Optimized Interleaving for Online Retrieval Evaluation

Filip Radlinski
Microsoft
Cambridge, UK
filiprad@microsoft.com

Nick Craswell
Microsoft
Bellevue, WA, USA
nickcr@microsoft.com

ABSTRACT

Interleaving is an online evaluation technique for comparing the relative quality of information retrieval functions by combining their result lists and tracking clicks. A sequence of such algorithms have been proposed [6, 13, 4], each being shown to address problems in earlier algorithms. In this paper, we formalize and generalize this process, while introducing a formal model: We identify a set of desirable properties for interleaving, then show that an interleaving algorithm can be obtained as the solution to an optimization problem within those constraints. Our approach makes explicit the parameters of the algorithm, as well as assumptions about user behavior. Further, we show that our approach leads to an unbiased and more efficient interleaving algorithm than any previous approach, using a novel log-based analysis of user search behavior.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Information Search and Retrieval.

Keywords: Interleaving, Evaluation, Web Search

1. INTRODUCTION

In most studies retrieval evaluation is performed using manual relevance judgments that assess the relevance of particular documents to particular queries, or by observing user behavior in an actual retrieval system. In both cases, the goals are clear: Sensitivity to small improvements in retrieval quality for a given cost of evaluation, and fidelity to the actual user experience were real users to directly compare particular retrieval systems.

The most common approach involves relevance judgments. Among other benefits, this most easily allows for reproducibility and reusability: A retrieval algorithm can be run on a document collection for a particular query set for which judgments are known. Performance can be measured using any number of metrics such as Mean Average Precision (MAP) [11], Discounted Cumulative Gain (DCG) [11], or Subtopic Recall [16]. Different researchers can apply the same retrieval algorithms to the same collection to reproduce results. Further, the data can be used to evaluate future retrieval methods on the same document collection and query set.

In contrast, online evaluation involves real users searching for actual and current information needs. The users can enter search terms, and documents are retrieved. Based on the user interface (in particular the captions shown by the retrieval system), users select results, reformulate or revise their information need, and continue with other tasks. Their behavior in selecting queries issued and documents clicked can be interpreted as feedback about the documents retrieved. Reproducing an evaluation requires showing new users similar results. If the new users are substantially different, or have substantially different needs or behavior, the outcome may change. Furthermore, a record of behavior given particular documents returned to users does not tell the researcher how the users would have behaved had other documents been shown, so observed behavior is not easily reusable when evaluating new retrieval methods.

However, online evaluation benefits from involving real users, particularly when conducted on real queries in situ. In that case there is no uncertainty in how a judge should interpret the query *sdsu*, nor how to trade off the relevance of new and old documents to the query *wsdm*. All aspects of the users' context and state of knowledge are present, some of which may be difficult to realistically capture in a test collection. Finally, as usage data can be collected essentially for free by any active information retrieval system, its cost can be much lower than that of obtaining sufficient relevance judgments from experts to detect small relevance improvements. For these reasons, we focus on online evaluation.

Among online evaluation approaches in the context of Web search, two methods dominate today: The first involves measuring properties of user responses to retrieval, such as the the time it takes users to click, or the positions of these clicks and other observable behavior [9, 14, 5]. This can be used to compute a score for a given retrieval algorithm that can be compared across systems. The second approach invovles showing users a *combination* of results retrieved by different ranked retrieval algorithms, and observing which results users select from this combination. This is usually termed *interleaved* evaluation. A number of authors have demonstrated the higher sensitivty of the interleaved approach [7, 13, 1], largely due to the within-user comparison that is taking place: The same user with the same information need at the same time is shown the best results proposed by both systems, and directly chooses between them.

WSDM'13, February 4–8, 2013, Rome, Italy.

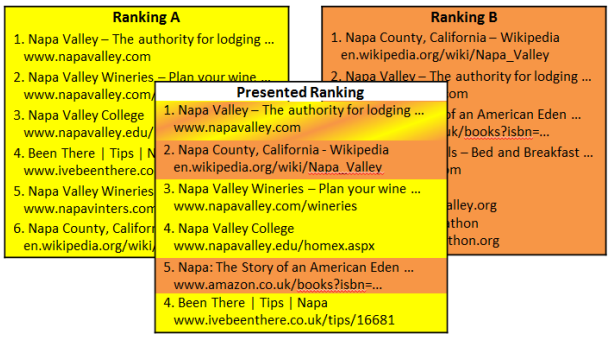

Figure 1: Illustrative example of interleaving. Rankings produced by two retrieval functions for the query *napa valley*, are combined into an interleaved combination.

In this paper, we address the question of how to obtain an optimal interleaving algorithm. In particular, a number of interleaving algorithms have been proposed, including Balanced Interleaving [6], Team Draft Interleaving [13] and Probabalistic Interleaving [4]. Further, additional variants of interleaving algorithms, for example involving how credit is assigned for clicks have been proposed [2, 15]. This leads to the question of what the best interleaving algorithm would be, and why. Also, the two most recent interleaving algorithms addressed unexpected flaws in previous algorithms. This suggests that analysis of interleavings algorithm is difficult, and that the properties encoded by each algorithm implicitly make assumptions that are difficult to verify. The designer of an algorithm risks creating new problems, even while fixing existing ones.

We thus invert the problem of deriving an interleaving algorithm: Starting with properties that we wish the algorithm to have, we formulate interleaving as the solution to an optimization problem. We then solve for the algorithm. This is the key contribution of our work. A second contribution is our evaluation method: Following a similar approach to Li et al. [10], we show how different interleaving algorithms can be evaluated on historical log data, without running a new algorithm on new users.

This paper is organized as follows. First, we present interleaved evaluation in detail. We then describe our approach, interleaving as the solution to an optimization problem. After formulating the optimization problem, we show theoretical guarantees and solve for two interleaving algorithms. We conclude with an evaluation comparing our approach with previous interleaving algorithms using a real usage data.

2. INTERLEAVING

An interleaving evaluation compares two retrieval functions R_A and R_B. Rather than showing the results of R_A to some users, and R_B to the rest, both results lists are generated for each query issued and a randomized combination is presented to users, as illustrated in Figure 1.

2.1 Goals of Interleaving

While mixing retrieval results from multiple rankings was first described by Kantor [8], the first interleaving algorithm was detailed and implemented by Joachims [6, 7]. He proposed that the comparison should satisfy:

(a) be blind to the user with respect to the underlying [retrieval functions],

(b) be robust to biases in the user's decision process that do not relate to retrieval quality,

(c) not substantially alter the search experience, and

(d) lead to clicks that reflect the user's preference.

2.2 Previous Algorithms

A number of interleaving algorithms have been proposed that attempt to satisfy these requirements. We focus on three: Balanced, Team Draft and Probabalistic interleaving. In all cases, we suppose that we are wishing to construct an interleaved list of results $A = (a_1, a_2, \ldots)$ and $B = (b_1, b_2, \ldots)$. We denote the interleaved list $I = (i_1, i_2, \ldots)$.

Balanced Interleaving

Balanced interleaving [6] creates a combined list I such that the top-k_I results of I includes the top-k_A of A and the top-k_B of B where for every k_I, values k_A and k_B differ by at most 1. The algorithm runs as follows: First, toss an unbiased coin. The outcome of this coin toss is $t = (A)$ or $t = (B)$. Let there be two pointers p_A and p_B that indicate the rank of the highest ranked document in A and B respectively that is not yet in I. Construct I by greedily appending a_{p_A} whenever $p_A < p_B$, or $p_A = p_B$ and $t = (A)$, and appending b_{p_B} otherwise. Recompute p_A and p_B after each document is added to I.

Team Draft Interleaving

Radlinski et al. proposed Team Draft interleaving [13] based on how sports teams are often assigned in friendly matches: Two captains (each with a preference order over players) toss a coin, then take turns picking players for their team.

In the retrieval setting, the algorithm proceeds similarly: In each iteration, toss an unbiased coin, t. If $t = (A)$, append the next available result from A (i.e. the next highest result in A that is not already in I) to I, followed by the next available result from B. If $t = (B)$, follow the same procedure but with B before A. This process continues, with a coin toss and two results added per iteration. The output is the interleaved list I and a record of which list provided each of the results in I.

Probabilistic Interleaving

Probabilistic interleaving, proposed by Hofmann et al. [4], is similar to Team Draft but allows a richer set of rankings to be constructed. First, rather than taking two results per coin toss, Probabilistic interleaving takes one. In an extreme case, I could be constructed entirely from A (with probability $0.5^{|I|}$), but in expectation A and B contribute the same number of results. Second, when selecting a result to append to I, Team Draft interleaving selects the next available result from a ranking. In Probabilistic Interleaving, results are instead selected with a probability decreasing with the position of the result in the ranking from which it is being selected. Thus, when selecting the first document in I from A, it is most likely that a_1 is selected, less likely that a_2 is selected, even less likely that a_3 is selected, and so forth.

Assigning Credit for Clicks

Having established these three list combination approaches, a way to interpret user clicks in each case is also required. Each interleaving algorithm was presented with a credit assignment rule, while other work has proposed alternate credit

assignment approaches (only). It is the combination of list interleaving and credit assignment that comprises an interleaving algorithm, and determines fidelity and sensitivity.

In Balanced interleaving, when clicks are observed, the minimum values of k_A and k_B to generate the list down to the lowest click are computed. Then, $k = max(k_A, k_B)$. The number of clicked documents present in the top k of A is compared to the number of clicked documents in the top k of B. The ranking with the higher count "wins" this ranking instance (also termed *impression*), otherwise it is a tie. If one of the retrieval functions wins more than half of non-tied impressions, Balanced interleaving concludes that this retrieval function is better.

In Team Draft interleaving, each document is assumed to "be on the team" of the ranking from which it was selected: Clicks on this document count as credit for that ranker (each team always has an equal number of documents). The ranker with more credit "wins" each impression. A refinement ignores clicks on results in any identical prefix of both A and B, increasing sensitivity of the algorithm [1].

In Probabalistic interleaving, credit assignment is more sophisticated: Given an interleaved ranking I, all the possible coin toss sequences that could have resulted in I are computed. For each possible sequence of coin tosses, the algorithm computes which retrieval function "wins" using the same logic as with Team Draft. Probabilistic interleaving then assigns a probabalistic outcome to each impression based on how likely each ranker is to have won each impressions across all possible consistent coin toss inputs.

Finally, an assumption implicit in the above is that all clicks on documents are treated equally. Yue et al. [15] showed how to learn a weight for each click to improve sensitivity, for example perhaps treating top-position clicks differently. Hofmann et al. [3] reduce bias by similarly reweighting clicks. Alternatively, Hu et al. [2] proposed to interpret clicks as pairwise preferences over documents.

2.3 Success Criteria and Breaking Cases

Given these alternatives, the question arises of what makes a successful interleaving algorithm. When considering success, we adopt the approach of [12], focusing on fidelity and sensitivity. An interleaving algorithm with good fidelity will tend to agree with other appropriate evaluation methods such as test collections or randomized A/B tests in identifying the better of two retrieval functions. A sensitive algorithm will make efficient use of data, coming to a statistically significant conclusion with fewer interleaved impressions.

Fidelity and sensitivity of interleaving algorithms can be affected by systematic problems in their design, known as breaking cases. Breaking cases are interesting in the current study since they indicate algorithmic problems that we wish to address. More generally, in a practical setting it is useful to know the breaking cases of an algorithm, since a particular comparison (with an 'unlucky' pair of retrieval functions) may systematically have that case more often, which can change or delay the outcome of the comparison.[1]

As shown in [13, 1], $A = (doc_1, doc_2, doc_3)$ and $B = (doc_3, doc_1, doc_2)$ is a breaking case for Balanced interleaving. In particular, a user who clicks at random on one of the documents in the interleaved list will prefer A two out of three times. Team Draft interleaving also draws the wrong conclusions for some pairs of rankings A and B [4, 1]. For instance, if the only relevant and clicked document is doc^*, in expectation Team Draft interleaving prefers neither ranking when $A = (doc_1, doc^*, doc_3)$ and $B = (doc_1, doc_2, doc^*)$, despite one clearly being better for users. Finally, while Probabilistic interleaving avoids these breaking cases, it can show rankings that are very dissimilar from A and B, potentially degrading the user experience.

[1]Note that while these constructed breaking cases are *possible* for the interleaving algorithms, in many real-world evaluations they are not prevalent enough to affect the outcome.

3. OPTIMIZING INTERLEAVING

We now turn to the question of deriving an interleaving algorithm as the solution to an optimization problem. We start by deriving the constraints on this problem.

3.1 Refining Interleaving Goals

Refining the desirable properties presented by Joachims [6], we propose to modify the two last conditions:

(c') The comparison does not substantially alter the search experience, presenting the user with one input ranking, or the other, or a ranking that is "in between" the two.

(d') An interleaved evaluation produces a preference for a ranker if and only if the users prefer the documents returned by that ranker. Specifically:

- d'.1 If document d is clicked, the input ranker that ranked d higher is given more credit for the click than the other ranker.
- d'.2 In expectation, a randomly clicking user does not create a preference for either input ranker.

These goals will be written formally in Section 3.2. However, we first consider the intuition.

There are two naturally competing goals in an online evaluation: Should rankings be shown to obtain maximally useful relevance information, or should rankings that minimally disrupt the user be shown? Similar to Joachims requirement (c), we argue for the latter as users can quickly abandon an information retrieval system that performs poorly – even if it is doing so for evaluation reasons. In our formulation, (c') aims to guarantee small relevance impact: (1) interleaving two identical lists must yield the same list; (2) if two lists start with the same k documents, the interleaved list must also start with those same k documents; (3) if document d_1 is ranked higher than d_2 in both input rankings, it is also ranked higher in the interleaved ranking; (4) any document shown in the top k by an interleaving algorithm must be in the top k of at least one of the input rankings.

The property (d') limits how credit can be assigned to rankers based on clicks. For example, if a ranking is improved by moving the only relevant document higher (and users click on relevant documents), then the interleaving algorithm must recognize this improvement. Similarly, if clicks are made randomly then there should not be any preference.

Sensitivity

A further goal is for the interleaving algorithm to be as sensitive as possible to changes in ranking quality. This means that it should require the fewest user queries (or impressions) possible to infer a statistically significant preference. While absent from Joachims' original explicit criteria, we will show how to incorporate this as well.

Given these goals, we now show how the interleaving algorithm can be written as the solution to an explcit optimization problem.

3.2 Optimization Framework

Suppose we have two input rankers, R_A and R_B, that we wish to compare. We use lowercase letters to denote documents returned by these rankers, and uppercase letters to denote (ordered) rankings of results. For example, $A(q) = (a_1, \ldots, a_n)$ denotes the results retrieved by R_A for query q. Let $A^k(q)$ denote the (unordered) set of top-k documents $\{a_1, \ldots, a_k\}$. Without loss of generality, for notational simplicity we present the optimization problem for a single fixed query, for example writing A instead of $A(q)$.

Our goal is to obtain an algorithm that, given any two rankings A and B, produces a distribution over interleaved rankings of documents $\mathcal{L}$. The parameter we learn, p_j, is the probability with which each ranking $L_j \in \mathcal{L}$ will be shown to users[2]. For any given $L = (l_1, l_2, \ldots) \in \mathcal{L}$, let $\delta_i(L)$ (subsequently δ_i) be the (real valued) credit assigned to R_A whenever document l_i is clicked. Thus δ_i is positive if ranker R_A receives credit for this document, and negative if ranker R_B receives credit.

We next address randomly clicking users: We would like that for any user who clicks at random there is no preference inferred by the interleaving algorithm for either input retrieval function. However, it is not clear how to formalize this requirement. Instead, we require this to be the case for a specific model random user:[3] Let a randomly clicking user be a user who (1) picks a random threshold k from any distribution, then (2) clicks on $\eta \geq 1$ documents in the top-k of the interleaved list chosen uniformly at random.

We can finally write a formal definition of our constraints, based on the intuition from Section 3.1:

c' The interleaving list L satisfies:

$$\forall k.\ \exists i, j.\ s.t.\ L^k = A^i \bigcup B^j \tag{1}$$

This simply requires that any prefix of L consists of all top i documents from ranking A and all top j documents from ranking B. This means that, for example, the top document of L must be either a_1 or b_1. Similarly, the top two must be $\{a_1, a_2\}, \{a_1, b_1\}$ or $\{b_1, b_2\}$.

d'.1 The credit function δ satisfies:

$$rank^*(l_i, A) < rank^*(l_i, B) \quad \Leftrightarrow \quad \delta_i > 0 \tag{2}$$
$$rank^*(l_i, A) > rank^*(l_i, B) \quad \Leftrightarrow \quad \delta_i < 0 \tag{3}$$

where $rank^*(d, R)$ is the rank of document d in R if $d \in R$, and $|R| + 1$ otherwise. Rank positions are numbered from top to bottom, so the highest position has the lowest rank.

d'.2 Given the probability with which each ranking L_j is shown to users,

$$\forall k, \forall \eta,\ E\left[\eta E\left[\delta_i\right]_{i \in 1 \ldots k}\right]_{L_j} = 0 \tag{4}$$

[2]For example, in Balanced interleaving, there would be one or two elements in $\mathcal{L}$, with equal values of p_j. For Team Draft interleaving, there are up to $2^{|L|/2}$.

[3]Other models of random user could be used instead of, or even in addition to, this model.

3.3 Problem Definition

The set of permissible interleaved rankings is:

$$\mathcal{L} = \{L : \forall k, \exists i, j.\ s.t.\ L_k = A_i \bigcup B_j\} \tag{5}$$

The only parameter of the algorithm is the probability p_i with which each ranking $L_i \in \mathcal{L}$ is shown to users. The values of p_i must satisfy

1. Each ranking L_i is shown with a valid probability:
$$p_i \in [0, 1] \tag{6}$$

2. The probabilities add to 1:
$$\sum_{i=1}^{|\mathcal{L}|} p_i = 1 \tag{7}$$

3. The expected credit from a random user is zero:
$$\forall k, \forall \eta,\ \sum_{n=1}^{|\mathcal{L}|} \left(p_n \eta \frac{1}{k} \sum_{i=1}^{k} \delta_i \right) = 0$$
which simplifies to:
$$\forall k,\ \sum_{n=1}^{|\mathcal{L}|} \left(p_n \sum_{i=1}^{k} \delta_i \right) = 0 \tag{8}$$

For any input rankings A and B, and credit function δ_i, the solution values of p_i completely define an interleaving algorithm. The p_i values indicate the probablity with which each ranking is shown to users, and δ_i determines credit assignment.

However, this problem is also usually underconstrained. To produce an interleaved list of length k, there are $k + 1$ constraints but up to 2^k parameters p_i (although in practice usually many fewer, as A and B in Equation 5 are often similar in most real-world comparisons). We can therefore further refine the algorithm by optimizing for sensitivity.

3.3.1 Ensuring Sensitivity

As formulated, this problem allows any ranking that shows any prefix combination of results from A and B to be shown to users. However, as noted earlier, interleaving is more sensitive than comparisons where each ranking is shown to half of users because the same user observes documents from *both* rankers simultaneously and chooses between them. We therefore propose to maximize a sensitivity term subject to the above constraints (Equations 5 through 8).

Intuitively, the solution is to ensure fairness at the impression level: For every ranking shown to users, both rankers should get approximately equal space. While our formulation ensures fairness in expectation, here we wish to maximize it for every impression shown to users. Our goal is also similar to that of Probabilistic interleaving [4], where each impressions is given a real valued score that depends on all possible team assignments that could have generated it, smoothing the preference inferred. However, we instead select impresions that are most sensitive to ranking changes.

We choose a simple model of sensitivity to retain tractability that is a special case of our model in Section 3.2. Suppose that users observe the result at position i with probability

$f(i)$. Assuming a single random click on an interleaved list L, the probability of ranker R_A winning the impression is

$$w_A(L) = \sum_{i:\delta_i>0} f(i), \tag{9}$$

although we now drop the parameter L from $w_A(L)$ for succinctness. Similarly, the probability of ranker R_B winning the impression, and of the rankers being tied can be written

$$w_B = \sum_{i:\delta_i<0} f(i) \tag{10}$$

$$w_T = \sum_{i:\delta_i=0} f(i) \tag{11}$$

Sensitivity can be defined as the uncertainty in the winner of a particular impression. Intuitively, sensitivity should be low if one of the rankers is bound to be preferred, or if the impression is always tied. Conversely, if each ranker would win 50% of impressions with random clicking, sensitivity should be high. We choose the following form of the sensitivity $s(L)$ of a particular ranking L:

$$s(L) = 0 \times w_T + entropy(w_A, w_B) \times (1 - w_T) \tag{12}$$

$$= -\frac{1 - w_T}{w_A + w_B} \log_2 \frac{w_A^{w_A} w_B^{w_B}}{(w_A + w_B)^{w_A + w_B}} \tag{13}$$

Note that if $w_T = 0$, this reduces to the entropy of a binomial outcome. For example, if for a given L, $w_A = w_B = 0.5$, the sensitivity is 1. On the other hand, if $w_T = 1$, $w_A = 1$ or $w_B = 1$, the sensitivity is zero.

The optimized interleaving algorithm thus reduces to maximizing the expected sensitivity (Equation 13) subject to the constraints presented in Equations 5 through 8[4].

An alternative formulation may be to minimize the variance in the expected outcome of the entire interleaving experiment across all impressions, although we leave assessing this to future work. We also note that different sensitivity criteria could be proposed, and be substituted into our approach. This would simply change the optimal probabilities for showing different ranking.

3.3.2 Secondary Goals

If we desire, we can also add extra criteria to optimize at the expense of sensitivity. For example, *all else being equal, minimize the frequency with which we show results from ranker R_B*. This might be required, for example, if we are performing high-risk comparisons of a known high quality ranker R_A with an experimental ranker R_B.

3.3.3 Credit as a Parameter

One particular difference in this formulation over previous approaches is that the credit function δ is an explicit parameter of the algorithm. This may seem a new parameter, but was in fact implicit in previous algorithms. For instance, Balanced, Team Draft and Probabalistic interleaving approaches each deal with multiple clicks differently. Yet this choice may have a non-trivial impact on the outcome of experiments, and the implicit behavior may not be desired. Rather, an explicit δ means the experimenter can choose the tradeoff, much as judgment based metrics make explicit the tradeoff between documents at different positions. We will consider some examples of credit functions below.

[4]For simplicity, in our evaluation we assume that $f(i)$ in Equations 9 through 11 is proportional to $1/i$

Probabilistic Interleaving encodes a similar idea – credit is assigned based on possible coin toss outcomes, and the closer a document is in R_A and R_B, the smaller its effect when clicked. However, instead of encoding credit in how documents are selected when interleaving, we make it explicit. Yue et al.'s [15] approach is also related, where the weight of each click is learned. Our approach is complimentary: An experimenter can specify a credit function based on where clicks happen, and the importance of e.g. duration of click may be learned using their approach.

3.4 Theoretical Guarantees

In this section we show the theoretical benefits of our approach over previous algorithms.

3.4.1 Allowed Rankings

We first consider the class of rankings allowed in $\mathcal{L}$, which are a superset of those produced by Team Draft and Balanced interleaving and a subset of those produced by Probabilistic interleaving.

PROPERTY 1. *The rankings produced by Balanced interleaving are in $\mathcal{L}$.*

PROOF INTUITION. *Balanced interleaving constructs an interleaving by maintaining pointers p_A and p_B to the highest ranked documents in each input ranking but that are not in the interleaved list. It then greedily selects the next document to add to the interleaved list based on the higher pointer, or the coin toss in the event of a tie. The value of i and j in Equation 5 is equal to $p_A - 1$ and $p_B - 1$ when result k is added by the Balanced interleaving algorithm.* □

PROPERTY 2. *The rankings produced by Team Draft interleaving are in $\mathcal{L}$.*

PROOF INTUITION. *Team Draft interleaving involves the two rankers greedily selecting the next available document to add to the interleaved list subject to the coin tosses. The ranks of the last document that is not available for each ranker correspond to the values of i and j in Equation 5.* □

PROPERTY 3. *The rankings produced by Probabilistic interleaving are a superset of $\mathcal{L}$.*

PROOF INTUITION. *Every prefix L^k involves the top i documents from ranking A and top j documents from ranking B. Clearly these could have been selected at random by the Probabilistic interleaving algorithm.* □

We also note that any ranking shown to users by our approach does not misorder more pairs of documents than would be misordered in expectation if half the users were shown the results from ranker A and half the users were shown the results from ranker B:

PROPERTY 4. *For every L^k prefix of length k of any $L \in \mathcal{L}$, the number of misordered pairs of documents between L^k and A^k plus the number of misordered pairs between L^k and B^k is less than or equal to the number of misordered pairs between A^k and B^k.*

PROOF INTUITION. *Let $rank^*(d, R)$ be the rank of document d in R if $d \in R$, and $|R| + 1$ otherwise. For any pair of documents $l_i, l_j \in \mathcal{L}^k$, with $i < j$, there are three cases. One of three cases must hold:*

1. $rank^*(l_i, A) \leq rank^*(l_j, A), rank^*(l_i, B) \leq rank^*(l_j, B)$ *This pair of documents is not misordered with respect to either A or B.*
2. $rank^*(l_i, A) > rank^*(l_j, A), rank^*(l_i, B) \leq rank^*(l_j, B)$ *This pair is misordered between L and A. However, it is also misordered between A and B.*
3. $rank^*(l_i, A) \leq rank^*(l_j, A), rank^*(l_i, B) > rank^*(l_j, B)$ *This pair is misordered between L and B. However, it is also misordered between A and B.*

It is not possible that $rank^*(l_i, A) > rank^*(l_j, A)$ and that $rank^*(l_i, B) > rank^*(l_j, B)$ because l_i could then not be at position i in L: Document l_j would have had to have been selected before l_i could have been selected because it precedes l_i in both A and B. Hence the total number of misordered documents must be bounded as required. □

3.4.2 Comparisons with Previous Algorithms

Balanced interleaving and Team Draft interleaving both have known breaking cases that Optimized interleaving solves:

PROPERTY 5. *The breaking cases with Balanced interleaving do not exist with Optimized interleaving.*

PROOF. *Balanced interleaving is biased when one of the rankings is preferred more often in expectation by a randomly clicking user. The randomly clicking user constraints ensure this does not occur with Optimized interleaving.* □

PROPERTY 6. *The breaking cases where Team Draft interleaving is not sensitive to actual relevance improvements with a single click do not exist with Optimized interleaving.*

PROOF. *The formulation of Optimized interleaving requires that the ranker that places a document higher receives more credit for that document whenever it is clicked. Hence promoting a relevant and clicked document will always be recognized by Optimized interleaving.* □

On the other hand, Probabilistic interleaving can present rankings that degrade the user experience more than Optimized interleaving. Letting $MOP(A, B)$ be the number of pairs of documents misordered between A and B,

PROPERTY 7. *If A and B agree on the order of any pair of documents, there exists a ranking R that can be shown to users by Probabilistic interleaving where $MOP(R, A) + MOP(R, B) > \max_{L \in \mathcal{L}} MOP(L, A) + MOP(L, B)$. In words, Probabilistic interleaving may show rankings that include more disagreements with both input rankers than the ranking with most disagreements shown by Optimized interleaving.*

PROOF OUTLINE. *As shown in Property 4, the number of misordered pairs in any ranking shown by Optimized interleaving is bounded by $MOP(A, B)$. Ranking A is at this bound, and $A \in \mathcal{L}$. Let a_i and a_j with $i < j$ be a pair of documents in the same order in B with smallest i and then smallest j given i. Consider the ranking R that reverses a_i and a_j in A. This ranking can be shown by Probabilistic interleaving. Moreover, $MOP(R, A) \geq 1$. Also, $MOP(R, B) \geq MOP(A, B)$ because only pairs of the form (a_i, a_k) or (a_k, a_j) with $i < k < j$ also change order when a_i and a_j are reversed. Now, it must be that case that $rank(a_k, B) < rank(a_i, B)$ because j has the smallest possible value. After the swap, R and B agree on the order of (a_i, a_k), but disagree on the order of (a_j, a_k) whereas the reverse was true previously. Hence $MOP(R, A) + MOP(R, B) \geq 1 + MOP(A, B) > MOP(A, B)$.* □

3.4.3 Existence of a Solution

It would be useful to know what further requirements on credit functions must be satisfied for there to always exist a solution to the optimization problem for any pair of input rankings A and B. It is clearly essential that for any k, there exist rankings in $L \in \mathcal{L}$ where $\Delta_k(L) = \sum_{i=1}^{k} \delta_i(L)$ are both positive and negative, or that all $\Delta_k(L)$ values are zero: Otherwise Equation 8 has no soluton. Empirically, for the Linear Rank Difference and Inverse Rank credit function examples below there is always a solution for rankings of length up to 10. However, we leave the question of general conditions as future work.

4. EXAMPLES

To obtain an interleaving algorithm, we need to specify a credit function. What form should this function take? We now present three examples that satisfy requirement (d'.1) in Section 3.2, followed by an example of how this impacts the interleaving algorithm obtained.

4.1 Example Credit Functions

Binary Preference

A particularly simple credit function would be to to give all the credit for any clicked document to the ranker that positioned this document higher:

$$\delta_i^{\text{Bin}} = \begin{cases} 1 & \text{if } rank^*(l_i, A) < rank^*(l_i, B) \\ -1 & \text{if } rank^*(l_i, A) > rank^*(l_i, B) \\ 0 & \text{otherwise} \end{cases}$$

Given this credit function, consider Equation 8, with $k = 3$ and input rankings $A = (d_1, d_2, d_3)$ and $B = (d_2, d_3, d_1)$. Under (c') there are three allowed rankings, the two original rankings plus the intermediate ranking (d_2, d_1, d_3). In all three cases a user who clicks randomly to $k = 3$ will assign two thirds of credit to input B. The equation becomes:

$$\sum_{n=1}^{|\mathcal{L}|} (p_n(1 - 1 - 1)) = 0$$

Together with Equation 7, we see that there is no valid solution for p_i. Hence this credit function would be biased and cannot be used. These input rankings are also a breaking case for Balanced interleaving. In fact, this credit function closely resembles that encoded by Balanced interleaving. Note that this credit function also violates the requirement introduced in Section 3.4.3.

Linear Rank Difference

Next consider an approach where the credit assigned to a clicked document is the additional effort that a user would have to spend to find the document in the other ranking:

$$\delta_i^{\text{Lin}} = rank^*(l_i, A) - rank^*(l_i, B), \tag{14}$$

Inverse Rank

A third way we could assign credit for clicks would be to give more weight to changes in rank at the top of a ranking. For example, we could use an inverse-rank credit scoring:

$$\delta_i^{\text{Rank}} = 1/rank^*(l_i, B) - 1/rank^*(l_i, A) \tag{15}$$

Table 1: Interleaved rankings $\mathcal{L}$ for $A = (a, b, c, d)$ and $B = (b, d, c, a)$. For two different credit functions, we show solution display probabilities p with cost constraints $\Delta_1 \ldots \Delta_4$ for each ranking, where $\Delta_j = \sum_{i=1}^{j} \delta_i$. Each allowed ranking has a total of 4 misordered pairs with respect to A and B. For comparison, we show the display probabilities for three other algorithms.

	Linear Rank Diff. Cost					Inverse Rank Cost					sensi-	Other algorithm				
Interleaved	Constraints					Constraints					tivity	ranking probabilities			Misordered Pairs	
Ranking L_i	Δ_1	Δ_2	Δ_3	Δ_4	p_i	Δ_1	Δ_2	Δ_3	Δ_4	p_i	$s(L_i)$	Bal.	TD	Prob.	$L_i \sim A$	$L_i \sim B$
(a, b, c, d)	3	2	2	0	0	3/4	1/4	1/4	0	0	0.83		25%	15.7%	0	4
(a, b, d, c)	3	2	0	0	25%	3/4	1/4	0	0	40%	0.87	50%	25%	18.0%	1	3
(b, a, c, d)	-1	2	2	0	0	-1/2	1/4	1/4	0	0	0.73		25%	11.5%	1	3
(b, a, d, c)	-1	2	0	0	35%	-1/2	1/4	0	0	35%	0.74	50%	25%	13.2%	2	2
(b, d, a, c)	-1	-3	0	0	40%	-1/2	-3/4	0	0	25%	0.60			10.8%	3	1
(b, d, c, a)	-1	-3	-3	0	0	-1/2	-3/4	-3/4	0	0	0.50			6.3%	4	0
other	*disallowed*					*disallowed*								24.3%	*average sum: 5.69*	

4.2 Illustrative Optimization Solutions

Table 1 compares the solutions produced using these credit functions with Balanced (Bal), Team Draft (TD) and Probabilistic (Prob) interleaving for one particular pair of rankings. We now walk through the example in the table.

The left column shows the allowed rankings $\mathcal{L}$ for this pair of input rankings. There are six of them. As shown above, the possible rankings produced by Team Draft and Balanced interleaving are subsets of these six. The *Other algorithm* column shows that each possible ranking for these algorithms is shown equally often. The rankings shown by Probabilistic interleaving are a superset of these six rankings. To illustrate the user impact of interleaving, the rightmost column shows the number of misordered pairs of documents in the interleaved list with respect to the input rankings A and B. While all six allowed rankings are "in between" the input rankings in terms of the total number of misordered pairs, Probabilistic interleaving shows other rankings 24.3% of the time, allowing possibilities such as showing a different document in the top position than was in either input.

Finally, the middle columns describe the optimization problem solved in more detail: The columns Δ_i each represent one constraint of the optimization problem. For instance, the columns Δ_2 show the values of $\delta_1 + \delta_2$ for each L_i. Here, we see that to ensure that in expectation there is no credit from a randomly clicking users given linear credit, the value of p_i in the top four rows must add up to 60% (thus ensuring that in expectation ranker R_A gets credit equal to $2 \times 60\%$ while ranker R_B in expectation gets identical credit in expectation, equal to $3 \times 40\%$). The column labeled $s(L_i)$ shows the sensitivity of each ranking.

5. EMPIRICAL EVALUATION

We finally turn to evaluation based on real user data. To be able to compare all the algorithms in a realistic setting without implementing and running all five at large scale, we devise a log-based approach that allows repeatable comparison of interleaving algorithms. Intuitively, we find queries where a sufficient number of distinct rankings was previously shown that we can imagine that two particular rankings were being interleaved, and take the rankings actually shown to measure the outcome of any interleaving algorithm that would have shown some subset of these rankings.

5.1 Log Collection Details

We would like to simulate the online case offline: Have two rankers providing rankings of documents, and observe these rankings interleaved using different algorithms. Hence we need a log of rankings shown to users and their responses. However, the Web is constantly changing, with new and ephemeral content. Many documents are ranked very few times, particularly lower down in query results. To get a suitable offline collection, we thus take all queries and all sets of top *four* results shown by a commerical search engine over the first six weeks of 2012. As Web search users most often click at top positions, these documents include most clicked documents. We keep all queries with at least four such distinct rankings, each shown at least 10 times, and with at least one user click on one of the top four documents each time[5]. Additionally, we require the distinct rankings not to have been the product of Web search personalization.

As interleaving involves a comparison of two rankings, for each query we must call two rankings A and B. We take the most frequent top-4 ranking as A, and the most dissimilar ranking (i.e. with a difference at the highest position) which was shown at least ten times as B. In the event of a tie in dissimilarity, the most frequent most dissimilar ranking is selected as B. For example, for the query *beef barley soup* we take $A = (d_1, d_2, d_3, d_4)$ and $B = (d1, d_3, d_2, d_4)$, where d_1 is a document on *cooks.com*, d_2 is a document on *southernfood.about.com*, d_3 is a document on *allrecipes.com* and d_4 is on *foodnetwork.com*.

Next, we further filter the logs, restricting ourselves to queries where all possible rankings produced by Team Draft interleaving were shown to users at least ten times, as well as both possible rankings produced by Balanced interleaving.

We end up with 64,251 distinct queries[6]. These can be interpreted as having run Balanced, Team Draft and Optimized interleaving by reweighting actual rankings shown to real users[7]. We can also evaluate Probabalistic interleaving over the subset of possible rankings that were shown.

Table 2 shows an example of the rankings observed for one particular query. On the left, we see which two rankings were selected as A and B. The four rankings produced by Team Draft interleaving, and two produced by Balanced interleaving, are included. For example, the row with (A) (B) in the

[5]These queries had some amount of instability in the top 4 positions due to changes in the Web, different rankers providing users with results, and potentially other reasons.

[6]As can be expected, all queries in this dataset are frequent. Further, we must assume that the intent of the queries does not change during data collection.

[7]For just one query there was no solution for the Inverse Rank credit function optimization since sufficient rankings were not shown to users. We considered this query a tie.

Table 3: Pearson correlation and directional agreement between each pair of interleaving algorithms.

Pearson correlation

	Bal.	TD.	Prob.	Opt:Linear	Opt:Inv
Balanced	-	0.94	0.90	0.93	0.84
Team Draft	0.94	-	0.92	0.93	0.91
Probabilistic	0.90	0.92	-	0.91	0.92
Opt:Delta	0.93	0.93	0.91	-	0.88
Opt:Inverse	0.84	0.91	0.92	0.88	-

Directional agreement

	Bal.	TD.	Prob.	Opt:Linear	Opt:Inv
Balanced	-	94%	87%	94%	92%
Team Draft	94%	-	88%	93%	91%
Probabilistic	87%	88%	-	88%	89%
Opt:Delta	94%	93%	88%	-	95%
Opt:Inverse	92%	91%	89%	95%	-

Team Draft column would be generated by Team Draft if R_A won the first coin toss, and R_B won the second coin toss. These rankings also include those produced by Probabilistic interleaving 63% of the time. We also see the solution produced by our optimization approach for two credit functions presented earlier. Both credit functions result in a different set of four rankings being shown to users than are selected by Team Draft interleaving, and these rankings are not shown equally often. In particular, A and B are not shown to users as they have low sensitivity (see Sec. 3.3.1).

5.2 Scoring Each Query

To compute the interleaving score of each query, we weight each observed ranking with the appropriate probability for each algorithm. We then sample the actually observed click patterns for the ranking, calculating a mean score that would have been observed per query had the query had been shown to many users (assuming the log data is representative).

5.3 Evaluation Results

We compare our optimization approach to previous interleaving algorithms, measuring agreement and sensitivity.

5.3.1 Correlation of Interleaving Algorithms

We first measure the agreement in the scores of the interleaving algorithms across queries. We would expect this to be high, as Team Draft and Balanced interleaving have been shown to both be reliable in previous online evaluations [7, 13, 1, 12]. Table 3 shows the Pearson correlation, as well as directional agreement of each pair of algorithms. As expected, both are very high.

The most interesting cases for further analysis are those where the algorithms disagree. The following four queries have the highest disagreement (most negative product of scores) between Balanced or Team Draft, and an Optimized algorithm. For each, we show the A and B rankings, which was preferred by each algorithm, and a manual assessment.

Balanced vs Optimized: Linear Rank Cost

`query` *shrimp stir fry with vegetables*
- `A` (1) cooks.com; (2) chinesefood.about.com; (3) myrecipes.com; (4) allrecipes.com
- `B` (4) allrecipes.com; (1) cooks.com; (2) chinesefood.about.com; (3) myrecipes.com;

`Preference` All algorithms prefer ranker A, except Balanced
`Notes` This is a classic case of Balanced bias, as ranking B just promotes the previously fourth result to the top
`Correct` Team Draft, Probabilistic, Opt:Linear and Opt:Inverse

Balanced vs Optimized: Inverse Cost

`query` *siberian rhubarb extract*
- `A` (1) drozfans.com; (2) herbalmenopauseremedy.com; (3) ezinearticles.com; (4) bizrate.com
- `B` (2) herbalmenopauseremedy.com; (3) ezinearticles.com; (4) bizrate.com; (4) herbalmenopausremedy.com

`Preference` All algorithms prefer ranker A, except Balanced
`Notes` Also a breaking case for Balanced, with ranking B removing the top result from ranking A.
`Correct` Team Draft, Probabilistic, Opt:Linear and Opt:Inverse

Team Draft vs Optimized: Linear Rank Cost

`query` *trutv originals*
- `A` (1) trutv.com homepage; (2) hollywoodreporter.com; (3) trutv.com/videos/originals; (4) closinglogos.com
- `B` (2) hollywoodreporter.com; (3) trutv.com/videos/originals; (1) trutv.com homepage; (4) closinglogos.com

`Preference` All algorithms except Team Draft prefer A
`Notes` This is the breaking case discussed for Team Draft in Section 2.3: Most clicks are on result (3)
`Correct` Balanced, Probabilistic, Opt:Linear and Opt:Inverse

Team Draft vs Optimized: Inverse Credit

`query` *publix.org oasis*
- `A` (1) yoomk.com; (2) publix.org; (3) publix.com; (4) publix.com (sub-page)
- `B` (2) publix.org; (3) publix.com; (4) publix.com (sub-page); (5) answers.yahoo.com

`Preference` Balanced and Team Draft prefer ranker A, the others prefer ranker B
`Notes` Most clicks are on publix.org, although a sufficient number are on yoomk.com when shown first to change the outcome for Balanced and Team Draft.
`Correct` Probabilistic, Opt:Linear and Opt:Inverse

Summary

To summarize these four most extreme cases, we see that Balanced was correct once, Team Draft twice, and Probabilistic, Optimized:Inverse and Optimized:Linear in all four.

5.3.2 Agreement with Expert Judgments

Although the previous four cases are informative, we must consider the agreement with expert judgments to have a generalized estimate of the correctness of the interleaving algorithms. We thus take all the queries in our log based set, and intersect them with a large set of previously judged queries. We find 1,664 queries in the intersection.

Table 4 shows the fraction of queries for which each interleaving algorithm agrees directionally with DCG@4, a popular judgment-based relevance metric [11], and the correlation between interleaving and DCG@4 values. Agreement and correlation of the algorithms is similar, but all the agreement values are low, in the mid-50% range. While at first surprising, note that we restrict ourselves to queries with some amount of instability, which is more likely for ambiguous queries. Easy-to-judge queries are often navigational, where the ranking produced by a commercial search engine is likely more stable. One way to bound the agreement is

Table 2: Data for one example query, *hrc*. d_1 is a webpage on http://www.hrc.army.mil/, d_2 is on http://www.hrc.org/, etc.

Label	Generated By: Team Draft	Balanced	Probabilistic	Ranking	Frequency Actually Shown	Optimization Solution: Delta Credit	Inverse Credit
R_A	(A) (A)	(A)	11.8%	d_1, d_2, d_3, d_4	>1000	-	-
	(A) (B)		9.9%	d_1, d_2, d_4, d_3	14	17%	23%
			9.9%	d_1, d_2, d_4, d_5	46	33%	27%
	(B) (A)		11.8%	d_2, d_1, d_3, d_4	19	33%	38%
	(B) (B)	(B)	9.9%	d_2, d_1, d_4, d_3	14	17%	12%
R_B			9.9%	d_2, d_1, d_4, d_5	179	-	-

Table 4: Directionary agreement and Pearson correlation between interleaving algorithms and manual judgments. The differences are not statistically significant. The correlations of Optimized:Linear and Optimized:Inverse are statistically significantly greater than zero ($p<0.01$).

Algorithm	Directional Agreement	Pearson Correlation
Balanced	52.0%	0.020
Team Draft	52.4%	0.046
Probabilistic	54.2%	0.039
Optimized:Linear	53.1%	0.078
Optimized:Inverse	53.2%	0.065
All algorithms	40.2%	
Any algorithm	63.2%	

to consider the fraction of queries where *any* of the interleaving algorithms agree directionally with DCG@4: This is 63.2% of queries. In fact, the following queries are those with highest average interleaving preference (across all five algorithms) yet disagreeing with judgments:

`tw` Judgments prefer the Wikipedia page about Taiwan. User prefer to click on the Twitter homepage.

`tea` Judgments prefer pages about the drink. Users prefer the Texas Education Agency.

`att webmail` Judgments also give credit to the main AT&T homepage, which is rarely clicked by users.

`mail` Judgments prefer mail.com, users prefer mail.yahoo.com.

`torrent` Judges assign some relevance to results about the BitTorrent protocol, while users almost only click on download sites.

`y` Judgments prefer the Wikipedia page for the letter Y, while users prefer yahoo.com.

We see that in all cases the interleaving evaluation may be correct, suggesting that the judgments used may not always reflect user needs for ambiguous queries.

5.3.3 Bias Analysis

We notice that our chosen (most frequent) ranking A is preferred for 53 to 58% of queries, ranking B is preferred for 37% to 41% of queries, with the two rankings tied for 4% to 8% (depending on the algorithm): With all five algorithms, ranking A is preferred on average. This provides a way to evaluate the bias of the different interleaving algorithms.

We summarize the queries by *pattern*, assigning a letter to each Web document such that A is always (a, b, c, d). We can then describe ranking B in terms of these letters[8]. For example, in our dataset, ranker B is (a, c, b, d) for 18,262 queries and (a, b, e, c) for 9,494 queries.

[8]The first document not returned by ranker A but returned by ranker B is assigned e, then f and so forth.

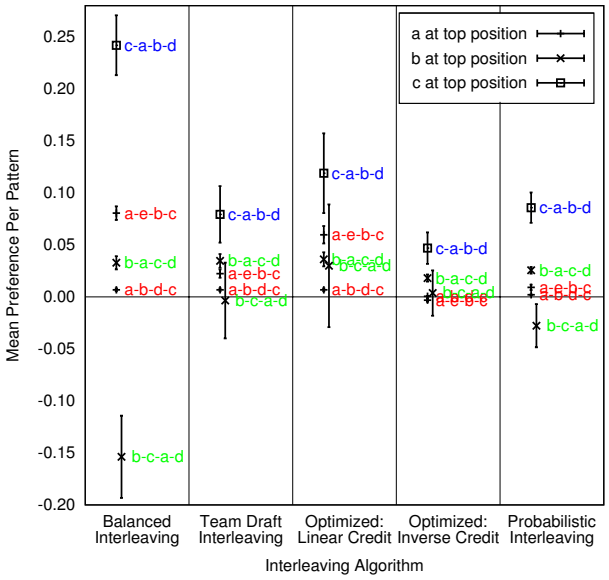

Figure 2: Per-pattern aggregate preference of different interleaving algorithms for selected patterns, showing the relative credit given to different ranking changes.

Figure 2 shows the mean interleaving preference for some example patterns. For instance, across all queries where ranker B mapped to (c, a, b, d) – i.e. the first document returned by ranker B is the same as the third document returned by ranker A, etc – Balanced interleaving generates a mean preference of 0.242. For each ranking pattern, each algorithm generates a preference in favor of ranker A (up) or ranker B (down). Because the most frequent ranking is most often best in our dataset, we expect all points to lie above zero. However, each interleaving algorithm encodes a different relative preference for each pattern, depending on the (previously implicit) credit function. For example, the bias in Balanced interleaving means that (b, c, a, d) is biased against ranker A, and (c, a, b, d) is biased for ranker A. We also see the relative magnitude of different changes. As expected, inserting an originally lower result at higher positions tends to cause a bigger preference.

5.3.4 Sensitivity Analysis

Finally, we compare the sensitivity of the different interleaving algorithms. Following [12, 4], we measure the sensitivity of each algorithm by sampling different numbers of

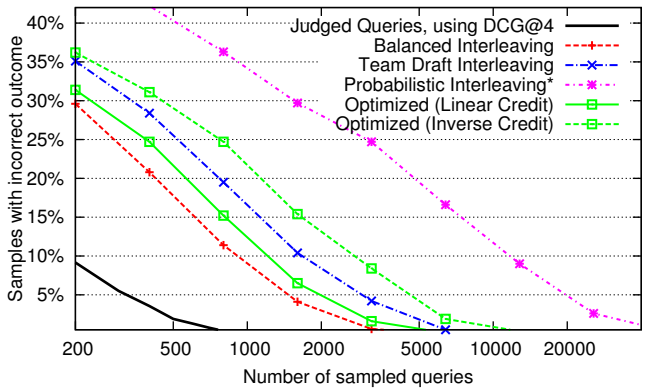

Figure 3: Fraction of bootstrapped simulations that prefer ranker A over ranker B for each interleaving algorithm, as a function of the number of queries sampled.

queries from our log[9] 10,000 times, and observing the fraction of samples for which ranker A is preferred overall. If one of the algorithms reliably prefers A with fewer impressions, it is more sensitive on this collection. Figure 3 shows that the optimized algorithms have comparable sensitivity to previous interleaving algorithms. In particular, we see that the Optimized algorithm with Linear Rank Credit is similarly sensitive to Balanced interleaving, while we showed above that it is less biased. We also note that the apparent poor sensitivity of Probabilistic interleaving may be caused by the nature of our dataset: Only a specific small fraction of the possible rankings that would have been produced by Probabilistic interleaving were present in our dataset, potentially disadvantaging this algorithm. Moreover, in opposition to other algorithms, probabalistic interleaving can make a probabilistic preference from even single-impression queries, where our evaluation technique cannot be applied.

6. CONCLUSION

In this paper we have presented a new approach to online evaluation with interleaving. We show that it is possible to postulate the requirements of interleaving by making explicit the assumptions about the value of user clicks, and solving for the rankings that should be shown to users.

Our key theoretical contribution is to show how to invert the question of obtaining interleaving algorithms: Starting with the desirable properties, we essentially solve for the algorithm. We then show that our interleaving algorithms are an improvement over previous ones. Empirically, we achieve similar correlation with judgments, and sensitivity only second to Balanced interelaving, which is known to be biased in some cases. We also avoid showing users poorer rankings than necessary while making it possible to optimize sensitivity of the algorithm directly.

Our work leaves open a number of questions. First, further theoretical analysis is necessary to show what further restrictions on the credit function optimized must exist to ensure there is always a solution to the optimization problem. Second, futher empirical evaluations by implementing this interleaving algorithm are required to confirm the sensitivity improvements. Finally, by making the credit function a parameter of the optimization framework, we leave its choice as an important parameter. A better understanding of the best credit function to use is clearly necessary.

[9]As the query set is already skewed away from a natural query distribution, queries were sampled uniformly.

Aknowledgments

We thank Martin Szummer, Emine Yilmaz, Michael Taylor and the anonymous reviewers for comments on this work.

7. REFERENCES

[1] O. Chapelle, T. Joachims, F. Radlinski, and Y. Yue. Large scale validation and analysis of interleaved search evaluation. *ACM Transactions on Information Science*, 30(1), 2012.

[2] J. He, C. Zhai, and X. Li. Evaluation of methods for relative comparison of retrieval systems based on clickthroughs. In *Proc. of CIKM*, 2009.

[3] K. Hofmann, F. Behr, and F. Radlinski. On caption bias in interleaving experiments. In *Proc. of CIKM*, 2012.

[4] K. Hofmann, S. Whiteson, and M. de Rijke. A probabilistic method for inferring preferences from clicks. In *Proc. of CIKM*, 2011.

[5] J. Huang, R. W. White, and S. Dumais. No clicks, no problem: Using cursor movements to understand and improve search. In *Proceedings of CHI*, 2011.

[6] T. Joachims. Optimizing Search Engines Using Clickthrough Data. In *Proceedings of KDD*, 2002.

[7] T. Joachims. Evaluating Retrieval Performance using Clickthrough Data. In J. Franke, G. Nakhaeizadeh, and I. Renz, editors, *Text Mining*. Physica/Springer Verlag, 2003.

[8] P. Kantor. National, language-specific evaluation sites for retrieval systems and interfaces. In *International Conference on Computer-Assisted Information Retrieval (RIAO)*, pages 139–147, 1988.

[9] D. Kelly and J. Teevan. Implicit feedback for inferring user preference: A bibliography. *SIGIR Forum*, 37(2), 2003.

[10] L. Li, W. Chu, J. Langford, and X. Wang. Unbiased offline evaluation of contextual-bandit-based news article recommendation algorithms. In *Proc. of WSDM*, 2011.

[11] C. D. Manning, P. Raghavan, and H. Schuetze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[12] F. Radlinski and N. Craswell. Comparing the sensitivity of information retrieval metrics. In *Proc. of SIGIR*, 2010.

[13] F. Radlinski, M. Kurup, and T. Joachims. How does clickthrough data reflect retrieval quality. In *Proc. of CIKM*, 2008.

[14] K. Wang, T. Walker, and Z. Zheng. PSkip: Estimating relevance ranking quality from web search clickthrough data. In *Proceedings of KDD*, 2009.

[15] Y. Yue, Y. Gao, O. Chapelle, Y. Zhang, and T. Joachims. Learning more powerful test statistics for click-based retrieval evaluation. In *SIGIR*, 2010.

[16] C. Zhai, W. W. Cohen, and J. Lafferty. Beyond independent relevance: Methods and evaluation metrics for subtopic retrieval. In *Proc. of SIGIR*, 2003.