Multileaved Comparisons for Fast Online Evaluation

Anne Schuth[1], Floor Sietsma[1], Shimon Whiteson[1],
Damien Lefortier[1,2], and Maarten de Rijke[1]

[1]University of Amsterdam, Amsterdam, The Netherlands
[2]Yandex LLC, Moscow, Russia
{anne.schuth,fsietsma,s.a.whiteson,derijke}@uva.nl, damien@yandex-team.ru

ABSTRACT

Evaluation methods for information retrieval systems come in three types: *offline evaluation*, using static data sets annotated for relevance by human judges; *user studies*, usually conducted in a lab-based setting; and *online evaluation*, using implicit signals such as clicks from actual users. For the latter, preferences between rankers are typically inferred from implicit signals via *interleaved comparison* methods, which combine a pair of rankings and display the result to the user. We propose a new approach to online evaluation called *multileaved comparisons* that is useful in the prevalent case where designers are interested in the relative performance of more than two rankers. Rather than combining only a pair of rankings, multileaved comparisons combine an arbitrary number of rankings. The resulting user clicks then give feedback about how all these rankings compare to each other. We propose two specific multileaved comparison methods. The first, called *team draft multileave*, is an extension of *team draft interleave*. The second, called *optimized multileave*, is an extension of *optimized interleave* and is designed to handle cases where a large number of rankers must be multileaved. We present experimental results that demonstrate that both team draft multileave and optimized multileave can accurately determine all pairwise preferences among a set of rankers using far less data than the interleaving methods that they extend.

Categories and Subject Descriptors

H.3 [**Information Storage and Retrieval**]: H.3.3 Information Search and Retrieval

Keywords

Information retrieval; Online evaluation; Interleaved comparisons

1. INTRODUCTION

Deployed search engines often have several teams of engineers tasked with developing potential improvements to the current production ranker. To determine whether the candidate rankers they develop are indeed improvements, such teams need experimental feedback about their performance relative to the production ranker.

CIKM'14, November 3–7, 2014, Shanghai, China.

ACM 978-1-4503-2598-1/14/11$15.00.
http://dx.doi.org/10.1145/2661829.2661952.

However, in order to develop and refine those candidate rankers in the first place, they also need more detailed feedback about how the candidate rankers compare *to each other*. For example, to explore a parameter space of interest, they may be interested in the relative performance of multiple rankers in that space.

Several existing approaches could be used to generate this feedback. Firstly, assessors could produce relevance assessments from which offline metrics (e.g., MAP, nDCG, ERR [23]) could be computed. However, offline metrics do not tell the whole story since relevance assessments come from assessors, not users. Secondly, online experiments could generate user feedback such as clicks from which rankers could be evaluated. In particular, *interleaved comparison* [15, 16] methods enable such evaluations with greater data efficiency than A/B testing [22]. But teams of engineers can easily produce enough candidate rankers that comparing all of them to each other using interleaving methods quickly becomes infeasible.

To address this difficulty, we propose a new evaluation paradigm, which we call *multileaved comparison*, that makes it possible to compare more than two rankers at once. Multileaved comparisons can provide detailed feedback about how multiple candidate rankers compare to each other using much less interaction data than would be required using interleaved comparisons.

In particular, we propose two specific implementations of multileaved comparisons. The first, which we call *team draft multileave* (TDM), builds off of *team draft* (TD) [22], an interleaving method that assigns documents in the interleaved list to a team per ranker. Surpisingly, only a minor extension to TD is necessary to enable it to perform multileaved comparisons, yielding TDM. However, despite its appealing simplicity, TDM has the important drawback that it requires multileavings, i.e., the result lists shown to the user, to be long enough to represent teams for each ranker.

Therefore, we propose a second method that we call *optimized multileave* (OM), which builds off of *optimized interleave* (OI) [21], an interleaved comparison method that uses a *prefix constraint* to restrict the allowed interleavings to those that are "in between" the two rankers and then solves an optimization problem to ensure unbiasedness and maximize sensitivity of the interleavings shown to users. OM requires deriving a new prefix constraint, new definitions of unbiasedness and sensitivity, a new credit function upon which these definitions depend, and a new sampling scheme to make optimization tractable. Because it avoids the limitations of TDM, OM is better suited to handle larger numbers of rankers.

We present experimental results on several datasets that aim to answer the following research questions.

RQ1 Can multileaved comparison methods identify preferences between rankers faster than interleaved comparison methods?

RQ2 Does OM scale better with the number of rankers than TDM?

RQ3 How does the sensitivity of multileaving methods compare to that of interleaving methods?

RQ4 Do multileaving methods improve over interleaving methods in terms of unbiasedness and online performance?

Our experimental results demonstrate that TDM and OM can accurately determine a set of pairwise preferences among a set of rankers using much less data than TD and OI, respectively.

The main contributions of this work are: (1) a novel ranker evaluation paradigm in which more than two rankers can be compared at once, (2) two implementations of this new paradigm, TDM and OM, and (3) a thorough experimental comparison of TDM and OM against each other and against TD and OI that shows that multileaved comparison methods can find preferences between rankers much faster than interleaved comparison methods. Our experiments also show that TDM outperforms OM unless the number of rankers becomes too large to handle for TDM, at which point OM performs better. Finally, our experiments show that, when the differences between evaluated rankers are varied, the sensitivity of TDM and OM is affected in the same way as for TD and OI.

2. RELATED WORK

Evaluation of information retrieval systems, i.e., rankers, has always been a central topic in IR research. Cranfield-style evaluation, as described by Cleverdon et al. [7], uses a fixed document collection, a fixed set of queries, and relevance judgments for the documents in the collection with respect to the queries. These judgements are produced by trained assessors. The relevance judgements are used to compute metrics, such as MAP, nDCG and ERR, for rankers; see [23]. We refer to this type of evaluation as *offline evaluation*, and it is still the predominant form of evaluation in, e.g., TREC-style competitions [26]. Obtaining reliable relevance judgments is typically expensive and time-consuming. However, once collected, performing repeatable experiments to compare existing rankers and try out new ones is fast and straightforward. Carterette and Allan [3] and Sanderson and Joho [24] discuss approaches to building test sets for evaluation at low cost; Azzopardi et al. [1], Berendsen et al. [2] go a step further and describe methods for automatically generating test collections and training material for learning-based rankers, respectively.

Another way to evaluate rankers is through *user studies*. Such studies are usually conducted in a lab setting [18]; as a consequence, they are expensive, hard to repeat and laborious to scale up.

In this paper, we focus on the more recent *online evaluation* paradigm, as described by Kohavi et al. [19], which relies on real users of a search engine. Online evaluation comes in several forms. One variant, called *A/B testing*, compares two rankers by showing ranker A to one group of users and ranker B to another group. Then, absolute click metrics are computed for ranker A and B, the outcome of which is used to select a winner. Carterette and Jones [4] studied the relationship between clicks coming from users and offline evaluations metrics. In particular, they were able to reliably predict nDCG from clicks.

Interleaved comparison [5, 15, 16] is a variant of online evaluation which has been shown to produce very reliable comparisons of rankers [22] using much less data then A/B testing. Interleaved comparison methods take as input two rankers and a query, and produce as output a combined result list to show to the user. The resulting clicks are then interpreted by the interleaving method to decide on a winning ranker. *Balanced interleave* (BI) [17] randomly selects a ranker to start with. Then, it takes the first document from this ranker and, alternating, each ranker contributes its next document. This document is added to the interleaving only if it was not yet present. BI can produce biased results: in comparisons of two very similar rankers, it can favor one ranker regardless of where the user clicks. This bias was subsequently fixed in *team draft* (TD) [22], which we discuss further in Section 4.1. Other methods include *document constraints* (DC) [9] and *probabilistic interleave* (PI) [11]. PI has the advantage that historical interaction data can be reused using importance sampling, for instance in an online learning to rank setting [12]. In principle, PI with importance sampling could also be used in our setting, in which multiple rankers must be compared. However, because PI relies on probabilistic rankers, it risks showing the user poor rankers that are not related to the original rankers to be interleaved, which can affect on-line performance [12]. *Optimized interleave* (OI) [21] addresses this issue by restricting the allowed interleavings to those that are the union of prefixes of the input rankings. In addition, it computes a probability distribution over these rankers that avoids bias and maximizes sensitivity. In this paper, we extend both TD and OI. Previously, TD was extended by Chuklin et al. [6] to handle non-uniform result lists that contain vertical documents such as images.

Also related is work on the *K-armed dueling bandit problem* [27]. Existing algorithms that aim at solving this problem (e.g., [28, 29]) all work by performing a series of pairwise interleaved comparisons, with a focus on finding the best ranker in a set of rankers. By contrast, in this paper, we show how multiple rankers can be compared at once and focus on the task of finding out how all rankers in a set relate to one another.

Our work differs from earlier work in that it does not rely on pairwise comparisons. As a result, when a set of rankers are evaluated, it is no longer necessary to separately compare each ranker pair. We obviate that need by introducing a new paradigm called *multileaved comparisons* that can evaluate a complete set of rankers in one comparison and thereby requires substantially less data.

3. PROBLEM DEFINITION

The problem we want to tackle can be formulated as follows: we have a set of rankers $\mathcal{R}$ whose performance we want to evaluate using click feedback. We may be interested in knowing how all rankers in $\mathcal{R}$ compare to each other, as doing so gives valuable feedback to the engineers who design new rankers. If we already have a working production ranker, we may also be interested in determining how each ranker in $\mathcal{R}$ compares to it.

In this paper, we focus on developing multileaved comparisons methods for the former task because it represents a scenario that is vital for enabling ranker development in deployed search engines. For completeness, in Section 6.6, we also evaluate our methods, designed to compare all rankers to each other, on the task variation in which they are asked to compare a set of rankers to a single production ranker.

To formalize the task of determining how all rankers in $\mathcal{R}$ compare to each other, we begin by defining ground truth as a *preference matrix* P, an $|\mathcal{R}| \times |\mathcal{R}|$ matrix in which each cell P_{ij} contains the difference in expected nDCG [14] between rankers R_i and R_j, normalized to lie between 0 and 1:

$$P_{ij} = 0.5(nDCG(R_i) - nDCG(R_j)) + 0.5,$$

where $nDCG(R_i)$ is the expected nDCG of ranker R_i across queries. The goal of an online evaluation method is then to use click feedback to learn a matrix $\hat{P}$ that approximates P. Its performance is thus measured using the error of $\hat{P}$ with respect to P. We propose a binary error metric that counts the number of times $\hat{P}$ is incorrect about which ranker has a higher expected nDCG:

$$E_{bin} = \frac{\sum_{i,j \in \mathcal{R} \wedge i \neq j} \text{sgn}(\hat{P}_{i,j} - 0.5) \neq \text{sgn}(P_{i,j} - 0.5)}{|\mathcal{R}| \cdot (|\mathcal{R}| - 1)},$$

Algorithm 1 Team draft multileave (TDM).

Require: set of rankings $\mathcal{R}$, multileaving length k.
$L \leftarrow [\,]$ *//initialize new multileaving*
$\forall R_x \in \mathcal{R} : T_x \leftarrow \emptyset$ *//initialize teams for each ranking*
while $|L| < k$ **do**
  select R_x randomly s.t. $|T_x|$ is minimized
  $p \leftarrow 0$
  while $R_x[p] \in L$ **and** $p < k - 1$ **do**
    $p \leftarrow p + 1$
  if $R_x[p] \notin L$ **then**
    $L \leftarrow L + [R_x[p]]$ *//append document to multileaving*
    $T_x \leftarrow T_x \cup \{R_x[p]\}$ *//add document to team*
return L, T

Algorithm 2 Prefix constraint sampling.

Require: set of rankings $\mathcal{R}$, multileaving length k, sample size η.
$\mathcal{L} \leftarrow \emptyset$ *//initialize empty set of multileavings*
while $|\mathcal{L}| < \eta$ **do**
  $L_i \leftarrow [\,]$ *//initialize new multileaving*
  while $|L_i| < k$ **do**
    select R_x randomly from $\mathcal{R}$
    $p \leftarrow 0$
    while $R_x[p] \in L_i$ **and** $p < k - 1$ **do**
      $p \leftarrow p + 1$
    if $R_x[p] \notin L_i$ **then**
      $L_i \leftarrow L_i + [R_x[p]]$ *//append document to multileaving*
  $\mathcal{L} \leftarrow \mathcal{L} \cup \{L_i\}$ *//add constructed multileaving to set*
return $\mathcal{L}$

where $\text{sgn}(\cdot)$ returns -1 for negative values, 1 for positive values and 0 otherwise, and the infix operator $\neq$ returns 1 whenever the signs are not equal.

4. METHOD

Using interleaving methods, learning $\hat{P}$ requires interleaving each ranker pair (R_i, R_j) separately to estimate each P_{ij}, which means that many interleavings are required for learning. The goal of multileaved comparison methods is to reduce the cost of learning by constructing *multileavings* that, by combining documents from all rankers $\mathcal{R}$, can learn about all cells in P at once.

We propose two variants of multileaved comparison: *team draft multileave* (TDM), explained in Section 4.1, and *optimized multileave* (OM), explained in Section 4.2. OM is designed to avoid a limitation of TDM on the number of rankers that it can compare using a single query.

4.1 Team draft multileave

The first variant of multileaved comparisons is based on *team draft* (TD) [22]. This interleaving method follows the analogy of selecting players (documents) for a team (ranking) for a friendly sports match. The construction of an interleaved list takes several rounds, until the interleaving is long enough. In each round, rankers select their most preferred document that is still available. It is added to their team and appended to the interleaving. The order in which rankers get to pick a document in a round is randomized. After a user interacts with documents in the interleaving, the team that owns a clicked document gets credit and the team with the most credit wins the comparison.

We propose *team draft multileave* (TDM), an extension that can compare more than two rankers at a time. Doing so is straightforward, as it only requires changing the number of teams that participate. TDM is described in Algorithm 1, which returns not only the multileaving, but also the teams to which the documents in the multileaving belong.

These team assignments are used after a user interacts with the interleaving to update the matrix $\hat{P}_{ij}$. We maintain an empirical mean for all $\hat{P}_{ij}$. We increase the preference $\hat{P}_{ij}$ if and only if there were more clicks on documents belonging to the team of ranker i than on documents belonging to the team of ranker j. Note that one reason why this may happen is that ranker j was not represented in the multileaving.

4.2 Optimized multileave

While TDM is a natural way of dealing with more than two rankers, it requires multileavings to be long enough to represent teams for each ranker. Therefore, we propose *optimized multileave* (OM), based on *optimized interleave* (OI) [21], which does not have this drawback and thus may scale better with the number of rankers.

We start in Section 4.2.1 by constructing combinations of documents from the different rankings that satisfy a generalization of the prefix constraint of [21]; this results in a set of allowed multileavings. Then we assign a probability to each of these multileavings that determines how often it is shown to users. This probability distribution over multileavings is computed by solving for the simplex and unbiasedness constraints in Section 4.2.2. Subsequently, the probability distribution over multileavings that maximizes sensitivity is selected in Section 4.2.3. When a multileaving is shown to a user, credit is assigned, according to credit functions in Section 4.2.4, to each of the original rankings based on which documents the user clicks. We explain each step in more detail in the following sections.

4.2.1 Allowed multileavings

The prefix constraint proposed in [21] states that any prefix (i.e., the top) of the constructed interleaving should be the union of prefixes of the two original rankings. We extend this to the case with more than two original rankings by defining the set of allowed multileavings $\mathcal{L}$ as follows:

$$\mathcal{L} = \{L_i : \forall k, \forall R_x \in \mathcal{R}, \exists m_x \text{ such that } L_i^k = \bigcup R_x^{m_x}\}. \quad (1)$$

Here, $\mathcal{R}$ is the set of original input rankings R_x that we want to compare, L_i^k is the top k documents of multileaving L_i, and $R_x^{m_x}$ is the top m_x documents in ranking R_x. Note that when there are only two rankings (A and B in the definition in [21]) in $\mathcal{R}$, then (1) coincides with the prefix constraint in [21].

Our constraint in (1) allows for at most $|\mathcal{R}|^{|L_i|}$ multileavings. Even with a relatively small $|\mathcal{R}|$ and $|L_i|$, this is more than can be handled by the optimization step described in the following sections. Therefore, we consider a sampling approach. Instead of materializing all multileavings allowed by (1), we construct only a small number of them using Algorithm 2. The result of this algorithm is a set $\mathcal{L}$ of multileavings that obey the prefix constraint (1) because documents from a ranking can be added to the multileaving only if all documents above it in the ranking have already been added.

The size of the set $\mathcal{L}$ of multileavings can be controlled by the parameter η. Keeping η small reduces the size of the resulting optimization problem but could introduce bias, since only a subset of allowed multileavings are considered. Besides that, due to the small number of multileavings considered, it may be the case that the optimization problem becomes overconstrained. As a result, it may no longer be possible to satisfy the unbiasedness constraint, leading to a second source of bias. We hypothesize, however, that this will not lead to severe degradation of the algorithm's performance, since ranker evaluation methods can perform well in practice even when they are biased [13].

4.2.2 Simplex and unbiasedness constraints

Every allowed multileaving $L_i \in \mathcal{L}$ is shown to the user with probability p_i. These probabilities have to satisfy a number of constraints. First of all, as in [21], they must satisfy the *simplex constraint* to form a valid probability distribution:

$$p_i \in [0, 1], \tag{2}$$

$$\sum_{i=1}^{|\mathcal{L}|} p_i = 1. \tag{3}$$

Furthermore, the multileavings satisfy the *unbiasedness constraint*: they should be shown to the user in such a way that none of the original rankings gets an unfair advantage. We instantiate this constraint by insisting that if the multileavings are presented to a randomly clicking user (according to the probability distribution), all original rankings receive the same expected credit.

In [21], a randomly clicking user is assumed to pick a number k, and clicks every result in the top k of the presented list with the same probability. When a user clicks in this way, none of the original rankings should be preferred and they should all receive the same expected credit. We adapted the resulting constraint for the multileave case. Here, given a multileaving L_i, let d_{ij} denote its j-th document and let $\delta(d_{ij}, R_x)$ be the credit assigned to ranker R_x when d_{ij} is clicked. The following constraint directly extends [21] and expresses that, for every k, there should be some constant c_k such that when the user clicks every document in the top k, every original ranking receives the same expected credit c_k:

$$\forall k, \exists c_k \text{ such that } \forall x, \sum_{i=1}^{|\mathcal{L}|} \left(p_i \sum_{j=1}^{k} \delta(d_{ij}, R_x) \right) = c_k. \tag{4}$$

4.2.3 Optimizing for sensitivity

Given the above constraints, multiple probability distributions over multileavings may still be possible, because the optimization problem may be underconstrained. Whether it is underconstrainted or overconstrained, however, depends on the number of sampled multileavings. As described in Section 4.2.1, if the number of samples is small, there might not even be a single solution to the optimization problem.

If the optimization problem is indeed underconstrained, there is the opportunity to prefer one probability distribution over multileavings over another. Following [21], we want to optimize the probabilities for maximal sensitivity. Intuitively, this means that probability distributions that distribute more mass to multileavings that can distinguish between rankers are preferred. We follow the alternative suggestion by [21], in that we minimize variance, as opposed to maximizing entropy.

The expected credit assigned to ranking R_x after the user clicks on documents in multileaving L_i is:

$$\mathbb{E}[\delta(R_x)] = \sum_{j=1}^{|L_i|} f(j) \cdot \delta(d_{ij}, R_x).$$

Here $f(j)$ is the probability with which a user clicks a document at position j. For simplicity and following [21], we assume that $f(j) = 1/j$. Given a multileaving L_i, we define the expectation over the variance in credit assigned to the different rankings as:

$$\mathbb{E}[Var_i] = \sum_{x=1}^{|\mathcal{R}|} \left(\left(\sum_{j=1}^{|L_i|} f(j) \cdot \delta(d_{ij}, R_x) \right) - \mu_i \right)^2, \tag{5}$$

$$\mu_i = \frac{1}{|\mathcal{R}|} \sum_{x=1}^{|\mathcal{R}|} \sum_{j=1}^{|L_i|} f(j) \cdot \delta(d_{ij}, R_x). \tag{6}$$

Then the aim of the optimization is to find the p_i's such that the sum of all variances is minimized:

$$\sum_{i=1}^{|\mathcal{L}|} p_i \cdot \mathbb{E}[Var_i]. \tag{7}$$

Note that we minimize the sum of all variances while taking all other constraints from Section 4.2.2 into account. In particular, if we did not ensure unbiasedness, we would find $p_i = 1$ for multileaving L_i with the lowest $\mathbb{E}[Var_i]$.

4.2.4 Assigning credit

We have not yet defined the credit function δ. This function is used in a number of places in the multileaved comparison method: (1) ensuring unbiasedness, (2) optimizing for sensitivity, and (3) determining the outcome. The credit function should assign credit to an input ranking, given a clicked document in a multileaving. However, in the optimization step, there is no observed click yet. There, we assume all documents are clicked.

Following [21], we define two possible credit functions. Intuitively, both assign more credit to rankings that rank clicked documents at a higher position. The first is *inverse rank* and analogous to the function with the same name in [21]:

$$\delta(d_{ij}, R_x) = \frac{1}{\text{rank}(d_{ij}, R_x)}. \tag{8}$$

Here, $\text{rank}(d_{ij}, R_x)$ is the rank of document d_{ij} in R_x if it is present in the ranking, and otherwise $|R_x| + 1$. Note that this is the rank in the full ranking R_x and not just the top k.

An alternative credit function is *negative rank*:

$$\delta(d_{ij}, R_x) = -\text{rank}(d_{ij}, R_x). \tag{9}$$

This credit function is analogous to the *linear rank difference* credit function from [21].[1] The difference between the credit functions in [21] and the ones defined here is that we cannot define them on a pair of rankings. Instead, our credit functions are defined as giving certain credit to a single ranking.

4.2.5 Optimized multileaved comparisons

Above, we described the ingredients of OM. Here and in Algorithm 3 we put them all together. In short, when a multileaved comparison is performed, the following happens. Each of the rankers that are to be compared generates a ranking, given the user's query. A set of multileavings is generated from these rankings using Algorithm 2. Then, a probability distribution over these multileavings is computed that obeys the unbiasedness constraints in Section 4.2.2. Following [21] we use a linear constraint optimization solver to find a distribution that satisfies these constraints. If there is more than one such distribution, we select the distribution that the minimizes variance in Section 4.2.3.[2] A single multileaving is sampled from this distribution and shown to the user who issued the query.

The user's clicks are used to assign credit to each ranker that participated in the comparison. As with TDM, we maintain an empirical mean for all $\hat{P}_{ij}$. We increase the preference $\hat{P}_{ij}$ if and only if the sum of credit for ranker i was larger than the sum of credit for ranker j.

[1] We use the term *negative rank* even when we refer to OI with *linear rank difference*.

[2] *Gurobi optimization toolkit* http://www.gurobi.com.

Algorithm 3 Optimized multileave (OM).

Require: set of rankings $\mathcal{R}$, multileaving length k, sample size η.

$\mathcal{L} \leftarrow prefix\_constraint\_sampling(\mathcal{R}, \eta)$ *// Algorithm 2*

$\mathcal{C} \leftarrow \emptyset$ *//initialize set of constraints*

$\forall L_i \in \mathcal{L} : \mathcal{C} \leftarrow \mathcal{C} \cup \{0 < p_i < 1\}$ *// add simplex constraints*

$\forall k \forall x : \mathcal{C} \leftarrow \mathcal{C} \cup \{\sum_{i=1}^{|\mathcal{L}|} \left(p_i \sum_{j=1}^{k} \delta(d_{ij}, R_x) \right) = c_k\}$ *// add unbiasedness constraints*

$\forall L_i \in \mathcal{L} : \mu_i \leftarrow \frac{1}{|\mathcal{R}|} \sum_{x=1}^{|\mathcal{R}|} \sum_{j=1}^{|L_i|} f(j) \cdot \delta(d_{ij}, R_x)$ *//compute means*

$s_i \leftarrow \sum_{x=1}^{|\mathcal{R}|} \left(\left(\sum_{j=1}^{|L_i|} f(j) \cdot \delta(d_{ij}, R_x) \right) - \mu_i \right)^2$ *// sensitivity*

$o \leftarrow \sum_{i=1}^{|\mathcal{L}|} p_i \cdot s_i$ *// optimization objective*

$p \leftarrow minimize(o, \mathcal{C})$ *// constrained optimization problem*

$L_i \leftarrow$ sample from $\mathcal{L}$ with probability p_i

return L_i

5. EXPERIMENTS

In this section, we detail our experimental setup.[3] We first describe the data sets that we use in Section 5.1. Then, in Section 5.2 we describe how we select rankers. In Section 5.3, we detail our click simulation framework, in Section 5.4 we describe our experiments, and in Section 5.5 we detail our parameter settings.

5.1 Data sets

Our experiments for RQ1, RQ2, and RQ4 are conducted on nine data sets that are distributed as LETOR 3.0 and 4.0 [20]. Each data set contains feature vectors representing the relationships between queries and documents. These feature vectors contain between 45 and 64 features. Examples of features are BM25, Language Modeling, and PageRank. Each of these features can be treated independently as rankers, by simply sorting on the feature value. While we use learning to rank data sets, we perform ranker evaluation rather than learning. The (manually assessed) relevance level of each document-query pair is also provided in the dataset. Finally, all data sets are pre-split by query for 5-fold cross validation. In the nine data sets, the following search tasks are implemented. The *OHSUMED* data set models a literature search task which is based on a query log of a search engine for the MedLine abstract database. This data set contains 106 queries that implement an informational search task. The remaining eight data sets are based on TREC Web track tasks run between 2003 and 2008. The datasets *HP2003*, *HP2004*, *NP2003*, and *NP2004* implement navigational tasks, homepage finding and named-page finding respectively. *TD2003* and *TD2004* implement an informational task: topic distillation. These last six data sets are based on the .GOV document collection, a crawl of the .gov domain, and contain between 50 and 150 queries and approximately 1000 judged documents per query. The more recent .GOV2 collection formed the basis of *MQ2007* and *MQ2008*; two data sets that contain 1700 and 800 queries respectively, but far fewer judged documents per query. The data sets *OHSUMED*, *MQ2007* and *MQ2008* are annotated with graded relevance judgments (3 grades, from 0, not relevant, to 2, highly relevant). The other data sets have binary relevance labels (grade 0 for not relevant, 2 for relevant).

5.2 Selecting rankers

For experiments aimed at answering RQ1, RQ2, and RQ4, we handpick a set of features that are known to perform well and treat each of them independently as a ranker. Among others, we select BM25, LMIR.JM, Sitemap, PageRank, HITS and TF.IDF. Most of our experiments are run with $|\mathcal{R}| = 5$ rankers; only those experiments that investigate the impact of the number of rankers use a different number of rankers. We compute nDCG [14] for each ranker to produce the ground truth P_{ij} for all ranker pairs i, j on the held-out test fold, as described in Section 3. Some average nDCG values of rankers that we use are 0.46 (BM25), 0.43 (Hyperlink based), 0.11 (PageRank), 0.50 (Sitemap), and 0.39 (LMIR.JM).

To answer RQ3, that is, to understand the impact of the difference between evaluated rankers on interleaving and multileaved comparison methods, we use synthetic data generated in a controlled way. We first generate, for each query, a ranking with 10 documents, 4 to 6 of them being relevant, using 3 grades for relevance labels as in, e.g., *OHSUMED* above. Then, we derive additional rankings by altering the initial ranking depending on the expected difference between them (see Section 6.3).

[3]Open source https://bitbucket.org/ilps/lerot.

Table 1: Instantiations of the cascade click model [8].

	$P(click = 1\|R)$			$P(stop = 1\|R)$		
	0	1	2	0	1	2
perfect	0.0	0.5	1.0	0.0	0.0	0.0
navigational	0.05	0.5	0.95	0.2	0.5	0.9
informational	0.4	0.7	0.9	0.1	0.3	0.5
random	0.5	0.5	0.5	0.0	0.0	0.0

5.3 Simulating clicks

To produce clicks, we use a click simulation framework that is analogous to [12], which is explained in [25]. The framework produces clicks based on a cascade click model also used by [10] that effectively explains the click behavior of web search users. The cascade click model explains position bias by assuming that users start examining a result from the top of the list. Then, when the user scans down the list, for each document they determine whether it looks promising enough to deserve a click. This is modeled with a click probability given some relevance label $P(click = 1|R)$. After a click, a user decides whether their information need has been satisfied with the document just clicked. We model this with a stop probability $P(stop = 1|R)$. Table 1 lists the instantiations of the click model used in our experiments. The *perfect* instantiation provides unrealistically reliable feedback, and is used to obtain an upper bound on performance. The second (*navigational*) and third (*informational*) instantiations reflect two types of search task also implemented by our data sets, as well as increasing levels of noise (i.e., smaller differences in click probabilities for different relevance levels). For each instantiation, the table provides the click and stop probabilities given a relevance grade R. For example, under the navigational model, simulated users would be very likely to click on a highly relevant document ($P(click = 1|2) = 0.95$), and very likely to stop examining documents once they clicked on such a document ($P(stop = 1|2) = 0.9$). Under the informational model, users are less likely to stop, and click probabilities for the different relevance grades are much more similar, resulting in a higher level of noise. The *random* instantiation of the click model is used to examine behavior of the evaluation methods when no information is present in the clicks. For data sets with binary relevance judgments, only the two extremes are used.

5.4 Experimental runs

Our experiments consider the following evaluation methods.

TD teamdraft interleave [22], pairwise comparisons (baseline).

TDM teamdraft multileave, our extension of TD that performs multileaved comparisons.

OI optimized interleave [21], pairwise comparisons (baseline).

OM optimized multileave, our extension of OI that performs multileaved comparisons. We experiment with several sample sizes η and the credit function.

Our experiments for RQ1, RQ2, and RQ4 are performed as follows. We select a set of rankers that to compare. We then repeatedly sample queries randomly with replacement from the pool of queries. This simulates a user arriving at our search engine and entering a query. We assume that there is no dependence between two consecutive queries. When a query has been selected, it is given to the online evaluation methods. For the pairwise (baseline) methods, we select a pair of rankers such that all pairs of ranker pairs i, j where $i \neq j$ are compared the same number of times. The multileaved comparison methods, on the other hand, compare all rankers at the same time. So, either an interleaving of two rankers or a multileaving of all rankers is shown to the user. We then simulate the user interacting with the result list and produce clicks according to the given instantiation of the click model. Using these clicks, for the pairwise (baseline) methods, we update $\hat{P}_{ij}$ only for the pair of rankers that we compared. For the multileaved comparison methods, we update all $\hat{P}_{ij}$ for all pairs of rankers. For RQ3, we follow the above approach as closely as possible. However, since there is no notion of a *ranker* that generalizes over queries, we repeatedly ($N = 100$) issue the same set of rankings to produce clicks with the click model in order to obtain a stable $\hat{P}_{ij}$.

The main objective for all experiments is to find the $\hat{P}_{ij}$ that minimizes the error metric E_{bin} when compared to ground truth P_{ij} computed using nDCG (see Section 3). We also investigate other properties. We measure the *bias* of each method by using a random instantiation of the click model and comparing with E_{bin} to a ground truth where $P_{ij} = 0.5$ for all pairs of rankers. We also measure online performance in terms of nDCG of the rankings presented to the user. Lastly, we measure the effect of the number of rankers we compare and the effect of the length of the result list. We test for significant differences using a two tailed t-test.

5.5 Parameter settings

For OM, we set the number of multileavings to $\eta = 1, 5, 10, 100$. For both OI and OM we test two types of credit function: *negative credit* and *inverse credit*. For OM, we use *inverse credit* by default and for OI we use *negative credit* unless stated otherwise as these performed best for the respective methods. For all experiments except those that investigate the effects of these parameters, the number of rankers is $|\mathcal{R}| = 5$ and the results lists length is $k = 10$.

6. RESULTS AND ANALYSIS

Here, we answer the research questions posed in Section 1.

6.1 Main result

Our main result is depicted in Fig. 1. It shows the error measured with E_{bin} for the two baseline interleaving methods OI and TD and for our two multileaving methods OM and TDM. These results are obtained by aggregating over all the datasets that we consider. Table 2 provides an alternative view on the same results by splitting them per dataset. We performed our analysis for three levels of increasing noise in the feedback: *perfect*, *navigational* and *informational* instantiations of the click model.

Interestingly, as can be seen in Fig. 1, the multileaved extensions of the interleaving methods converge to an error close to their interleaving counterparts. Both OI and OM have difficulties coping with noise in user feedback: the error to which these methods converge increases when the noise increases. This is in contrast with TD and TDM: with increasing noise they are capable of learning the ranker preference almost as well as with the *perfect* click model.

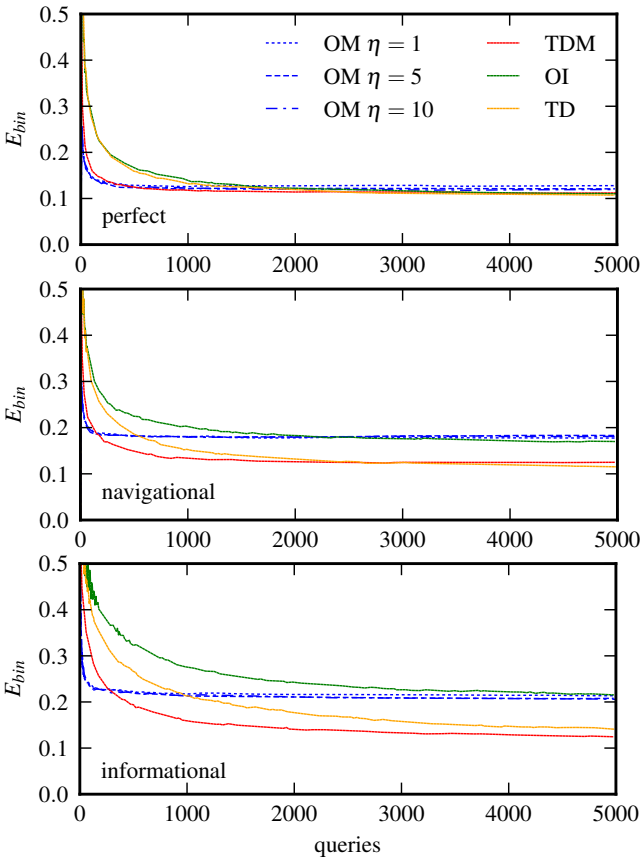

Figure 1: Average E_{bin} error of interleaved and multileaved comparisons. Averaged over 25 repetitions, 9 datasets with 5 folds each. The plots depict error for three instantiations of the click model: perfect, navigational and informational. Result list length $l = 10$ and number of rankers $|\mathcal{R}| = 5$.

In response to RQ1, Fig. 1 shows clearly that the error of both of our multileaving methods drops much faster than their interleaving counterparts. This indicates that multileaved comparison methods can learn preferences between multiple rankers with far less data (i.e., queries and clicks) than interleaved comparison methods.

Under perfect feedback, TDM and OM learn ranker preferences equally fast. When noise increases, OM initially learns these preferences faster than TDM does. Under noisy feedback, TDM keeps improving the learned preferences long after OM has plateaued.

Table 2 shows the error E_{bin} at 500 queries. We choose a rather low number of queries to emphasize learning speed. Note that the rightmost column is equal to the E_{bin} values in a slice of Fig. 1 after 500 queries. For the multileaving methods, each $\hat{P}_{ij}$ has had 500 updates by then. The interleaving methods only performed 50 updates of $\hat{P}_{ij}$ for each pair of rankers. The results show that, in general, the multileaving methods have significantly less error than the interleaving methods. In particular, OM has less error than OI does in 24 out 27 experiments. The exception to this rule are the three experiments on MQ2007. TDM has less error than TD in 22 out of 27 experiments. In two experiments, TDM has a significantly higher error; those experiments are on perfect and navigational instantiations of the click model on the TD2003 data set. In both these exceptions convergence was reached far before 500 queries for all methods. While the multileaving methods still converged faster, they did so to a slightly higher error.

For OM we see in both Table 2 and Fig. 1 that η, the sample size, does not seem have a large effect on the error. Therefore, with a

Table 2: E_{bin} at 500 queries. Averaged over 25 repetitions and 5 folds. Standard deviation is between brackets. Per data set and instantiation of the click model, we print the best method in bold. Statistically significant improvements (losses) over the respective baselines are indicated by $^{\triangle}$ ($p < 0.05$) and $^{\blacktriangle}$ ($p < 0.01$) ($^{\triangledown}$ and $^{\blacktriangledown}$).

Method	HP2003	HP2004	MQ2007	MQ2008	NP2003	NP2004	OHSUMED	TD2003	TD2004	Average
perfect click model										
OM $\eta = 1$	**0.000** (0.00) $^{\blacktriangle}$	**0.000** (0.00) $^{\blacktriangle}$	0.324 (0.13) $^{\blacktriangledown}$	0.043 (0.05) $^{\blacktriangle}$	0.067 (0.07) $^{\blacktriangle}$	0.088 (0.09) $^{\triangle}$	0.340 (0.15) $^{\blacktriangle}$	0.194 (0.13) $^{\blacktriangle}$	**0.096** (0.07) $^{\blacktriangle}$	0.128 (0.15)
OM $\eta = 5$	**0.000** (0.00) $^{\blacktriangle}$	**0.000** (0.00) $^{\blacktriangle}$	0.285 (0.11) $^{\triangledown}$	**0.040** (0.05) $^{\blacktriangle}$	0.068 (0.07) $^{\blacktriangle}$	**0.084** (0.08) $^{\blacktriangle}$	0.341 (0.14) $^{\blacktriangle}$	0.195 (0.13) $^{\blacktriangle}$	0.106 (0.08) $^{\triangle}$	0.124 (0.15) $^{\triangle}$
OM $\eta = 10$	**0.000** (0.00) $^{\blacktriangle}$	**0.000** (0.00) $^{\blacktriangle}$	0.297 (0.11) $^{\blacktriangledown}$	0.049 (0.06) $^{\blacktriangle}$	0.071 (0.07) $^{\blacktriangle}$	0.090 (0.10) $^{\triangle}$	0.338 (0.14) $^{\blacktriangle}$	**0.186** (0.12) $^{\blacktriangle}$	0.107 (0.08) $^{\triangle}$	0.126 (0.15) $^{\triangle}$
OM $\eta = 100$	**0.000** (0.00) $^{\blacktriangle}$	**0.000** (0.00) $^{\blacktriangle}$	0.233 (0.11)	0.049 (0.05) $^{\blacktriangle}$	**0.067** (0.06) $^{\blacktriangle}$	0.093 (0.10)	0.338 (0.15) $^{\blacktriangle}$	0.193 (0.12) $^{\blacktriangle}$	0.114 (0.08)	0.136 (0.15)
OI	0.014 (0.04)	0.035 (0.05)	0.254 (0.12)	0.155 (0.10)	0.111 (0.09)	0.116 (0.09)	0.440 (0.16)	0.243 (0.12)	0.131 (0.10)	0.167 (0.16)
TDM	0.005 (0.02)	0.018 (0.04)	**0.166** (0.08) $^{\blacktriangle}$	0.050 (0.07) $^{\blacktriangle}$	0.086 (0.06) $^{\blacktriangle}$	0.097 (0.10)	**0.265** (0.19) $^{\blacktriangle}$	0.270 (0.10) $^{\triangledown}$	0.159 (0.07)	**0.124** (0.13)
TD	0.007 (0.03)	0.024 (0.04)	0.305 (0.13)	0.134 (0.09)	0.114 (0.09)	0.122 (0.10)	0.350 (0.16)	0.235 (0.12)	0.143 (0.10)	0.159 (0.15)
navigational click model										
OM $\eta = 1$	**0.005** (0.02) $^{\blacktriangle}$	**0.006** (0.02) $^{\blacktriangle}$	0.530 (0.10) $^{\blacktriangledown}$	0.138 (0.09) $^{\blacktriangle}$	**0.062** (0.05) $^{\blacktriangle}$	**0.080** (0.07) $^{\blacktriangle}$	0.430 (0.11) $^{\blacktriangle}$	0.228 (0.15) $^{\triangle}$	0.159 (0.12)	0.182 (0.20)
OM $\eta = 5$	0.011 (0.03) $^{\triangle}$	0.011 (0.03) $^{\blacktriangle}$	0.518 (0.12) $^{\blacktriangledown}$	0.132 (0.09) $^{\blacktriangle}$	0.068 (0.06) $^{\blacktriangle}$	**0.080** (0.07) $^{\blacktriangle}$	0.438 (0.12) $^{\blacktriangle}$	**0.227** (0.15) $^{\triangle}$	**0.154** (0.11)	0.182 (0.20)
OM $\eta = 10$	0.018 (0.04)	0.012 (0.03) $^{\blacktriangle}$	0.503 (0.10) $^{\blacktriangledown}$	0.134 (0.10) $^{\blacktriangle}$	0.063 (0.06) $^{\blacktriangle}$	0.081 (0.08) $^{\blacktriangle}$	0.419 (0.13) $^{\blacktriangle}$	0.237 (0.14)	0.158 (0.11)	0.181 (0.19)
OM $\eta = 100$	0.013 (0.03)	0.019 (0.04) $^{\blacktriangle}$	0.462 (0.11) $^{\blacktriangledown}$	0.137 (0.09) $^{\blacktriangle}$	0.068 (0.06) $^{\blacktriangle}$	0.082 (0.08) $^{\blacktriangle}$	0.436 (0.13) $^{\blacktriangle}$	0.253 (0.16)	0.167 (0.10)	0.211 (0.20)
OI	0.022 (0.04)	0.044 (0.06)	0.398 (0.14)	0.229 (0.12)	0.116 (0.08)	0.137 (0.09)	0.635 (0.16)	0.269 (0.12)	0.166 (0.10)	0.224 (0.21)
TDM	0.021 (0.04)	0.026 (0.04)	**0.190** (0.09) $^{\blacktriangle}$	**0.082** (0.08) $^{\blacktriangle}$	0.086 (0.07) $^{\blacktriangle}$	0.106 (0.10) $^{\blacktriangle}$	**0.330** (0.17) $^{\blacktriangle}$	0.308 (0.14) $^{\triangledown}$	0.188 (0.08)	**0.149** (0.15)
TD	0.017 (0.04)	0.030 (0.05)	0.322 (0.14)	0.198 (0.11)	0.126 (0.09)	0.154 (0.10)	0.386 (0.16)	0.272 (0.14)	0.169 (0.10)	0.186 (0.16)
informational click model										
OM $\eta = 1$	0.089 (0.03) $^{\blacktriangle}$	0.071 (0.05) $^{\blacktriangle}$	0.635 (0.12) $^{\blacktriangledown}$	**0.169** (0.10) $^{\blacktriangle}$	**0.064** (0.05) $^{\blacktriangle}$	**0.083** (0.07) $^{\blacktriangle}$	0.397 (0.09) $^{\blacktriangle}$	0.289 (0.17) $^{\blacktriangle}$	0.213 (0.10) $^{\blacktriangle}$	0.223 (0.20) $^{\blacktriangle}$
OM $\eta = 5$	0.095 (0.02) $^{\blacktriangle}$	0.081 (0.05) $^{\blacktriangle}$	0.602 (0.13) $^{\blacktriangledown}$	0.170 (0.12) $^{\blacktriangle}$	0.070 (0.06) $^{\blacktriangle}$	0.090 (0.07) $^{\blacktriangle}$	0.383 (0.10) $^{\blacktriangle}$	0.289 (0.16) $^{\blacktriangle}$	0.199 (0.10) $^{\blacktriangle}$	0.220 (0.20) $^{\blacktriangle}$
OM $\eta = 10$	0.096 (0.02) $^{\blacktriangle}$	0.087 (0.04) $^{\blacktriangle}$	0.583 (0.15) $^{\blacktriangledown}$	0.186 (0.11) $^{\blacktriangle}$	0.072 (0.05) $^{\blacktriangle}$	0.086 (0.07) $^{\blacktriangle}$	0.380 (0.11) $^{\blacktriangle}$	0.294 (0.16) $^{\blacktriangle}$	0.199 (0.09) $^{\blacktriangle}$	0.220 (0.19) $^{\blacktriangle}$
OM $\eta = 100$	0.100 (0.00) $^{\blacktriangle}$	0.101 (0.03) $^{\blacktriangle}$	0.518 (0.13) $^{\blacktriangledown}$	0.178 (0.11) $^{\blacktriangle}$	0.080 (0.06) $^{\blacktriangle}$	0.095 (0.07) $^{\blacktriangle}$	0.393 (0.11) $^{\blacktriangle}$	**0.282** (0.15) $^{\blacktriangle}$	0.200 (0.09) $^{\blacktriangle}$	0.243 (0.18) $^{\blacktriangle}$
OI	0.202 (0.12)	0.198 (0.13)	0.421 (0.15)	0.318 (0.14)	0.186 (0.11)	0.246 (0.11)	0.674 (0.14)	0.382 (0.13)	0.280 (0.13)	0.323 (0.20)
TDM	**0.060** (0.07) $^{\blacktriangle}$	**0.066** (0.06) $^{\blacktriangle}$	**0.276** (0.16) $^{\blacktriangle}$	0.177 (0.12) $^{\blacktriangle}$	0.139 (0.10) $^{\blacktriangle}$	0.147 (0.09) $^{\blacktriangle}$	**0.366** (0.19) $^{\blacktriangle}$	0.317 (0.13)	**0.198** (0.11) $^{\blacktriangle}$	**0.194** (0.16) $^{\blacktriangle}$
TD	0.120 (0.10)	0.131 (0.09)	0.419 (0.15)	0.307 (0.14)	0.190 (0.10)	0.207 (0.11)	0.438 (0.17)	0.352 (0.16)	0.242 (0.14)	0.267 (0.17)

Table 3: E_{bin} when the number of rankers $|\mathcal{R}|$ is varied. Result list length $k = 10$, averaged over 10 repetitions and 5 folds of the NP2003 data set.

Method	$\|\mathcal{R}\| = 3$	$\|\mathcal{R}\| = 5$	$\|\mathcal{R}\| = 7$	$\|\mathcal{R}\| = 10$
OM $\eta = 10$	0.144 (0.16)	0.154 (0.12)	0.111 (0.06)	0.116 (0.04)
TDM	0.191 (0.18)	0.192 (0.09)	0.190 (0.06)	0.203 (0.05)
OI	0.189 (0.18)	0.200 (0.08)	0.255 (0.06)	0.316 (0.04)
TD	0.143 (0.13)	0.214 (0.09)	0.246 (0.05)	0.284 (0.04)

surprisingly small number of samples, effective and computationally efficient multileaving is possible. Consequently, in most of the analyses that follow, we report only on OM with $\eta = 10$.

6.2 Scaling the number of rankers

The motivation for performing *multileaved* comparisons lies in the fact that it is possible to compare multiple rankers at once. Most of our experiments in this paper use a set of 5 rankers but, in response to RQ2, in this section we analyze what happens when the number of rankers being compared increases.

Table 3 lists how each method performs when the number of rankers to be compared varies. We kept the result list length fixed at $k = 10$. Both interleaving methods OI and TD are impacted greatly when the number of rankers increases. This is largely due to the fact that many more comparisons are needed and as such each $\hat{P}_{ij}$ receives fewer updates. By contrast, OM and TDM do not show significant degradation when the number of rankers increases.

We suspect that there may be an interaction between the number of rankers that are compared and the length of the result list shown

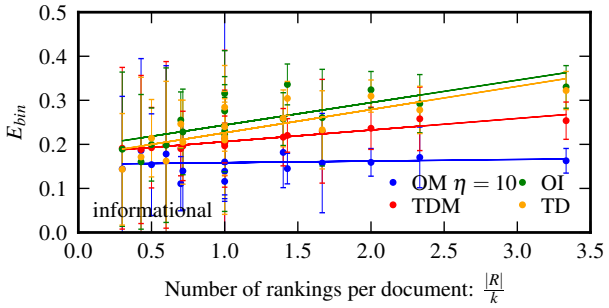

Figure 2: Scaling with the number of rankers. Average E_{bin} against the number of rankers x per result list length k. Computed on all combinations of $|\mathcal{R}| = 3, 5, 7, 10$ and $k = 3, 5, 7, 10$. Averaged over 10 repetitions and 5 folds of the NP2003 data set. Standard deviation is indicated with error bars and lines are fitted using least squares.

to the user. Depending on the method, the result list length may limit the number of rankers that can be represented at once. We experimented with several settings where we varied the number of rankers to be compared and the result list length. We considered all combinations of $|\mathcal{R}| = 3, 5, 7, 10$ rankers and lengths $k = 3, 5, 7, 10$. Because of computational limitations, we had to limit ourselves to a single data set, a single user model, with fewer repetitions and fewer queries. We selected the NP2003 dataset with the *informational* instantiation of the click model with 10 repetitions and 2.5K queries.

In Fig. 2, we plot the error E_{bin} against the number of rankers per documents in the result list. The four rightmost data points, for

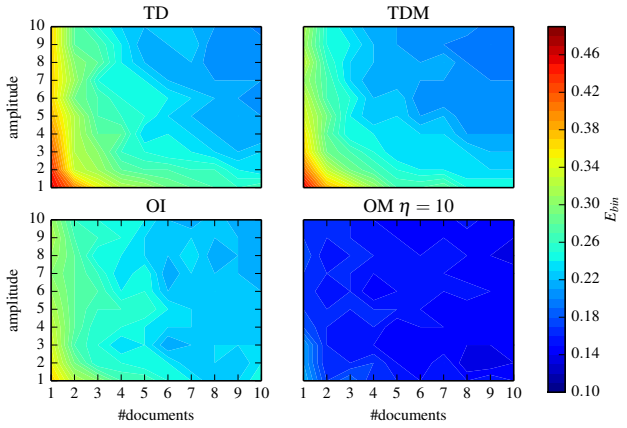

Figure 3: The effect of differences between rankings, the number of moved documents and the amplitude of the move is controlled. E_{bin} at 500 queries, 100 issues, averaged over 125 repetitions. We used the informational click model.

instance, were produced using 10 rankers and result lists of length 3 only. The leftmost points are from the opposite scenario: 3 rankers were compared with document lists of length 10. Note that there are many ways in which $\frac{|\mathcal{R}|}{k}$ can be equal to 1, and that therefore there is a relatively wide spread of error.

We fitted lines for each evaluation method using least squares. Though these lines are not perfect fits, they give a useful indication of the behavior of the methods when the ratio between the number of rankers and the number of documents increases. Fig. 2 shows that the multileaving methods can cope better with an increase in this ratio than the interleaving baselines. The performance of OM is not impacted by an increase of this ratio; the two interleaving methods almost double their error when the ratio increases from $\frac{3}{10}$ to $\frac{10}{3}$.

While Table 3 shows that TDM is not impacted by the number of rankers, in Fig. 2, we see that the error for TDM does increase when the ratio of rankers per result list length goes up. We attribute this to the fact that team draft methods always assign a document in an interleaving to a single input ranker. When there are (many) more rankers than documents to which they can be assigned, then most rankers cannot be distinguished from one another. Consequently, not all $\hat{P}_{ij}$ can be updated per comparison.

6.3 Sensitivity

In this section, we investigate RQ3. We study the impact of the difference between evaluated rankers on interleaving and multileaved comparison methods using synthetic data as discussed in Section 5.2. We consider cases when the position of one or more document(s) changes from one ranking to another (we also investigated cases when one or more document(s) are replaced by new ones and obtained similar results). In doing so, we control two things: the *number* of documents moved as well as the *amplitude* of the move, i.e., how far away is the moved document located from its original position. While we only control the difference w.r.t. a single ranking and not between all pairs of rankings, by increasing the number and amplitude of the changes, we increase the space of possible rankings, effectively increasing the chance of them being different from each other.

For each interleaving and multileaved comparison method, we look at the impact on E_{bin} at 500 queries of the difference between rankings using the *informational* click model, with $|\mathcal{R}| = 5$ rankers, result lists of length $k = 10$ and 100 issues of each query. Results are depicted in Fig. 3 as a heat map of E_{bin} depending on the number of documents moved and the amplitude of the move. We

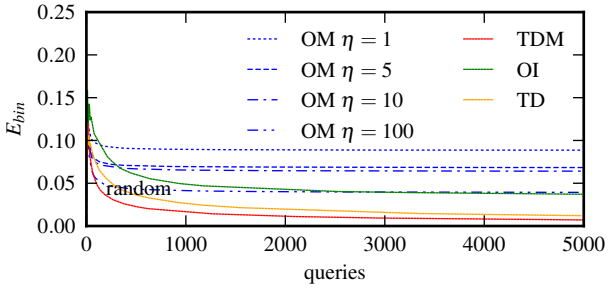

Figure 4: Incorrectly identified preferences under a random click model, with $|\mathcal{R}| = 5$ rankers and result list of length $k = 10$. Measured as E_{bin} versus a ground truth with no preferences, $P_{ij} = 0.5$ for all i, j. Averaged over 25 repetitions, 9 dataset with each 5 folds.

observe that E_{bin} decreases as the difference between rankings increases (whether this is the number of moves or the amplitude of the moves) in the same way for all methods, which means that differences between rankers affect all methods in the same way. We also observe that OM performs much better than other methods, which is in line with Fig. 1 at the 100 query issue point.

Returning to RQ3, these results show that the sensitivity of multileaving methods is affected in the same way as for interleaving methods when the differences between rankers vary. Interestingly, this means that multileaved methods can distinguish between rankers just as well as interleaving methods even when the differences between them is very small. Hence, multileaved comparison methods can be used to explore a parameter space using very small steps.

6.4 Bias

Next, we address RQ4. We evaluate fidelity requirement (2) from [13] which states that, under random clicks, rankers should tie in expectation. TD was designed to fulfill this requirement. We run experiments with the *random* instantiation of the click model (see Section 5.3). When a user clicks on a result list without any preference for relevant documents, an online evaluation method that interprets these clicks should not detect any preferences among rankers. We measure how many preferences each comparison method detects when exposed to a random user by comparing the $\hat{P}_{ij}$ of the method to a ground truth that consists of $P_{ij} = 0.5$ for all i, j using E_{bin}.

The result is shown in Fig. 4. For all methods, the error quickly drops to rather low values. Both TD and TDM steadily converge to values near 0. Within a few hundred queries, their error is below 5%. In the long run, neither method detects differences among rankers when it should not. OI takes much longer to drop below 5% and plateaus higher than both team draft methods. For OM, it turns out that the number of multileavings that is sampled, η (see Section 4.2.1) has a big impact on the bias of the method. The larger the sample size, the less bias the OM method has. A more elaborate explanation of this effect can be found in Section 6.7. It may come as a surprise that both OI and OM have such a large bias since both these methods explicitly restrict themselves to producing unbiased result lists. The fact that the error increases when η goes up (see Table 2) can be explained by a bias-variance trade-off: when η goes up, the bias goes down at the cost of variance that is introduced.

6.5 Online performance

A general concern with online ranker evaluation is that users may be confronted with inferior systems. The degree to which this happens may vary per evaluation method. Again, in response to RQ4, we measure online performance of the four evaluation methods using

Table 4: Online performance measured with nDCG (higher is better). Averaged over 5K queries, 25 repetitions and 5 folds. Standard deviation is between brackets. Per data set, we print the best value in bold.

Method	HP2003	HP2004	MQ2007	MQ2008	NP2003	NP2004	OHSUMED	TD2003	TD2004	total
OM $\eta = 1$	0.522 (0.01)	0.465 (0.01)	0.289 (0.01)	0.377 (0.02)	0.500 (0.02)	0.445 (0.03)	0.396 (0.02)	0.183 (0.03)	0.180 (0.01)	0.373 (0.12)
OM $\eta = 5$	0.491 (0.01)	0.430 (0.01)	0.289 (0.00)	0.374 (0.02)	0.463 (0.02)	0.410 (0.02)	0.396 (0.02)	0.174 (0.02)	0.172 (0.01)	0.355 (0.11)
OM $\eta = 10$	0.486 (0.01)	0.425 (0.01)	0.289 (0.00)	0.373 (0.02)	0.460 (0.02)	0.407 (0.02)	0.394 (0.02)	0.173 (0.02)	0.170 (0.01)	0.353 (0.11)
TDM	0.536 (0.01)	**0.476** (0.01)	0.288 (0.01)	0.376 (0.02)	**0.513** (0.02)	**0.457** (0.03)	**0.398** (0.02)	**0.196** (0.03)	0.184 (0.01)	**0.380** (0.13)
OI	**0.539** (0.01)	0.476 (0.01)	**0.297** (0.00)	**0.383** (0.02)	0.506 (0.02)	0.432 (0.03)	0.388 (0.02)	0.173 (0.03)	**0.188** (0.01)	0.376 (0.13)
TD	0.493 (0.01)	0.446 (0.01)	0.293 (0.00)	0.380 (0.02)	0.482 (0.02)	0.419 (0.03)	0.394 (0.02)	0.166 (0.02)	0.175 (0.01)	0.361 (0.12)

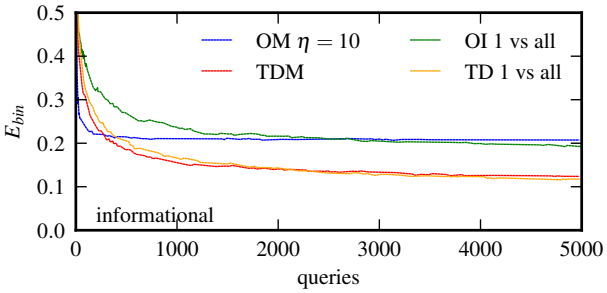

Figure 5: One rankers versus many rankers, measured with E_{bin} of $\hat{P}_{ij}$ against P_{ij} where we keep i fixed. Averaged over 25 repetitions, 9 data sets and 5 folds.

nDCG [14]. Table 4 lists the nDCG for each evaluation method measured on the result list that was actually shown to the user. On average, TDM produces the highest online performance, i.e., users were the least affected by the evaluation in which they participated.

Interestingly, for OM, the nDCG score goes down when the sample size η goes up. This may be due to the fact that, when the number of sampled multileavings goes up, the optimization problem is less overconstrained. As a consequence, it is easier to satisfy the unbiasedness constraint. Less biased multileavings are more "in between" the input rankings and therefore they do not represent a strong preference for one ranker. Such multileavings turn out to have a lower nDCG. TDM does not suffer from this problem. On some data sets, in particular HP2003, HP2004, NP2003 and NP2004, for OM the online performance drops considerably when η goes up. Incidentally, on these data sets, the error also increases when η goes up (see Table 2); less biased multileavings have a lower online performance.

6.6 Comparing to a production ranker

Though we focus on efficiently comparing all rankers to each other, other variants are also useful in practice, as detailed in Section 3. Here, we investigate how online evaluation methods perform on one such variant: comparing a set of rankers to a single benchmark, e.g., a production ranker. Though our multileaving methods were not specifically designed for this variant, we can measure their performance on it by computing the error E_{bin} of $\hat{P}_{ij}$ against P_{ij} where we keep i fixed. We perform this experiment on the *informational* instantiation of the click model and we average over 25 repetitions, 9 data sets and 5 folds.

Fig. 5, which presents the result of this analysis, shows that multileaving methods outperform the interleaving methods. OM, in particular, continues to learn much more quickly than the alternatives. Unsurprisingly, when comparing Fig. 5 to Fig. 1, we see that the advantage of multileaving methods over interleaving methods diminishes when the task changes from learning all cells in $\hat{P}$ to learning just one row and column in $\hat{P}$. Note that the multileaving methods do still learn all cells in $\hat{P}$.

Table 5: Overconstrainedness of OM $\eta = 10$ averaged over 10 repetitions and 5 folds of the NP2003 data set.

k	$\lvert\mathcal{R}\rvert = 3$	$\lvert\mathcal{R}\rvert = 5$	$\lvert\mathcal{R}\rvert = 7$	$\lvert\mathcal{R}\rvert = 10$
3	0.393 (0.21)	0.976 (0.03)	0.999 (0.00)	1.000 (0.00)
5	0.934 (0.18)	0.996 (0.01)	0.999 (0.00)	1.000 (0.00)
7	0.984 (0.05)	0.997 (0.00)	0.999 (0.00)	1.000 (0.00)
10	0.995 (0.01)	0.998 (0.00)	1.000 (0.00)	1.000 (0.00)

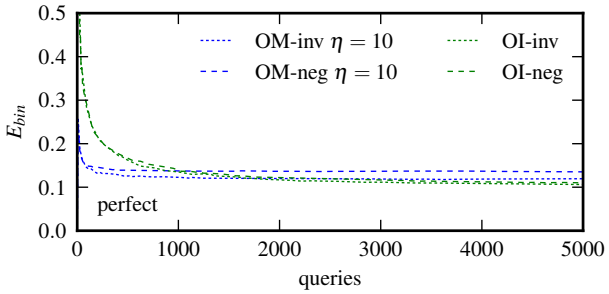

Figure 6: Impact of *negative* and *inverse* credit functions (see Section 4.2.4) in OI and OM on the perfect click model. Averaged over 25 repetitions, 9 dataset with 5 folds each.

6.7 Parameters of OM and OI

In this section, we investigate some of the design choices made when extending OI to OM; where possible, we do so by comparing to the impact of our same choices on OI.

As described in Section 4.2.1, we had to restrict the number of multileavings we can consider in the optimization problem of OM. As we saw in Section 6.4 and to a lesser extent in Section 6.1, the number of sampled multileavings η does have an impact on the performance of OM. We hypothesized that this is due to the optimization problem of OM becoming overconstrained when the number of multileavings is small. When we investigate this effect, we find the following. For smaller sample sizes, $\eta = 1, 5, 10$, the problem was almost always overconstrained on all of the nine data sets. With $\eta = 100$, the problem was overconstrained in 85% of the multileaved comparisons. For OI, we confirm the claim by Radlinski and Craswell [21] that the optimization problem is usually underconstrained: we found that the problem was overconstrained in only 1% of the interleavings.

The above findings were all for the scenario with $\lvert\mathcal{R}\rvert = 5$ rankers and $k = 10$ documents in the result lists. In Table 5, we see what happens when we vary $\lvert\mathcal{R}\rvert$ and k and keep $\eta = 10$. Computational limitations prevented us from evaluating what would happen with values larger than $\eta = 100$. As long as the number of rankers is small and the length of the multileaving is short, a small number of samples is enough to avoid having an overconstrained problem.

In Fig. 6, we analyze the impact of the credit function (see Section 4.2.4) on OI and OM. We see that OM performs best when using

the *inverse credit function* while the effect of the credit function on OI is smaller than on OM. The observed degraded performance of the negative credit function for OM is explained by the fact that this credit function assumes a linear relation between the rank and credit. This effect is stronger in OM because the credit function does not model the difference but rather absolute values.

7. CONCLUSION

We presented a new paradigm for online evaluation of information retrieval systems. We have shown that it is possible to extend interleaved comparison methods to variants that, instead of comparing two rankers, compare multiple rankers at a time. We introduced two implementations of this paradigm that extend state-of-the-art interleaving methods to their multileaving counterparts. One is *team draft multileave* (TDM) and is an extension of *team draft*. The second is *optimized multileave* (OM) and extends *optimized interleave*. We have shown in extensive experiments that both multileaving methods have their merits. OM learns preferences between rankers very quickly while TDM learns them slightly more slowly, though faster than either of the interleaving methods. However, TDM learns more accurate preferences in the long run than OM or either of the interleaving methods to which we compare. On the other hand, OM scales much better than TDM when the number of rankers increases. Thus, depending on the number of rankers to be compared, one might prefer one multileaving algorithm over the other but both should be preferred over interleaving algorithms when more than two rankers are to be compared.

As to future work, currently, in TDM, when documents belonging to the team of a ranker are clicked, preferences for this ranker over other rankers without clicks are inferred, even when those other rankers are not even represented by a team in the multileaving. This may happen when the number of rankers to be compared is larger than the number of documents in the multileaving. We aim to develop a variant of TDM that avoids this problem. Another future direction is to customize TDM and OM to tasks other than comparing all rankers in a set to each other. When comparing all rankers to a production ranker, as we do in Section 6.6, the definitions of unbiasedness and sensitivity could be adjusted to take into account the restricted goal of this task variant. In addition, multileaved comparison methods could form the basis of a new approach to tackling the K-armed dueling bandit problem, in which the best ranker among a set is sought. By measuring the uncertainty associated with each $\hat{P}_{ij}$, such a method could gradually exclude rankers from the multileaving that are deemed unlikely to be the best, thereby homing in on the most promising rankers. Finally, we are interested in integrating multileaving methods into learning methods analogous to the *dueling bandits gradient descent* method [27].

Acknowledgements. We would like to thank Filip Radlinski for discussions about optimized interleave. This research was supported by the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreements nr 288024 (LiMoSINe) and nr 312827 (VOX-Pol), the Netherlands Organisation for Scientific Research (NWO) under project nrs 727.-011.005, 612.001.116, 612.066.930, HOR-11-10, 640.006.013, the Center for Creation, Content and Technology (CCCT), the Dutch national program COMMIT, by the ESF Research Network Program ELIAS, the Elite Network Shifts project funded by the Royal Dutch Academy of Sciences (KNAW), the Netherlands eScience Center under project number 027.012.105, the Yahoo! Faculty Research and Engagement Program, the Microsoft Research PhD program, and the HPC Fund.

REFERENCES

[1] L. Azzopardi, M. de Rijke, and K. Balog. Building simulated queries for known-item topics: an analysis using six European languages. In *SIGIR '07*. ACM, 2007.

[2] R. Berendsen, M. Tsagkias, W. Weerkamp, and M. de Rijke. Pseudo test collections for training and tuning microblog rankers. In *SIGIR '13*. ACM, 2013.

[3] B. Carterette and J. Allan. Incremental test collections. In *CIKM*. ACM, 2005.

[4] B. Carterette and R. Jones. Evaluating search engines by modeling the relationship between relevance and clicks. In *NIPS '07*, 2008.

[5] O. Chapelle, T. Joachims, F. Radlinski, and Y. Yue. Large-scale validation and analysis of interleaved search evaluation. *ACM Trans. Inf. Syst.*, 30(1), 2012.

[6] A. Chuklin, A. Schuth, K. Hofmann, P. Serdyukov, and M. de Rijke. Evaluating aggregated search using interleaving. In *CIKM '13*. ACM, 2013.

[7] C. W. Cleverdon, J. Mills, and M. Keen. Factors determining the performance of indexing systems. Aslib cranfield project, Cranfield: College of Aeronautics, 1966.

[8] F. Guo, C. Liu, and Y. M. Wang. Efficient multiple-click models in web search. In *WSDM '09*. ACM, 2009.

[9] J. He, C. Zhai, and X. Li. Evaluation of methods for relative comparison of retrieval systems based on clickthroughs. In *CIKM '09*. ACM, 2009.

[10] K. Hofmann. *Fast and Reliably Online Learning to Rank for Information Retrieval*. PhD thesis, University of Amsterdam, 2013.

[11] K. Hofmann, S. Whiteson, and M. de Rijke. A probabilistic method for inferring preferences from clicks. In *CIKM '11*. ACM, 2011.

[12] K. Hofmann, A. Schuth, S. Whiteson, and M. de Rijke. Reusing historical interaction data for faster online learning to rank for IR. In *WSDM '13*. ACM, 2013.

[13] K. Hofmann, S. Whiteson, and M. de Rijke. Fidelity, soundness, and efficiency of interleaved comparison methods. *ACM Trans. Inf. Syst.*, 31(3):Article 18, 2014.

[14] K. Järvelin and J. Kekäläinen. Cumulated gain-based evaluation of IR techniques. *ACM Trans. Inf. Syst.*, 20(4):422–446, 2002.

[15] T. Joachims. Optimizing search engines using clickthrough data. In *KDD '02*. ACM, 2002.

[16] T. Joachims. Evaluating retrieval performance using clickthrough data. In *Text Mining*. Physica/Springer, 2003.

[17] 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 Trans. Inf. Syst.*, 25(2), 2007.

[18] D. Kelly. Methods for evaluating interactive information retrieval systems with users. *Found. & Tr. Inform. Retr.*, 3(1–2):1–224, 2009.

[19] R. Kohavi, R. Longbotham, D. Sommerfield, and R. M. Henne. Controlled experiments on the web: survey and practical guide. *Data Mining and Knowledge Discovery*, 18(1):140–181, 2009.

[20] T.-Y. Liu, J. Xu, T. Qin, W. Xiong, and H. Li. LETOR: Benchmark dataset for research on learning to rank for information retrieval. In *LR4IR '07*, 2007.

[21] F. Radlinski and N. Craswell. Optimized interleaving for online retrieval evaluation. In *WSDM '13*. ACM, 2013.

[22] F. Radlinski, M. Kurup, and T. Joachims. How does clickthrough data reflect retrieval quality? In *CIKM '08*. ACM, 2008.

[23] M. Sanderson. Test collection based evaluation of information retrieval systems. *Found. & Tr. Inform. Retr.*, 4(4):247–375, 2010.

[24] M. Sanderson and H. Joho. Forming test collections with no system pooling. In *SIGIR '04*. ACM, 2004.

[25] A. Schuth, K. Hofmann, S. Whiteson, and M. de Rijke. Lerot: An online learning to rank framework. In *LivingLab '13*. ACM, 2013.

[26] E. M. Voorhees and D. K. Harman. *TREC: Experiment and Evaluation in Information Retrieval*. MIT Press, 2005.

[27] Y. Yue, J. Broder, R. Kleinberg, and T. Joachims. The K-armed dueling bandits problem. *J. Comp. and System Sciences*, 78(5), 2009.

[28] M. Zoghi, S. Whiteson, M. de Rijke, and R. Munos. Using confidence bounds for efficient on-line ranker evaluation. In *WSDM '14*. ACM, 2014.

[29] M. Zoghi, S. Whiteson, R. Munos, and M. de Rijke. Relative upper confidence bound for the K-armed dueling bandit problem. In *ICML '14*, 2014.