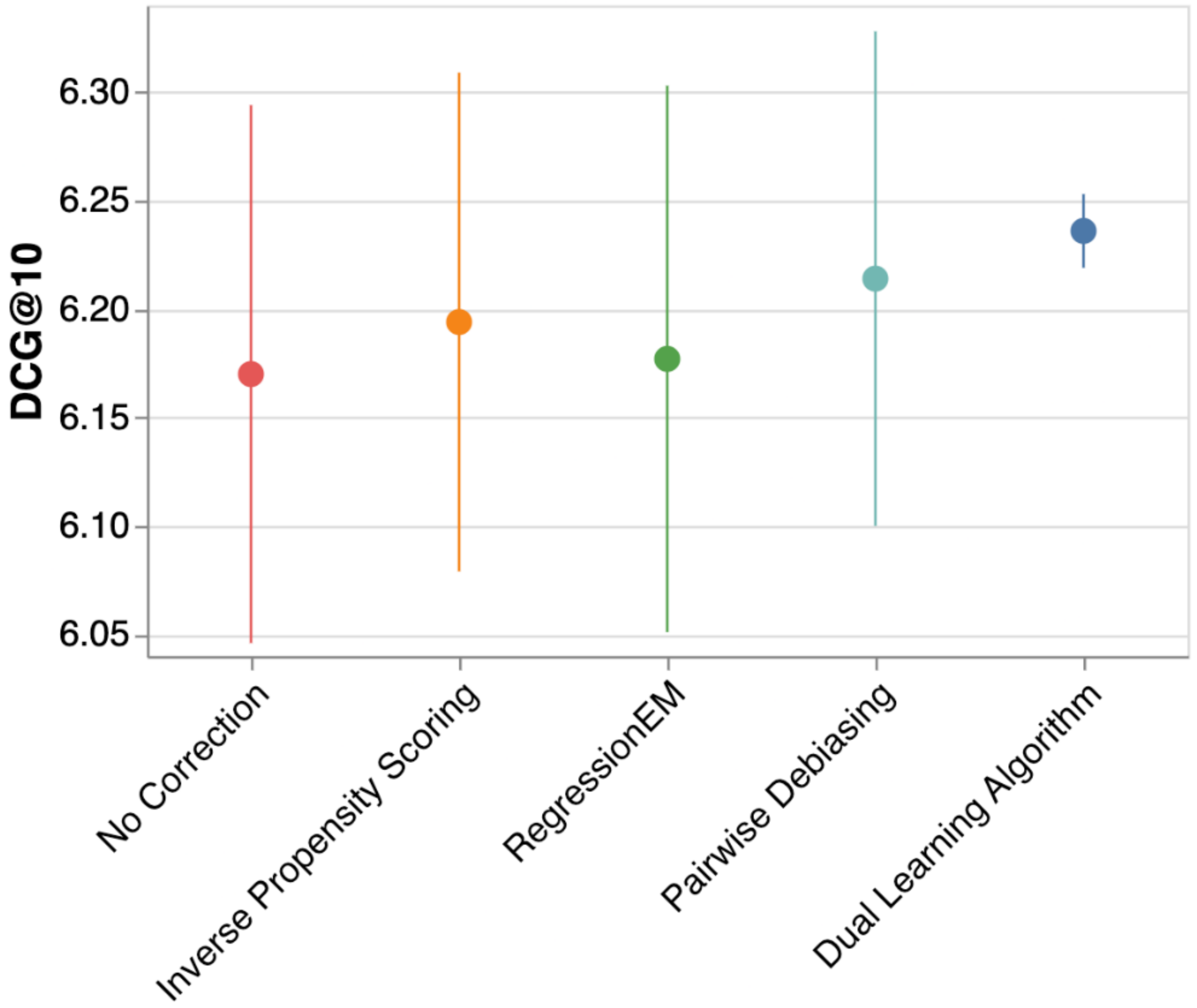
Most (academic) work in unbiased learning to rank is evaluated in **semi-synthetic simulation** [1]:

- Real queries/documents but synthetic clicks
- But does ULTR work in reality?

Baidu-ULTR [2] is the first large-scale web search dataset with real clicks for offline evaluation (≈380M queries, ≈1.2B user sessions)

[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 MonoBERT cross-encoders trained on the Baidu-ULTR dataset [1].
Models were trained on user clicks and evaluated on 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 [1]** warrants more scrutiny
- **WSDM Cup participants reported much higher ranking performance [2]**, in fact, we find all reported results are **outperformed by random shuffling**
- The authors did **not properly estimate position bias** and **we found dataset artifacts** (20% of the dataset consists of two docs)
- The original work focused on **pointwise ranking methods [1]** but many ULTR methods were proposed in pairwise / listwise settings.

[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.

Research questions

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

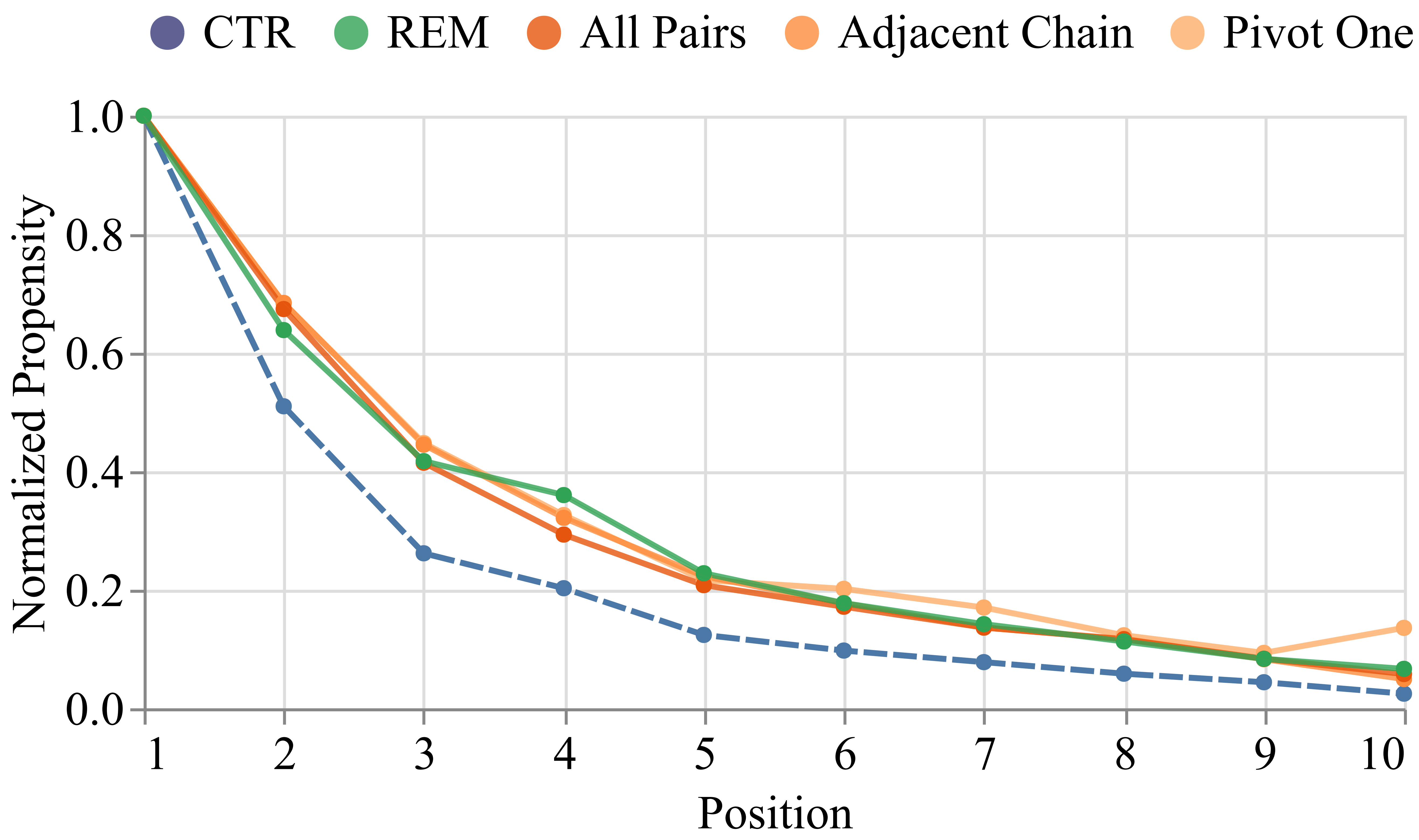
The Baidu-ULTR dataset

- **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, …
- **Presentation features:** item type, height, position, …

In this work, we focus on query-document text, position, and clicks.

Baidu search engine in 03/2024

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.

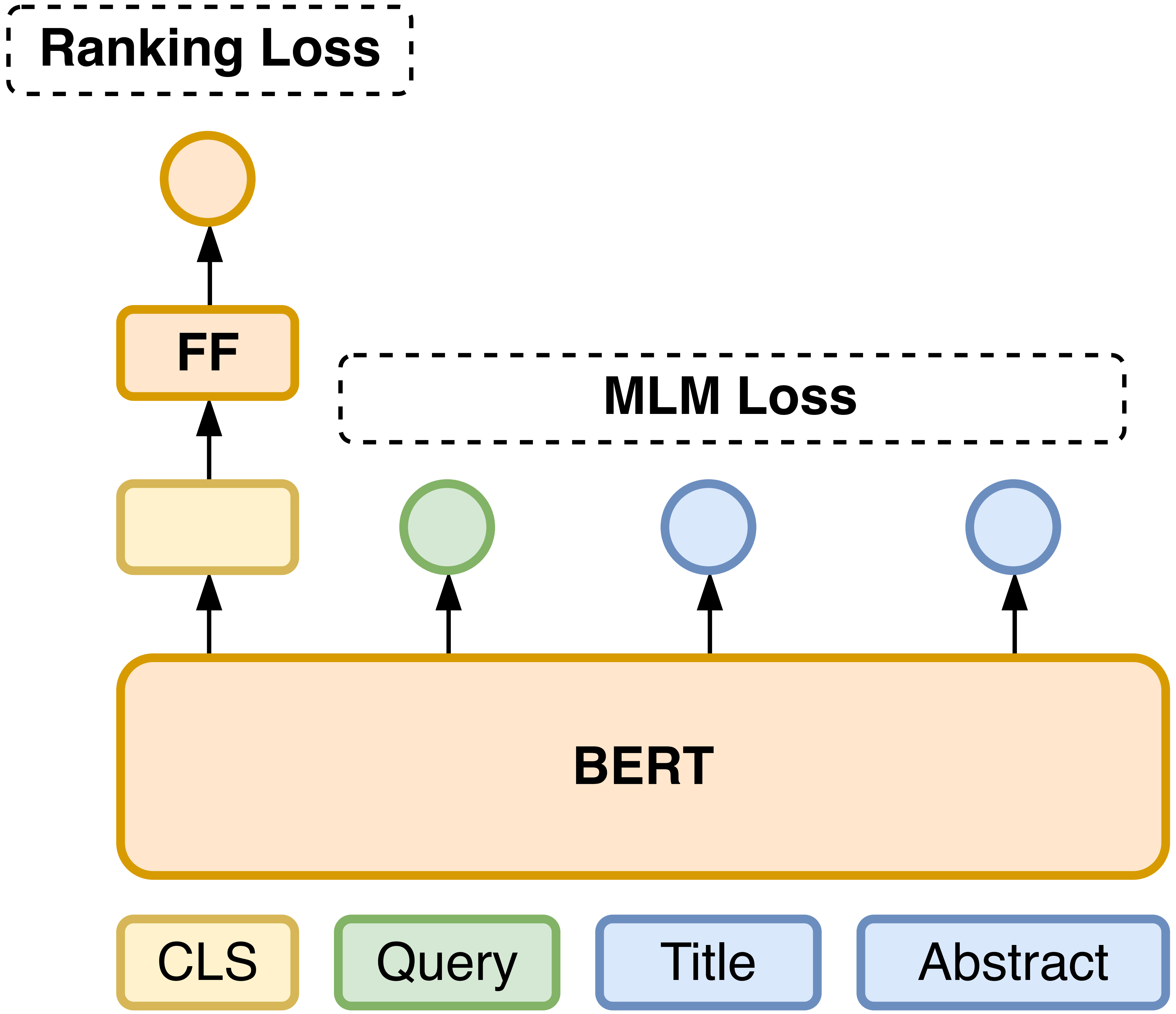
Therefore, we expect ULTR methods to improve ranking performance on this dataset.

Experimental Setup

Baidu Cross Encoder Setup

MonoBERT cross encoder

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



Losses

- **Ranking loss:** binary cross-entropy on clicks
- **MLM loss:** 30% masking rate

Reranking Dataset

To conduct more experiements, we create a smaller reranking dataset (≈2.3M sessions) using pre-computed query-document embeddings.

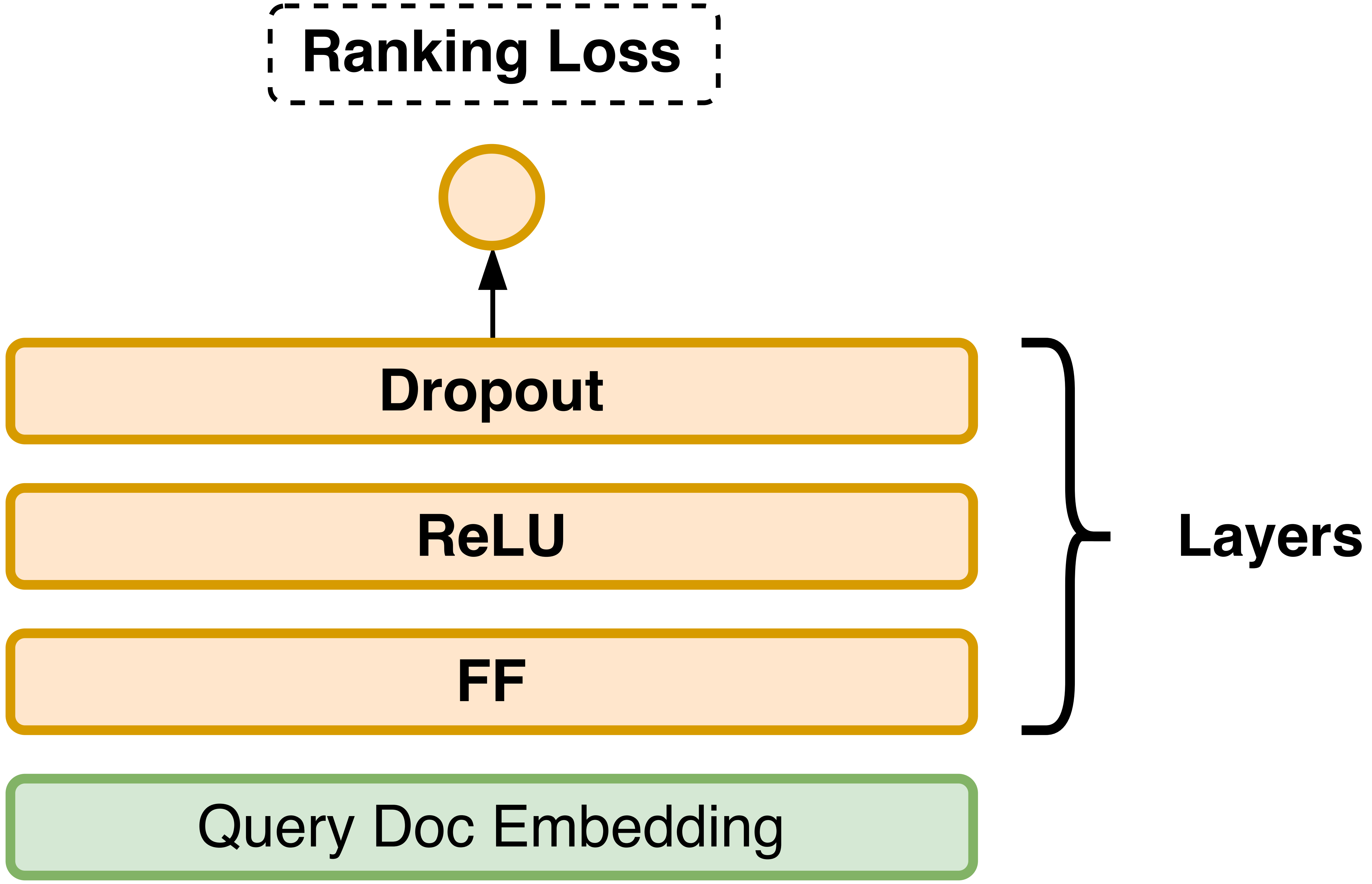
Pre-computed **query-document embeddings:**

- **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

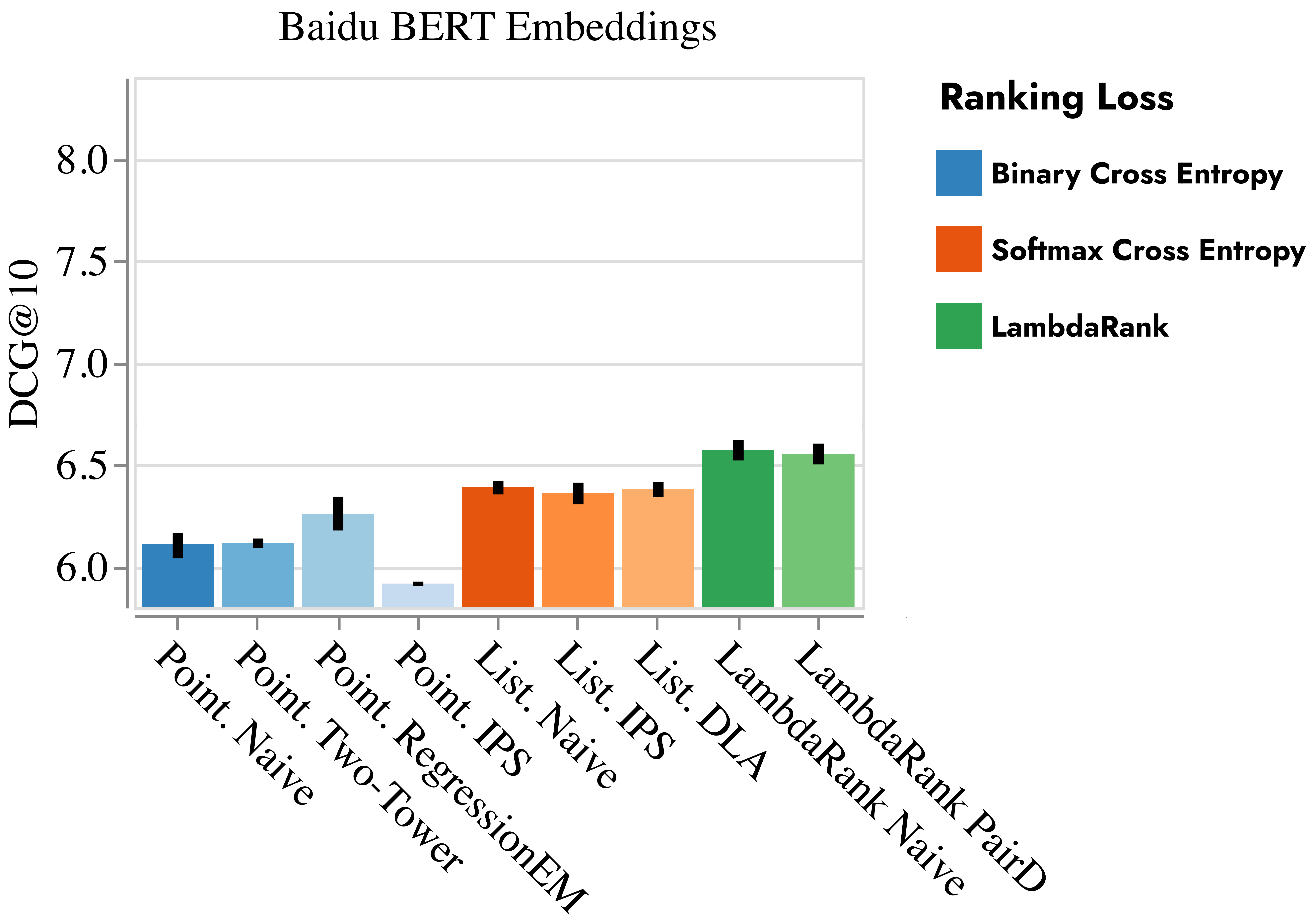
Feed forward ReLU networks
- 64 - 1024 hidden dims
- 2 - 5 layers
- Optional dropout
- Log1p normalization for LTR

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

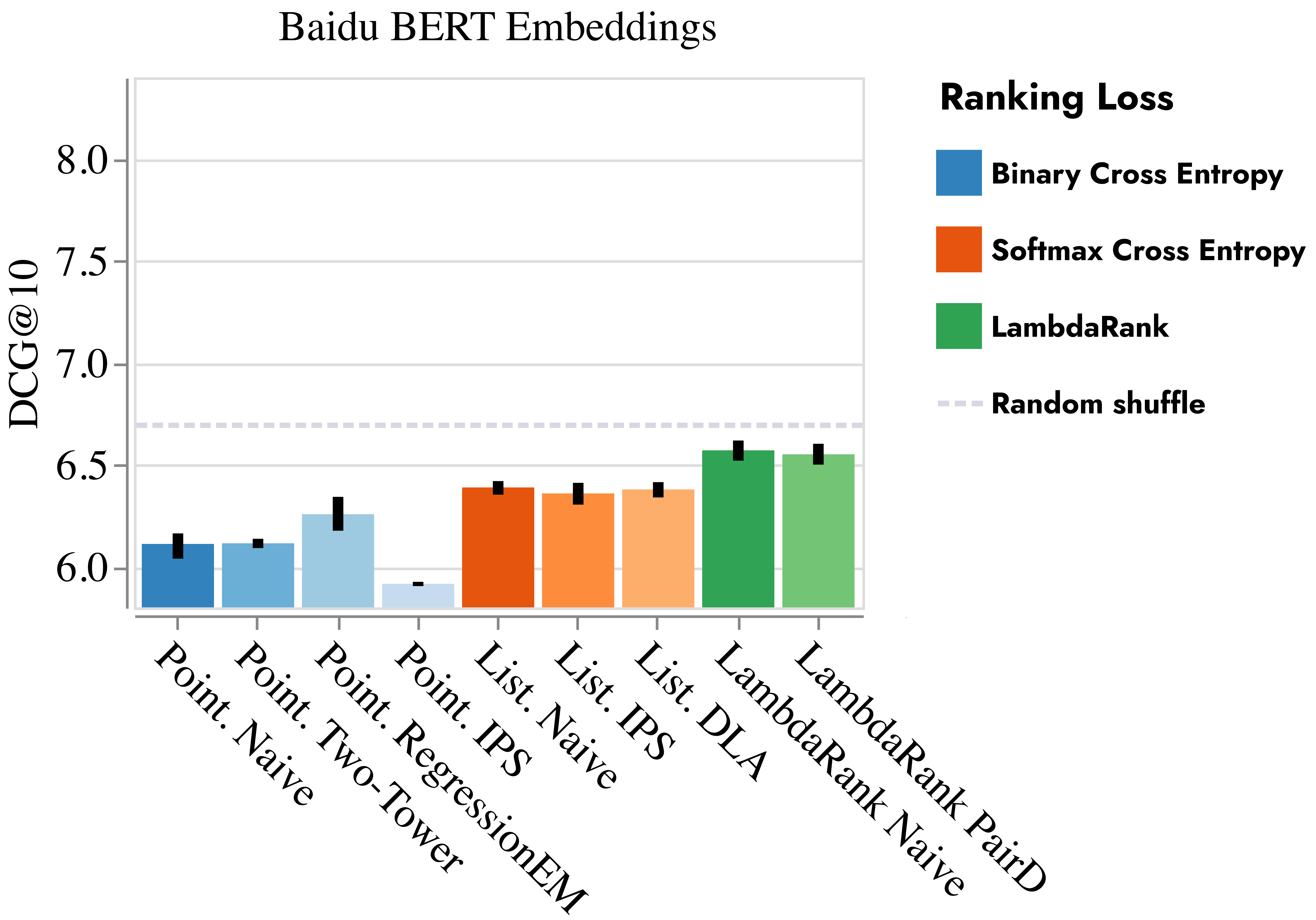


Results

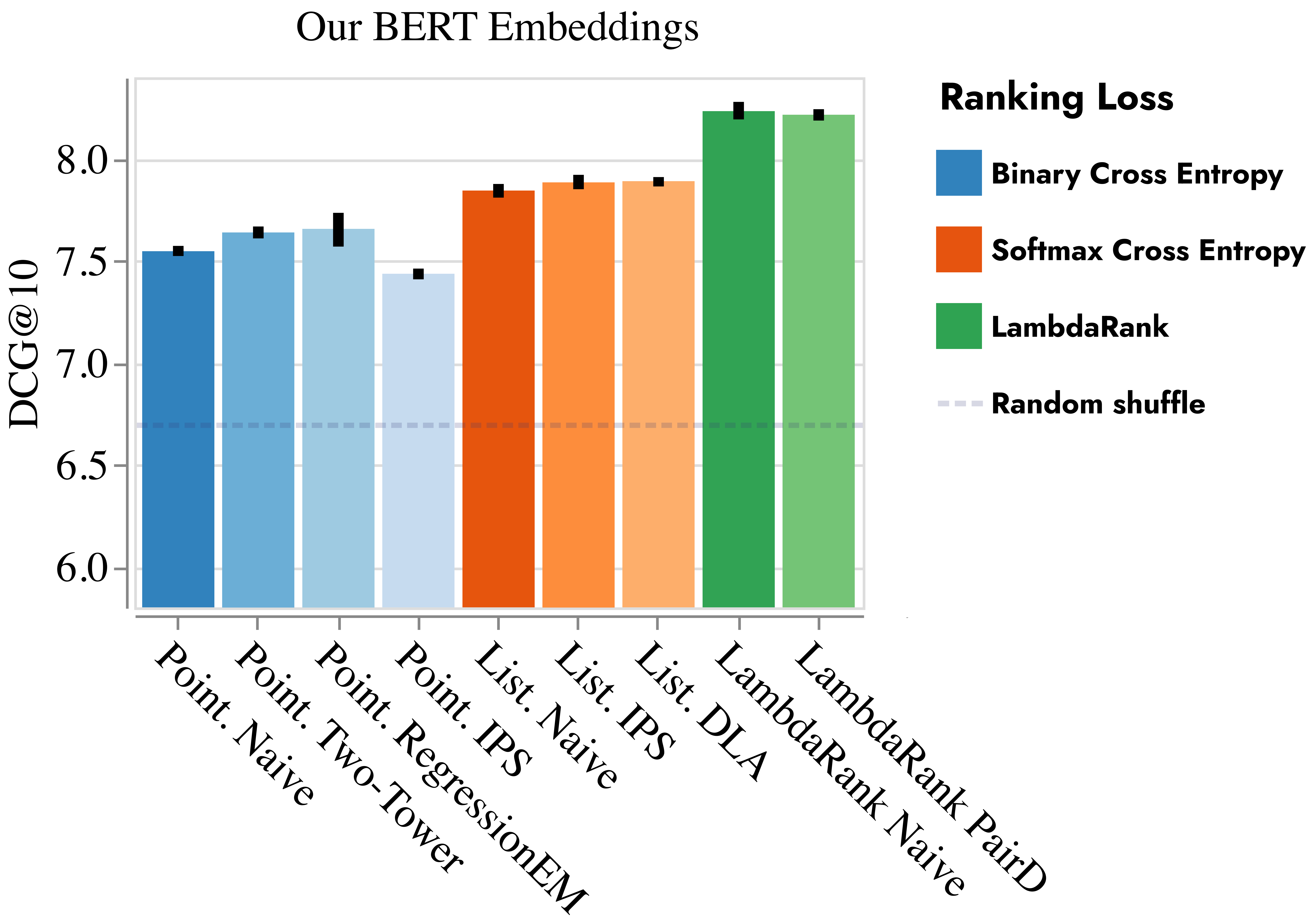
We can reproduce the Baidu NeurIPS results



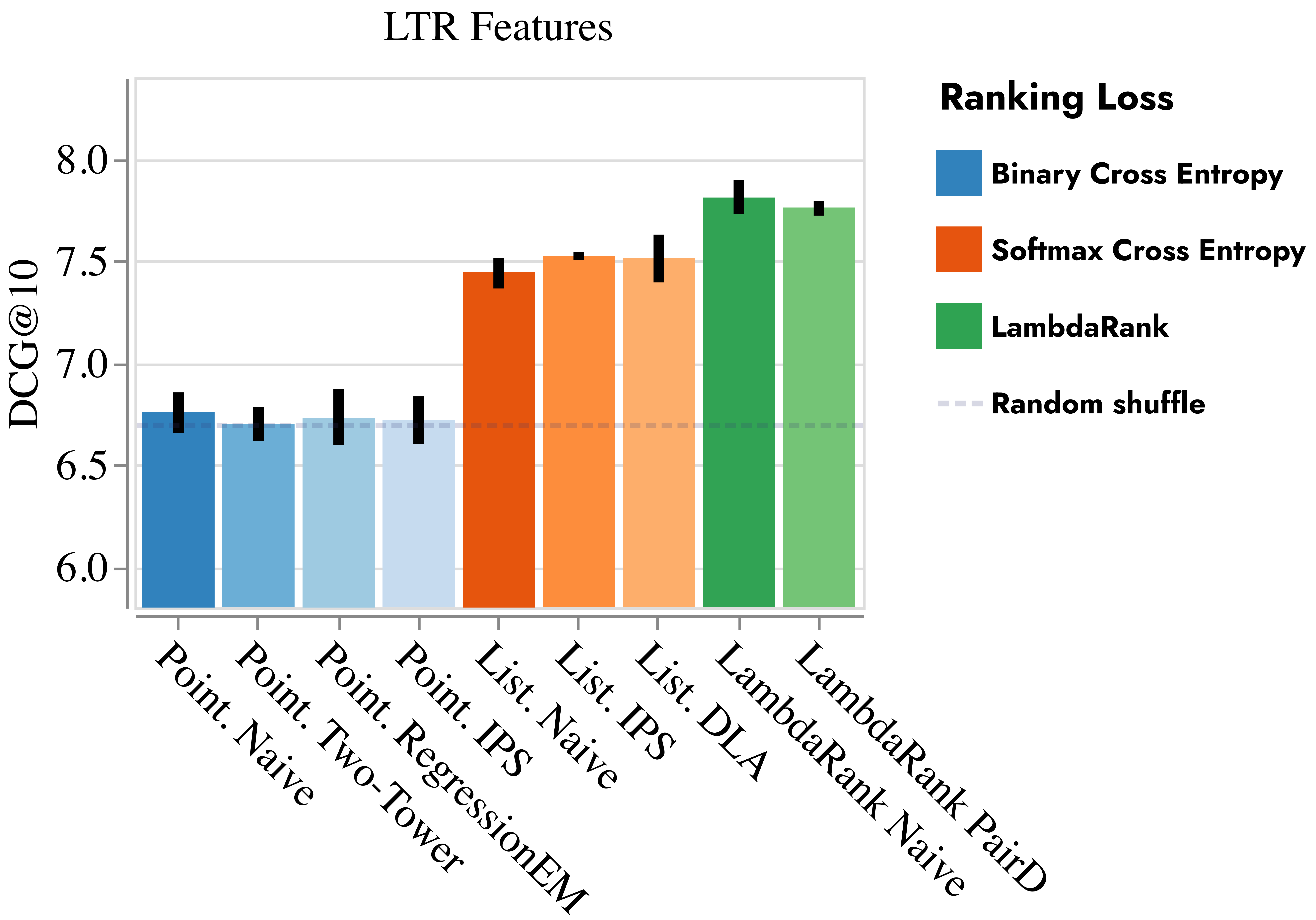
We can reproduce the Baidu NeurIPS results



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?

What if we directly train MonoBERT with ULTR?

- We train **three pointwise** and **three listwise** MonoBERTs from scratch
- ULTR leads to stark model differences when applied during pre-training
- IPS-based methods degrade ranking performance
- And again, listwise outperforms pointwise

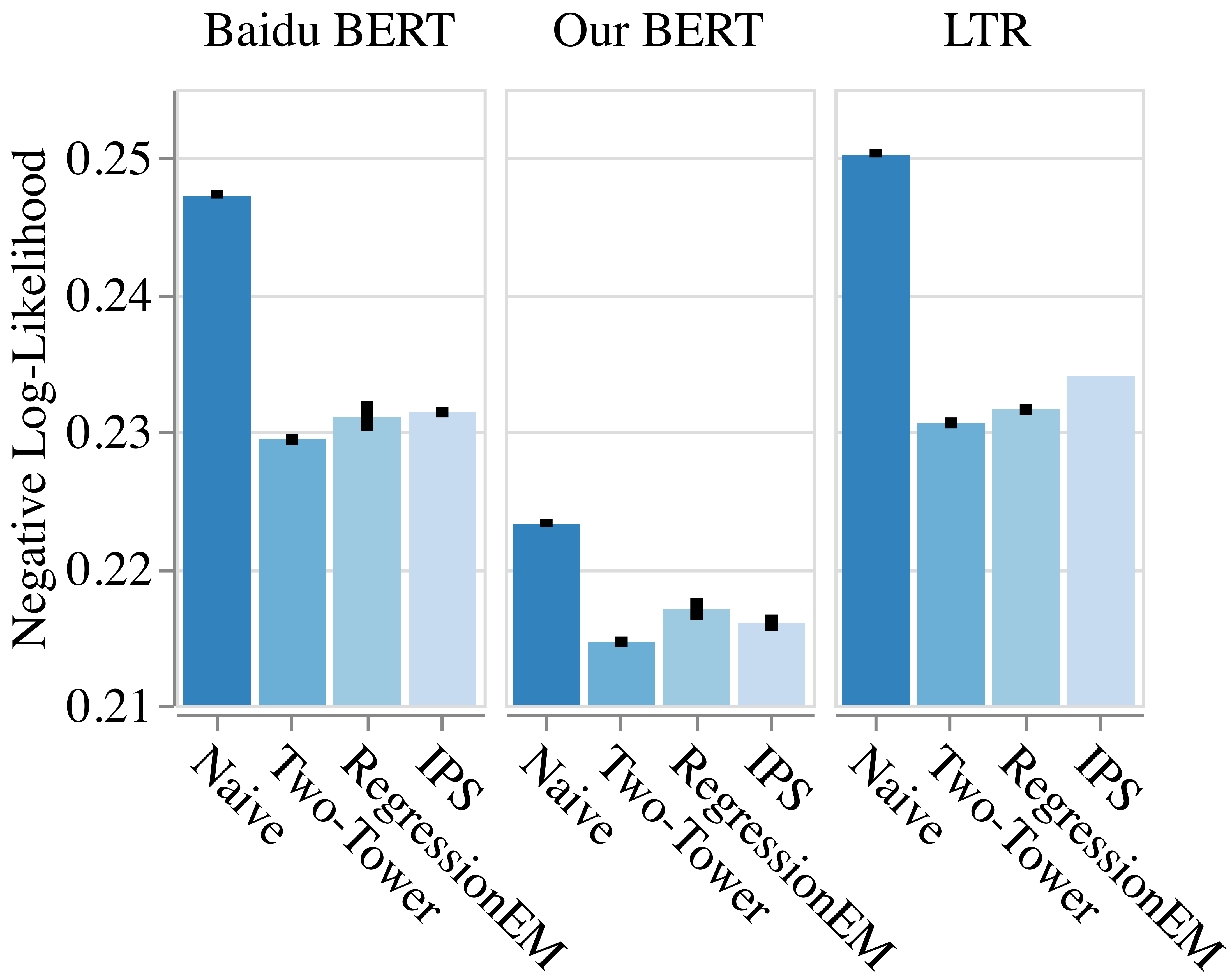
Model	DCG@10 ↑	NLL ↓
Pointwise Naive	7.251	0.227
Pointwise Two-Tower	7.456	0.217
Pointwise IPS	6.296	0.317
Listwise Naive	8.478	-
Listwise IPS	7.450	-
Listwise DLA	7.802	-

The interactions between ULTR and transformers need further investigation

Discussion

Ranking vs. click prediction

- **ULTR consistently leads to better click prediction**
- **But better click prediction does NOT imply better ranking performance**
- BM25 alone achieves a DCG@10≈9.54, better than any BERT model trained on clicks



Why might ULTR not improve ranking performance?

- No position bias (unlikely, given our analysis)
- **More complex user behavior**
- Identifiability issues when estimating position bias
- **Distribution shift between training and testing:**
  Training on top-10 vs. testing on up to top-1000 items
- Training on data collected from a strong logging policy [1]
- **Potential disagreement between users and annotators**

[1] Deffayet, Romain, et al. An offline metric for the debiasedness of click models. In SIGIR 2023.

Implications for the field

Our results confirm the original authors, common ULTR methods lead to (at best) marginal improvements on the largest public ULTR dataset.

- Our results call for adjusting simulation setups to reflect real-world challenges
- Interaction between ULTR methods 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

- **Data:** We publish our three smaller, cleaned, and pre-processed Baidu-ULTR reranking datasets with BERT embeddings and LTR features:

Load train / test click dataset:

```
from datasets import load_dataset

dataset = load_dataset(
    "philipphager/baidu-ultr_baidu-mlm-ctr",
    name="clicks",
    split="train", # ["train", "test"]
    cache_dir="~/.cache/huggingface",
)

dataset.set_format("torch") #  [None, "numpy", "torch", "tensorflow", "pandas", "arrow"]
```

Our BERT embeddings for Baidu ULTR on HuggingFace [1]

[1] https://huggingface.co/datasets/philipphager/baidu-ultr_uva-mlm-ctr

Contributions

- **Data:** We publish our smaller, cleaned, and pre-processed Baidu-ULTR reranking datasets with BERT embeddings and LTR features
- **Methods:** We publish Jax implementations of six standard ULTR methods
- **Models:** We train six MonoBERT models from scratch, releasing their weights
- **Bias estimation:** We publish code for four position bias estimators

Philipp Hager, p.k.hager@uva.nl

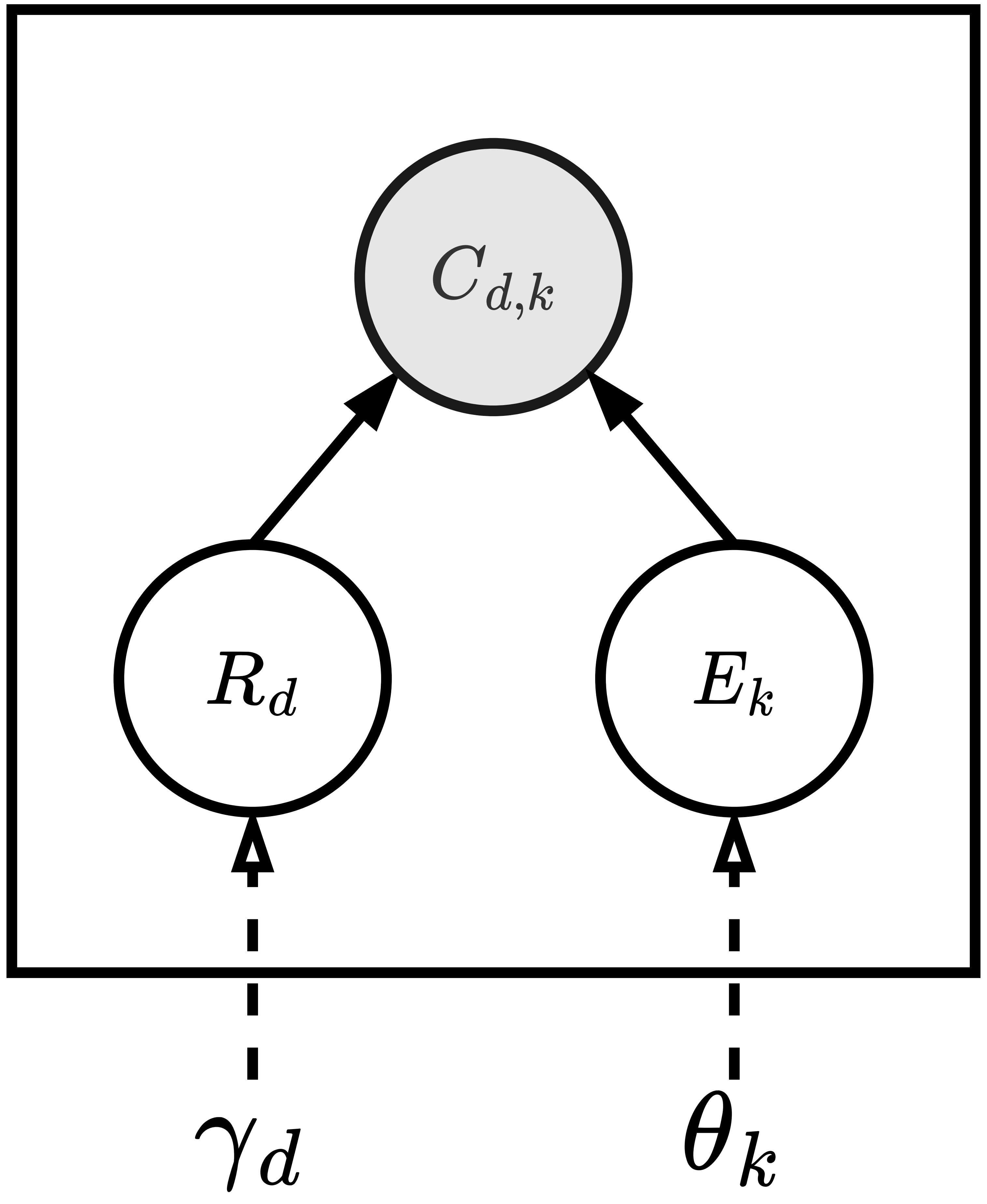
Backup

Position Based Model

PBM

Users click on examined and relevant items:

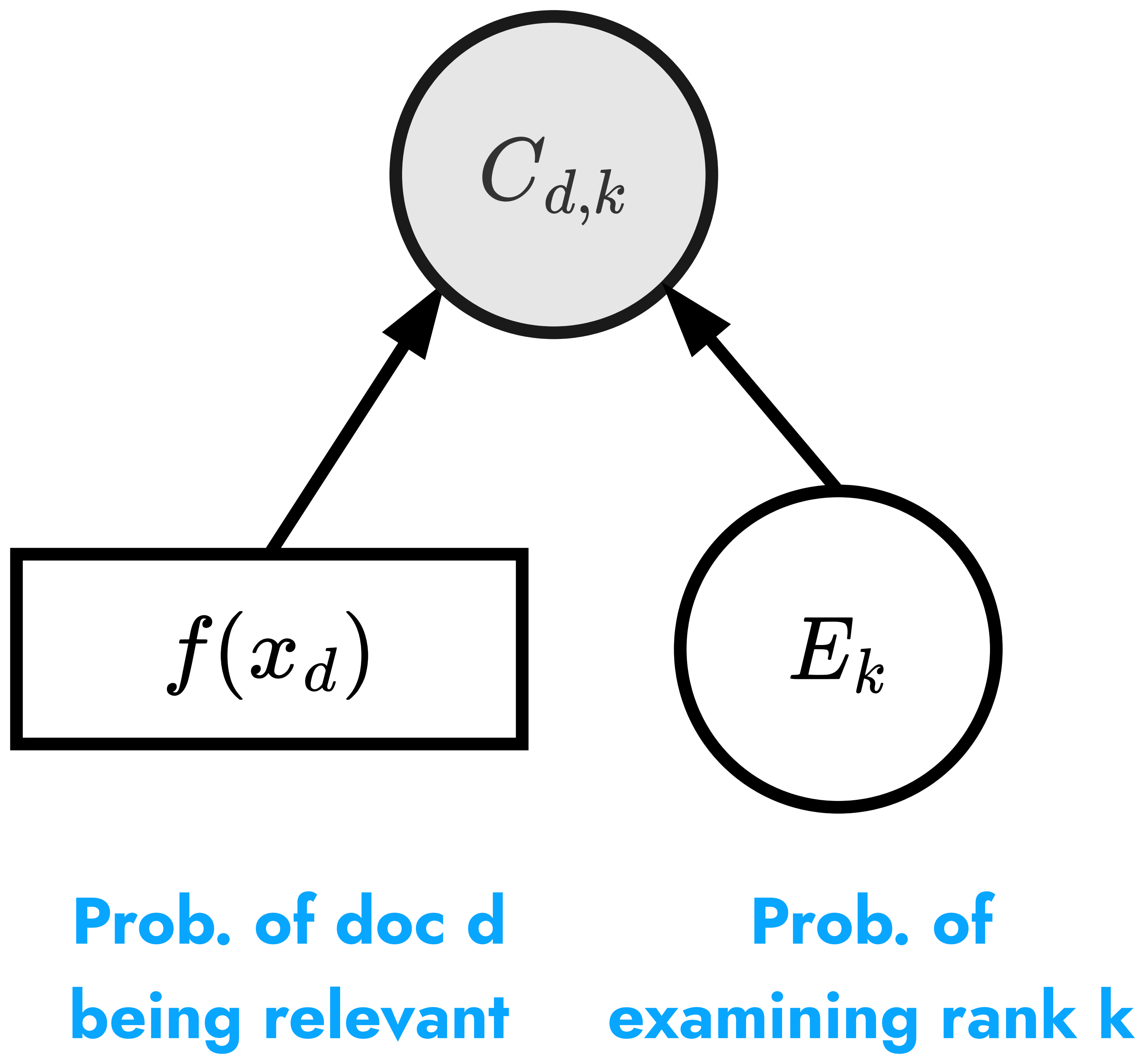
$$P(C = 1 \mid d, k) = \underbrace{P(E = 1 \mid k)}_{\text{Prob. of examining rank k}} \cdot \underbrace{P(R = 1 \mid d)}_{\text{Prob. of doc d being relevant}}$$



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.



[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.

The curious case of two documents

MD5 hashes of query/title/abstract revealed:

- 13% of documents have a "-" as a title: ***unavailable content***
- 9% of documents share the same title: ***what other people searched***

Pairwise Debiasing / Unbiased LambdaMART

Estimates propensity ratios for clicked and non-clicked documents:

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

$\tilde{e}^{-}(k)$ 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.