# causal inference in recommendation system

Warwick NLP

Hanqi Yan

Basic Idea:

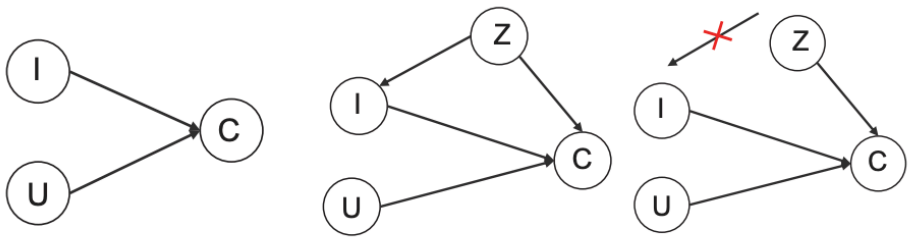
a. **Adjust confonding by causal inference**
b. modify original data to counterfactual
c. prompt-based

1. Causal Intervention for Leveraging **Popularity Bias** in Recommendation
2. Be Causal: De-biasing **Social Network** Confounding in Recommendation
3. Deconfounded Recommendation for Alleviating **Bias Amplification**

4. A Model-Agnostic Causal Learning Framework for Recommendation using Search Data

paper1

Popularity Bias




(a) Causal graph of traditional methods. (b) Causal graph that considers item popularity. (c) We cut off $Z \rightarrow I$ for model training.

**Figure 1: Causal graphs to describe the recommendation process. U: user, I: exposed item, C: interaction probability, Z: item popularity. We identify $Z$ as the confounder between $I$ and $C$, and propose deconfounded training with $P(C|do(U,I))$ as the interest matching.**

paper3

Imbalanced distribution (major/popular item groups) in historical data

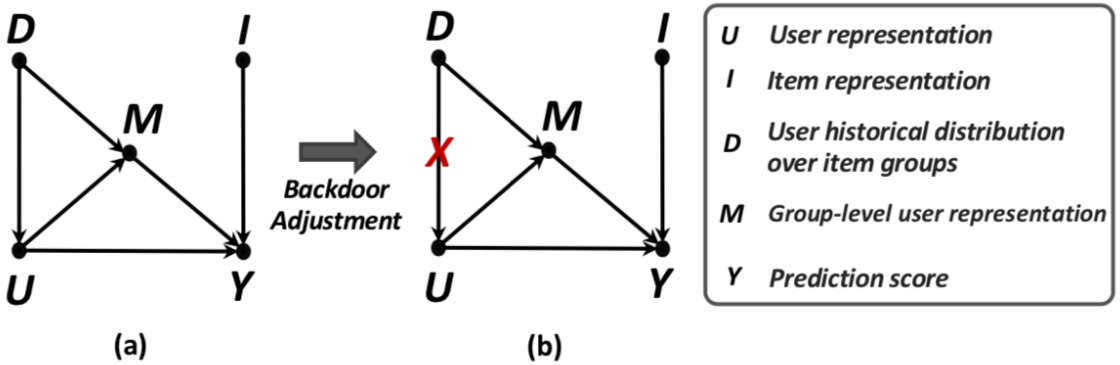



**Figure 2: (a) The causal graph of conventional RS. (b) The causal graph used in DecRS.**

paper2

Social Network introduced exposure bias

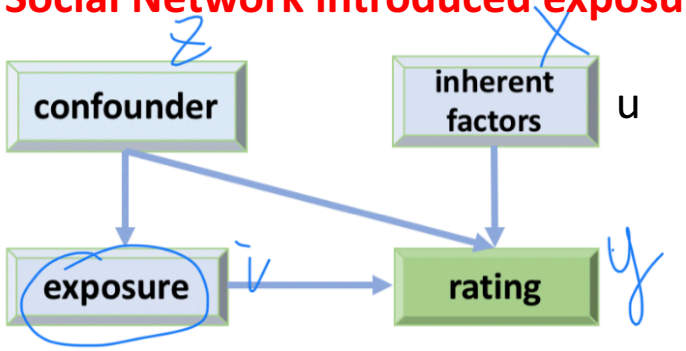



**Figure 3: The causal graph for recommendation.**

## Key points/ Common things:

- **Step1**: Construct Causal graph: Identify bias/confounder
- **Step2**: Derived Unbiased/ Deconfoundeing system by formulation
- **Step3**: Propose training(unbiased) and inference(incorporate confounder) strategy

**+ Separate module for modeling confounder**

# Causal Intervention for Leveraging Popularity Bias in Recommendation

# Modeling the item popularity

## local popularity

item frequency in historical data at t timestamp

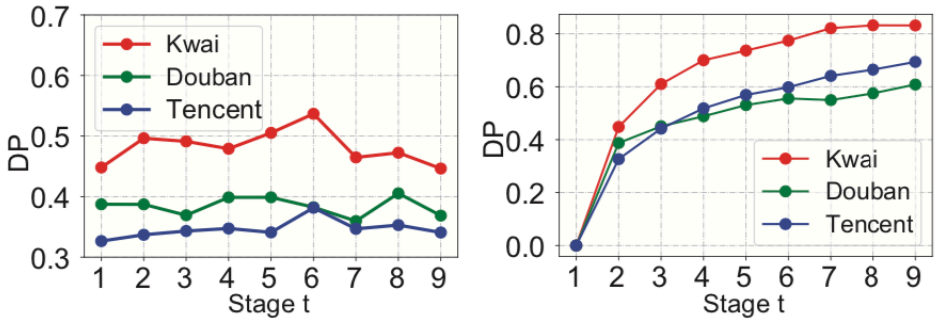
Let $\mathcal{D}$ denote the historical data, which is sequentially collected through $T$ stages, *i.e.*, $\mathcal{D} = \{\mathcal{D}_1 \cup \cdots \cup \mathcal{D}_T\}$; $\mathcal{U} = \{u_1, \ldots, u_{|\mathcal{U}|}\}$ and $\mathcal{I} = \{i_1, \ldots, i_{|\mathcal{I}|}\}$ denote all users and items, respectively. Through learning on historical data, the recommender system is expected to capture user preference and serves well for the next stage $T+1$. That is, it aims to obtain high recommendation accuracy on $\mathcal{D}_{T+1}$. In this work, we are interested in the factor of item popularity, defining the *local popularity* of item $i$ on the stage $t$ as:

confounder z

$$\boxed{m_i^t} = D_i^t / \sum_{j \in \mathcal{I}} D_j^t, \tag{1}$$

where $D_i^t$ denotes the number of observed interactions for item $i$ in $\mathcal{D}_t$. We can similarly define the global popularity of an item $m_i$ based on its interaction frequency in $\mathcal{D}$, but we think the local popularity has a larger impact on the system's exposure mechanism and user decision, since the system is usually periodically retrained and the most recent data has the largest impact.

## Drift of Popularity




(a) $DP(t, t+1)$ (b) $DP(1, t)$

**Figure 2: Popularity drift between: (a) two successive stages $DP(t, t+1)$; (b) the first and present stage $DP(1, t)$.**

stages. First, we represent each stage $t$ as a probability distribution over items: $[m_1^t, \ldots, m_{|\mathcal{I}|}^t]$, where each entry denotes an item's frequency in the stage. Then, we use the Jensen-Shannon Divergence (JSD) [15] to measure the similarity between two stages:

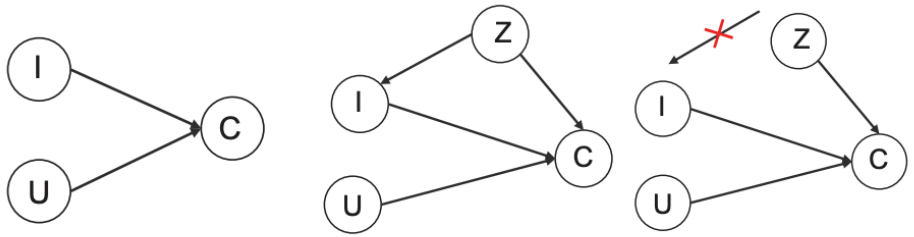
popularity shift

$$DP(t, s) = JSD\left([m_1^t, \ldots, m_{|\mathcal{I}|}^t], [m_1^s, \ldots, m_{|\mathcal{I}|}^s]\right), \tag{2}$$

where $t$ and $s$ are the two stages. Similar to JSD, the range of DP is $[0, log(2)]$, and a higher value indicates a larger popularity drift.

Basic Version: **P**opularity **D**eConfounding (**PD**)

Deconfounded Training




(a) Causal graph of traditional methods. (b) Causal graph that considers item popularity. (c) We cut off $Z \rightarrow I$ for model training.

**Key idea: Performing *do* ($I$ ) forces to remove the impact of $I$ 's parent nodes**

$$
\begin{aligned}
P(C|do(U,I)) &\overset{(1)}{=} P_{G'}(C|U,I) \\
&\overset{(2)}{=} \sum_z P_{G'}(C|U,I,z)P_{G'}(z|U,I) \\
&\overset{(3)}{=} \sum_z P_{G'}(C|U,I,z)P_{G'}(z) \\
&\overset{(4)}{=} \sum_z P(C|U,I,z)P(z),
\end{aligned}
\tag{3}
$$

where $P_{G'}(\cdot)$ denotes the probability function evaluated on $G'$.

[3]Note that since there is no backdoor path between $U$ and $C$ in our causal graph, it holds that $P(C|U, do(I)) = P(C|do(U), do(I)) = P(C|do(U,I))$. Here we use $P(C|do(U,I))$ as the intervention model for brevity.

## Step 1. Estimating $P(C|U,I,Z)$.

**Traditional RS (BPR loss)**

$$
\max_{\Theta} \sum_{(u,i,j)\in\mathcal{D}} \log\sigma\left(P_\Theta\left(c=1|u,i,m_i^t\right) - P_\Theta\left(c=1|u,j,m_j^t\right)\right),
$$

**Separate the UI and popularity**

$$
P_\Theta(c=1|u,i,m_i^t) = ELU'(f_\Theta(u,i)) \times (m_i^t)^\gamma, \tag{5}
$$

where $f_\Theta(u,i)$ denotes any user-item matching model and we choose the simple *Matrix Factorization* (MF) in this work; hyper-parameter

## Step 2. Estimating $\sum_z P(C|U,I,z)P(z)$.

$$
\begin{aligned}
P(C|do(U,I)) &= \sum_z P(C|U,I,z)P(z) \\
&= \sum_z ELU'(f_\Theta(u,i)) \times z^\gamma P(z) \\
&= ELU'(f_\Theta(u,i)) \sum_z z^\gamma P(z) \\
&= ELU'(f_\Theta(u,i)) E(Z^\gamma)
\end{aligned}
$$

where $E(Z^\gamma)$ denotes the expectation of $Z^\gamma$. Note that the expectation of one variable is a constant. Since $P(C|do(U,I))$ is used to rank items for a user, the existence of $E(Z^\gamma)$ does not change the ranking. We can thus use $ELU'(f_\Theta(u,i))$ to estimate $P(C|do(U,I))$.

# Advanced Version: **P**opularity **D**eConfounding and Adjusting (**PDA**)

**Seek better usage of popularity**

Add popularity z during inference

mendation policy with this bias. To achieve the target, we just need to do the intervention of $Z = \tilde{z}$ for model inference:

$$P(C|do(U = u, I = i), do(Z = \tilde{z})) = P_{\Theta}(c = 1|u, i, \tilde{m}_i), \tag{7}$$

where $\tilde{m}_i$ denotes the popularity value of $\tilde{z}$. This intervention probability directly equals the conditional probability since there is no

$$\tilde{m}_i = m_i^T + \alpha(m_i^T - m_i^{T-1}), \tag{8}$$

where $m_i^T$ is the popularity value of the last stage, and $\alpha$ is the hyper-parameter to control the strength of popularity drift in predicting

stage optimizes Equation (4) and the inference stage can adjust the popularity bias as:

$$PDA_{ui} = ELU'(f_{\Theta}(u, i)) \times \boxed{(\tilde{m}_i)^{\tilde{\gamma}}.} \tag{9}$$

$\tilde{\gamma}$ denotes the popularity smoothing hyper-parameter used for model inference, which can be different from that used in training.

**Algorithm 1:** PD/PDA

**Input:** dataset $\mathcal{D} = \{(u, i, m_i^t)\}$; hyper-parameter $\gamma$;
predicted popularity $\{\tilde{m}_i\}$; *mode*: PD or PDA

**while** *stop condition is not reached* **do**
  Update model parameters $\Theta$ by optimizing Equation (4);
  **if** *mode==PD* **then** validate model with $ELU'(f_\Theta(u, i))$;
  **else** validate model with Equation (9) (simplify $\tilde{\gamma} = \gamma$);
**end**
**if** *mode==PD* **then** recommend items using $ELU'(f_\Theta(u, i))$;
**else** recommend items using Equation (9) (simplify $\tilde{\gamma} = \gamma$);

$$\max_{\Theta} \sum_{(u,i,j)\in\mathcal{D}} \log\sigma\left(P_\Theta\left(c = 1|u, i, m_i^t\right) - P_\Theta\left(c = 1|u, j, m_j^t\right)\right),$$

Train as p(c|U,I,z)

$$\theta$$

```
def create_bpr_loss_with_pop_global(self, users, pos_items, neg_items): # this global does not refer to global popularity, just a name
    pos_scores = tf.reduce_sum(tf.multiply(users, pos_items), axis=1)   #users, pos_items, neg_items have the same shape
    neg_scores = tf.reduce_sum(tf.multiply(users, neg_items), axis=1)
    # item stop

    pos_scores = tf.nn.elu(pos_scores) + 1
    neg_scores = tf.nn.elu(neg_scores) + 1
    pos_scores_with_pop = tf.multiply(pos_scores,self.pos_pop)
    neg_scores_with_pop = tf.multiply(neg_scores,self.neg_pop)

    maxi = tf.log(tf.nn.sigmoid(pos_scores_with_pop - neg_scores_with_pop)+1e-10)
    self.condition_ratings = (tf.nn.elu(self.batch_ratings) + 1 ) * tf.squeeze(self.pos_pop)
    self.mf_loss_ori = tf.negative(tf.reduce_mean(maxi))
    mf_loss = self.mf_loss_ori
    # regular
    regularizer = tf.nn.l2_loss(users) + tf.nn.l2_loss(pos_items) + tf.nn.l2_loss(neg_items)
    regularizer = regularizer/self.batch_size

    reg_loss = self.decay * regularizer
    return mf_loss, reg_loss
```

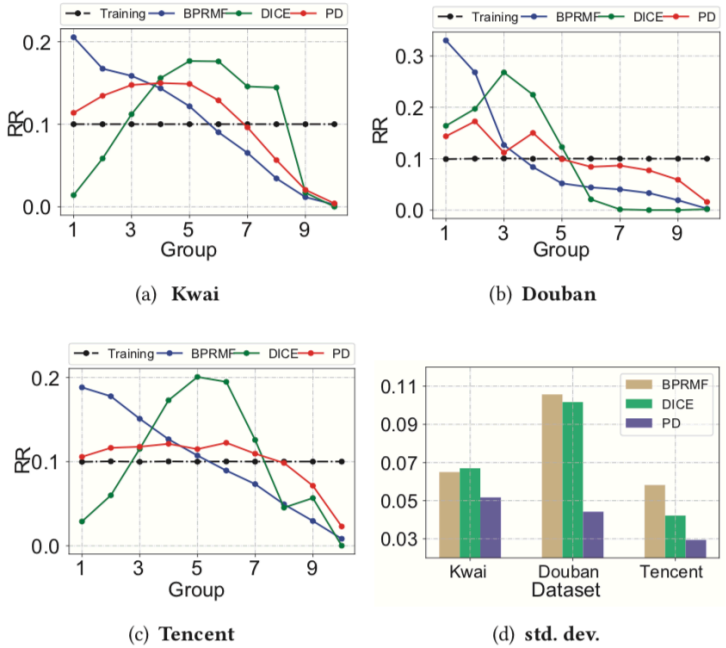
```
'''PD/PDG result without injecting popularity'''
self.batch_ratings = tf.matmul(user_embedding, pos_item_embedding, transpose_a=False, transpose_b = True)
```

’’’ PDA with injecting popularity ’’’

```
self.condition_ratings = (tf.nn.elu(self.batch_ratings) + 1 ) * tf.squeeze(self.pos_pop)
```

# why it works?

- There seems a discrepancy between training and inference with PD.
  - We claim that PD can alleviate the amplification of popularity bias, i.e., removing the bad effect of popularity bias. We conduct an analysis of the recommendation lists to show that deconfounded recommendation does not favor popular items as much as conventional recommendations
  -




(a) **Kwai** (b) **Douban** (c) **Tencent** (d) **std. dev.**

**Figure 4: Recommendation rate(RR) over item groups.**

# Deconfounded Recommendation for Alleviating Bias Amplification

**Imbalanced distribution (major/popular item groups) in historical data**

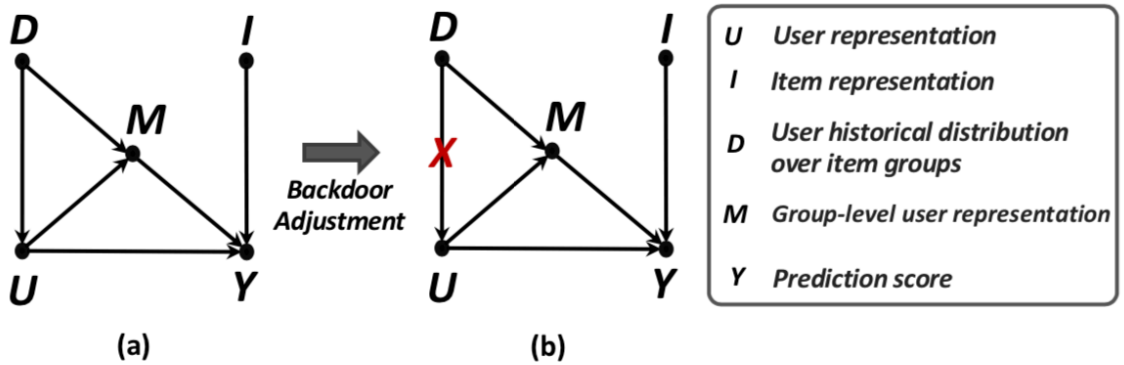



**Figure 2: (a) The causal graph of conventional RS. (b) The causal graph used in DecRS.**

*2.2.1 Backdoor Adjustment.* According to the theory of backdoor adjustment [25], the target of DecRS is formulated as: $P(Y|do(U = \boldsymbol{u}), I = \boldsymbol{i})$ where $do(U = \boldsymbol{u})$ can be intuitively seen as cutting off the edge $D \rightarrow U$ in the causal graph and blocking the effect of $D$ on $U$ (*cf.* Figure 2(b)). We then derive the specific expression of backdoor adjustment. Formally,

$$P(Y|do(U = \boldsymbol{u}), I = \boldsymbol{i})$$

$$= \sum_{\boldsymbol{d} \in \mathcal{D}} P(\boldsymbol{d}|do(U = \boldsymbol{u}))P(Y|do(U = \boldsymbol{u}), \boldsymbol{i}, M(\boldsymbol{d}, do(U = \boldsymbol{u}))) \quad (2a)$$

$$= \sum_{\boldsymbol{d} \in \mathcal{D}} P(\boldsymbol{d})P(Y|do(U = \boldsymbol{u}), \boldsymbol{i}, M(\boldsymbol{d}, do(U = \boldsymbol{u}))) \quad (2b)$$

$$= \sum_{\boldsymbol{d} \in \mathcal{D}} P(\boldsymbol{d})P(Y|\boldsymbol{u}, \boldsymbol{i}, M(\boldsymbol{d}, \boldsymbol{u})), \quad (2c)$$

how to estimate p(d)?

$$P(Y|do(U = \boldsymbol{u}), I = \boldsymbol{i}) \approx f(\boldsymbol{u}, \boldsymbol{i}, M(\boldsymbol{\boxed{\sum_{d \in \tilde{\mathcal{D}}} P(d)d}}, \boldsymbol{u})). \quad (5)$$

**Modeling p(d)**

- *Sampling of D.* To estimate the distribution of $D$, we sample users' historical distributions over item groups in the training data, which comprise a discrete set $\tilde{\mathcal{D}}$. Formally, given a user $u$, $\boldsymbol{d}_u = [p_u(g_1), ..., p_u(g_N)] \in \tilde{\mathcal{D}}$ and each click frequency $p_u(g_n)$ over group $g_n$ is calculated by

$$p_u(g_n) = \sum_{i \in \mathcal{I}} p(g_n|i)p(i|u) = \frac{\sum_{i \in \mathcal{H}_u} q^i_{g_n}}{|\mathcal{H}_u|}, \quad (3)$$

根据统计 train data

**Add confounder z during inference**

$$Y_{u,i} = (1 - \hat{\eta}_u) * Y^{RS}_{u,i} + \hat{\eta}_u * Y^{DE}_{u,i}, \quad (11)$$

where $Y_{u,i}$ is the inference score for user $u$ and item $i$, $Y^{RS}_{u,i}$ and $Y^{DE}_{u,i}$ are the prediction scores from the conventional RS and DecRS,

# Be Causal: De-biasing Social Network Confounding in Recommendation

## Social Network introduced exposure bias

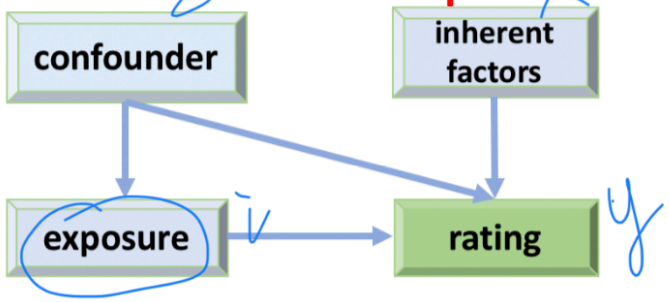



Figure 3: The causal graph for recommendation.

## Modeling exposure bias. Zu

$$\mathcal{L}_z = -\sum_{u \in G} \log P(N_s(u)|Z_u)$$
$$= \sum_{u \in G} \left[ \log \sum_{v \in G} \exp(Z_v \cdot Z_u) - \sum_{u_i \in N_s(u)} Z_{u_i} \cdot Z_u \right] \quad (1)$$

The final output $Z_u \in \mathbb{R}^d$ sufficiently explores diverse neighborhoods of each user, which thus represents to what extent the exposure for a user is influenced by his friends in graph $G$.

## Exposure Assignment Learning a_{u,i}

defines whether the item $i$ is exposed ($a_{ui} = 1$) or unexposed ($a_{ui} = 1$) to user $u$, i.e., $a_{ui} = 1$. Based on the informative confounder learned from social network, we propose the notation of *propensity* to capture the exposure from the causal inference language.

Definition 1 (Propensity). *Given an observed rating $y_{ui} \in$ rating and confounder $Z_u$ in (1), the propensity of the corresponding exposure for user–item pair $(u, i)$ is defined as*

$$\pi(a_{ui}; Z_u) = P(a_{ui} = 1 | y_{ui} \in rating; Z_u) \quad (2)$$

## Incorporate the confounder for inference

$$\hat{y}_{ui} = \sum_{u,i \in O} U^\top I + W_u{}^\top Z_u + \epsilon_{ui}, \quad \epsilon_{ui} \sim \mathcal{N}(0, 1) \quad (13)$$

**the extent the user is exposed to the item**

where $W_u$ is coefficient that describes how much the confounder $Z_u$ contributes to the rating. To define the unbiased loss function for the biased observations $y_{ui}$, we leverage the IPS strategy [29] to weight each observation with *Propensity*. By Definition 1, the intuition of the inverse propensity is to down-weight the commonly observed ratings while up-weighting the rare ones.

$$\mathcal{L}_y = \frac{1}{|O|} \sum_{u,i \in O} \frac{(y_{ui} - \hat{y}_{ui})^2}{\pi(a_{ui}; Z_u)} \quad (14)$$

**uplift weight for rare exposed sample**

3.5.2 *Optimization.* To this end, the objective function of our DENC method to predict ratings could be derived as:

$$\mathcal{L} = \mathcal{L}_y + \lambda_a \mathcal{L}_a + \lambda_z \mathcal{L}_z + \lambda_d \mathcal{L}_d + \mathcal{R}(\Omega) \quad (15)$$

**Ld** assignment independent of confounder

and items, i.e., $\Phi_{(a)} = \left[U_1^{(a)}, \cdots, U_M^{(a)}, I_1^{(a)}, \cdots, I_M^{(a)}\right] \in \mathbb{R}^{k_d \times 2M}$, $\Phi_{(\hat{a})} = \left[U_1^{(\hat{a})}, \cdots, U_M^{(\hat{a})}, I_1^{(\hat{a})}, \cdots, I_M^{(\hat{a})}\right] \in \mathbb{R}^{k_d \times 2M}$. Figure 3 also vice versa. That means $U \in \mathbb{R}^{k_d}$ and $I \in \mathbb{R}^{k_d}$ should be independent of the exposure assignment, i.e., $U^{(a)} \perp\!\!\!\perp U^{(\hat{a})}$ or $I^{(a)} \perp\!\!\!\perp I^{(\hat{a})}$. Accordingly, minimizing the discrepancy between $\Phi_{(a)}$ and $\Phi_{(\hat{a})}$ : the

Based on Eq. (7), the overall exposure $P(a_{ui})$ in Eq. (6) can be written as the function of parameters $\Theta = \{W_0, b_0\}$ and $Z_u$, i.e., :erm

$$\mathcal{L}_a = \sum_{u,i} -\log P(a_{ui}; Z_u, \Theta) \quad (8) \quad (9)$$

**La:** MLE for exposure modeling

# A Model-Agnostic Causal Learning Framework for Recommendation using Search Data

## Key points/ Common things:

- **Step1**: Construct Causal graph: Identify bias/confounder
- **Step2**: Derived Unbiased/ Deconfoundeing system by formulation
- **Step3**: Propose training(unbiased) and inference(incorporate confounder) strategy

**+ Separate module for modeling confounder**

# Causal Intervention for Leveraging Popularity Bias in Recommendation

# Should look beyond the RS, to general NLP

**Causal Inference in Natural Language Processing: Estimation, Prediction, Interpretation and Beyond**

Amir Feder[1*], Katherine A. Keith[2], Emaad Manzoor[3], Reid Pryzant[4], Dhanya Sridhar[5], Zach Wood-Doughty[6], Jacob Eisenstein[7], Justin Grimmer[4], Roi Reichart[1], Margaret E. Roberts[8], Brandon M. Stewart[9], Victor Veitch[7,10], and Diyi Yang[11]

[1]Technion - Israel Institute of Technology
[2]University of Massachusetts Amherst
[3]University of Wisconsin - Madison
[4]Stanford University
[5]Columbia University
[6]Johns Hopkins University
[7]Google Research
[8]University of California San Diego
[9]Princeton University
[10]University of Chicago
[11]Georgia Tech

**Go to PDF**

# Causal Inference & NLP

**NLP helps causality.** Causal inference with text data involves several challenges that distinct from typical causal inference settings: text is high-dimensional, needs sophisticated modeling to measure semantically meaningful factors like topic, and demands careful thought to formalize the intervention that a causal question corresponds to. The developments in NLP around modeling language, from topic models to contextual embeddings, offer promising ways to extract the information we need from text to estimate causal effects.

**Causality can help NLP.** To address the robustness and interpretability challenges posed by NLP methods, we need new criteria to learn models that go beyond exploiting correlations. For example, we want predictors that are invariant to certain changes that we make to text, such as changing the format while holding fixed the ground truth label. There is considerable promise in using causality to develop new criteria in service of building robust and interpretable NLP methods. In contrast to the well-studied area of causal inference with text, this area of causality and NLP research is less well-understood, though well-motivated by recent empirical successes. In Section 4, we cover the existing research and review the challenges and opportunities around using causality to improve NLP.

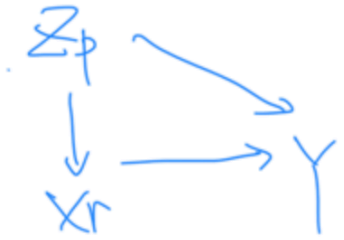
# My problem Set UP

Y is rating for a new user-item pair
X is treatment
Z is confounder

$S_2$: Review Text is treatment. $X_r$

Confounder is Personality. $Z_p$ { score tendency. expression habit

Motivation: even review is the same.
rating can be different.

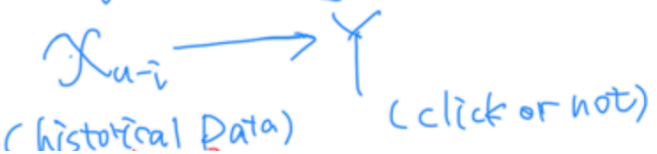



Review Text is confounder. $Z_r$

1. reflect aspect-level preference → interest → I
2. Score tendency → Y

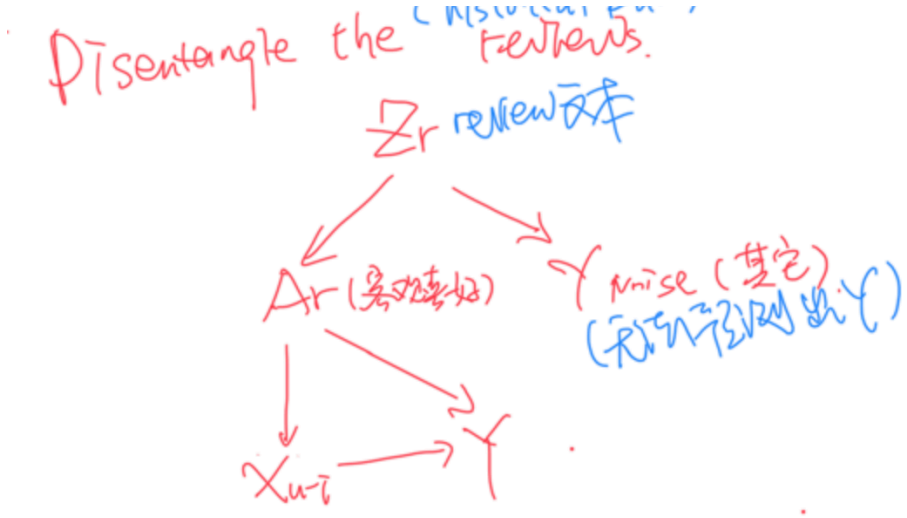
aspect-level Interest

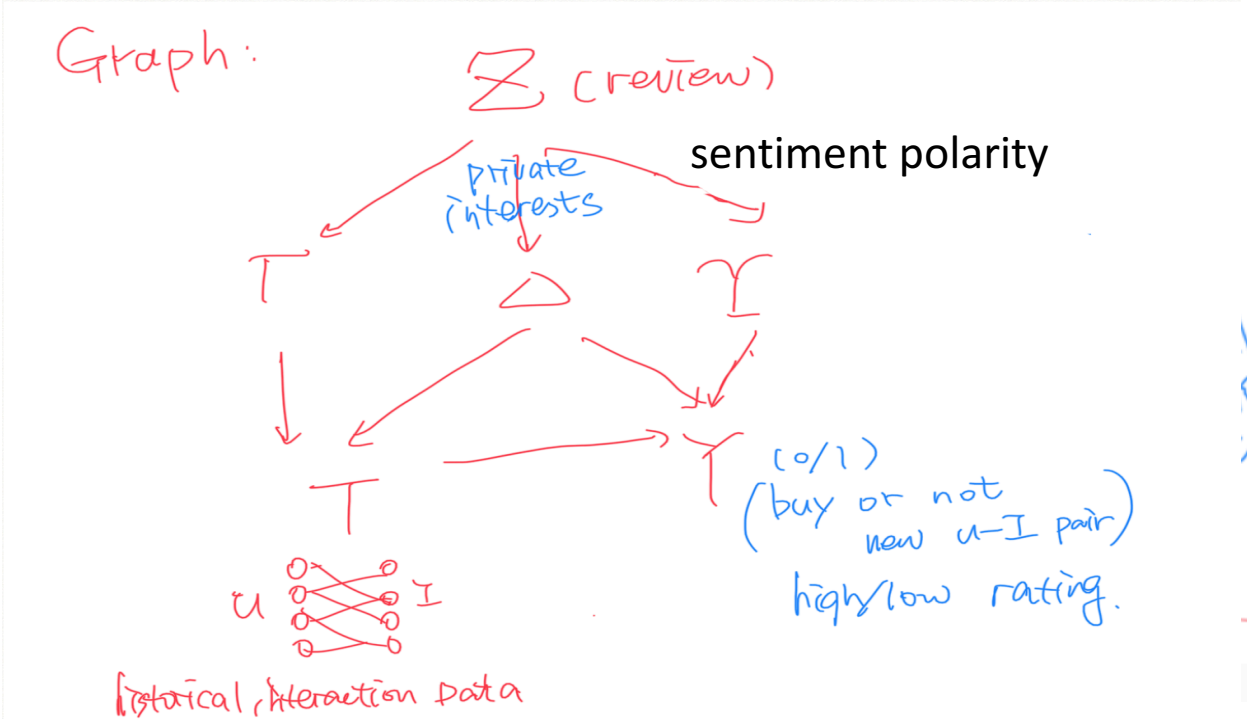
$Z_r$ (historical review) 混杂了 { interest 表达 ✓ $A_r$
语言习惯. ✗ $Y_{noise}$
(评分分数).



**Disentangle Z** high-level methodology

LEARNING DISENTANGLED REPRESENTATIONS FOR COUNTERFACTUAL REGRESSION, ICLR 2020[link]




| | |
|---|---|
| $\gamma\Gamma$ | \gamma \Gamma |
| $\delta\Delta$ | \delta \Delta |
| $\upsilon\Upsilon$ | \upsilon \Upsilon |

# Counterfactual Explainable Recommendation

*aspect-level explanation