InfoRank: Unbiased Learning-to-Rank via Conditional Mutual Information Minimization

Jiarui Jin[1], Zexue He[2], Mengyue Yang[3], Weinan Zhang[1],
Yong Yu[1], Jun Wang[3], Julian McAuley[2]

[1]Shanghai Jiao Tong University, [2]University of California San Diego,
[3]University College London

Slides: https://jinjiarui.github.io/files/www24_oral.pdf

Content

- Problem Background
  - Biases in Learning-to-Rank
  - Unbiased Learning-to-Rank
- Architecture
  - Causality in Ranking
  - InfoRank
- Experiment
  - Simulator
  - Results
- Conclusion

Content

- **Problem Background**
  - Biases in Learning-to-Rank
  - Unbiased Learning-to-Rank
- Architecture
  - Causality in Ranking
  - InfoRank
- Experiment
  - Simulator
  - Results
- Conclusion

University of California San Diego

Shanghai Jiao Tong University

Biases in Learning-to-Rank

- **Explicit** feedback: relevance scores.
  - Predict relevance probability $P(r)$

query q

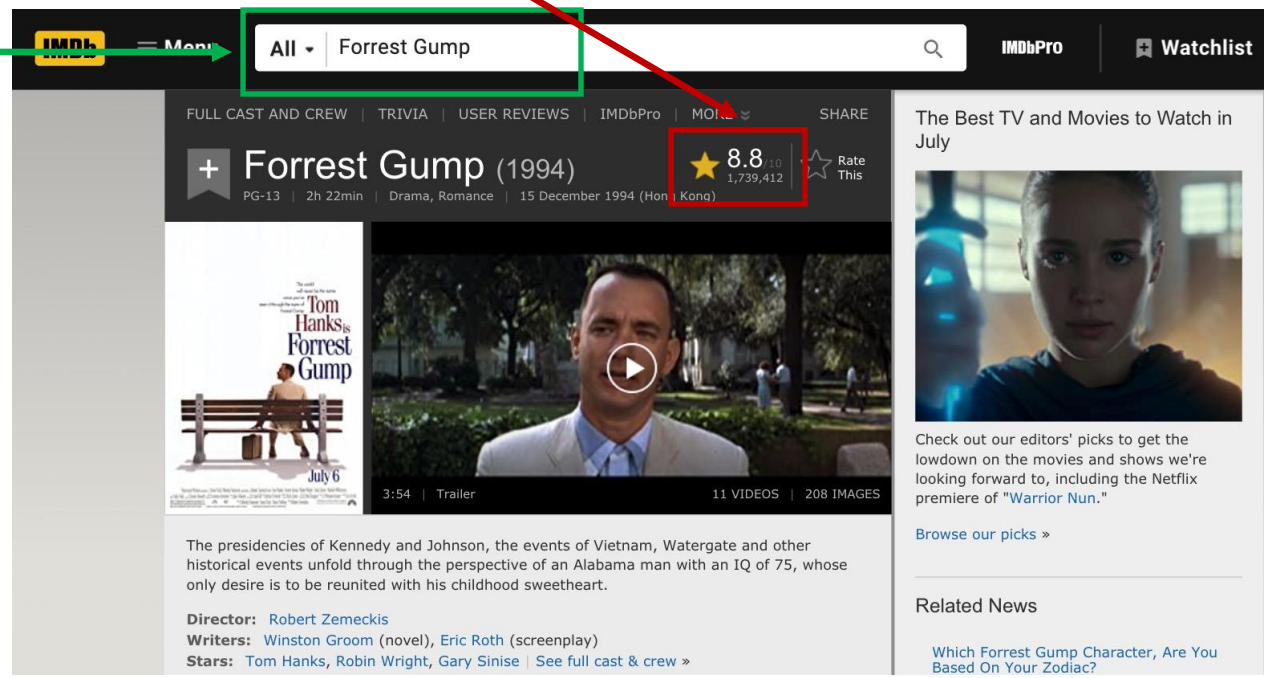


Figure: IMDB Movie

Biases in Learning-to-Rank

- **Implicit** feedback: click scores.
  - Predict click probability $P(c)$

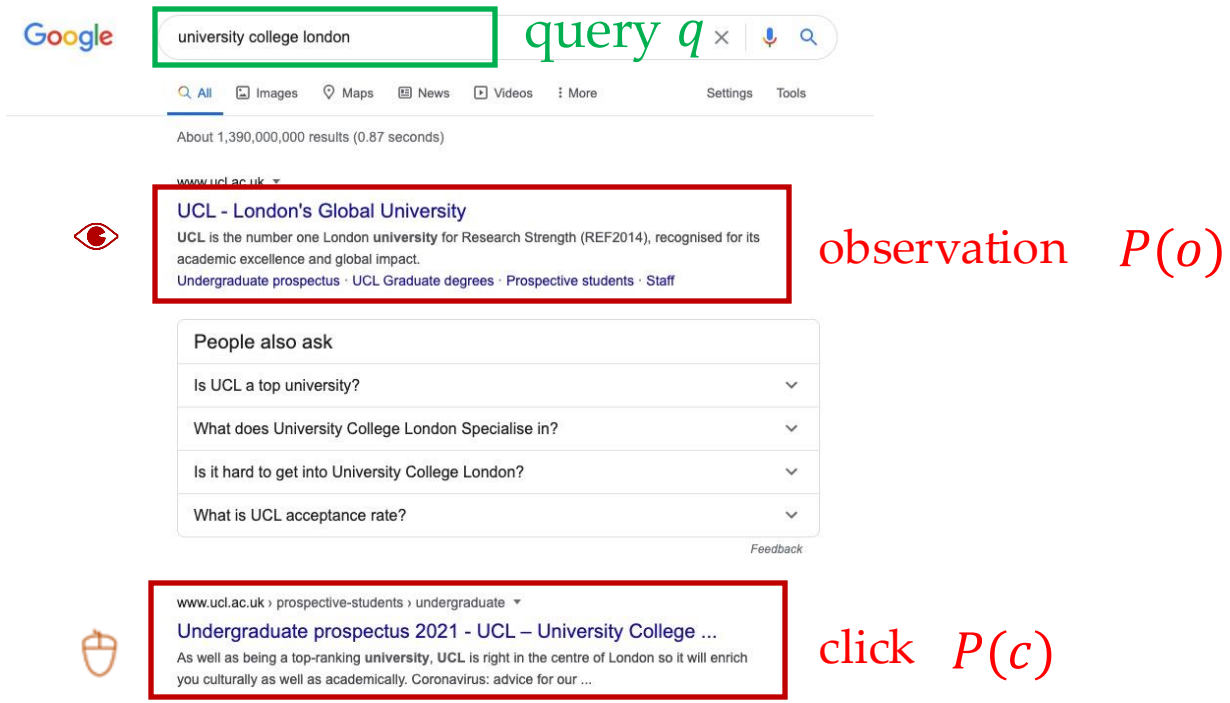


Figure: Google Search

Biases in Learning-to-Rank

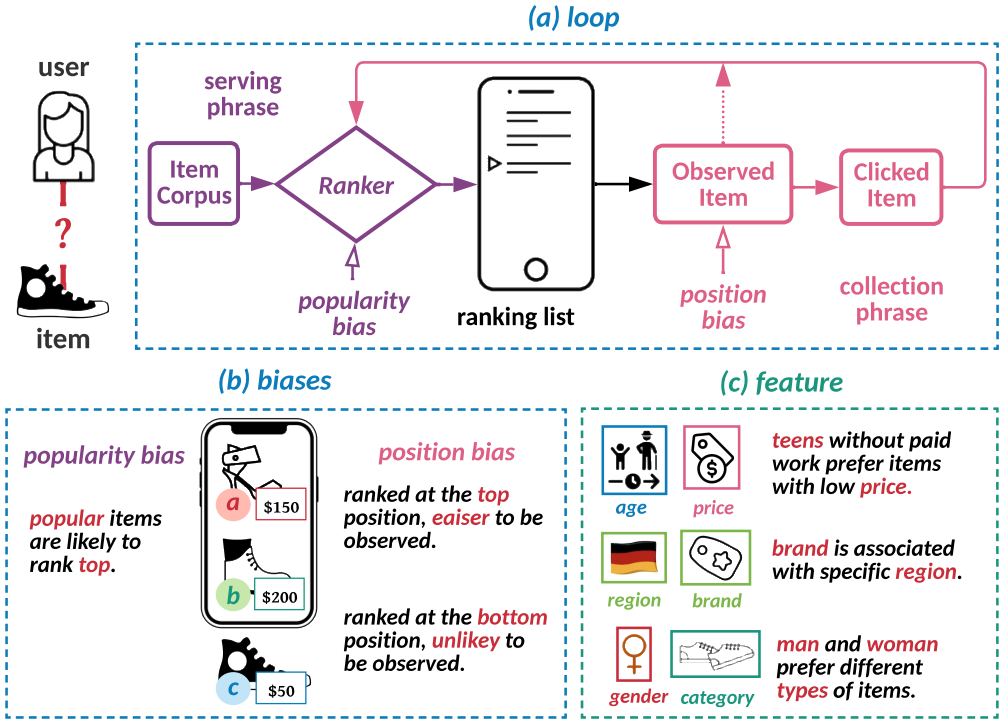


Figure: An illustrated example of the feedback loop. These biases tend to be amplified within the feedback loop, resulting in a "rich-get-richer" dilemma.

- **Position Bias:** users typically peruse presented item lists from top to bottom, with their attention diminishing rapidly along the way. Consequently, higher-ranked items receive more exposure and greater opportunities for observation and subsequent clicks. **Position bias** manifests during **the collection of user feedback**.
- **Popularity Bias**: this bias prompts ranking systems to recommend popular items more frequently than their popularity would warrant. **Popularity bias** occurs when **the system returns ranked lists for user service**.
- In both cases, blindly optimizing ranking performance based on implicit feedback data may inadvertently reinforce the existing presentation or popularity order rather than learning personalized relevance.

Unbiased Learning-to-Rank

- **Biased** ranking models trains a ranker f that assigns a relevance score to each query-item pair by using the click data as the supervision. The risk function is:

$$\mathcal{F}(f) = \sum_{u} \sum_{d \in \mathcal{D}_u} \Delta(f(x), c)$$

where $\Delta(f(x), c)$ denotes a point-wise loss function, x denotes the feature and c represents clicks.

- **Position Bias**: Conventional debiasing methods typically introduce an additional relevance factor, denoted as r. These methods estimate r instead of c. They leverage the insight that a user clicks on an item only when it has been both observed and perceived as relevant:

$$P(C = 1|X = x) = P(R = 1|X = x) \cdot P(O = 1|X = x)$$

Unbiased ranking infers relevance from click data and generate a ranked list based on $P(R = 1|X = x)$ different from biased ranking using $P(C = 1|X = x)$.

University of California San Diego

Shanghai Jiao Tong University

Unbiased Learning-to-Rank

- **Popularity Bias**: As previous debiasing algorithms rely on past user feedback, particularly clicks, to estimate popularity for an item d, we employ the notations $(\mathcal{C}, \mathcal{O}, \mathcal{R})$ to represent the set of prior feedback (clicks, observations, relevance). When a ranker can be considered free from popularity bias if the relevance estimation remains independent of the item's past click history. Alternatively, following the idea of collaborative filtering, relevance estimation could consider the item's

$$P(R = 1|\mathcal{C} = \{c = 1\}_d, X = x) = P(R = 1|\mathcal{R} = \{r = 1\}_d, X = x)$$

$\{c = 1\}_d$ and $\{r = 1\}_d$ do not encompass the "current" click and relevance to be estimated.

Content

- Problem Background
  - Biases in Learning-to-Rank
  - Unbiased Learning-to-Rank
- **Architecture**
  - Causality in Ranking
  - InfoRank
- Experiment
  - Simulator
  - Results
- Conclusion

University of California San Diego

Shanghai Jiao Tong University

Causality in Ranking

- Our idea is to summarize these biases into a single observation factor: (i) User observation feedback is influenced by the position of items (**Position Bias**); (ii) The system generates ranked lists based on observations or further clicks (**Popularity Bias**). We consider the observation factor as the "sensitive attribute". An ideal ranker should adhere to the following principle:

$$P(R = r|O = 1, X = x) = P(R = r|O = 0, X = x)$$

  holds for any relevance score $r \in \{0,1\}$ and any observation value o $\in \{0,1\}$ attenable by O.

- Our evaluation metric can be defined as:

$$\Delta\text{CI} := |P(R = 1|O = 1, X = x) - P(R = 1|O = 0, X = x)|$$

- If a ranker is a ranker free from the effect of the observation factor, $\Delta\text{CI} = 0$.

Causality in Ranking

- **Position Bias.** Convolutional debiasing approaches only introduce the observation factor ($P(C = 1|X = x) = P(R = 1|X = x) \cdot P(O = 1|X = x)$), but is not suffice, because the relevance estimation can still be affected by whether it has been observed or not. To achieve this, we incorporate our conditional independence ($P(R = r|O = 1, X = x) = P(R = r|O = 0, X = x)$) into it as:

$$P(R = 1|X = x) = P(R = 1|O = o, X = x)$$

where $o \in \{0,1\}$.

Causality in Ranking

- **Popularity Bias.** From our conditional independence ($P(R = r|O = 1, X = x) = P(R = r|O = 0, X = x)$), we can derive $P(\mathcal{C} = \{c = 1\}_d|\mathcal{O} = \{o = 1\}_d, X = x) = P(\mathcal{R} = \{r = 1\}_d|X = x) \cdot P(\mathcal{O} = \{o = 1\}_d, X = x)$. It implies that given the features of an item d, its previous clicks (i.e., $\{c = 1\}_d$) only occurs when d is both relevant ($\{r = 1\}_d$) and observed ($\{o = 1\}_d$) by users. Following this, we can proceed to derive:

$$P(R = 1|\mathcal{C} = \{c = 1\}_d, X = x) = \frac{P(R = 1|\mathcal{O} = \{o = 1\}_d, X = x)}{P(R = 1|X = x)} P(R = 1|\mathcal{R} = \{r = 1\}_d, X = x)$$

  where $o \in \{0,1\}$.

- Reinforce $O = 1$ and $R = 1$'s independence conditioned on $X = x$ can lead to an approximation where $\frac{P(R = 1|\mathcal{O} = \{o = 1\}_d, X = x)}{P(R = 1|X = x)}$ approaches 1. The remaining part $P(R = 1|\mathcal{R} = \{r = 1\}_d, X = x)$ reflects the ranker's inductive capacity (learning from historical records $\mathcal{R} = \{r = 1\}_d$ to infer $R = 1$).

InfoRank Architecture

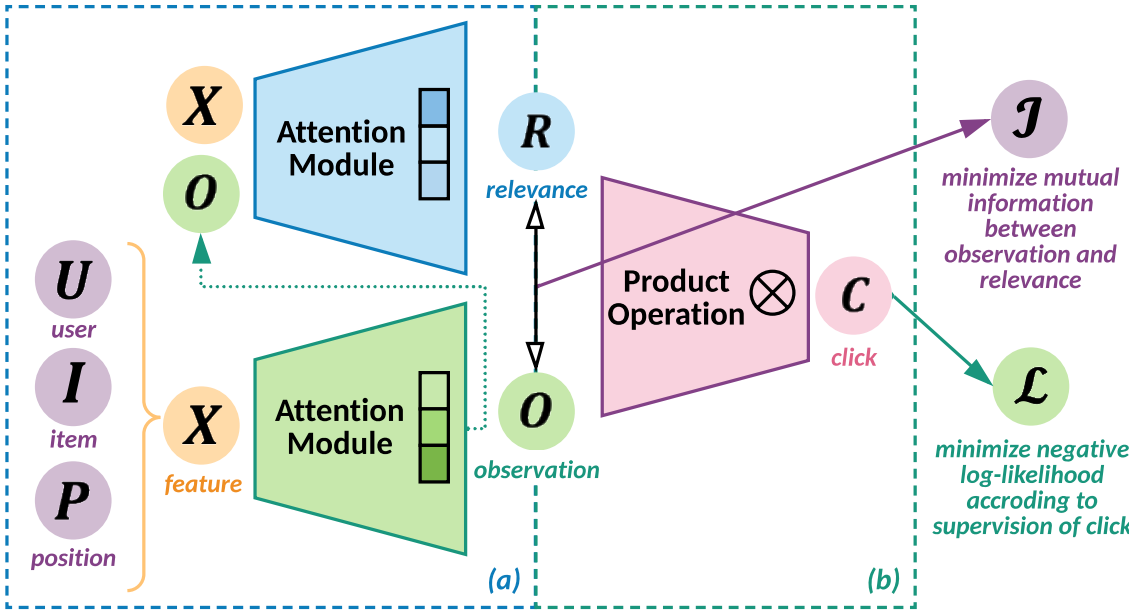


Figure: The overall architecture of InfoRank.

- **Unbiased Estimation for R and O.** We leverage an attention mechanism to mine correlations between user-item features. Formally, for the h-th head, we have:

$$\beta_{ij}^{(h)} = \left(x_i W_T^{(h)}\right) \cdot \left(x_j W_T^{(h)}\right)^{\mathrm{T}}$$

where x_i and x_j are the i-th and j-th feature. $W_{\cdot}^{(h)}$s are trainable weights and $\beta_{ij}^{(h)}$ determines the correlations between x_i and x_j. We subsequently normalize this value within the feature scope as:

$$\alpha_{ij}^{(h)} = \mathrm{softmax}(\beta_{ij}^{(h)}) = \frac{\exp\left(\beta_{ij}^{(h)}/\iota\right)}{\sum_{j=0}^{N-1} \exp\left(\beta_{ij}^{(h)}/\iota\right)}$$

where ι denotes temperature.

InfoRank Architecture

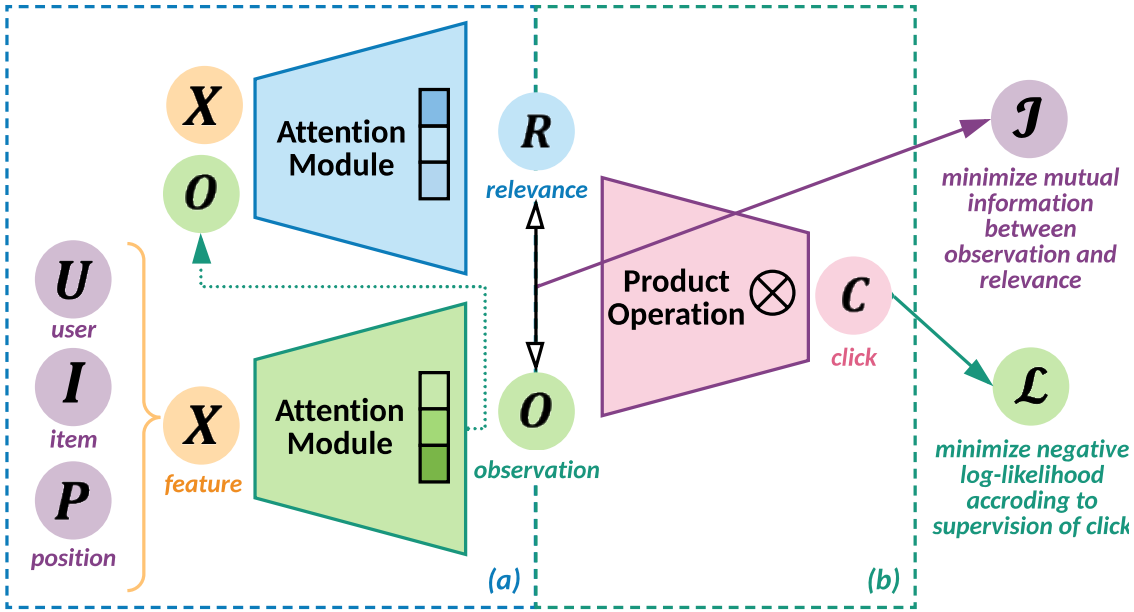


Figure: The overall architecture of InfoRank.

- **Unbiased Estimation for R and O.** We jointly attend on the feature scope from different representation subspaces to learn stably as:

$$\omega_i = \sigma(W_q \cdot \left(\frac{1}{H} \sum_{h=0}^{H-1} \sum_{j=0}^{N-1} \alpha_{ij}^{(h)} \left(x_J W_C^{(h)} \right) \right) + b_q)$$

where H is the number of attention heads, and $W.$, $b.$ are trainable parameters. We further integrate this information with attention vector w to obtain feature embedding p:

$$p = \frac{1}{N} \sum_{i=0}^{N-1} w^{\mathrm{T}} \cdot \tanh(W_p \cdot \omega_i + b_p)$$

p is further fed into two separated MLP modules activated by a sigmoid function without parameter sharing to obtain the estimation of $P(R = 1|O = o, X = x)$ and $P(O = 1X = x)$.

InfoRank Architecture

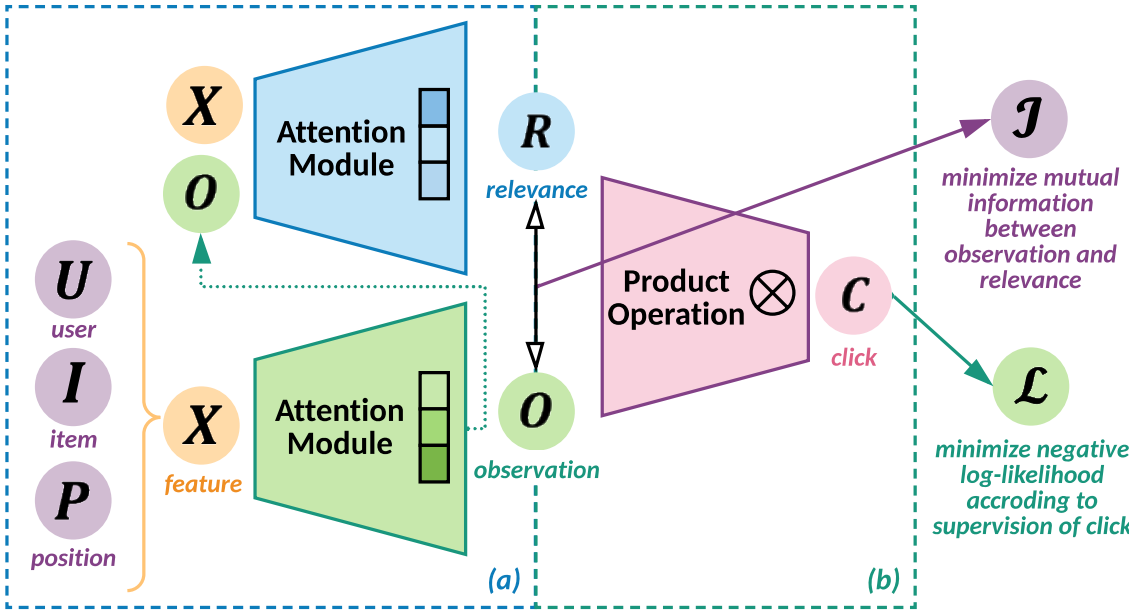


Figure: The overall architecture of InfoRank.

- **Estimating C.** Given the distribution $P(O = 1X = x)$ and the conditional distribution $P(R = 1|O = o, X = x)$, we can compute $P(C = 1X = x)$ as:

$$P(C = 1|X = x) = P(R = 1|O = o, X = x) \cdot P(O = 1|X = x)$$

- **Conditional Mutual Information**. We establish the following proposition:

PROPOSITION 3.1. *Given that relevance, click, and observation variables are binary (i.e., $R, C, O \in \{1, 0\}$), for any user-item pair with feature X, the following statements are equivalent:*

- *The relevance R and observation O are conditionally independent given X. In other words, $P(R, O|X) = P(R|X) \cdot P(O|X)$. That is, $P(R|O = 1, X) - P(R|O = 0, X) = 0$.*
- *The conditional mutual information between relevance R and observation O (later defined in Eq. (13)) is zero, i.e., $\mathcal{I}(R; O|X) = 0$.*
- *The conditional independence score ΔCI is zero.*

InfoRank Architecture

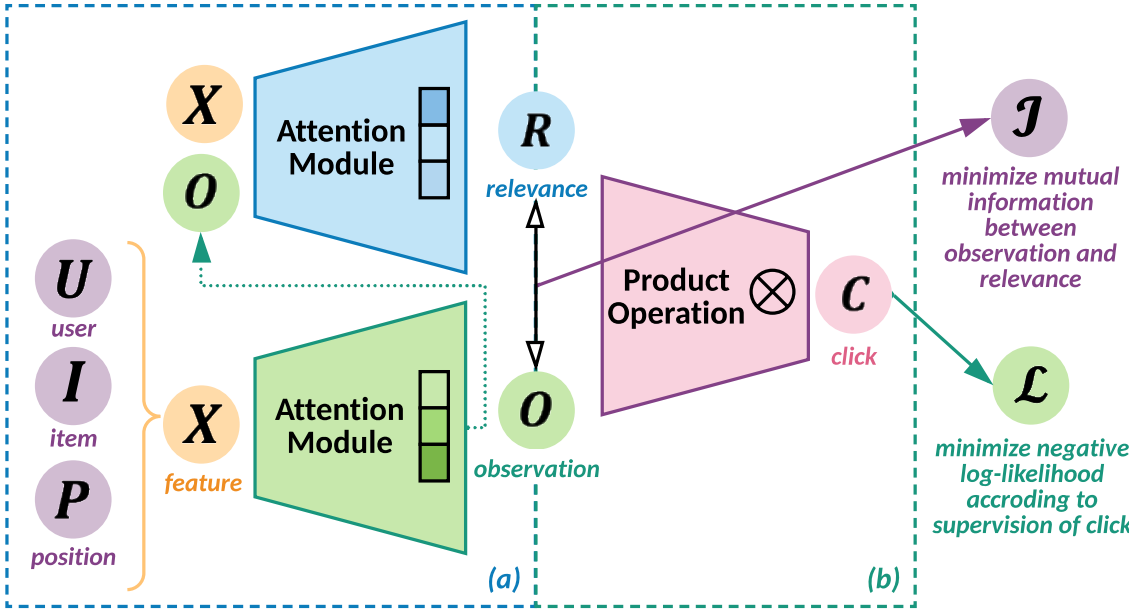


Figure: The overall architecture of InfoRank.

- **Conditional Mutual Information**. We design $\mathcal{I}$ to minimize the conditional mutual information:

$$\mathcal{I} := \mathcal{I}(R; O|X = x)$$

$$= \sum_{R,O} P(R|O, X = x) \cdot P(O|X = x) \cdot \ln \frac{P(R|O, X = x)}{P(R|X = x)}$$

We then derive $P(R|X = x)$ using:

$$P(R|X = x) = \sum_{O} P(R|O, X = x) \cdot P(O|X = x)$$

- **Optimization Function**. We combine the supervisions over click data and conditional mutual information minimization to derive:

$$\text{argmin}_\theta \mathcal{L} + \eta \cdot \mathcal{I}$$

where η is the hyper-parameter for balance. $\mathcal{L}$ can be defined as:

$$\mathcal{L} = -\sum_{(c,x)\in\mathcal{D}} (c \cdot \log P(\hat{c}|x) + (1 - c) \cdot \log(1 - P(\hat{c}|x)))$$

InfoRank Architecture

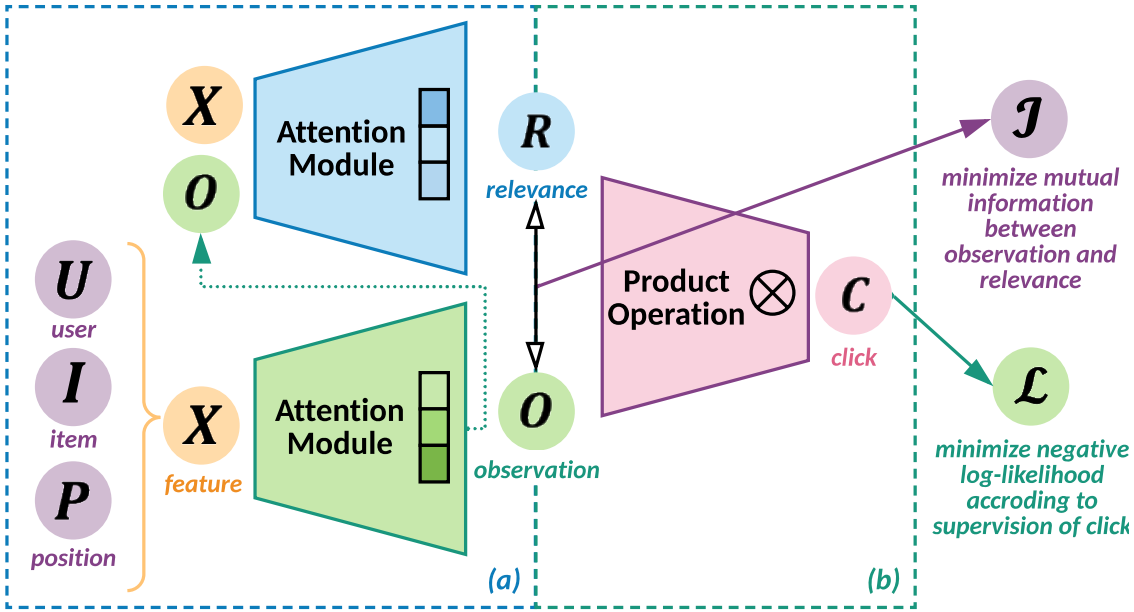


Figure: The overall architecture of InfoRank.

Algorithm 1 InfoRank

INPUT: implicit feedback dataset $\mathcal{D} = \{(\boldsymbol{x}, c, o)\}$;
OUTPUT: unbiased ranker f_θ with parameter θ

Initialize all parameters.
repeat
  Randomly sample a batch $\mathcal{B}$ from $\mathcal{D}$
  for each data point $(\boldsymbol{x}, c, o)$ in $\mathcal{B}$ **do**
    Calculate $P(R = 1|O = o, X = \boldsymbol{x})$ (and $P(O = 1|X = \boldsymbol{x})$) using Eqs. (8), (9), and (11).
    Compute $P(C = 1|X = \boldsymbol{x})$ using Eq. (12).
  end for
  Compute $\mathcal{L}$ and $\mathcal{I}$ according to Eqs. (16) and (13).
  Update θ by minimizing Eq. (17).
until convergence

Using the estimation of $P(C = 1X = x)$ for training and using the estimation of $P(R = 1|O = o, X = x)$ for inference.

Content

- Problem Background
  - Biases in Learning-to-Rank
  - Unbiased Learning-to-Rank
- Architecture
  - Causality in Ranking
  - InfoRank
- **Experiment**
  - Simulator
  - Results
- Conclusion

University of California San Diego

Shanghai Jiao Tong University

Simulator

- Evaluation of unbiased learning-to-rank algorithms treats their click data as relevance data and generates corresponding click data by using click models.
- Our click models includes PBM, UBM, and CCM:
- PBM (Position Based Model) simulates user browsing behavior based on the assumption that the bias of an item only depends on its position, which can be formulated as $P(o_i) = \rho_i^\tau$, where ρ_i represents position bias at position i and $\tau \in [0, +\infty)$ is a parameter controlling the degree of position bias. The position bias ρ_i is obtained from an eye-tracking experiment in [Joachims et al., 2005] and the parameter τ is set as 1.
- UBM (User Browsing Model) assumes that the observation probability depends not only on the rank of an item $d_{i'}$ but also on the rank of the previously clicked item $d_{i'}$ as $P(o_i = 1 | c_{i'} = 1, c_{i'+1} = 0, \dots, c_{i-1} = 0) = \gamma_0$. We get γ_0 from the eye-tracking experiments in [Dupret et al., 2008].
- CCM (Cascade Click Model) assumes that the user browses search results in a sequential order from top to bottom. User browsing behavior is conditioned on both current and past items, as $P(c_i = 1 | o_i = 0) = 0$, $P(c_i = 1 | o_i = 1, r_i) = P(r_i)$, $P(o_{i+1} = 1 | o_i = 0) = 0$, $P(o_{i+1} = 1 | o_i = 1, c_i = 0) = \gamma_1$, $P(o_{i+1} = 1 | o_i = 1, c_i = 1, r_i) = \gamma_2 \cdot \big(1 - P(r_i)\big) + \gamma_3 \cdot P(r_i)$. The parameters are obtained from an experiment in [Guo et al., 2009].

[Joachims et al., 2005] Thorsten Joachims, et al. Accurately Interpreting Click-Through Data as Implicit Data. SIGIR 2005.

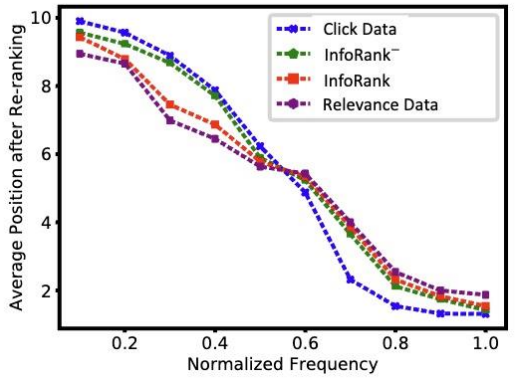
[Dupret et al., 2008] Georges E Dupret et al. A User Browsing Model to Predict Search Engine Click Data from Past Observation. SIGIR 2008.

[Guo et al., 2009] Fan Guo, et al. Click Chain Model in Web Search. WWW 2009.

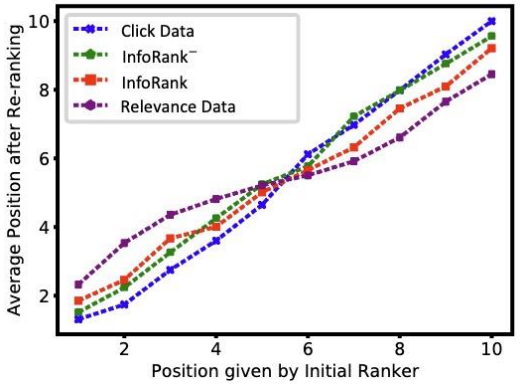
Results

Ranker	Debiasing Method	Yahoo (UBM)				LETOR (UBM)				Adressa (UBM)			
		MAP	N@3	N@5	N@10	MAP	N@3	N@5	N@10	MAP	N@3	N@5	N@10
InfoRank (Ranking)	Labeled Data	.856	.755	.760	.795	.695	.381	.468	.563	.821	.714	.727	.754
	InfoRank (Debiasing)	**.845***	**.736***	**.739***	**.779***	**.650***	**.380***	**.460***	**.541***	**.801***	**.691***	**.715***	**.739***
	Regression-EM	.837	.683	.692	.731	.634	.374	.442	.535	.794	.673	.706	.731
	Randomization	.835	.680	.689	.728	.630	.368	.437	.515	.792	.668	.695	.728
	Click Data	.823	.670	.678	.720	.622	.356	.428	.489	.782	.648	.677	.707
LambdaMART	Labeled Data	.854	.745	.757	.790	.685	.380	.461	.558	.814	.709	.722	.747
	Ratio-Debiasing	.832	.712	.722	.755	.631	.365	.421	.506	.791	.669	.702	.730
	Regression-EM	.827	.680	.693	.741	.628	.356	.411	.490	.785	.650	.681	.711
	Randomization	.824	.675	.687	.725	.624	.346	.407	.482	.784	.648	.678	.705
	Click Data	.814	.666	.673	.712	.614	.339	.396	.473	.779	.635	.664	.694
DNN	Labeled Data	.831	.685	.705	.737	.678	.364	.454	.551	.802	.700	.722	.745
	InfoRank (Debiasing)	.828	.683	.696	.734	.637	.360	.416	.499	.786	.667	.692	.725
	Dual Learning	.825	.680	.693	.730	.625	.352	.410	.487	.784	.663	.688	.720
	Regression-EM	.823	.676	.689	.726	.618	.347	.400	.479	.779	.656	.675	.713
	Randomization	.822	.677	.686	.724	.617	.346	.397	.477	.777	.644	.664	.701
	Click Data	.817	.665	.671	.710	.612	.335	.387	.469	.775	.633	.659	.688

Ranker	Debiasing Method	Yahoo (PBM)			Yahoo (CCM)			LETOR (PBM)			LETOR (CCM)		
		MAP	N@5	N@10	MAP	N@5	N@10	MAP	N@5	N@10	MAP	N@5	N@10
InfoRank (Ranking)	Labeled Data	.856	.760	.795	.856	.760	.795	.695	.468	.563	.695	.468	.563
	InfoRank (Debiasing)	**.849***	**.732***	**.772***	**.846***	**.712***	**.758***	**.681***	**.457***	**.559***	**.658***	**.455***	**.539***
	Regression-EM	.841	.715	.740	.822	.685	.734	.675	.453	.552	.652	.450	.534
	Randomization	.840	.704	.736	.817	.679	.728	.671	.450	.551	.649	.449	.531
	Click Data	.831	.682	.725	.808	.658	.710	.647	.445	.510	.640	.439	.498
LambdaMART	Labeled Data	.854	.757	.790	.854	.757	.790	.685	.461	.558	.685	.461	.558
	Ratio-Debiasing	.836	.728	.764	.828	.691	.738	.648	.446	.513	.644	.440	.502
	Regression-EM	.830	.700	.743	.816	.675	.727	.636	.436	.509	.634	.431	.497
	Randomization	.827	.690	.728	.814	.673	.722	.633	.433	.498	.628	.427	.493
	Click Data	.820	.672	.716	.804	.653	.706	.630	.424	.494	.625	.418	.488
DNN	Labeled Data	.831	.705	.737	.831	.705	.737	.678	.454	.551	.678	.454	.551
	InfoRank (Debiasing)	.829	.703	.736	.828	.692	.735	.651	.446	.547	.650	.444	.531
	Dual Learning	.828	.697	.734	.823	.681	.731	.645	.437	.528	.638	.430	.525
	Regression-EM	.829	.699	.736	.819	.678	.728	.635	.426	.500	.628	.417	.490
	Randomization	.825	.693	.732	.816	.674	.726	.630	.419	.495	.625	.415	.487
	Click Data	.819	.667	.711	.801	.650	.705	.629	.419	.492	.621	.409	.480



Average position after re-ranking of items at each normalized frequency by different debiasing methods together with InfoRank and InfoRank⁻ on Yahoo.



Average position after re-ranking of items at each original position by different debiasing methods together with InfoRank and InfoRank⁻ on Yahoo.

University of California San Diego

Shanghai Jiao Tong University

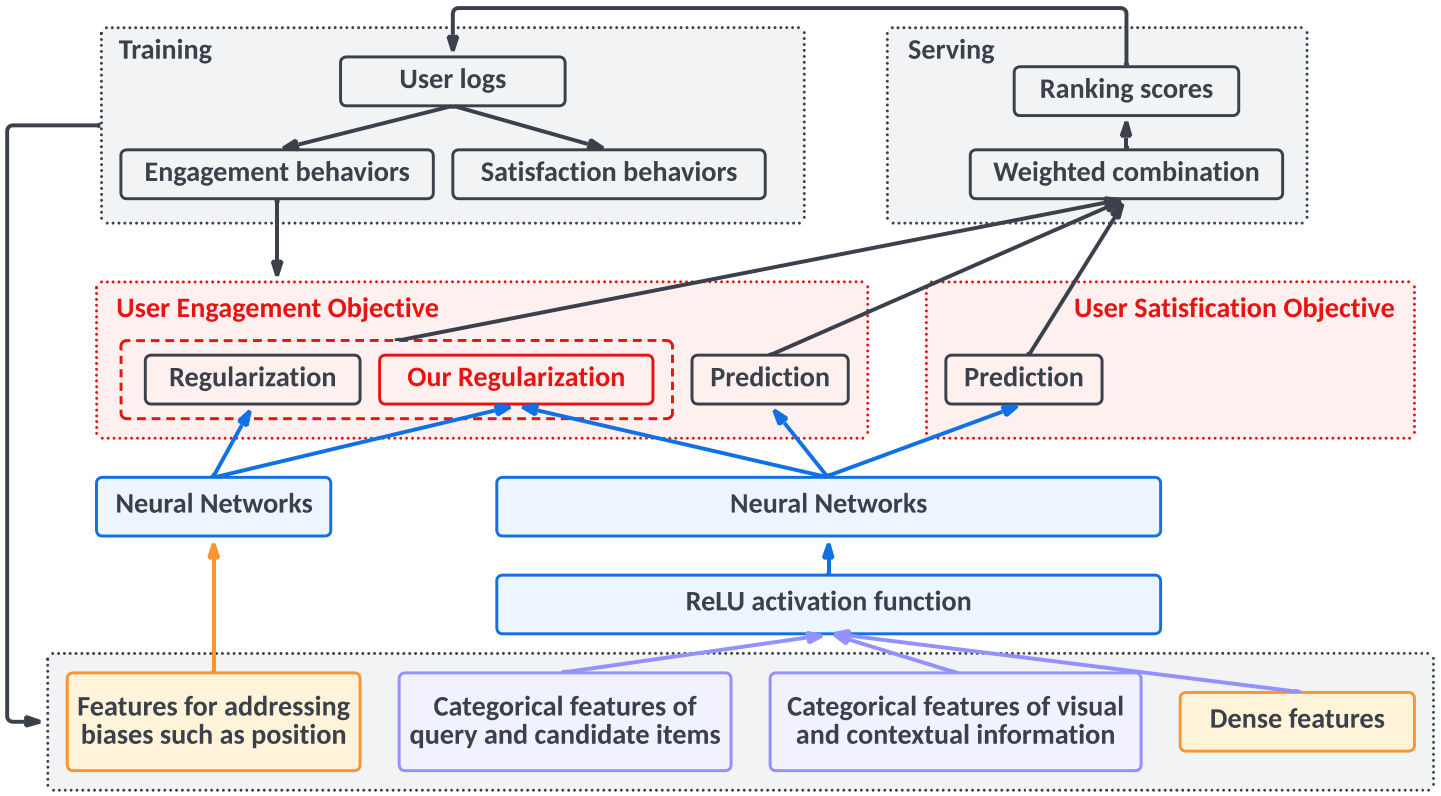
Content

- Problem Background
  - Biases in Learning-to-Rank
  - Unbiased Learning-to-Rank
- Architecture
  - Causality in Ranking
  - InfoRank
- Experiment
  - Simulator
  - Results
- **Conclusion**

University of California San Diego

Shanghai Jiao Tong University

Deployment Feasibility



- In many tower-based recommendation platform, existing learning-to-rank platforms apply a shadow tower responsible for generating observation estimations, as position significantly affects the observation of an item.
- Therefore, to integrate the proposed mutual information minimization into the pipeline, we only need to adapt their regularization term to ours. This modification mainly involves switching from the existing "regularization" term to our proposed "conditional mutual information regularization" term.

Conclusion

- We propose to summarize the position bias and popularity bias into a single observation factor.
- We derive the conditional mutual information minimization to push the conditional independence between the relevance estimation and observation estimation to simultaneously address the position bias and popularity bias.
- Our InfoRank framework can be seamlessly applied into existing learning-to-rank platforms.

Thanks for Your Listening