Active Exploration for Learning Rankings from Clickthrough Data

Filip Radlinski, Thorsten Joachims
Cornell University
Motivation

- Problem: Rank documents for web search.
- There are many machine learning techniques for learning to rank.
- Sometimes, we can get expert judgments for training.
- Usually, easier to get training data from users.
  - e.g. for personalized search or for a specific document collection.
- How do we get the most useful training data from users, to quickly learn a good ranking?
Formalization

- Say we have a corpus: $C = \{d_1, \ldots, d_{|C|}\}$
- Say we are given a query $q$
- For this query, we want to learn the relevance for each $d_i$

Some ranking function provides initial estimates.

**Goal:** Learn the relevances with as little training data as possible.
Search is a *Process*

**Goal:** Learn the relevances with as little training data as possible.

- Search involves a three step process:
  1. Given relevance estimates, pick a ranking to display to users. *(our focus in this talk)*
  2. Given a ranking, users provide feedback:
     User clicks provide pairwise relevance judgments. *[Joachims et al, ‘07]*
  3. Given feedback, update the relevance estimates.
     *(we’ll briefly cover this)*
Plan of Attack

• What we know:
  (1) We have an estimate of the relevance of each result.
  (2) We get pairwise comparisons of the top few results.
  (3) We do not have absolute relevance information.

Goal: Learn the document relevances quickly.

• We need to address four questions:
  (1) How to represent knowledge about doc relevance.
  (2) How to maintain this knowledge as we collect data.
  (3) Given our knowledge, what is the best ranking?
  (4) What rankings do we show users to get useful data?
**Representing Relevance**

- Given a query, \( q \), let \( M^* = (\mu_1^*, \ldots, \mu_{|C|}^*) \in \mathcal{M} \) be the true relevance values of the documents.

- We model our knowledge of \( M^* \) with a Bayesian framework:
  \[
P(M|D) = \frac{P(D|M)P(M)}{P(D)}
  \]

- Assume \( P(M|D) \) is spherical multivariate normal:
  \[
P(M|D) = \mathcal{N}(\nu_1, \ldots, \nu_{|C|}; \sigma_1^2, \ldots, \sigma_{|C|}^2)
  \]
### (1) Representing Relevance

- Given a fixed query, \( P(M|D) \) maintains our knowledge about relevance as clicks are observed.

- This tells us which documents we are sure about, and which ones need more data.
(I) Representing Relevance

- Given a fixed query, $P(M|D)$ maintains our knowledge about relevance as clicks are observed.
- This tells us which documents we are sure about, and which ones need more data.
(1) Representing Relevance

• Given a fixed query, $P(M|D)$ maintains our knowledge about relevance as clicks are observed.

• This tells us which documents we are sure about, and which ones need more data.
(I) Representing Relevance

- Given a fixed query, $P(M|D)$ maintains our knowledge about relevance as clicks are observed.
- This tells us which documents we are sure about, and which ones need more data.
(2) Maintaining $P(M|D)$

- A standard approach to model noisy pairwise judgments is the Bradley-Terry model:

$$P(d_i \succ d_j) = \frac{\text{rel}(d_i)}{\text{rel}(d_i) + \text{rel}(d_j)}$$

[Bradley & Terry, ’52]

- Adding a Gaussian prior, we can apply an off-the-shelf algorithm to maintain $P(M|D)$: Glicko Rating System, commonly used for chess.

$$\nu_i \leftarrow \nu_i + \frac{q}{\sigma_i^2 + \frac{1}{\delta^2}} g(\sigma_j^2)(s_i - E[s|\nu_i, \nu_j, \sigma_j^2])$$

$$\sigma_i^2 \leftarrow \left(\frac{1}{\sigma_i^2} + \frac{1}{\delta^2}\right)^{-1}$$

[Glickman, ’99]
(3) Ranking, i.e. Inference

- We want to assign relevances $M = (\mu_1, \ldots, \mu_{|C|})$ such that $\mathcal{L}(M, M^*)$ is small, but $M^*$ is unknown.

- We want $\arg\min_M E_{M^* \sim P(M|D)}[\mathcal{L}(M, M^*)]$.

- If loss is pairwise decomposable, we can write:

$$\sum_{i=1}^{|C|} \sum_{j=i+1}^{|C|} E_{M^* \sim P(M|D)}[\mathcal{L}^{pair}(M, M^*, i, j)]$$

- Recall: $P(M|D) = \mathcal{N}(\nu_1, \ldots, \nu_{|C|}; \sigma_1^2, \ldots, \sigma_{|C|}^2)$

$\Rightarrow$ For misordered pairs & squared error the minimizer is the mode of $P(M|D)$. We’ll use that.
(4) Getting Useful Data

- We could just present the mode ranking. But then the data we get would always be about the documents already ranked highly.

TRADEOFF:
Show best ranking ⇔ Collect useful data

- Instead, we optimize the ranking shown to users.
- WLOG, we consider modifications that pick the top two documents to minimize future loss, then append the mode ranking.
Exploration Strategies

Expected Loss:

\[
\sum_{i=1}^{\left| C \right|} \sum_{j=i+1}^{\left| C \right|} E_{M^* \sim P(M|D)} \left[ L_{pair}(M, M^*, i, j) \right]
\]

Strategies:

- **Passive**: Present the mode ranking.
- **Random**: Pick top two randomly.
- **Largest Expected Loss**: Select pair with largest contribution to the loss.
- **One Step Lookahead**: Select pair with largest expectation reduction in \( L_{pair} \)

Radlinski & Joachims, *Active Exploration for Learning Rankings from Clickthrough Data*, KDD ’07
Loss Function

What loss function do we want to optimize for?

1. The loss for ranking a less relevant document above a more relevant document should be larger if the documents are presented higher.

2. The loss should be larger if error in relative relevance is larger.

\[
\mathcal{L}^{\text{pair}} = e^{-r_{ij}} \left( (\mu_i - \mu_j) - (\mu_i^* - \mu_j^*) \right)^2 \mathbf{1}_{\text{misordered}}
\]

\begin{align*}
(1) & & (2) & & \text{(hinge; 1)} \\
\end{align*}
Performance: Synthetic

- We simulated users with a simple user model, measuring the effect of parameter settings & noise.
- Performance of different strategies on synthetic data:
Performance: TREC data

- Gaussian prior over relevance no longer holds.
- Loss looks similar as for synthetic data.
- Mean Average Precision also improves quickly.
Alternative Loss Functions?

- $\mathcal{L}^{\text{pair}}$ is clearly related to MAP, but is it best?

<table>
<thead>
<tr>
<th>Loss Function</th>
<th>MAP Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>$e^{-r_{ij}} \left( (\mu^<em>_i - \mu^</em><em>j) - (\nu_i - \nu_j) \right)^2 \mathbf{1}</em>{\text{misordered}}$</td>
<td>$0.481 \pm 0.017$</td>
</tr>
<tr>
<td>$\left( (\mu^<em>_i - \mu^</em><em>j) - (\nu_i - \nu_j) \right)^2 \mathbf{1}</em>{\text{misordered}}$</td>
<td>$0.281 \pm 0.017$</td>
</tr>
<tr>
<td>$e^{-r_{ij}} \left( (\mu^<em>_i - \mu^</em>_j) - (\nu_i - \nu_j) \right)^2$</td>
<td>$0.287 \pm 0.012$</td>
</tr>
<tr>
<td>$e^{-r_{ij}} \mathbf{1}_{\text{misordered}}$</td>
<td>$0.337 \pm 0.020$</td>
</tr>
</tbody>
</table>

- Optimizing for the ordering performs much worse than optimizing for relevance estimates!
Conclusions

• We have formalized the ranked relevance elicitation problem.

• We have demonstrated that by using active exploration, the ranking improves quickly.

• We presented a number of exploration strategies.

• The method works even if the Gaussian assumptions are violated, and for noisy clicks.