terms, this means that given enough training data, the learning algorithm is guaranteed to find the best system in $\mathcal{S}$.

5. FEEDBACK PROPENSITY MODELS

In Section 4, we showed that the relevance signal r_i, the observation pattern o_i, and the propensities of the observations $Q(o_i(y) = 1|\boldsymbol{x}_i, \bar{\boldsymbol{y}}_i, \mathrm{r}_i)$ are the key components for unbiased LTR from biased observational feedback. We now outline how these quantities can be elicited and modeled in a typical search-engine application. However, the general framework of Section 4 extends beyond this particular application, and beyond the particular feedback model below.

5.1 Position-Based Propensity Model

Search engine click logs provide a sample of query instances $\boldsymbol{x}_i$, the presented ranking $\bar{\boldsymbol{y}}_i$ and a (sparse) click-vector where each $\mathrm{c}_i(y) \in \{0, 1\}$ indicates whether result y was clicked or not. To derive propensities of observed clicks, we will employ a click propensity model. For simplicity, we consider a straightforward examination model analogous to [17], where a click on a search result depends on the probability that a user examines a result (i.e., $\mathrm{e}_i(y)$) and then decides to click on it (i.e., $\mathrm{c}_i(y)$) in the following way:

$$P(\mathrm{e}_i(y) = 1|\,\mathrm{rank}(y|\bar{\boldsymbol{y}})) \cdot P(\mathrm{c}_i(y) = 1|\,\mathrm{r}_i(y), \mathrm{e}_i(y) = 1).$$

In this model, examination depends only on the rank of y in $\bar{\boldsymbol{y}}$. So, $P(\mathrm{e}_i(y) = 1|\,\mathrm{rank}(y|\bar{\boldsymbol{y}}_i))$ can be represented by a vector of examination probabilities p_r, one for each rank r. These examination probabilities can model presentation bias documented in eye-tracking studies [11], where users are more likely to see results at the top of the ranking than those further down.

For the probability of click on an examined result $P(\mathrm{c}_i(y) = 1|\,\mathrm{r}_i(y), \mathrm{e}_i(y) = 1)$, we first consider the simplest model where clicking is a deterministic noise-free function of the users private relevance assessment $\mathrm{r}_i(y)$. Under this model, users click if and only if the result is examined and relevant ($\mathrm{c}_i(y) = 1 \leftrightarrow [\mathrm{e}_i(y) = 1 \wedge \mathrm{r}_i(y) = 1]$). This means that for examined results (i.e., $\mathrm{e}_i(y) = 1$) clicking is synonymous with relevance ($\mathrm{e}_i(y) = 1 \rightarrow [\mathrm{c}_i(y) = \mathrm{r}_i(y)]$). Furthermore, it means that we observe the value of $\mathrm{r}_i(y)$ perfectly when $\mathrm{e}_i(y) = 1$ ($\mathrm{e}_i(y) = 1 \rightarrow o_i(y) = 1$), and that we gain no knowledge of the true $\mathrm{r}_i(y)$ when a result is not examined ($\mathrm{e}_i(y) = 0 \rightarrow o_i(y) = 0$). Therefore, examination equals observation and $Q(o_i(y)|\boldsymbol{x}_i, \bar{\boldsymbol{y}}_i, \mathrm{r}_i) \equiv P(\mathrm{e}_i(y)|\,\mathrm{rank}(y|\bar{\boldsymbol{y}}_i))$.

Using these equivalences, we can simplify the IPS estimator from (4) by substituting p_r as the propensities and by using $\mathrm{c}_i(y) = 1 \leftrightarrow [o_i(y) = 1 \wedge \mathrm{r}_i(y) = 1]$

$$\hat{R}_{IPS}(S) = \frac{1}{n}\sum_{i=1}^{n}\sum_{y:\mathrm{c}_i(y)=1}\frac{\mathrm{rank}(y|S(\boldsymbol{x}_i))}{p_{\mathrm{rank}(y|\bar{\boldsymbol{y}}_i)}}. \qquad (5)$$

$\hat{R}_{IPS}(S)$ is an unbiased estimate of $R(S)$ under the position-based propensity model if $p_r > 0$ for all ranks. While absence of a click does not imply that the result is not relevant (i.e., $\mathrm{c}_i(y) = 0 \not\rightarrow \mathrm{r}_i(y) = 0$), the IPS estimator has the nice property that such explicit negative judgments are not needed to compute an unbiased estimate of $R(S)$ for the loss in (2). Similarly, while absence of a click leaves us unsure about whether the result was examined (i.e., $\mathrm{e}_i(y) =?$), the IPS estimator only needs to know the indicators $o_i(y) = 1$ for results that are also relevant (i.e., clicked results).

Finally, note the conceptual difference in how we use this standard examination model compared to most prior work. We do not try to estimate an average relevance rating $\mathrm{rel}(\boldsymbol{x}, y)$ by taking repeat instances of the same query $\boldsymbol{x}$, but we use the model as a propensity estimator to de-bias individual observed user judgments $\mathrm{r}_i(y)$ to be used directly in ERM.

5.2 Incorporating Click Noise

In Section 5.1, we assumed that clicks reveal the user's true r_i in a noise-free way. This is clearly unrealistic. In addition to the stochasticity in the examination distribution $P(\mathrm{e}_i(y) = 1|\,\mathrm{rank}(y|\boldsymbol{y}))$, we now also consider noise in the distribution that generates the clicks. In particular, we no longer require that a relevant result is clicked with probability 1 and an irrelevant result is clicked with probability 0, but instead, for $1 \geq \epsilon_+ > \epsilon_- \geq 0$,

$$P(\mathrm{c}_i(y) = 1|\,\mathrm{r}_i(y) = 1, o_i(y) = 1) = \epsilon_+,$$
$$P(\mathrm{c}_i(y) = 1|\,\mathrm{r}_i(y) = 0, o_i(y) = 1) = \epsilon_-.$$

The first line means that users click on a relevant result only with probability ϵ_+, while the second line means that users may erroneously click on an irrelevant result with probability ϵ_-. An alternative and equivalent way of thinking about click noise is that users still click deterministically as in the previous section, but based on a noisily corrupted version $\tilde{\mathrm{r}}_i$ of r_i. This means that all reasoning regarding observation (examination) events o_i and their propensities p_r still holds, and that we still have that $\mathrm{c}_i(y) = 1 \rightarrow o_i(y) = 1$. What does change, though, is that we no longer observe the "correct" $\mathrm{r}_i(y)$ but instead get feedback according to the noise-corrupted version $\tilde{\mathrm{r}}_i(y)$. What happens to our learning process if we estimate risk using (5), but now with $\tilde{\mathrm{r}}_i$?

Fortunately, the noise does not affect ERM's ability to find the best ranking system given enough data. While using noisy clicks leads to biased empirical risk estimates w.r.t. the true r_i (i.e., $\mathbb{E}[\hat{R}_{IPS}(S)] \neq R(S)$), in expectation this bias is order preserving for $R(S)$ such that the risk minimizer remains the same.

$$\begin{aligned}
&\mathbb{E}[\hat{R}_{IPS}(S_1)] > \mathbb{E}[\hat{R}_{IPS}(S_2)] \\
\Leftrightarrow\ &\mathbb{E}_{\boldsymbol{x},\mathrm{r},\bar{\boldsymbol{y}}}\left[\mathbb{E}_o\mathbb{E}_{\mathrm{c}|o}\left[\sum_{y:\mathrm{c}(y)=1}\frac{\mathrm{rank}(y|S_1(\boldsymbol{x})) - \mathrm{rank}(y|S_2(\boldsymbol{x}))}{p_{\mathrm{rank}(y|\bar{\boldsymbol{y}})}}\right]\right] > 0 \\
\Leftrightarrow\ &\mathbb{E}_{\boldsymbol{x},\mathrm{r}}\left[\sum_y \mathrm{P}(\mathrm{c}(y) = 1|o(y) = 1, \mathrm{r}(y))\delta\,\mathrm{rank}(y|\boldsymbol{x})\right] > 0 \\
\Leftrightarrow\ &\mathbb{E}_{\boldsymbol{x},\mathrm{r}}\left[\sum_y \delta\,\mathrm{rank}(y|\boldsymbol{x}) \cdot (\epsilon_+\,\mathrm{r}(y) + \epsilon_-(1 - \mathrm{r}(y)))\right] > 0 \\
\Leftrightarrow\ &\mathbb{E}_{\boldsymbol{x},\mathrm{r}}\left[\sum_y \delta\,\mathrm{rank}(y|\boldsymbol{x}) \cdot ((\epsilon_+ - \epsilon_-)\,\mathrm{r}(y) + \epsilon_-)\right] > 0 \\
*\Leftrightarrow\ &\mathbb{E}_{\boldsymbol{x},\mathrm{r}}\left[\sum_y \delta\,\mathrm{rank}(y|\boldsymbol{x}) \cdot (\epsilon_+ - \epsilon_-)\,\mathrm{r}(y)\right] > 0 \\
\Leftrightarrow\ &\mathbb{E}_{\boldsymbol{x},\mathrm{r}}\left[\sum_y \delta\,\mathrm{rank}(y|\boldsymbol{x}) \cdot \mathrm{r}(y)\right] > 0 \\
\Leftrightarrow\ &R(S_1) > R(S_2),
\end{aligned}$$

where $\delta\,\mathrm{rank}(y|\boldsymbol{x})$ is short for $\mathrm{rank}(y|S_1(\boldsymbol{x})) - \mathrm{rank}(y|S_2(\boldsymbol{x}))$ and we use the fact that $\epsilon_- \sum_{y\in\bar{\boldsymbol{y}}} \delta\,\mathrm{rank}(y|\boldsymbol{x}) = 0$ in the step marked $*$. This implies that our propensity-weighted ERM

is a consistent approach for finding a ranking function with the best true $R(S)$,

$$\begin{aligned}\hat{S} &= \text{argmin}_{S\in\mathcal{S}}\left\{R(S)\right\} \\ &= \text{argmin}_{S\in\mathcal{S}}\left\{\mathbb{E}[\hat{R}_{IPS}(S)]\right\}, \end{aligned} \quad (6)$$

even when the objective is corrupted by click noise as specified above.

5.3 Propensity Estimation

As the last step of defining the click propensity model, we need to address the question of how to estimate its parameters (i.e. the vector of examination probabilities p_r) for a particular search engine. The following shows that we can get estimates using data from a simple intervention similar to [27], but without the strong negative impact of presenting uniformly random results to some users. This also relates to the Click@1 metric proposed by [3].

First, note that it suffices to estimate the p_r up to some positive multiplicative constant, since any such constant does not change how the IPS estimator (5) orders different systems. We therefore merely need to estimate how much p_r changes relative to p_k for some "landmark" rank k. This suggests the following experimental intervention for estimating p_r: before presenting the ranking to the user, swap the result at rank k with the result at rank r. If we denote with y' the results originally in rank k, our click model before and after the intervention indicates that

$$\begin{aligned} P(\mathrm{c}_i(y')=1|\text{no-swap}) &= p_k \cdot P(\mathrm{c}_i(y')=1|\,\mathrm{e}_i(y')=1) \\ P(\mathrm{c}_i(y')=1|\text{swap-k-and-r}) &= p_r \cdot P(\mathrm{c}_i(y')=1|\,\mathrm{e}_i(y')=1) \end{aligned}$$

where

$$\begin{aligned} &P(\mathrm{c}_i(y')=1|\,\mathrm{e}_i(y')=1) \\ &\quad= \sum_{v\in\{0,1\}} P(\mathrm{c}_i(y')=1|\,\mathrm{r}_i(y')=v, \mathrm{e}_i(y')=1)\cdot P(\mathrm{r}_i(y')=v) \end{aligned}$$

is constant regardless of the intervention. This means that the clickthrough rates $P(\mathrm{c}_i(y')=1|\text{swap-k-and-r})$, which we can estimate from the intervention data, are proportional to the parameters p_r for any r. By performing the swapping intervention between rank k and all other ranks r, we can estimate all the p_r parameters.

This swap-intervention experiment is of much lower impact than the uniform randomization proposed in [27] for a different propensity estimation problem, and careful consideration of which rank k to choose can further reduce impact of the swap experiment. From a practical perspective, it may also be unnecessary to separately estimate p_r for each rank. Instead, one may want to interpolate between estimates at well-chosen ranks and/or employ smoothing. Finally, note that the intervention only needs to be applied on a small subset of the data used for fitting the click propensity model, while the actual data used for training the ERM learning algorithm does not require any interventions.

5.4 Alternative Feedback Propensity Models

The click propensity model we define above is arguably one of the simplest models one can employ for propensity modeling in LTR, and there is broad scope for extensions.

First, one could extend the model by incorporating other biases, for example, trust bias [11] which affects perceived relevance of a result based on its position in the ranking. This can be captured by conditioning the click probabilities also on the position $\mathrm{P}(\mathrm{c}_i(y')=1|\,\mathrm{r}_i(y'),\mathrm{e}_i(y')=1,\text{rank}(y|\bar{\boldsymbol{y}}_i))$. We have already explored that the model can be extended to include trust bias, but it is omitted due to space constraints. Furthermore, it is possible to model saliency biases [30] by replacing the p_r with a regression function.

Second, we conjecture that a wide range of other click models (e.g., cascade model [5] and others [5, 3, 1, 4]) can be adapted as propensity models. The main requirement is that we can compute marginal click probabilities for the clicked documents in hindsight, which is computationally feasible for many of the existing models.

Third, we may be able to define and train new types of click models. In particular, for our propensity ERM approach we only need the propensities $Q(o_i(y)=1|\boldsymbol{x}_i,\bar{\boldsymbol{y}}_i,\mathrm{r}_i)$ for observed and relevant documents to evaluate the IPS estimator, but not for irrelevant documents. This can be substantially easier than a full generative model of how people reveal relevance judgments through implicit feedback. In particular, this model can condition on all the revealed relevances $\mathrm{r}_i(y_j)$ in hindsight, and it does not need to treat them as latent variables.

Finally, the ERM learning approach is not limited to binary click feedback, but applies to a large range of feedback settings. For example, the feedback may be explicit star ratings in a movie recommendation system, and the propensities may be the results of self-selection by the users as in [20]. In such an explicit feedback setting, o_i is fully known, which simplifies propensity estimation substantially.

6. PROPENSITY-WEIGHTED SVM-RANK

We now derive a concrete learning method that implements propensity-weighted LTR. It is based on SVM-Rank [9, 10], but we conjecture that propensity-weighted versions of other LTR methods can be derived as well.

Consider a dataset of n examples of the following form. For each query-result pair $(\boldsymbol{x}_j, y_j)$ that is clicked, we compute the propensity $q_i = Q(o_i(y)=1|\boldsymbol{x}_i,\bar{\boldsymbol{y}}_i,\mathrm{r}_i)$ of the click according to our click propensity model. We also record the candidate set Y_j of all results for query $\boldsymbol{x}_j$. Typically, Y_j contains a few hundred documents – selected by a stage-one ranker [26] – that we aim to rerank. Note that each click generates a separate training example, even if multiple clicks occur for the same query.

Given this propensity-scored click data, we define Propensity SVM-Rank as a generalization of conventional SVM-Rank. Propensity SVM-Rank learns a linear scoring function $f(\boldsymbol{x},y)=w\cdot\phi(\boldsymbol{x},y)$ that can be used for ranking results, where w is a weight vector and $\phi(\boldsymbol{x},y)$ is a feature vector that describes the match between query $\boldsymbol{x}$ and result y.

Propensity SVM-Rank optimizes the following objective,

$$\begin{aligned} \hat{w} &= \text{argmin}_{w,\xi}\ \frac{1}{2}w\cdot w + \frac{C}{n}\sum_{j=1}^{n}\frac{1}{q_j}\sum_{y\in Y_j}\xi_{jy} \\ s.t. \quad & \forall y\in Y_1\backslash\{y_1\}: w\cdot[\phi(\boldsymbol{x}_1,y_1)-\phi(\boldsymbol{x}_1,y)]\geq 1-\xi_{1y} \\ & \quad\vdots \\ & \forall y\in Y_n\backslash\{y_n\}: w\cdot[\phi(\boldsymbol{x}_n,y_n)-\phi(\boldsymbol{x}_n,y)]\geq 1-\xi_{ny} \\ & \forall j\forall y: \xi_{jy}\geq 0. \end{aligned}$$

C is a regularization parameter that is typically selected via cross-validation. The training objective optimizes an

upper bound on the regularized IPS estimated empirical risk of (5), since each line of constraints corresponds to the rank of a relevant document (minus 1). In particular, for any feasible (w, ξ)

$$\begin{aligned}\text{rank}(y_i|\boldsymbol{y}) - 1 &= \sum_{y \neq y_i} \mathbb{1}_{w \cdot [\phi(\boldsymbol{x}_i, y) - \phi(\boldsymbol{x}_i, y_i)] > 0} \\ &\leq \sum_{y \neq y_i} \max(1 - w \cdot [\phi(\boldsymbol{x}_i, y_i) - \phi(\boldsymbol{x}_i, y)], 0) \\ &\leq \sum_{y \neq y_i} \xi_{iy}.\end{aligned}$$

We can solve this type of Quadratic Program efficiently via a one-slack formulation [10], and we are using SVM-Rank[1] with appropriate modifications to include IPS weights $1/q_j$. The resulting code will be available online.

In the empirical evaluation, we compare against the naive application of SVM-Rank, which minimizes the rank of the clicked documents while ignoring presentation bias. In particular, Naive SVM-Rank sets all the q_i uniformly to the same constant (e.g., 1).

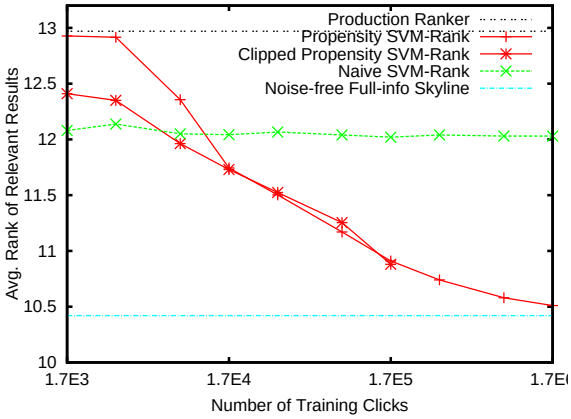

Figure 1: Test set performance in terms of (2) for Propensity SVM-Rank with and without clipping compared to SVM-Rank naively ignoring the bias in clicks ($\eta = 1$, $\epsilon_- = 0.1$). The skyline is a Ranking SVM trained on all data without noise in the full-information setting, and the baseline is the production ranker S_0.

7. EMPIRICAL EVALUATION

We take a two-pronged approach to evaluating our approach empirically. First, we use synthetically generated click data to explore the behavior of our methods over the whole spectrum of presentation bias severity, click noise, and propensity misspecification. Second, we explore the real-world applicability of our approach by evaluating on an operational search engine using real click-logs from live traffic.

7.1 Synthetic Data Experiments

To be able to explore the full spectrum of biases and noise, we conducted experiments using click data derived from the Yahoo Learning to Rank Challenge corpus (set 1). This corpus contains a large number of manually judged queries, where we binarized relevance by assigning $\mathrm{r}_i(y) = 1$ to all documents that got rated 3 or 4, and $\mathrm{r}_i(y) = 0$ for ratings $0, 1, 2$. We adopt the train, validation, test splits in the corpus. This means that queries in the three sets are disjoint, and we never train on any data from queries in the test set. To have a gold standard for reporting test-set performance, we measure performance on the binarized full-information ratings using (2).

To generate click data from this full-information dataset of ratings, we first trained a normal Ranking SVM using 1 percent of the full-information training data to get a ranking function S_0. We employ S_0 as the "Production Ranker", and it is used to "present" rankings $\bar{\boldsymbol{y}}$ when generating the click data. We generate clicks using the rankings $\bar{\boldsymbol{y}}$ and ground-truth binarized relevances from the Yahoo dataset according to the following process. Depending on whether we are generating a training or a validation sample of click data, we first randomly draw a query $\boldsymbol{x}$ from the respective full-information dataset. For this query we compute $\bar{\boldsymbol{y}} = S_0(\boldsymbol{x})$ and generate clicks based on the model from Section 5. Whenever a click is generated, we record a training example with its associated propensity $Q(o(y) = 1|\boldsymbol{x}, \bar{\boldsymbol{y}}, \mathrm{r})$. For the experiments, we model presentation bias via

$$Q(o(y) = 1|\boldsymbol{x}, \bar{\boldsymbol{y}}, \mathrm{r}) = p_{rank(y|\bar{\boldsymbol{y}})} = \left(\frac{1}{rank(y|\bar{\boldsymbol{y}})}\right)^{\eta}. \quad (7)$$

The parameter η lets us control the severity of the presentation bias. We also introduce noise into the clicks according to the model described in Section 5. When not mentioned otherwise, we use the parameters $\eta = 1$, $\epsilon_- = 0.1$, and $\epsilon_+ = 1$, which leads to click data where about 33% of the clicks are noisy clicks on irrelevant results and where the result at rank 10 has a 10% probability of being examined. We also explore other bias profiles and noise levels in the following experiments.

In all experiments, we select any parameters (e.g., C) of the learning methods via cross-validation on a validation set. The validation set is generated using the same click model as the training set, but using the queries in the validation-set portion of the Yahoo dataset. For Propensity SVM-Rank, we always use the (unclipped) IPS estimator (5) to estimate validation set performance. Keeping with the proportions of the original Yahoo data, the validation set size is always about 15% the size of the training set.

The primary baseline we compare against is a naive application of SVM-Rank that simply ignores the bias in the click data. We call this method *Naive SVM-Rank*. It is equivalent to a standard ranking SVM [9], but is most easily explained as equivalent to Propensity SVM-Rank with all q_j set to 1. Analogously, we use the corresponding naive version of (5) with propensities set to 1 to estimate validation set performance for Naive SVM-Rank.

7.2 How does ranking performance scale with training set size?

We first explore how the test-set ranking performance changes as the learning algorithm is given more and more click data. The resulting learning curves are given in Figure 1, and the performance of S_0 is given as a baseline. The click data has presentation bias according to (2) with $\eta = 1$

[1]https://www.joachims.org/svm_light/svm_rank.html