Offline Comparative Evaluation with Incremental, Minimally-Invasive Online Feedback

Ben Carterette*
Department of Computer and Information Sciences
Newark, DE, USA
carteret@udel.edu

Praveen Chandar
Spotify
New York City, NY, USA
praveenr@spotify.com

ABSTRACT

We investigate the use of logged user interaction data—queries and clicks—for offline evaluation of new search systems in the context of counterfactual analysis. The challenge of evaluating a new ranker against log data collected from a static production ranker is that new rankers may retrieve documents that have never been seen in the logs before, and thus lack any logged feedback from users. Additionally, the ranker itself could bias user actions such that even documents that have been seen in the logs would have exhibited different interaction patterns had they been retrieved and ranked by the new ranker. We present a methodology for incrementally logging interactions on previously-unseen documents for use in computation of an unbiased estimator of a new ranker's effectiveness. Our method is very lightly invasive with respect to the production ranker results to insure against users becoming dissatisfied if the new ranker is poor. We demonstrate how well our methods work in a simulation environment designed to be challenging for such methods to argue that they are likely to work in a wide variety of scenarios.

KEYWORDS

Measurement, Performance, Experimentation

ACM Reference Format:
Ben Carterette and Praveen Chandar. 2018. Offline Comparative Evaluation with Incremental, Minimally-Invasive Online Feedback. In *SIGIR '18: The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, July 8–12, 2018, Ann Arbor, MI, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3209978.3210050

1 INTRODUCTION

Implicit feedback such as user clicks has long been an important signal for measuring the effectiveness of search engines. Such signals have advantages over more traditional explicit feedback such as relevance judgments in that they are available at low cost for engines running at scale; they are collected in a natural setting; they reflect a more personal notion of relevance than typically used in relevance judging. Thus they have become widely-used metrics for online A/B testing and interleaving experiments.

However, there are problems with purely online evaluation. There is the risk of user attrition if results being deployed are worse than users expect. In addition, the time it takes to get sufficient feedback may be prohibitive—in large-scale web search engines it may not be a problem, but in settings where traffic is much lower or in which there are daily or weekly periodicity effects, it could take several weeks to obtain enough of the implicit feedback to be confident in a decision. Finally, users' behavior may be biased in unpredictable and hard-to-control ways.

Thus there has been recent interest in using historical log data for offline learning and evaluation. This has its own challenges. In particular, due to strong biases for users to click top-ranked results and for the engine itself to influence user clicks, a click can rarely be interpreted sans context. Furthermore, when new approaches to ranking result in entirely new documents being retrieved, historical log data has limited use in evaluation.

Some of the most recent work on this topic has investigated it from the perspective of *counterfactual analysis*, which asks the question "what would users have done had they seen this alternative ranking?" For example, one recent work by Joachims et al. [15] uses inverse propensity weighting to train rankers from historical log data. The method requires some degree of randomization of results to remove position bias; while this randomization carries the same risk of user attrition mentioned above, the method Joachims et al. present is meant to minimize the amount of perturbation necessary so that most users will see most results as intended.

In this paper we build on Joachims' et al.'s work to extend it to evaluating new rankers that can retrieve never-before-seen documents. Since these rankers may retrieve more (or fewer) relevant documents than the production ranker, we must assume that they are capable of biasing user behavior just as display position is. And since they may retrieve entirely new documents, we need a way to collect some incremental feedback on those documents without perturbing the production ranker results too much so that we are able to accurately compare rankers' effectiveness.

Our proposed methodology requires a set of alternate rankers along with a production ranker to curate a reusable offline dataset. The curated dataset consists of a set of logs with user feedback along with propensity estimates to debias the logs. In spirit, the idea is similar to how TREC creates a resuable test collection using a pool of rankers [30]. The primary contribution of this work are these methods to curate an offline dataset, which allow comparisons of new rankers to each other and to the production ranker.

As a secondary contribution, we introduce a full simulation environment for rigorous testing of offline evaluation methods that

*Current Affiliation: Spotify

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
SIGIR '18, July 8–12, 2018, Ann Arbor, MI, USA
© 2018 Association for Computing Machinery.
ACM ISBN 978-1-4503-5657-2/18/07…$15.00
https://doi.org/10.1145/3209978.3210050

use historical log data. Our simulations allow the generation of runs of varying effectiveness and user models that can produce very noisy click data—as real users do. We demonstrate the effectiveness of our methods even in this adversarial simulation environment.

This paper is organized as follows. In Section 2 we discuss related work on online evaluation and counterfactual analysis. In Section 3 we describe our propensity weight-based method. In Section 4 we present our simulation models, and Section 5 presents experimental results and analysis. We conclude in Section 6.

2 RELATED WORK

In this section, we discuss prior work on evaluating retrieval systems online and offline using implicit feedback. We start by discussing the use of clicks for IR evaluation, then briefly describe the challenges in using clicks for evaluation and efforts to overcome them. Finally, we discuss the relation of our work to counterfactual inference and *off-policy* evaluation in reinforcement learning.

2.1 Online evaluation using implicit feedback

Implicit feedback such as clicks, dwell times, etc. can be obtained at a low cost when users are interacting with the search system in a natural environment, thus making them a valuable resource to be used as proxy for relevance. In recent years, substantial effort has gone into leveraging clicks for the evaluation and optimization of retrieval systems. Joachims et al. [13] were among the first to propose a methodology to incorporate click data in learning to rank algorithms. Early efforts in using clicks for evaluation include models to combine clicks along with editorial judgments to predict the performance of new retrieval systems [1, 3, 23]. A/B testing has also long been in use in online web search and recommendation [17].

While implicit feedback has several advantages, incorporating clicks in practice is challenging, as pointed out by a landmark study conducted by Joachims et al. [14]. They investigated the accuracy of clicks through a user study and concluded that clicks can be valuable although biased and care must be taken to handle certain biases. Studying bias in click data has received considerable attention with researchers pointing out different types of biases including presentation bias [14], attractiveness bias [33], and trust bias [16]. Along with different biases, missing judgments is a major problem in using historical click data for offline evaluation of new systems [31].

2.2 Handling bias in implicit feedback

A commonly used approach to handle bias in click data is to model user's behavior when interacting with a ranked list (i.e. predict the likelihood of clicking on a document given a ranked list). These models can also be used to minimize the effect of the bias in click data. Early work on click modeling includes work by Richardson et al. [28] and Craswell et al. [8]. Richardson et al. primarily used two features—document position and query-document relevance—to propose a position bias model, whereas Craswell et al. [8]'s model assumes a user who scans the ranked list from the top to bottom and clicks on a relevant document when encountered. Since then several click models have been proposed, most of them based on probabilistic graphical models [10, 5] (also see Chuklin et al. [6] for a complete overview).

Randomization is another technique used to handle bias in click data. Radlinski et al. [26] introduced a minimally invasive technique to infer unbiased preference by randomly swapping documents at different ranks. Similarly, interleaving approaches that merge documents originating from different rankers were proposed by Chapelle et al. [4] . The interleaving methods consists of two step: (1) merging two or more rankers before presenting to the user; (2) interpreting interactions on the interleaved results. Several approaches to merge results from different rankers and corresponding methods to estimate system performance has been proposed in the literature, including *team-draft interleaving* [27], probabilistic interleaving [12], etc. (see [11] for a complete overview). The main difference between our work and the interleaving techniques is that our randomization is less invasive; the chances of a bad ranker affecting user experience is lower in our case compared to interleaving. Also, the primary goal of interleaving methods is to reliably detect preferences between different rankers in an online setting whereas our goal is to de-bias the production logs in order to use them in an offline setting.

2.3 Counterfactual evaluation

The problem of evaluating retrieval system using historical click logs is similar to *off-policy* evaluation in reinforcement learning. In this setting, the goal is to evaluate new retrieval systems (often referred to as *target policy*) that retrieve a ranked list different from a production system $\mathcal{S}_0$ using the production logs. In other words, the value of the *target policy* $\mathcal{S}_1$ must be estimated given a set of logs consisting of a query, ranked list and a reward $\langle q, S, r \rangle$. Common methods used to solve this estimation problem include directly estimating the reward r given set of logs (click modeling) [6], de-biasing the logs using randomization and Inverse Propensity Weighting [19, 2], and a hybrid method that combines both (often referred to as *doubly robust estimation* [9]).

More recently, Joachims et al. [15] proposed an approach based on counterfactual inference to deal with position bias in production logs. They computed propensity scores for each clicked rank position in the logs and use the propensities to learn a model that is robust to biased feedback. A more invasive randomization scheme was proposed by Wang et al. [31] in the context of personal search. All of these approaches are based on Inverse Propensity Weighting that was originally developed in the field of causal inference [29]. Our paper builds on these works to estimate propensities using randomization; however, unlike prior work, our approach can be used to evaluate new systems that could rank document unseen before in the production logs.

2.4 Relation to offline evaluation

In our work, the way we prioritize new document for feedback is similar in spirit to the pooling techniques and low cost evaluation methods proposed in offline IR evaluation literature. Pooling techniques propose a way to carefully choose a subset of documents to be judged by a human assessors and broadly they fall into two categories: static-sampling based [25, 24, 32] and active-sampling pooling [18, 20, 21, 7]. Our methods are similar to static methods; however, in our case there are no human assessor and we rely on click information that are noisy and incomplete.

3 INVERSE PROPENSITY SCORES

As mentioned above, our work is based on inverse propensity weighting, in particular the method introduced by Joachims et al. [15] that uses click feedback on individual documents and estimates propensities from per-rank marginal clickthrough rates. We adopt the following notation:

- $\mathcal{L}$ is a log;
- $\ell \in \mathcal{L}$ is a line in the log consisting of the tuple $\langle q_\ell, S_\ell, s_\ell \rangle$;
  - q_ℓ is a query or more generally a *context* consisting of a query and user profile;
  - S_ℓ is a vector of document IDs (the ranked list);
  - s_ℓ is a vector of observed rewards (clicks on documents in S_ℓ);
- $\text{rank}(d|S, q)$ or $\text{rank}(d|\mathcal{S}, q)$ is the rank of document d in the ranked list S (produced by ranker $\mathcal{S}$) for context q;
- $f(\cdot)$ is a function that will be applied to ranks (see below);
- p_r is the propensity of users to click at rank r, that is, the unbiased marginal click-through rate on rank r.

The effectiveness of a ranker $\mathcal{S}$ can be estimated using inverse propensity scoring (IPS):

$$\hat{V}_{IPS}(\mathcal{S}) = \frac{1}{|\mathcal{L}|} \sum_{\ell \in \mathcal{L}} \sum_{d \in S_\ell} \frac{f(\text{rank}(d|\mathcal{S}, q_\ell)) \cdot s_d}{p_{\text{rank}(d|S_\ell, q_\ell)}}$$

The only difference from Joachims et al., other than notation, is that they used $\text{rank}(d|\mathcal{S}, q)$ in the numerator directly, whereas we are applying a function to the rank. The function represents the contribution of the rank of the document to a (linear and additive) IR effectiveness measure such as precision or DCG. Assuming clicks are positively correlated with relevance, it is straightforward to show that the expected value of $\hat{V}_{IPS}(\mathcal{S})$ is proportional to the effectiveness measure that $f(\text{rank})$ represents; the proof directly follows that of Joachims et al. so we omit it.

3.1 Evaluating a new ranker

For the sake of simplicity, suppose there is a static "production ranker" $\mathcal{S}_0$ that has been in use for some time and has generated a large log. With that log and proper propensity estimation methods (discussed in more detail below), $\hat{V}_{IPS}(\mathcal{S}_0)$ can be used to compute an unbiased estimator of $\sum f(\text{rank})$ for *any* re-ranking of the documents that $\mathcal{S}_0$ ranks for the contexts represented in the log.

But consider a new ranker $\mathcal{S}_1$ that could potentially rank new documents for which we have no previous historical interaction data. In this case the estimator above would be biased; even assuming propensities do not change with the ranker, the new ranker can be evaluated using only the documents it retrieved in common with the production ranker and therefore its effectiveness will be relatively underestimated.

As well, there is no reason to assume that propensities would not change with the ranker. In the formulation above, the propensities are marginal clickthrough rates over all contexts. Since our question is whether we should replace the production ranker with a newly-developed ranker, and the counterfactual is what users *would have done* had we taken that action, we must consider propensities as a function of system effectiveness as well as rank position.

To address the latter problem, we define propensity as $p_{\mathcal{S},r}$: the propensity to click on documents retrieved by ranker $\mathcal{S}$ at rank r, and re-write the estimator as:

$$\hat{V}_{IPS}(\mathcal{S}) = \frac{1}{|\mathcal{L}|} \sum_{\ell \in \mathcal{L}} \sum_{d \in S_\ell} \frac{f(\text{rank}(d|\mathcal{S}, q_\ell)) \cdot s_d}{p_{\mathcal{S},\text{rank}(d|S_\ell, q_\ell)}}$$

Then to estimate these propensities, we introduce a new logging policy described below. When implemented, this new policy will also provide logged data that we can use to compute an unbiased estimator of the new ranker.

3.2 Estimating propensities

Estimating propensities requires unbiased data. They can not generally be estimated from raw search logs, as there are at least two major sources of bias present:

(1) position bias, which means that users are much more likely to click on documents near the top of the ranking, regardless of relevance, than documents ranked lower;
(2) system effectiveness bias, which means clickthrough rates may be biased by how effective the ranker is at retrieving relevant documents.

(Note that system effectiveness bias does not mean that better rankers result in higher clickthrough rates; better rankers could actually lead to lower clickthrough rates as users are more likely to be satisfied by a single result.)

In order to unbias results, we follow Joachims et al. in introducing some small amount of randomization into search results. The way this randomization is done is through a *logging policy*.

3.2.1 Logging policies. We present two different policies for randomizing results for logging. The first directly follows Joachims et al.: we select one rank as an "anchor" and then randomly select another rank with which to swap documents. This ensures that every document retrieved will be shown at the anchor rank the same number of times (in expectation), allowing us to compute the unbiased clickthrough rate for that ranker and that rank position. Using that data we can compute unbiased clickthrough rates for any rank position. We call this the "swap policy" and apply it to the production ranker results such that a small percentage of overall traffic sees results randomized in this way. Traffic selected for the swap policy is logged in a separate "swap log".

Specifically, the propensity at the anchor rank is computed from the swap log as:

$$p_{\mathcal{S}_0,k} = \frac{\text{total clicks on all documents at } k}{|\mathcal{L}|}$$

At each rank $i \neq k$, there are two possible documents that can appear: the one originally ranked at position i, and the one originally ranked at position k. Measuring the clickthrough rate on these two documents at both ranks i and k (weighting by the relative number of times they appeared at those ranks), then taking the ratio of those two quantities, gives an estimate of the relative decrease in clickthrough rate from rank i to k is the propensity for rank i.

The second policy, which is our contribution, is the "insertion policy", used to collect feedback on previously-unseen documents. Similar to the swap policy, we select an anchor rank; this is the only position in the ranked list that will be changed. We then select a document unranked by the production ranker but ranked by one of the new rankers we want to evaluate to place in that

position, replacing the document at the anchor rank with that one. We compute unbiased clickthrough rate at this position using the weighted sum of logged clicks on documents ranked by the new ranker (weighted by how often they appeared, as those that it has in common with the production ranker will have been seen more than its new documents). We use this to similarly compute unbiased clickthrough rates for all other positions for the same ranker. A small percentage of overall traffic will see results randomized this way, and that traffic logged in a separate "insertion log".

Since precise estimates may require a great deal of log data, we smooth them using simple Laplacian plus-one smoothing. This acts as a regularization that overestimates propensity at low ranks.

3.3 Using logged data to estimate effectiveness

After the swap and insertion policies have been active for some time, we will have three separate logs:

(1) the production ranker logs, which we treat as static such that the ranked results for a given context are identical every time that context appears;
(2) the swap log, in which each line has at most one pair of documents swapped: the one at the anchor rank with one at another randomly-selected rank;
(3) the insertion log, in which each line has at most one new document at the anchor rank, selected randomly from those retrieved by a new ranker.

Let $\mathcal{L}, \mathcal{L}_S, \mathcal{L}_I$ denote these three logs (respectively). For the purposes of computing the IPS estimator for the production ranker $\mathcal{S}_0$, $\mathcal{L}$ and $\mathcal{L}_S$ can be combined, as they rank the same set of documents for each context. The insertion log $\mathcal{L}_I$ contains new documents not previously seen in $\mathcal{S}_0$ ranked results.

The production ranker, or any new ranker that simply reranks documents ranked by the production ranker $\mathcal{S}_0'$, can be evaluated using the IPS estimator:

$$\hat{V}_{IPS}(\mathcal{S}_0) = \frac{1}{|\mathcal{L} \cup \mathcal{L}_S|} \sum_{\ell \in \mathcal{L} \cup \mathcal{L}_S} \sum_{d \in S_\ell} \frac{f(\text{rank}(d|\mathcal{S}_0, q_\ell)) \cdot s_d}{p_{\mathcal{S}_0, \text{rank}(d|S_\ell, q_\ell)}}$$

A new ranker $\mathcal{S}_1$ that can take new actions (rank new documents) is evaluated by using the logs $\mathcal{L}$ and $\mathcal{L}_S$ to estimate the contribution of all documents that it ranked in common with the production ranker, then adding to that an estimator based on $\mathcal{L}_I$ to include the contribution of the new documents it ranked.

$$\hat{V}_{IPS}(\mathcal{S}_1) = \frac{1}{|\mathcal{L} \cup \mathcal{L}_S|} \sum_{\ell \in \mathcal{L} \cup \mathcal{L}_S} \sum_{d \in S_\ell} \frac{f(\text{rank}(d|\mathcal{S}_1, q_\ell)) \cdot s_d}{p_{\mathcal{S}_1, \text{rank}(d|S_\ell, q_\ell)}} + \frac{1}{|\mathcal{L}_I|} \sum_{\ell \in \mathcal{L}_I} \sum_{d \in S_\ell} \frac{f(\text{rank}(d|\mathcal{S}_1, q_\ell)) \cdot s_d}{p_{\mathcal{S}_1, \text{rank}(d|S_\ell, q_\ell)} \cdot \pi_{d, q_\ell}}$$

Note that $\text{rank}(d|\mathcal{S}_1, q_\ell)$ requires logic for what to do when document d on line ℓ of the log has not been ranked by $\mathcal{S}_1$. We define it to be 0 in these cases.

Note also that we have introduced a new parameter: π_{d,q_ℓ} is the *inclusion probability* of document d ranked by $\mathcal{S}_1$ for query q_ℓ but not by $\mathcal{S}_0$ for the same query. Since such documents can only appear at the anchor rank, any given line of the log can only have one new document. If we do not re-weight documents to account for this, these documents will have an outsized effect on the estimator. Thus π_{d,q_ℓ} is the probability that document d is selected for insertion from all the new documents ranked for q_ℓ. (For documents at ranks other than the anchor rank, the inclusion probability would be 1.)

This raises another question: how do we assign π_{d,q_ℓ}? We address that in Section 3.5 below; before that, we describe the $f()$ function in more detail.

3.4 Reward

Joachims et al. reward a click on a document by the rank at which that document occurs, so that the expected value of the estimator is the sum of the ranks of relevant documents. We can generalize by applying a function to the rank, similar to the way IR effectiveness measures are based on functions of the ranks at which relevant documents appear.

In particular, if we define $f(\text{rank})$ as:

$$f_{\text{P@K}}(\text{rank}) = \tfrac{1}{K} \text{ if rank} \leq K\text{; 0 otherwise}$$

then the expected value of the estimator will be proportional to precision at K. Likewise, if we define $f(\text{rank})$ as:

$$f_{\text{DCG}}(\text{rank}) = \tfrac{1}{\log_2(\text{rank}+1)} \text{ if rank} \leq K\text{; 0 otherwise}$$

then its expected value is proportional to DCG@K. (Note that it is hard to capture nDCG: since we have no explicit relevance judgments we cannot compute the ideal DCG and therefore cannot normalize DCG values. We may be able to estimate the ideal DCG using clicks from the swap and insertion logs but we leave this for future work.)

3.5 Sampling documents

Finally we turn to the inclusion probabilities π_{d,q_ℓ} introduced in Section 3.3. We consider two ways to estimate these:

Uniform Sampling — just selects a document to insert uniformly at random from those in the set of documents retrieved by new rankers and not by the production ranker for each query q_ℓ.

Informative Sampling — works by assigning sampling probabilities to documents according to their effect on the difference in effectiveness between two rankers. Consider two rankers $\mathcal{S}_0$ and $\mathcal{S}_1$ that we are evaluating by DCG@10. We are interested in the comparative evaluation: which ranker is more effective? In other words, is the difference in DCG greater than zero, or less?

$$\sum_{k=1}^{10} f_{\text{DCG}}(k) \cdot rel_{d_{S_0,k}} - \sum_{k=1}^{10} f_{\text{DCG}}(k) \cdot rel_{d_{S_1,k}} > 0$$

From this formulation we can see that a document which has been ranked at the same position in both rankers has no effect on the difference in DCG. The effect of a document ranked at position k_0 in $\mathcal{S}_0$ and k_1 in $\mathcal{S}_1$ is $f_{\text{DCG}}(k_0) - f_{\text{DCG}}(k_1)$.

We compute this quantity for every document, for every pair of runs, and take the maximum value for each document.

$$w_d = \max_{S_i, S_j} |f_{\text{DCG}}(k_i) - f_{\text{DCG}}(k_j)|$$

Then the sampling probability for d is $\frac{w_d}{\sum_d w_d}$.

4 SIMULATION MODELS

We design our experiments around simulation of queries, rankers, and user interactions. This allows us to control for more potential confounding factors in a more precise way. Of course, simulation models and parameters could be confounding factors themselves. We intend to be clear about our decisions and implementations so that even if the reader disagrees, they are able to re-do our experiments with their own choices.

4.1 Simulating queries

The primary characteristic of queries that concerns us is the relevance of documents retrieved for the query. This is very simple to simulate; we just pick a random number of documents that rankers could retrieve—essentially a simulated pool—then from that set pick a random subset to be relevant. We generate pools of size 10 to 100 (chosen uniformly at random) and use 0.25 as a parameter for assigning relevance to documents in the pool.

4.2 Simulating rankers

When simulating rankers, we want to make sure we are generating some that are similar in effectiveness and some that are further apart. To simulate ranker $\mathcal{S}_j$, we first select a parameter η_j uniformly at random from the set $\{2^0, 2^1, 2^2, 2^3, 2^4\}$. This parameter will directly influence the ranker's mean effectiveness. For each query q in the set of simulated queries described above, we sample a value $\eta_{j,q}$ from a normal distribution centered on η_j with variance proportional to $\sqrt{\eta_j}$. This parameter will influence the ranker's effectiveness on this query and ensure that no two rankers are exactly identical.

We then choose a value K, the max rank to which the ranker will be evaluated. For each rank $k \in [1, K]$, we sample a relevance value from a multinomial distribution with probabilities proportional to $\{0 + \eta_{j,q}, 1 + \eta_{j,q}, 2 + \eta_{j,q}, \ldots\}$. Thus the larger $\eta_{j,q}$ is, the "flatter" the multinomial distribution is, creating a greater chance to sample a lower relevance grade. In other words, η_j is inversely correlated with ranker effectiveness. After sampling a relevance grade, we sample (uniformly without replacement) a document with that grade to place at rank k. The empirical mean precisions at 10 of rankers generated this way are $0.60, 0.57, 0.53, 0.50, 0.49$ for the five respective η values.

With this process, any two runs i, j with $\eta_i \neq \eta_j$ will show a statistically significant difference in effectiveness over enough queries. Because of simulated noise on individual queries, two runs with $\eta_i = \eta_j$ are not likely to have identical mean effectiveness over a query set, but should not be found statistically significantly different (this would be a Type I error). Two rankers with "adjacent" η parameters (that is, $|\log_2 \eta_i - \log_2 \eta_j| = 1$) will have enough variance that they may not be easily distinguishable. Thus we can analyze results on pairs of rankers that are more different versus pairs that are more similar in effectiveness.

Table 1 shows empirical statistical significance rates (by a paired, two-tailed t-test at the 0.05 level) for pairs of runs with all combinations of η parameters. As expected, the rate of statistical significance when ηs are equal is within the Type I error rate of 0.05. Adjacent η values give higher significance rates, and distant η values give rates of 1.

η	2^0	2^1	2^2	2^3	2^4
2^0	0.00	0.99	1.00	1.00	1.00
2^1	–	0.05	0.81	1.00	1.00
2^2	–	–	0.02	0.67	1.00
2^3	–	–	–	0.04	0.34
2^4	–	–	–	–	0.04

Table 1: Rate of detecting statistical significance in pairs of rankers generated with η parameters given on the column and row.

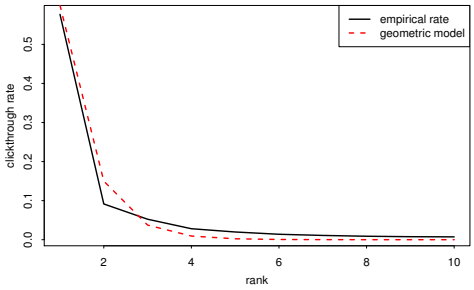

Figure 1: Observed clickthrough rates in the Yahoo Webscope log compared to a geometric model of user examination and clicks.

4.3 Simulating users

Our user model is based on the rank-biased precision model introduced by Moffat et al. [22]. We simulate users proceeding down a ranked list position-by-position, inspecting the result at each position, deciding whether to click it or not, and then deciding whether to continue to the next result or abandon the current query.

The probability of seeing a result is geometric in its rank:

$$P(\text{examine result at rank } k|\theta) = \theta^{k-1} \cdot (1 - \theta)$$

We model the decision of a user to click or not as binomial conditional on the relevance of the result: $P(\text{click}|R \geq 1)$ versus $P(\text{click}|R = 0)$.

Then we have:

$$P(\text{click at rank } k|\theta, R_{d_k}) = P(click|R_{d_k}) \cdot \theta^{k-1}$$

In this work we set $\theta = 0.25$, $P(click|R_{d_k} \geq 1) = 0.4$ and $P(click|R_{d_k} = 0) = 0.2$. Note that this is significantly more noise than previous papers on this topic have used in simulations. Our geometric model with high probability of abandonment means that very few users will see results below rank 5, and our low click probabilities mean that even fewer will click those results. We chose to use the geometric model due to its simplicity and ability to reflect real user behavior (see below). Alternatively, we could have used any of the user models proposed in [6].

4.3.1 User model validation. We use the Yahoo Webscope L18 search log data[1], which consists of anonymized queries, ranked results, and clicked positions, to validate our model and parameters.

We found $\theta \approx 0.25$ to be the best fit parameter for the geometric family. Figure 1 compares the best-fitting geometric model with the observed clickthrough rates.

[1] Available at http://webscope.sandbox.yahoo.com, under Language Data

Users recorded in the Webscope log are likely to click on documents that have been rated "bad" by editors. In particular, "bad" documents at rank 1 have an empirical clickthrough rate of 0.5 and 0.14 at rank 2 (note that this is a good fit to the geometric model with $\theta = 0.25$. Similarly, users are more than willing to pass over non relevant documents; documents rated "excellent" have a clickthrough rate of 0.44 at rank 1 compared to 0.12 at rank 2 (again note the correspondence to the geometric model). Overall clickthrough rate for relevant documents (of any grade, including perfect) at rank 1 is 0.59, dropping to 0.1 at rank 2, because perfect documents account for a very large share of clicks.

It is surprising that "bad" documents actually have higher clickthrough rates than "excellent" documents at the same rank (we decline to hypothesize why this might be the case). Since the goal of our experiments is to estimate effectiveness measures in the presence of noise and missing feedback, we do require some positive correlation between relevance and clicks. For this reason we choose not to model the relatively high clickthrough rate on "bad" documents. Despite this, we believe that our model is sufficiently simulating the amount of noise present in real online evaluation settings, even if not simulating users in those settings with high fidelity.

4.4 Simulating an online testing environment

Finally, we simulate an online testing environment as follows. One of the simulated rankers is chosen as the production ranker (essentially randomly; there is no guarantee that any of the new rankers will be more effective than the production ranker). The simulation then proceeds in a loop. At each iteration, one query is selected randomly from the set to simulate a user submitting that query to the ranker. For the first 100,000 iterations, there is a 1% chance that the query is diverted to the swap policy and the user sees ranked results with two positions swapped. The other 99% of the time the user will see the original ranked results.

After the first 100,000 iterations, there remains a 1% chance of diverting to the swap policy, but there is an additional 1% chance of diverting to the insertion policy. In this case, the query is run against all of the simulated new rankers[2]. All new documents—those retrieved by new rankers but not by the production ranker—are identified and then one is chosen for insertion using one of the methods described in Section 3.5 above. This document is placed at anchor rank, replacing the document from the production ranker.

We use rank 2 for the anchor rank in both the swap and insertion policies. We believe this is better than using rank 1 because it does not risk replacing "perfect" documents with a poorer result.

4.5 Discussion

Simulations have disadvantages and advantages. One advantage of our simulation approach is that we can use the models to compute "gold standard" propensities and IPS estimators to compare our own to, and we can compute effectiveness measures on rankers to compare the IPS estimators to. This allows us to argue about the effectiveness of our approach in settings with similar degrees of noise and query abandonment.

[2]In practice all results have been pre-generated and cached for fast retrieval.

In particular, we can compute propensity at any rank k for any ranker S as:

$$p_{\mathcal{S},k} = \frac{R_{\mathcal{S}}}{N} p(click|R \geq 1)\theta^{k-1} + \frac{N - R_{\mathcal{S}}}{N} p(click|R = 0)\theta^{k-1}$$

and compare our estimates against these values. Here $R_{\mathcal{S}}$ is the number of relevant documents retrieved by $\mathcal{S}$ (across all queries) and N is the total number of documents retrieved (which in our simulations is the same for all rankers). This means propensity is directly and positively correlated to ranker effectiveness.

This raises a disadvantage of our simulation model, which is that in real rankers there is probably not a direct positive correlation between propensity and effectiveness. Furthermore, it is likely that user abandonment and click probabilities are influenced by other relevant documents in the ranking; we likewise do not capture this. Note that previous work on this topic also does not capture these factors; this is something we intend to investigate in future work.

Another possible disadvantage is that we do not simulate "head queries" vs "tail queries", as all queries are equally likely to be sampled. We consider them to be more-or-less representative of "torso queries", i.e. those that are not unlikely to appear in a stream and do not exhibit the typical characteristic of head queries of very high clickthrough rate at rank 1. Our approach should have no issue with tail queries; for those queries there will be little information in the log for the production and new rankers alike. Furthermore, preliminary experiments suggest that a more realistic distribution for query sampling is actually *easier* for our methods.

In general, we believe that the simulation and evaluation parameters we have chosen create an experimental environment that is more challenging for IPS methods than would actually be found in reality. We claim that if our methods do well in this environment, they are likely to work well in real settings as well.

5 EXPERIMENTS AND RESULTS

We experiment by simulating the full test environment described above. Table 2 summarizes simulation parameters; for justifications of these choices see Section 4. We evaluate rankers by precision@K and DCG@K and use Kendall's tau rank correlation to compare the ranking of rankers by those two measures to the ranking by the corresponding IPS estimate. We performed at least 10 iterations for each of the experiment below and report the mean value.

We have no clear baseline to compare to, since as far as we know this is the first work to investigate the evaluation of previously unseen rankers using log data without explicit relevance judgments. Looking at evaluation over the first 100,000 log lines is the best comparison to previous work such as Joachims et al., since in that epoch we are applying their method directly. Thus our first experiment validates that we have implemented that work correctly and compares our user model to theirs.

5.1 Evaluating production rankings and re-rankings

Joachims et al.'s work can be used to evaluate re-rankings of logged results. To verify that we have correctly implemented that work, we show results when estimating DCG@5 for 10 different re-ranking algorithms. We compare results using our geometric user model with

component	parameter	value
queries	number of queries	1000
	pool size C_q	$C_q \in \{10 \ldots 100\}$
	number relevant	$\sim \text{Binom}(C_q, 0.25)$
users	θ	0.25
	$p(click \mid R \geq 1)$	0.4
	$p(click \mid R = 0)$	0.2
traffic	% to swap policy	1%
	% to insertion policy	1%
logs	lines before starting insertions	100,000
	total log lines	9,000,000
policies	anchor rank	2
	swap rank selection	uniformly random

Table 2: Simulation parameters used in experiments.

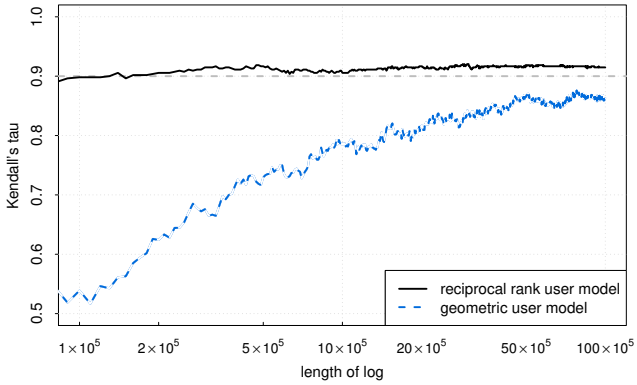

Figure 2: Kendall's tau correlations between DCG@5 and its IPS estimator for 10 rankers that re-rank the same 5 documents. This figure compares taus under two different user simulation models: the reciprocal rank model used by Joachims et al. and our geometric model.

noisy clicks to their user model, which has propensities inversely proportional to rank.

As Figure 2 shows, both models achieve higher correlations to the "true" ranking of rankers with more log data, though the reciprocal rank model starts out very high and only increases marginally with more data. The geometric model requires much more data to accurately estimate propensities because they are much smaller at low ranks than the inverse rank model. So while our results are in some sense "worse" in this figure, it is entirely because we chose to use a much more difficult user model—a choice we did not have to make. We argue that this allows us to make stronger statements about the generalizability of both methods to a variety of scenarios.

5.2 Evaluating new rankings

Our main experimental results are about evaluating new rankings.

Figure 3 shows results when the goal is to evaluate runs by precision at ranks 3, 5, and 10; Figure 4 shows the same for DCG at 3, 5, and 10. All six plots show the same pattern: an initial period of moderate but stable tau correlation when no documents are inserted. Then, as new documents are inserted, we observe a steep drop in tau values due to lack of sufficient data for the inserted documents. Finally, as more and more data is acquired the tau values increases

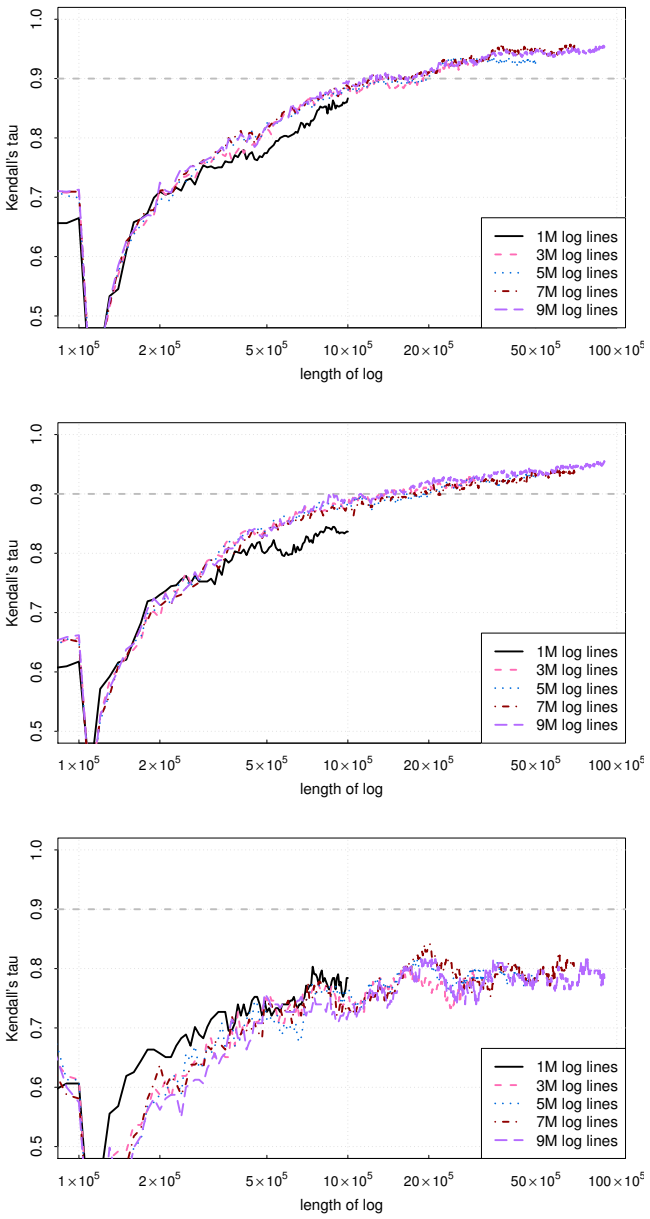

Figure 3: Kendall's tau correlation for the ranking by precision at K (top $K = 3$, middle $K = 5$, bottom $K = 10$) versus the ranking by the $\hat{V}_{\text{IPS}}$ estimate of precision. Each line represents a maximum log size to use; for example, the 1M line uses propensities estimated from one million log lines and ends after the millionth line. Correlation is computed every 10,000 log lines.

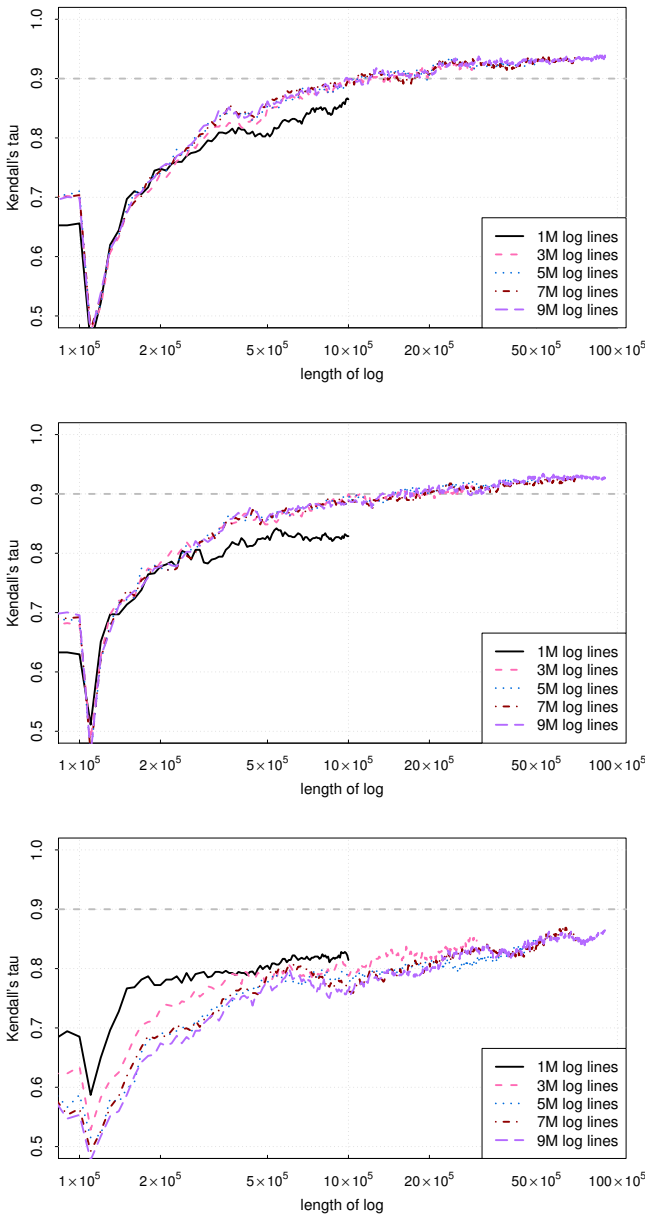

Figure 4: Kendall's tau correlation for the ranking by DCG at K (top $K = 3$, middle $K = 5$, bottom $K = 10$) versus the ranking by the $\hat{V}_{\text{IPS}}$ estimate of DCG. Each line represents a maximum log size to use; for example the 1M line uses propensities estimated from one million log lines and ends after the millionth line. Correlation is computed every 10,000 logs.

fast and eventually slows down as sufficient data is acquired for the inserted documents. Each plot shows results for logs of size varying from one million lines (about 20,000 swaps and insertions) up to nine million lines (about 180,000 swaps and insertions). For ranks 3 and 5, it is clear that one million is not enough, but by the

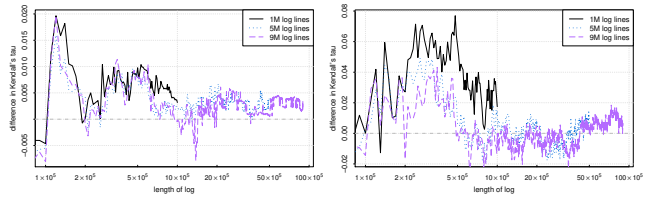

Figure 5: Differences in Kendall's tau correlation for the ranking by precision@3 versus the ranking by the $\hat{V}_{\text{IPS}}$ estimator of precision, comparing a weighted sampling function to uniform sampling. Left: using the precision-based weighting function increases Kendall's tau by a small amount. Right: using the DCG-based weighting function increases Kendall's tau by a larger amount.

time the log size is three million lines, a correlation to ground truth of 0.9 has been achieved, and more lines than that seems to make little difference beyond marginally increasing the correlation.

Effectiveness at rank 10 is much harder to estimate due to the very low propensity at that rank. It is easier to estimate DCG@10 than precision@10; this is because DCG@10 weights rank 10 less than precision@10 does. It is interesting that the smallest log gives the best results early on. This seems to be because the smaller log gives more "conservative" propensity estimates (due to smoothing) that have less chance of severely mis-ranking the systems.

5.2.1 Effect of weighted sampling for insertions. Figure 5 shows the effect of using weighted sampling for insertions as described in Section 3.5. While there is a positive effect on Kendall's tau correlations in the first million log lines, it is not large and highly variable, and the effect all but disappears after the first million lines. DCG-based sampling gives better results initially than precision-based sampling, but also hurts correlations more after the first one million lines.

5.2.2 Analysis by magnitude of difference in effectiveness. As mentioned in Section 4.2 above, our rankers are generated specifically so that some pairs will be very close in effectiveness (not statistically distinguishable) and others will have a large enough difference in magnitude to be statistically significant. In this section we break rankers out by their η parameter in order to investigate how much of the errors in rank correlation are due to the fact that rankers with identical ηs are essentially tied.

Figure 6 shows accuracy of the IPS prediction of the sign of the difference in effectiveness as a function of length of the log for one experiment (corresponding to "5M log lines" in the middle plot in Figure 3). Each line corresponds to pairs of rankers with equal or "adjacent" η parameters (that is, $|\log_2 \eta_i - \log_2 \eta_j| \leq 1$), as these are the pairs most likely to be mis-ranked. Pairs of rankers with larger differences are very unlikely to be mis-ranked; accuracy on those pairs exceeds 99% even with very few log lines.

It is clear from this figure that errors in rank correlation are primarily driven by pairs of rankers that are not statistically differentiable from each other, and secondarily by pairs of rankers with higher variability in effectiveness.

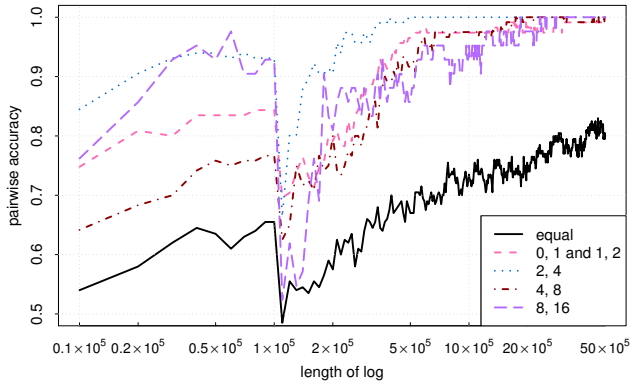

Figure 6: Accuracy of the IPS estimator at predicting the sign of the difference in precision@5 as log size increases (up to 5,000,000 lines). Each line shows accuracy for pairs of rankers with "adjacent" η parameters as given in the legend.

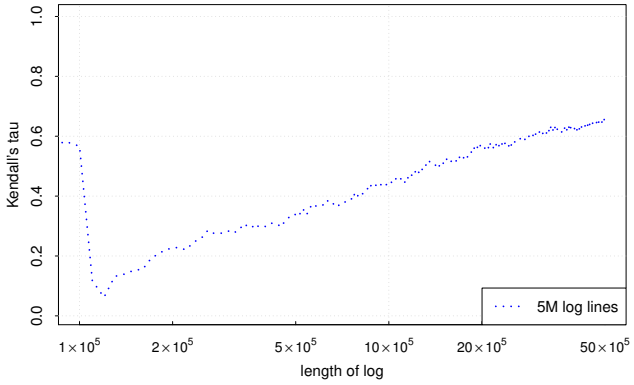

Figure 7: Kendall's tau correlation for 10 rankers ranked by precision@5 versus the IPS estimate of precision@5 based on a log of five million lines. Rankers were simulated to have on average only one document in common for each query.

5.2.3 Analysis by similarity of retrieved documents. New rankers are often quite similar to old rankers, retrieving many documents in common. Our results above use rankers that have about 60-70% of documents in common for each query. Figure 7 shows results when rankers have on average only one document in common. Since there are so many new documents to get feedback on, it takes much longer for the correlation to recover from the initial drop, only reaching its original value after more than two million lines. Thus decisions about how much traffic to route to the insertion policy should be partially based on how similar the rankers are in terms of the documents they retrieve.

5.2.4 Propensity estimation error. Figure 8 compares estimated propensity to the true value calculated as described in Section 4.5 and shows how the estimates improve with more data. With only 100,000 lines, estimates at ranks 1–5 are good, but propensities are

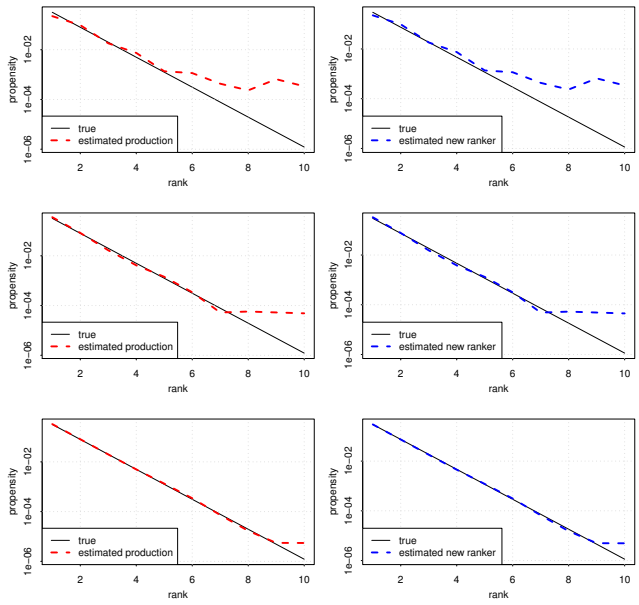

Figure 8: Comparisons of "true" propensity to estimated propensity. The upper two plots show propensity estimated after 100,000 log lines; the middle two after 1,000,000 log lines; and the lower two after 9,000,000 log lines. The left three plots show propensity for the production ranker; the right three for one of the new rankers (selected randomly).

highly overestimated after that due to smoothing and lack of data. After 1,000,000 lines, estimates are quite good down to rank 7, and after 9,000,000 lines estimates are accurate to rank 9. This shows that the amount of data required for good estimates increases faster than linearly with rank; we estimate it would take another several million lines to have a good estimate at rank 10.

5.2.5 Evaluating real rankers. Finally, in order to validate our methodology using "real" ranking algorithms, we evaluated nine learning-to-rank algorithms based on a labeled dataset. We used the MSLR-WEB10K dataset that contains editorial judgments for query-document pairs on a 4 point scale with 10K queries[3]. All nine rankers were trained using the *train* set and the results on the *test* set were used for simulations [4]. The average overlap between the systems is about 0.42. One ranker was treated as the production system and logs were generated using the methodology in Section 4.4. The experiment was repeated 10 times with each of the nine rankers as the production system. Figure 9 shows the Kendall's tau correlation for nine systems ranked by DCG at 5 using editorial judgments versus system ranking by the $\hat{V}_{\text{IPS}}$ estimate of DCG. Since these systems are less similar to each other than our simulated rankers, it takes longer to recover from the initial drop when beginning the insertion policy than in Figures 3 and 4, but not as long as Figure 7.

[3] https://www.microsoft.com/en-us/research/project/mslr/

[4] Multiple Additive Regression Trees (MART, RankNet, RankBoost, AdaRank, Coordinate Ascent, LambdaMART, ListNet, Random Forests and Linear Regression were trained using using RankLib with the default setting for the parameters

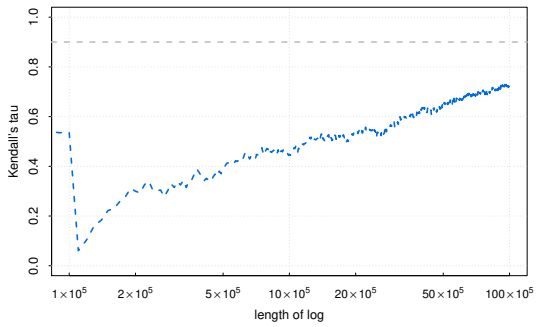

Figure 9: Kendall's tau correlations for DCG@5 versus its IPS estimate for "real" rankers trained on the MSLR-WEB10K dataset.

6 CONCLUSION

Building on the work of Joachims et al., we have introduced a method for collecting incremental implicit online feedback for the purpose of offline evaluation of rankers that can retrieve never-before-seen documents. Our method attempts to answer the counterfactual question "how would users have interacted with a new ranker had we fully deployed it?" without the risk of A/B testing or even interleaving a poorly-performing ranker. The amount of perturbation of production ranker results is minimal, as we only ever change one or two rank positions.

We tested our method through simulation and showed that, under some basic assumptions about the relationship between relevance and clicks, we are able to reliably distinguish rankers even in a difficult, noisy experimental setting with about 1 million log lines, of which only 2% go to light perturbation policies—though the total amount depends on how similar the rankers are to each other. More intelligent prioritization of documents to sample for feedback further improves the signal although only by a small amount.

We also showed the limits of the method: for measures like precision@10 that weight all ranks equally, the high variance in propensity estimates at low ranks make it much more difficult to distinguish between rankers. Furthermore, a user model that has a very small chance of generating clicks at low ranks makes the propensity estimation at these ranks much harder.

We intend to continue this work by further relaxing assumptions and improving the models. In particular, it has been observed that the relevance of results at higher ranks can affect user behavior at lower ranks; this has implications for propensity estimation. It may also be the case that more intelligent swapping and insertion policies can result in faster conclusions about differences between rankers. Most importantly, we plan to test these methods in a real on-line setting, specifically in rankers designed for music retrieval and recommendation.

REFERENCES

[1] R. Agrawal, A. Halverson, K. Kenthapadi, N. Mishra, and P. Tsaparas. *Generating labels from clicks.* WSDM '09. 2009.
[2] L. Bottou, J. Peters, J. Q. Candela, D. X. Charles, M. Chickering, E. Portugaly, D. Ray, P. Y. Simard, and E. Snelson. Counterfactual reasoning and learning systems - the example of computational advertising. *Journal of Machine Learning Research ()*, 2013.
[3] B. Carterette and R. Jones. Evaluating search engines by modeling the relationship between relevance and clicks. In J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis, editors, *NIPS*, pages 217–224. 2008.
[4] O. Chapelle, T. Joachims, F. Radlinski, and Y. Yue. Large-scale validation and analysis of interleaved search evaluation. *Trans. Inf. Systems*, 30(1):6:1–41, 2012.
[5] O. Chapelle and Y. Zhang. *A dynamic bayesian network click model for web search ranking*. 2009.
[6] A. Chuklin, I. Markov, and M. de Rijke. Click Models for Web Search. *Synthesis Lectures on Information Concepts, Retrieval, and Services*, 7(3):1–115, 2015.
[7] G. V. Cormack, C. R. Palmer, and C. L. a. Clarke. *Efficient construction of large test collections.* ACM, New York, New York, USA, Aug. 1998.
[8] N. Craswell, O. Zoeter, M. J. Taylor, and B. Ramsey. An experimental comparison of click position-bias models. *WSDM*, page 87, 2008.
[9] M. Dudík, D. Erhan, J. Langford, and L. Li. Doubly Robust Policy Evaluation and Optimization. *Statistical Science*, 29(4):485–511, 2014.
[10] G. E. Dupret and B. Piwowarski. A user browsing model to predict search engine click data from past observations. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '08, pages 331–338, 2008.
[11] K. Hofmann, L. Li, and F. Radlinski. Online Evaluation for Information Retrieval. *Foundations and Trends® in Information Retrieval*, 10(1):1–117, 2016.
[12] K. Hofmann, S. Whiteson, and M. de Rijke. A probabilistic method for inferring preferences from clicks. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, CIKM '11, pages 249–258, 2011.
[13] T. Joachims. Optimizing search engines using clickthrough data. *conference on Knowledge discovery and data mining*, 2002.
[14] T. Joachims, L. a. Granka, B. Pan, H. Hembrooke, F. Radlinski, and G. Gay. Evaluating the accuracy of implicit feedback from clicks and query reformulations in Web search. *ACM Transactions on Information Systems*, 25(2):7–es, 2007.
[15] T. Joachims, A. Swaminathan, and T. Schnabel. Unbiased Learning-to-Rank with Biased Feedback. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining - WSDM '17*, 2017.
[16] M. T. Keane and M. O'Brien. Click Models for Web Search. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, volume 28, 2006.
[17] R. Kohavi, R. Longbotham, D. Sommerfield, and R. M. Henne. Controlled experiments on the web: Survey and practical guide. *Data Min. Knowl. Discov.*, 18(1):140–181, 2009.
[18] D. Li and E. Kanoulas. Active Sampling for Large-scale Information Retrieval Evaluation. *arXiv.org*, pages 49–58, Sept. 2017.
[19] L. Li, S. Chen, J. Kleban, and A. Gupta. Counterfactual Estimation and Optimization of Click Metrics in Search Engines - A Case Study. pages 929–934, 2015.
[20] A. Lipani, J. R. M. Palotti, M. Lupu, F. Piroi, G. Zuccon, and A. Hanbury. Fixed-Cost Pooling Strategies Based on IR Evaluation Measures. *ECIR*, 2017.
[21] D. E. Losada, J. Parapar, and A. Barreiro. Feeling lucky? - multi-armed bandits for ordering judgements in pooling-based evaluation. *SAC*, pages 1027–1034, 2016.
[22] A. Moffat and J. Zobel. Rank-biased precision for measurement of retrieval effectiveness. *ACM Trans. Info. Sys.*, 27(1):1–27, 2008.
[23] U. Ozertem, R. Jones, and B. Dumoulin. Evaluating new search engine configurations with pre-existing judgments and clicks. In *Proceedings of the 20th International Conference on World Wide Web*, WWW '11, pages 397–406, 2011.
[24] V. Pavlu and J. Aslam. A practical sampling strategy for efficient retrieval evaluation. 2007.
[25] V. Pavlu, E. Yilmaz, J. A. Aslam, and H. Ave. A Statistical Method for System Evaluation Using Incomplete Judgments. pages 541–548, 2006.
[26] F. Radlinski and T. Joachims. Minimally invasive randomization for collecting unbiased preferences from clickthrough logs. In *Proceedings of the 21st National Conference on Artificial Intelligence - Volume 2*, AAAI'06, 2006.
[27] F. Radlinski, M. Kurup, and T. Joachims. How does clickthrough data reflect retrieval quality? In *Proceedings of the 17th ACM Conference on Information and Knowledge Management*, CIKM '08, pages 43–52, 2008.
[28] M. Richardson, E. Dominowska, and R. Ragno. *Predicting clicks: estimating the click-through rate for new ads.* 2007.
[29] P. R. ROSENBAUM and D. B. RUBIN. The central role of the propensity score in observational studies for causal effects. *Biometrika*, 70(1):41–55, 1983.
[30] E. M. Voorhees and D. K. Harman. *TREC : Experiment and Evaluation in Information Retrieval.* MIT Press, 2005.
[31] X. Wang, M. Bendersky, D. Metzler, and M. Najork. Learning to Rank with Selection Bias in Personal Search. pages 115–124, 2016.
[32] E. Yilmaz, E. Kanoulas, and J. A. Aslam. A simple and efficient sampling method for estimating AP and NDCG. In *the 31st annual international ACM SIGIR conference*, page 603, New York, New York, USA, July 2008. ACM Request Permissions.
[33] Y. Yue, R. Patel, and H. Roehrig. Beyond position bias: Examining result attractiveness as a source of presentation bias in clickthrough data. In *Proc. of the 19th International Conference on World Wide Web*, WWW, pages 1011–1018, 2010.