Toward building self-training search systems

Hang Li
Bytedance technology
Talk outline

• *Evolution of Search Technologies*
• Self-Training Search System
• Unbiased Learning to Rank
• Previous Work
• Our Work: Unbiased LambdaMART
• Conclusions
EVOLUTION OF SEARCH TECHNOLOGIES

1970
Library Search

1990
Web Search

2010
Natural Language Dialogue

From My Keynote at CCIR 2015:
Natural Language Dialogue – Future Way of Accessing Information
WEB SEARCH TECHNOLOGIES

• IR Models, e.g., BM25, LM4IR
• Link Analysis, e.g., PageRank
• Query Understanding, e.g., informational vs navigational
• Learning to Rank, e.g., LambdaMART
• Click Models, e.g., Position based Model
• Deep Matching Models, e.g., DSSM
• Unbiased Learning to Rank

This Talk
Talk outline

• Evolution of Search Technologies
• *Self-Training Search System*
• Unbiased Learning to Rank
• Previous Work
• Our Work: Unbiased LambdaMART
• Conclusions
Web search system

Currently Not An Autonomous Learning System

- Query Understanding
- Ranking
- Crawling
- Matching
- Document Understanding
- Indexing
- Indexing
Recommender system

Ranking → Selection

User Profiles → User IDs

Items

Autonomous Learning System
Future question answering system

Information and Knowledge Acquisition

- Encoding
- Storing
- Decoding
- Retrieving

Neural Symbolic Processing

Information and Knowledge

Ideally, Autonomous Learning System

Unstructured Data

Structured Data
Future question answering system

Information and Knowledge Access

- Encoding
- Decoding
- Storing
- Retrieving

Ideally, Autonomous Learning System

Neural Symbolic Processing

Information and Knowledge
SELF TRAINING SEARCH SYSTEM

Autonomous Learning System

$q$

$\{d\}$

$\{c\}$

$\{(q, d, c)\}$: click data

$P$ (Prediction): $f(q, d)$: ranking model

$M$ (Matching): $m(q, d)$: matching model

$G$ (Query): $g(q)$: query model

$h(d)$: document model

Learning
Opportunities and Challenges

• **Opportunities**
  - Click Data Usually Represents Users’ Implicit Relevance Feedback
  - Easy to Collect with Low Cost
  - Better System Adaptation if Self-Trained with Click Data

• **Challenges**
  - Click Data is Noisy
  - Click Data Has Biases, Including Position Bias, Presentation Bias
  - Click Data May Contain Spam
Talk outline

• Evolution of Search Technologies
• Self-Training Search System
• *Unbiased Learning to Rank*
• Previous Work
• Our Work: Unbiased LambdaMART
• Conclusions
Learning to rank

• Learning to Rank = Learning Ranking Model from Data
• Ranking Model: \( f(q, d) \) or \( f(x) \)
• Features: \( m(q, d), g(q), h(d), \cdots \)
• Training Data: \( \{(q, d, r)\} \), Usually Labeled by Humans
• Three Approaches with Different Types of Loss Functions
  • Pointwise Loss Function: \( L(f(x_i), r^+_i) \)
  • Pairwise Loss Function: \( L(f(x_i), f(x_j), r^+_i, r^-_j) \)
  • Listwise Loss Function: \( L(f(x_1), \cdots, f(x_k), r_1, \cdots, r_k) \)
Lambda-MART

• State-of-the-Art Learning to Rank Algorithm
• Model: Gradient Boosting or MART (Multiple Additive Regression Trees)
• Lambda Function: Gradient of Pairwise Loss Function
  • \( \lambda_i = \sum_{j:(d_i,d_j) \in I_q} \lambda_{ij} - \sum_{j:(d_j,d_i) \in I_q} \lambda_{ji} \)
  • \( \lambda_{ij} = \frac{-\sigma}{1+e^{\sigma(f(x_i)-f(x_j))}} |\Delta Z_{ij}| \)
• Iteratively Learn Additive Regression Trees based on Lambda Function
Unbiased learning to rank

• Unbiased Learning to Rank = Learning Ranking Model from Debiased Click Data

• In Self-Training Search System
  • Create Initial Ranker $f^{(0)}$
  
  • Repeat
    • Collect Click Data $\{(q, d, c)\}$
    • Conduct Debiasing of Click Data $\{(q, d, c)\} \rightarrow \{(q, d, r)\}$
    • Train New Ranker $f^{(i)}$ with Debiased Click Data

• Key Question: How to Eliminate Biases (Position Bias, Presentation Bias)
Debiased Click data as training data

Query

Click Data

Click

Implicit Relevance Judgment

Debiasing

Training

Ranking Model

Ranking List of Documents
Position bias

- Eye Tracking Experiment (Joachims et al 2005)
- Results on Top Positions Receive More Attention and More Clicks
- Number of Clicks Decreases from Top to Bottom

Figure 1: Percentage of times an abstract was viewed/clicked depending on the rank of the result.
Talk outline

• Evolution of Search Technologies
• Self-Training Search System
• Unbiased Learning to Rank
• Previous Work
• Our Work: Unbiased LambdaMART
• Conclusions
Click models

- Model: $P(q, d, c, r, b)$
- Query, Document, Click Variables $(q, d, c)$ Are Observable
- Relevance, Bias Variables Are Hidden
- Position based Model, Cascade Model, User Browsing Model, Dynamic Bayesian Network Model, etc
- Bayesian Learning: $P(r|q, d, c) \cdot P(b|q, d, c)$
- Using Click Models for Debiasing Can Be Suboptimal
Online randomization

Example: Swap Results at Two Positions

\[ P(c|i, r) = P(c|i)P(c|r) \]
\[ P(c|j, r) = P(c|j)P(c|r) \]
\[ P(c|j) = \frac{P(c|j, r)}{P(c|i, r)} P(c|i) \]
\[ \propto \frac{P(c|j, r)}{P(c|i, r)} \]
Unbiased learning to rank

- **Wang et al. 2016**
  - Employed Pointwise “Inverse Propensity Weighting” Principle, Estimated Position Bias Using Online Randomization

- **Joachims et al. 2017**
  - Proved Pointwise IPW, Estimated Position Bias Using Online Randomization

- **Wang et al. 2018**
  - Proposed Method Directly Estimate Position Bias from Click Data, Using Pointwise IPW Principle

- **Ai et al. 2018**
  - Proposed Joint Learning of Position Bias Model and Ranking Model from Click Data, Again Using Pointwise IPW Principle
Talk outline

• Evolution of Search Technologies
• Self-Training Search System
• Unbiased Learning to Rank
• Previous Work
• Our Work: Unbiased LambdaMART
• Conclusions
Co-authors

Ziniu Hu
UCLA

Yang Wang
ByteDance AI Lab

Qu Peng
ByteDance
Unbiased learning to rank

• Currently Only Dealing with Position Bias
• Click Data \((q, d, c)\) → Relevance Data \((q, d, r)\)
• Train Ranker \(f(q, d) = f(x)\)
• Biased: Click = Relevant, Unclick = Irrelevant
• Unbiased: Click = Relevant, Unclick = Irrelevant, *But with Debiasing*
• Debiasing (Position Bias): Inverse Propensity Weighting Principle
Pointwise unbiased Learning to rank
(previous work)

• Pointwise Loss Function (Pointwise Approach)
• Biased: Click = Relevant, Unclick = Irrelevant
• Unbiased (Position Bias): Click = Relevant, Unclick = Irrelevant, with Debiasing
  • Inverse Propensity Weighting Principle: Pointwise Loss Divided by Bias
  • Theoretical Guarantee: Unbiased Estimate of Pointwise Relevance Loss
  • Explanation: Clicks at Higher Positions Are Less Relevant and Are Panelized
• Debiasing and Learning Can Be Jointly or Separately Conducted
Conventional Learning to Rank

\[
\int L(f(x_i), r_i^+)dP(x_i, r_i^+)
\]

\[
\arg\min_{f} \sum_{q} \sum_{d_i \in D_q} L(f(x_i), r_i^+)
\]

Biased Learning to Rank

\[
\int L(f(x_i), c_i^+)dP(x_i, c_i^+)
\]

\[
\arg\min_{f} \sum_{q} \sum_{d_i \in D_q} L(f(x_i), r_i^+)
\]

Note that position information is omitted from loss function for ease of explanation.
Bias

\[ P(c_i^+|x_i) = t_i^+ P(r_i^+|x_i) \]

\[ t_i^+ = P(c_i^+|r_i^+), \text{ if } c^+ \Rightarrow r^+ \]

\[ P(c_i^+|x_i) = P(c_i^+|r_i^+)P(r_i^+|x_i) \]
Bias: ration between click probability and relevance probability

\[ P(c_i^+ | x_i) = t_i^+ P(r_i^+ | x_i) \]

Unbiased Learning to Rank

\[ \int \frac{L(f(x_i), c_i^+)}{t_i^+} dP(x_i, c_i^+) \]

Inverse Propensity Weighting

\[ = \int \frac{L(f(x_i), c_i^+)}{P(c_i^+ | x_i)/P(r_i^+ | x_i)} dP(x_i, c_i^+) \]

Unbiased Estimate

\[ = \int L(f(x_i), r_i^+) dP(x_i, r_i^+) \]

\[ \arg\min_f \sum_q \sum_{d_i \in D_q} \frac{L(f(x_i), c_i^+)}{t_i^+} \]
Pairwise unbiased Learning to rank
(Our work)

• Pairwise Loss Function (Pairwise Approach)
• Biased: Click = Relevant, Unclick = Irrelevant
• Unbiased (Position Bias): Click = Relevant, Unclick = Irrelevant, \textit{with Debiasing}
  • Inverse Propensity Weighting Principle: Pairwise Loss Divided by Click Bias and Unclick Bias
  • Theoretical Guarantee: Unbiased Estimate of Pairwise Relevance Loss
  • Explanation: Click Bias Has Intuitive Explanation, Unclick Bias May Not
• Debiassing and Learning Can be Jointly or Separately Conducted
Conventional Learning to Rank

\[
\int L(f(x_i), r_i^+, f(x_j), r_j^-) dP(x_i, r_i^+, x_j, r_j^-)
\]

\[
\text{argmin}_f \sum_q \sum_{(d_i, d_j) \in I_q} L(f(x_i), r_i^+, f(x_j), r_j^-)
\]

Biased Learning to Rank

\[
\int L(f(x_i), c_i^+, f(x_j), c_j^-) dP(x_i, c_i^+, x_j, c_j^-)
\]

\[
\text{argmin}_f \sum_q \sum_{(d_i, d_j) \in I_q} L(f(x_i), c_i^+, f(x_j), c_j^-)
\]
Propensities: Ratio between Click Probability and Relevance Probability, 
Ratio between Unclick Probability and Irrelevance Probability

\[ P(c_i^+ | x_i) = t_i^+ P(r_i^+ | x_i) \quad P(c_i^- | x_i) = t_i^- P(r_i^- | x_i) \]

Unbiased Learning to Rank

\[
\int \frac{L(f(x_i), c_i^+, f(x_j), c_j^-))}{t_i^+ \cdot t_j^-} dP(x_i, c_i^+, x_j, c_j^-) \\
= \int \frac{L(f(x_i), c_i^+, f(x_j), c_j^-))}{P(c_i^+ | x_i)/P(r_i^+ | x_i)P(c_j^- | x_j)/P(r_j^- | x_j)} dP(x_i, c_i^+, x_j, c_j^-) \\
= \int L(f(x_i), r_i^+, f(x_j), r_j^-))dP(x_i, r_i^+, x_j, r_j^-)
\]

Inverse Propensity Weighting

Unbiased Estimate

\[
\argmin_f \sum_q \sum_{(d_i,d_j) \in I_q} \frac{L(f(x_i), c_i^+, f(x_j), c_j^-))}{t_i^+ \cdot t_j^-}
\]
Pairwise debiasing and unbiased lambda-mart

- Pairwise Debiasing: General Algorithm of Debiasing for Pairwise Learning to Rank Algorithms
- Unbiased LambdaMART: Combining Pairwise Debiasing and LambdaMART
- Based on Inverse Propensity Principle on Pairwise Loss
- Algorithm of Pairwise Debiasing: Iteratively Conducting Debiasing of Click Data and Learning of Ranker
Pairwise debiasing

\[
\min_{f, t^+, t^-} \mathcal{L}(f, t^+, t^-) = \min_{f, t^+, t^-} \sum_q \sum_{(d_i, d_j) \in I_q} \frac{L(f(x_i), c_i^+, f(x_j), c_j^-)}{t_i^+ t_j^-} + \| t^+ \|_p^p + \| t^- \|_p^p
\]

s.t. \( t_1^+ = 1, t_1^- = 1 \)

• Initialize Propensities
• Repeat
  • Fix Propensities \( t_1^+ \) and \( t_1^- \), Estimate Ranking Model \( f \)
  • Fix Ranking Model \( f \), Estimate Propensities \( t_1^+ \) and \( t_1^- \)
Pairwise debiasing

Estimating Biases (Closed Form Solution)

\[
t_i^+ = \left[ \frac{\sum q \sum_j: (d_i,d_j) \in I_q (L(f^*(x_i), c_i^+, f^*(x_j), c_j^-)/(t_j^-)^*)}{\sum q \sum_j: (d_1,d_k) \in I_q (L(f^*(x_1), c_1^+, f^*(x_k), c_k^-)/(t_k^-)^*)} \right]^{\frac{1}{p+1}}
\]

\[
t_j^- = \left[ \frac{\sum q \sum_i: (d_i,d_j) \in I_q (L(f^*(x_i), c_i^+, f^*(x_j), c_j^-)/(t_i^+)^*)}{\sum q \sum_k: (d_k,d_1) \in I_q (L(f^*(x_k), c_k^+, f^*(x_1), c_1^-)/(t_k^+)^*)} \right]^{\frac{1}{p+1}}
\]

Learning Ranker with Pairwise Learning to Rank Algorithm

\[
\frac{\partial L(f,(t^+)^*,(t^-)^*)}{\partial f} = \sum q \sum (d_i,d_j) \in I_q \frac{1}{(t^+)^*(t^-)^*} \frac{\partial L(f,(c^+)^*,(c^-)^*)}{\partial f}
\]
Unbiased lambda mart

Algorithm 1 Unbiased LambdaMART

Require: click dataset $\mathcal{D} = \{(q, D_q, C_q)\}$; hyper-parameters $p, M$

Ensure: unbiased ranker $f$; propensities $t^+$ and $t^-$

1: Initialize all propensities as 1;
2: for $m = 1$ to $M$ do
3: for each query $q$ and each document $d_i$ in $D_q$ do
4: Calculate $\tilde{\lambda}_i$ with $(t^+)^*$ and $(t^-)^*$ using (36) and (37);
5: end for
6: Re-train ranker $f$ with $\tilde{\lambda}$ using LambdaMART algorithm
7: Re-estimate propensities $t^+$ and $t^-$ with ranker $f^*$ using (31) and (32)
8: end for
9: return $f$, $t^+$, and $t^-$;
Experimental results

<table>
<thead>
<tr>
<th>Ranker</th>
<th>Debiasing Method</th>
<th>MAP</th>
<th>NDCG@1</th>
<th>NDCG@3</th>
<th>NDCG@5</th>
<th>NDCG@10</th>
</tr>
</thead>
<tbody>
<tr>
<td>LambdaMART</td>
<td>Labeled Data (Upper Bound)</td>
<td>0.854</td>
<td>0.745</td>
<td>0.745</td>
<td>0.757</td>
<td>0.790</td>
</tr>
<tr>
<td></td>
<td>Pairwise Debiasing</td>
<td><strong>0.836</strong></td>
<td><strong>0.717</strong></td>
<td><strong>0.716</strong></td>
<td><strong>0.728</strong></td>
<td><strong>0.764</strong></td>
</tr>
<tr>
<td></td>
<td>Regression-EM [22]</td>
<td>0.830</td>
<td>0.685</td>
<td>0.684</td>
<td>0.700</td>
<td>0.743</td>
</tr>
<tr>
<td></td>
<td>Randomization</td>
<td>0.827</td>
<td>0.669</td>
<td>0.678</td>
<td>0.690</td>
<td>0.728</td>
</tr>
<tr>
<td></td>
<td>Click Data (Lower Bound)</td>
<td>0.820</td>
<td>0.658</td>
<td>0.669</td>
<td>0.672</td>
<td>0.716</td>
</tr>
<tr>
<td>DNN</td>
<td>Labeled Data (Upper Bound)</td>
<td>0.831</td>
<td>0.677</td>
<td>0.685</td>
<td>0.705</td>
<td>0.737</td>
</tr>
<tr>
<td></td>
<td>Dual Learning Algorithm [1]</td>
<td>0.828</td>
<td>0.674</td>
<td>0.683</td>
<td>0.697</td>
<td>0.734</td>
</tr>
<tr>
<td></td>
<td>Regression-EM</td>
<td>0.829</td>
<td>0.676</td>
<td>0.684</td>
<td>0.699</td>
<td>0.736</td>
</tr>
<tr>
<td></td>
<td>Randomization</td>
<td>0.825</td>
<td>0.673</td>
<td>0.679</td>
<td>0.693</td>
<td>0.732</td>
</tr>
<tr>
<td></td>
<td>Click Data (Lower Bound)</td>
<td>0.819</td>
<td>0.637</td>
<td>0.651</td>
<td>0.667</td>
<td>0.711</td>
</tr>
<tr>
<td>RankSVM</td>
<td>Labeled Data (Upper Bound)</td>
<td>0.815</td>
<td>0.631</td>
<td>0.649</td>
<td>0.675</td>
<td>0.707</td>
</tr>
<tr>
<td></td>
<td>Regression-EM</td>
<td>0.815</td>
<td>0.629</td>
<td>0.648</td>
<td>0.674</td>
<td>0.705</td>
</tr>
<tr>
<td></td>
<td>Randomization [14]</td>
<td>0.810</td>
<td>0.628</td>
<td>0.644</td>
<td>0.672</td>
<td>0.707</td>
</tr>
<tr>
<td></td>
<td>Click Data (Lower Bound)</td>
<td>0.811</td>
<td>0.614</td>
<td>0.629</td>
<td>0.658</td>
<td>0.697</td>
</tr>
</tbody>
</table>

Unbiased LambdaMART Significantly Outperforms Existing Methods
Experimental results

• At Commercial Search Engine
• AB Testing: Unbiased LambdaMART vs. LambdaMART + Click Data
• Increasing Click Ratio at Positions 1, 3, 5 by 2.64%, 1.21%, 0.80%
Open questions

• How to Deal with Presentation Bias
• How to Combat Click Spam
• How to Extend to Reinforcement Learning (Exploration and Exploitation)
Talk outline

• Evolution of Search Technologies
• Self-Training Search System
• Unbiased Learning to Rank
• Previous Work
• Our Work: Unbiased LambdaMART
• Conclusions
Concluding remarks

• Three Paradigms in Information Retrieval: Library Search, Web Search, Natural Language Dialogue
• Web Search Technologies Are Still Evolving
• Self-Training Search Systems Are Ideal
• Key Technologies: Unbiased Learning to Rank
• Possible to Jointly Conducting Debiasing of Click Data and Learning of Ranker
• We Propose General Algorithm Pairwise Debiasing, and Specific Implementation Unbiased LambdaMART
Thank you!
lihang.lh@bytedance.com