

Stat-weight: Improving the Estimator of Interleaved Methods Outcomes with Statistical Hypothesis Testing

Alessandro Benedetti^[0000-0002-8781-8619] and
Anna Ruggero^[0000-0003-1733-1132]

Sease Ltd., London, UK
a.benedetti@sease.io, a.ruggero@sease.io

Abstract. Interleaving is an online evaluation approach for information retrieval systems that compares the effectiveness of ranking functions in interpreting the users’ implicit feedback. Previous work such as Hofmann et al (2011) [10] has evaluated the most promising interleaved methods at the time, on uniform distributions of queries. In the real world, ordinarily, there is an unbalanced distribution of repeated queries that follows a long-tailed users’ search demand curve. The more a query is executed, by different users (or in different sessions), the higher the probability of collecting implicit feedback (interactions/clicks) on the related search results. This paper first aims to replicate the Team Draft Interleaving accuracy evaluation on uniform query distributions and then focuses on assessing how this method generalizes to long-tailed real-world scenarios. The reproducibility work raised interesting considerations on how the winning ranking function for each query should impact the overall winner for the entire evaluation. Based on what was observed, we propose that not all the queries should contribute to the final decision in equal proportion. As a result of these insights, we designed two variations of the Δ_{AB} score winner estimator that assign to each query a credit based on statistical hypothesis testing. To replicate, reproduce and extend the original work, we have developed from scratch a system that simulates a search engine and users’ interactions from datasets from the industry. Our experiments confirm our intuition and show that our methods are promising in terms of accuracy, sensitivity, and robustness to noise.

Keywords: Interleaving · Statistical Significance · Real-world Query Distribution.

1 Introduction

Online evaluation is used to estimate the best ranking function for an information retrieval system, directly addressing a live instance with real users and data. It compares ranking functions through the interpretation of the users’ behavior, represented by the collected interactions with the system under evaluation. This is called implicit feedback [27].

Online evaluation is widely used because implicit feedback is generally cheap and easy to obtain, and it allows businesses to calculate useful metrics on end-users that interact directly with the ranking function (metrics like the rate of clicked documents).

This paper focuses on interleaved evaluation approaches, specifically, the Team-Draft Interleaving (TDI) method [21], applied to the comparison of two ranking functions.

Interleaving is an alternative to AB testing [23]. It avoids the principal source of variance caused by the separation of the users in groups and the consequent necessary combination of the results [23]. In AB testing one group is exposed to the control system and another group is exposed to the variant. Each group sees just a single ranking function during the evaluation.

On the other hand, interleaved methods, when responding to a query, create a unique list with search results impartially picked from the two ranking functions. This interleaved list is presented to the end-users transparently, so they don't know which result item is coming from which ranking function.

In previous works interleaved methods have been evaluated with a uniform distribution of queries [10]. Experimental setups have simulated users submitting a query by randomly sampling from a set of available queries (with replacement).

In real-world applications, the same query is executed multiple times by different users and by the same users in different sessions. The number of collected implicit feedback is not uniformly distributed across the query set. This paper aims to replicate and then reproduce some of the experiments from one of the most prominent surveys on interleaved methods [10], investigating the effect that different query distributions have on the accuracy of TDI, extending and generalizing its evaluation to different settings commonly found in production systems.

Specifically, three research questions arise:

- *RQ1*: Is it possible to replicate the original paper experiments?
- *RQ2*: How does the original work generalize in the real-world scenario where queries have a long-tailed distribution?
- *RQ3*: Does applying statistical hypothesis testing improve the TDI evaluation accuracy in such a scenario?

Thanks to the insights collected during the reproducibility work we designed two novel methods that enrich the TDI scoring estimator with a preliminary statistical analysis: *stat-pruning* and *stat-weight*. The idea is to weigh differently the contribution of each query to the final winner of the evaluation. Such contribution should be proportional to the statistical significance of the observations collected for the query.

We present a set of experiments using a large set of explicitly labeled data and a framework, developed from scratch, to simulate the implicit feedback with user clicks under different conditions. These experiments show some interesting perspectives on the original work replicability, confirm that TDI generalizes quite accurately to the considered real-world scenario, and validate the intuition that

our statistical analysis-based methods can enrich TDI to bring better accuracy in identifying the best ranking function.

The concepts 'ranking function', 'ranking model', and 'ranker' are used interchangeably.

The paper is organized as follows: Section 2 presents the related work. Section 3 details the experimental setup, datasets, and runs used for replication and reproduction. Section 4 introduces the theory behind the proposed improvements and describe our the *stat-pruning* and *stat-weight* implementation. Section 5 discusses the experiments' runs and the obtained results. The paper's conclusions and future directions are listed in Section 6.

2 Related Work

Evaluation of Information Retrieval systems follows two approaches: offline and online evaluation.

For the offline, the most commonly used is the Cranfield framework [3]: an evaluation method based on explicit relevance judgments. The experimental collection is composed of triples that represent: a document, a query, and the relevance of that document for that query. The relevance judgments are provided by a team of trained experts and this is why this process is expensive. Collecting these judgments requires a lot of effort and there is the possibility that they do not reflect the same document relevance perceived by the common users.

Users' interactions are easier to obtain and come with a minimal cost. Being performed directly by the end-users, they can be used to represent their intent closely, bypassing the domain experts' indirection. Implicit feedback is a very promising approach but, as a drawback, it could be noisy and therefore requires some further elaborations. One type of noise introduced is click position bias. Users tend to click documents in the top positions of the search result list, independently of the relevance of the result. Many papers focus on this topic and develop approaches to make interleaving fairer [1,2,21,24]. Implicit feedback is collected in real-time and it's at the base of the interleaving process. Despite the fact that the most common method of online evaluation is still AB testing, interleaving is experiencing a growing interest in research. This type of testing uses a smaller amount of traffic, with respect to AB testing, without losing accuracy in the obtained result [1,16,23]. Interleaving was introduced the first time by Joachims [15,14] and from then, many other authors proposed their changes and improvements [1,20,21,24,22].

Team Draft Interleaving (TDI from now on) is among the most successful and used interleaving approaches [21] because of its simplicity of implementation and good accuracy. It is based on the strategy that captains use to select their players in team matches. TDI produces a fair distribution of ranking functions' elements in the final interleaved list. It also showed to overcome issues of a previously implemented approach, Balanced interleaving, in determining the winning model [1].

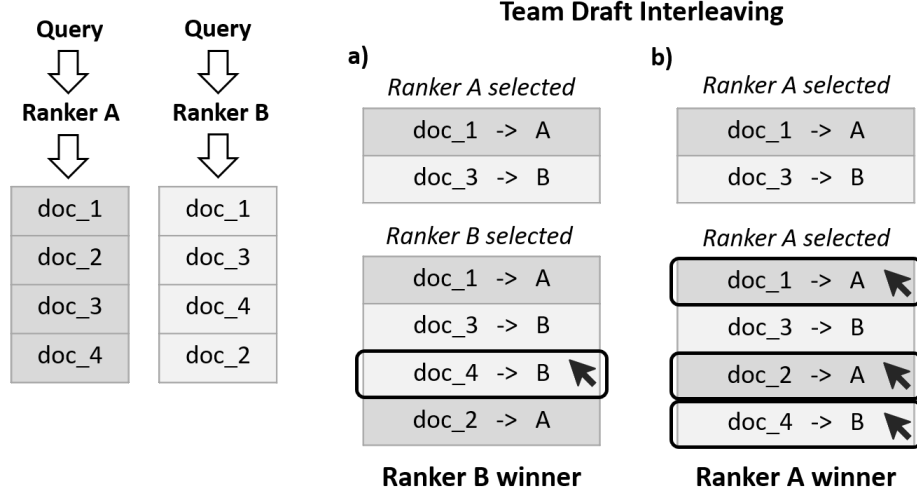


Fig. 1. Example of the Team Draft Interleaving method. Two combinations (ABBA and ABAB are shown out of the four combinations possible)

Team Draft Interleaving considers two ranking models: *rankerA* and *rankerB* (see Figure 1). For a given query, each ranker returns its ranked list of documents $l_a = (a_1, a_2, \dots)$ and $l_b = (b_1, b_2, \dots)$. The algorithm creates a unique ranked list $I = (i_1, i_2, \dots)$. This list is created by interleaving elements from the two lists l_a and l_b as described by Chapelle et al.[1]. Each element i_j is labelled *TeamA* if it is selected from l_a and *TeamB* if it is selected from l_b .

The list I is returned to the user, who interacts with the search results of interest. Let us consider clicks as a target interaction type for our analysis: given a click c , c_j is the position of the clicked search result in the ranked list I . Iterating on all clicks, the number of clicks collected by each ranker is computed as in Chapelle et al. [1]:

$$h_a = |\{c_j : i_{c_j} \in TeamA\}|$$

$$h_b = |\{c_j : i_{c_j} \in TeamB\}|$$

If $h_a > h_b$ then *rankerA* is the winner for that query; if $h_a < h_b$ then *rankerB* is the winner; otherwise it is a tie.

To assess the overall winner between *rankerA* and *rankerB*, the Δ_{AB} score is computed as [1]:

$$\Delta_{AB} = \frac{wins(A) + \frac{1}{2}ties(A, B)}{wins(A) + wins(B) + ties(A, B)} - 0.5 \quad (1)$$

Where:

- $wins(A)$ is the number of queries in which *rankerA* is the winner

- $wins(B)$ is the number of queries in which *rankerB* is the winner
- $ties(A, B)$ is the number of queries in which the two rankers equalize

A Δ_{AB} score < 0 means *rankerB* is the overall winner, a Δ_{AB} score $= 0$ means a tie, a Δ_{AB} score > 0 means *rankerA* is the overall winner.

Other types of interleaved methods are Document Constraint [8], Probabilistic Interleaving [10] and Optimized Interleaving [20].

Document constraint infers preference relations between pairs of documents, estimated from their clicks and ranks. The method compares the inferred constraints to the original result lists and assesses how many constraints are violated by each. The list that violates fewer constraints is deemed better. This method demonstrated to be more reliable than either balanced interleave or team draft on synthetic data, but it’s more computationally expensive.

In probabilistic interleaving, both the choice of the model that contributes to the interleaving list and the document to put in the list, are selected based on probability. This approach shows higher reliability, efficiency, and robustness to the noise with respect to the others.

Optimized interleaving proposes to formulate interleaving as an optimization problem that is solved to obtain the interleaved lists that maximize the expected information gained from users’ interactions.

To conclude this Section it’s worth mentioning that a generalized form of the Team Draft Interleaving has been proposed [17] and that additional research has been performed by Hofmann et al. with a new interleaving approach that aims to reduce the bias related to the way results are presented to the users [9] and studies on the fidelity, soundness, and efficiency of interleaved methods [11].

3 Experiments

We present a set of experiments designed to answer our three research questions. This paper aims to replicate and then reproduce under different scenarios experiment 1 from one of the most prominent surveys on interleaved methods [10].

In the first set of experiments, we address *RQ1* with the same settings and data as the original work (uniform query distribution). We compare a large number of ranker pairs over simulated user clicks to examine if the accuracy of the TDI method matches the published results from the original research. We discuss the assumptions and details that we found to hold up, and the ones that could not be confirmed. In addition we assess how the traditional TDI accuracy compares with two novel methods for Δ_{AB} score calculations (*stat-pruning* and *stat-weight*) under the uniform distribution conditions. It has been chosen to reproduce this experiment with the TDI method for the simple implementation and the easier reproduction process. Even if probabilistic interleaving has been shown to be the best method, TDI maintains a good trade-off between performance and simplicity.

In the second set of experiments, we address *RQ2* and *RQ3* introducing experimental scenarios with a long-tailed query distribution. We compare a large

number of ranker pairs over different real-world query distributions extracted from anonymized query logs. The aim is to examine how well the traditional TDI method generalizes in these real-world scenarios and how *stat-pruning* and *stat-weight* perform for comparison.

Finally, the last set of experiments introduces a realistic click model simulator to assess how well TDI, *stat-pruning* and *stat-weight* methods respond to noise.

The datasets used are detailed in 3.1. The experimental setup is described in 3.2 and the experiment runs are explained in 3.3.

3.1 Datasets

All experiments make use of the MSLR-WEB30k Microsoft learning to rank (*MSLR*) dataset¹. This dataset represents queries and documents by IDs. It consists of feature vectors extracted from query-document pairs along with relevance judgment labels. The relevance judgments are obtained explicitly from a retired labeling set of a commercial web search engine, which takes 5 values from 0 (irrelevant) to 4 (perfectly relevant). The features describe aspects of the query-document pair widely used in the research community such as the length of a field of the document or the term frequency of the query terms in a field of the document. In the data files, each row corresponds to a query-document pair. The first element is the relevance label of the pair, the second is the query id, and the following elements are the features.

The experiments use the training set of fold 1. This set contains 18,919 queries, with an average of 119.96 judged documents per query. To define the ranking functions to compare in our experiments, we leverage the features provided for the documents of the dataset. Specifically, from each feature, we define a ranker (identified by the feature id) that sorts the search results descending by the feature value.

The experiments involving the long-tailed distributions make use of an industrial dataset we called *long-tail-1*. It consists of a list of query executions extracted from the query log of a commercial search engine over a period of time. Each query is associated with the number of times it is executed by different users. The amount of users collected per query is capped to 1,000. This threshold ensures maintaining the long-tailed distribution while keeping a sustainable computational cost.

From this dataset, we derive the long tail in Figure 2.

3.2 Experimental Setup

To replicate the original experiments we designed and developed a system that simulates a search engine with users submitting queries and interacting with the results (clicks). The code references are in the appendix.

The experiments are designed to evaluate the interleaved methods' ability to establish the better of two ranking functions based on (simulated) user clicks.

¹ download from <http://research.microsoft.com/en-us/projects/mslr/default.aspx>.

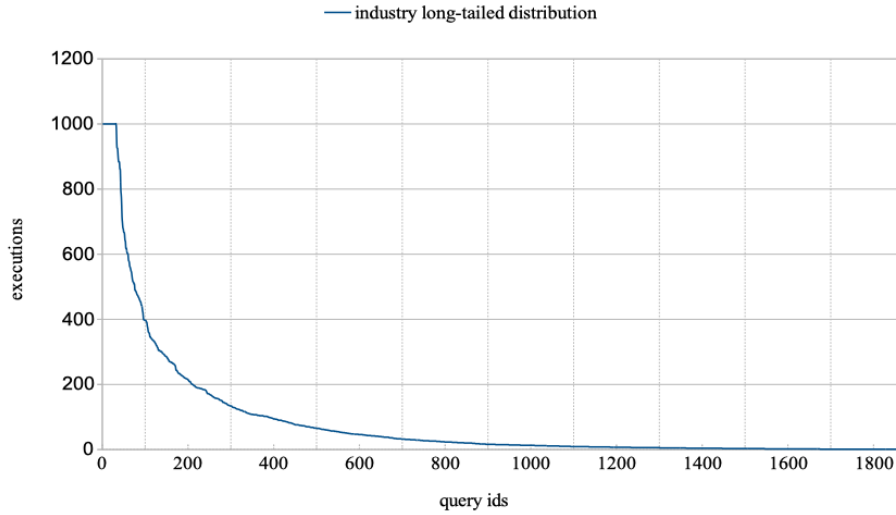


Fig. 2. Total unique queries: 1 861, Total executions:156 550

When a query is submitted to the simulated search engine it returns the pre-calculated list of matching search results with explicit relevance judgments from the dataset *MSLR*. The result set is ranked by a TDI interleaved evaluation of two ranking functions.

Each experiment run repeats a number of simulations proportional to the number r of different ranking functions we want to include in the evaluation. Each ranking function is identified by an incremental id that is aligned with the id of the feature of reference. When an experiment evaluates r ranking functions, it evaluates the first r , ordered by ascending id. Specifically, given a set of ranking functions, the number of simulations s in the run is the number of unique pairs in the set, where the pairs are subject to the commutative property ($AB = BA$). The user interactions are simulated using a query distribution and a click model.

The system simulates a user submitting a query from the set of available queries in the distribution (in long-tailed distributions each query is submitted multiple times). The search engine responds with an interleaved result list that is presented to the user. The simulation models the assumption that more relevant documents are more likely to be clicked. The user clicks are randomly generated following the probability distribution that the click model assigns to the relevance judgments provided.

Once the simulation completes, the ranking function preference of each click collected is evaluated and the Δ_{AB} score is computed to establish the winner by the interleaving evaluation chosen. The ground truth winner is calculated as the ranking function with the best average Normalised Discounted Cumulative Gain (NDCG from now on) [13,12] on the considered query set. The NDCG calculation is based on the explicit relevance judgments provided with the dataset *MSLR*.

Depending on the run, to determine the NDCG we use the complete search results list for a query or a top-k (cut-off at k). The winning ranker identified by the Δ_{AB} score is compared to the ground truth winner for the pair, when they match we have a correct guess. To assess the accuracy of the interleaved evaluation method we count the number of correct guesses over the total number of simulations s in the run showing at least one click.

Below we describe the query distributions, the click models, and the NDCG we used in our experiments.

Query Distributions The query distribution in input to the simulation establishes the number of queries that the user submits to the system. We use two types of query distributions in our experiments: uniform and long-tailed.

In the uniform query distribution, each unique query is executed a constant number of times. When considering q queries in a run, we refer to the first q query ids, in the same order as they appear in the dataset *MSLR* rows.

In the long-tailed query distribution, each query is executed a variable number of times. Starting from the *long-tail-1* distribution from the industry, we scaled down the number of queries and their executions by a factor $u \leq 1$ (see Figure 3). The first reason for that is to experiment how the evaluation methods

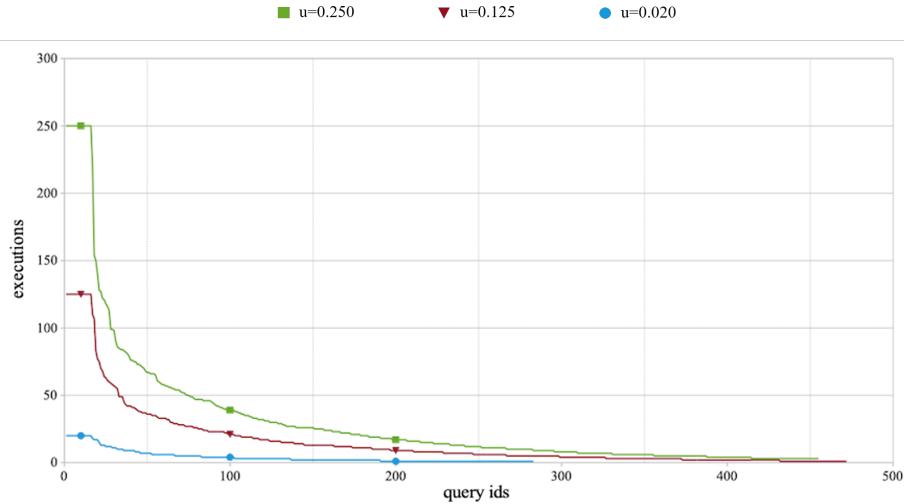


Fig. 3. Long-tailed query distributions used in the experiments

perform with different instances of realistic long-tailed distributions. The second reason is to act within the computational limits of our experimental setup. For each ranker pair to interleave, we simulate a number of query executions taken from the query distribution. The closer u gets to 1, the higher the total number of executions (see Table 1). Higher the number of executions higher the cost in

Table 1. Scaling the query executions

u	<i>unique queries total executions</i>	
0.020	283	1247
0.125	472	7681
0.250	455	14449

terms of memory and time consumption. Running the full simulation, on all the rankers, with the original long-tailed distribution would have required too much time and memory in our setup.

Click Models The click model in input to the simulation establishes the probability of a search result to be clicked given its explicit relevance label (ground truth). Using different click models makes it possible to study the evaluation methods at different levels of noise in user clicks.

The click model simulates user interactions according to the Dependent Click Model (DCM) [6,7], an extension of the cascade model [4].

According to this model, users scan result lists from the top-ranking result to the last. For each document they see, they decide to click it or not, depending on its perceived relevance (e.g., based on the title, thumbnail, and content snippets). After clicking on a document, users assess the full relevance of the search result and if the information need is satisfied then they stop scanning the result list. Otherwise, they progress.

Table 2. Click models used in the experiments [10]

<i>perfect model</i>					
$R(d)$	0	1	2	3	4
$P(c \mid R(d))$	0.00	0.20	0.40	0.80	1.00
$P(s \mid R(d))$	0.00	0.00	0.00	0.00	0.00
<i>realistic model</i>					
$R(d)$	0	1	2	3	4
$P(c \mid R(d))$	0.05	0.10	0.20	0.40	0.80
$P(s \mid R(d))$	0.00	0.20	0.40	0.60	0.80

For each document viewed by the user, $P(c \mid R(d))$ is the probability that a click is performed given the relevance label of the examined document $R(d)$. $P(s \mid R(d))$ is the probability that a user stops scanning the result list because it has satisfied the information need after clicking a document with the relevance label $R(d)$.

We use the two-click models proposed by the original research [10]. The realistic click model presents a noisier click behavior. Furthermore, the overall expected clicks quantity is lower for the realistic click model because the click

probabilities are consistently lower and the stop probabilities are higher. For these reasons, the realistic click model increases the difficulty for the interleaved methods to correctly guess the best ranking function. Table 2 lists the probabilities defined for the two models.

NDCG The NDCG metrics we used in our experiments are the complete NDCG (the complete search results list for a query) and the NDCG@10 (cut-off at 10).

Using NDCG@10 is quite common in the industry as many search engines show 10 documents on their first page and it aligns with the decision of having the simulated users click only the top-10 results. When comparing the complete NDCG with NDCG@10 we noticed that the average difference between the pair of rankers to evaluate is smaller, making it more difficult for the interleaved methods to correctly guess the best ranker (see Table 3 and Table 4).

Table 3. complete NDCG and NDCG@10 for the first 10 rankers, averaged over the 1000 queries

<i>ranker</i>	<i>NDCG</i>	<i>NDCG@10</i>
1	0.550	0.195
2	0.578	0.260
3	0.603	0.284
4	0.585	0.261
5	0.550	0.192
6	0.550	0.195
7	0.578	0.260
8	0.603	0.284
9	0.585	0.261
10	0.550	0.192

Table 4. average difference in NDCG over the 9180 pairs of rankers

<i>NDCG</i>	<i>NDCG@10</i>
0.026	0.046

3.3 Runs

We divided the runs into four groups: replication, uniform query distribution, long-tailed query distribution, and realistic click model. A seed is set at the beginning of each run so that it is possible to reproduce all the random choices of the experiment reliably in repeated executions. Using different seeds will result in slightly different results. An interesting future study could be to execute each run

multiple times with different seeds and explore such differences from a statistical perspective.

Replication The scope of this set of runs is to replicate the *experiment 1* from the original research and answer *RQ1*. As a baseline, to reduce the computational stress we evaluate the Team Draft Interleaving method only. We define a ranker for each of the 136 individual features provided with the *MSLR* dataset. We exhaustively compare all 9,180 distinct pairs derived from the 136 rankers. The query distribution used is uniform and for each ranker pair, the user submits 1,000 queries. The query set consists of the first distinct 1,000 queries as occurring in the *MSLR* dataset rows. The click model used is the *perfect model*. The results report the percentage of pairs for which the TDI method correctly identified the better ranker.

- **Run 1:** exactly replicates the original experiment, users click on the top-10 results for each query, to determine NDCG for the ground truth we use the complete set of documents provided with the dataset *MSLR*, i.e., no cut-off is used.

We observed some inconsistencies with the original work results so we added to this group two additional runs:

- **Run 2:** users click on the complete list of search results for each query. To determine NDCG for the ground truth we use the complete set of documents provided with the dataset *MSLR*, i.e., no cut-off is used.
- **Run 3:** users click on the top-10 results for each query. To determine NDCG for the ground truth we calculate NDCG@10.

Uniform query distribution The scope of this set of runs is to evaluate how the *stat-pruning* and *stat-weight* methods compare with the TDI baseline. We exhaustively compare all 9,180 distinct pairs derived from the 136 rankers. The query distribution used is uniform, each run uses a different number q of queries. The query set consists of the first distinct q queries as occurring in the *MSLR* dataset rows. The click model used is the *perfect model*. Users click on the top-10 results for each query. Unless stated otherwise, to determine NDCG for the ground truth we calculate NDCG@10. The results compare the percentage of pairs for which the TDI, *stat-pruning* and *stat-weight* methods correctly identified the better ranker.

- **Run 4:** the query set consists of the first distinct 1,000 queries as occurring in the *MSLR* dataset. Each query is executed once.
- **Run 5:** the query set consists of the first distinct 100 queries as occurring in the *MSLR* dataset. Each query is executed once.

- **Run 6:** the query set consists of the first distinct 100 queries as occurring in the *MSLR* dataset. Each query is executed 10 times.
- **Run 7:** the query set consists of the first distinct 100 queries as occurring in the *MSLR* dataset. Each query is executed 10 times. To determine NDCG for the ground truth we use the complete set of documents provided with the dataset *MSLR*, i.e., no cut-off is used.

Long-tailed query distribution The scope of this set of runs is to evaluate how the *stat-pruning* and *stat-weight* methods compare with the TDI baseline over long-tailed query distributions and answer *RQ2* and *RQ3*. The click model used is the *perfect model*. Each run uses a different long-tailed query distribution, see Figure 2 and Table 1 for reference. Users click on the top-10 results for each query. Unless stated otherwise, to determine NDCG for the ground truth we calculate NDCG@10. The results compares the percentage of pairs for which the TDI, *stat-pruning* and *stat-weight* methods correctly identified the better ranker.

- **Run 8:** the query set consists of 283 unique queries repeated following the long-tailed distribution with $u = 0.020$. We exhaustively compare all 9,180 distinct pairs derived from the 136 rankers.
- **Run 9:** the query set consists of 472 unique queries repeated following the long-tailed distribution with $u = 0.125$. We exhaustively compare all 9,180 distinct pairs derived from the 136 rankers.
- **Run 10:** the query set consists of 455 unique queries repeated following the long-tailed distribution with $u = 0.250$. We exhaustively compare all 2,415 distinct pairs derived from the first 70 rankers.
- **Run 11:** the query set consists of 283 unique queries repeated following the long-tailed distribution with $u = 0.020$. We exhaustively compare all 9,180 distinct pairs derived from the 136 rankers. To determine NDCG for the ground truth we use the complete set of documents provided with the dataset *MSLR*, i.e., no cut-off is used.

Realistic click model The scope of this set of runs is to evaluate how the *stat-pruning* and *stat-weight* methods compare with the TDI baseline over long-tailed query distributions with noisier clicks. We exhaustively compare all 9,180 distinct pairs derived from the 136 rankers. The click model used is the *realistic model*. The query distribution used is long-tailed. The query set consists of 283 unique queries repeated following the long-tailed distribution with $u = 0.020$, see Figure 2 and Table 1 for reference. Users click on the top-10 results for each query. The results compares the percentage of pairs for which the TDI, *stat-pruning* and *stat-weight* methods correctly identified the better ranker.

- **Run 12:** to determine NDCG for the ground truth we calculate NDCG@10.
- **Run 13:** to determine NDCG for the ground truth we use the complete set of documents provided with the dataset *MSLR*, i.e., no cut-off is used.

4 Improving the Overall Winner Decision

The research questions $RQ1$ and $RQ2$ are addressed by the experiments in Sections 3 and 5, here we want to focus on $RQ3$ which requires a deeper analysis.

In TDI, for each query in the dataset, a winner between two ranking functions is estimated. The estimation is based on the number of interactions (clicks for example) that prefer a ranking function or the other. In the interleaved result list, an interaction with a document that was picked from the ranked list A (the original ranked list produced by *rankerA*), shows a preference for *rankerA* and vice versa for *rankerB*.

The reliability of each winner is not assessed. All the winners (i.e., all the queries) are considered equal when aggregating the results to establish the overall winning ranker (Equation 1).

This may include preferences that are obtained with few clicks or preferences that are not strong enough given the number of clicks collected.

To mitigate this problem, this paper proposes two variations for the Δ_{AB} score: *stat-pruning* and *stat-weight*. They rely on an additional phase in the TDI evaluation, which assigns to each query a credit inversely proportional to the probability of obtaining by chance at least the same number of clicks, assuming the two rankers are equivalent. This credit per query affects the overall winner calculation. It should reduce the impact of queries with a weak winning ranker leading to a more accurate overall winner estimation.

4.1 Statistical hypothesis testing

While performing our reproducibility research we observed two problems:

- some queries have many interactions, but a very weak preference for the winning ranker
- some queries have a decent preference for the winning ranker but few interactions (the long tail)

In classic TDI, each query has the same weight when calculating the Δ_{AB} score. The overall winner decision may be polluted by the aforementioned queries.

Previous works have explored the possibility of assigning a different credit to each click [20]. The approach we suggest is to assign a different credit to each query.

The idea is to exploit statistical hypothesis testing to estimate if the observations for a query are reliable and to what extent. This additional phase is executed after the computation of h_a and h_b , and before the computation of the Δ_{AB} score.

The theory behind our approach is statistical hypothesis testing [25]. A statistical test verifies or contradicts a null hypothesis based on the collected samples. A result has statistical significance when it is very unlikely to have occurred given the null hypothesis [18].

The *p-value* of an observed result is the probability of obtaining a result at least as extreme, given that the null hypothesis is true. The result is statistically significant, by the standards of the study, when

$$p\text{-value} \leq \alpha$$

Such a scenario leads to the rejection of the null hypothesis and acceptance of the alternate hypothesis. The significance level, denoted by α is assigned at the beginning of a study [5].

Accordingly for this test, we need a null hypothesis, a p-value, and a significance level.

Our null hypothesis is that the two ranking functions we are comparing are equivalent and have the same chance to win in a query i.e. the probability of both ranking functions winning is 0.5.

For each query:

- n is the total number of clicks collected
- the winning ranker is the ranker that collected more clicks
- k is the clicks collected by the winning ranker.
- p is 0.5 (null hypothesis).

Given we are limiting our evaluation to two ranking functions:

$$k \geq \frac{n}{2}$$

When $k = \frac{n}{2}$, there is a draw, the query doesn't show any preference for the ranking function since each ranker collects the same amount of clicks.

The *p-value* is calculated through a binomial distribution as the probability of obtaining exactly that number of clicks k assuming the null hypothesis is true:

$$P(X = k) = \binom{n}{k} p^k (1-p)^{n-k} \quad (2)$$

When $k > \frac{n}{2}$, the query shows a preference for a ranking function. We are testing whether the clicks are biased towards the winning ranking function, so a single-tailed test is used.

The *p-value* is calculated through a binomial distribution as the probability, for the winning model, to obtain at least that number of clicks k assuming the null hypothesis is true:

$$\begin{aligned} P(X \geq k) &= 1 - P(X < k) \\ &= 1 - \sum_{i=0}^{k-1} \binom{n}{i} p^i (1-p)^{n-i} \end{aligned} \quad (3)$$

4.2 Stat-pruning

The first approach we designed is the simplest and most aggressive: the statistical significance of each query is determined by comparing the *p-value* with a

significance level $\alpha = 0.05$. This is the standard threshold used in most statistical tests.

If the *p-value* is below the threshold, the result is considered significant. The queries not reaching significance are discarded from the Δ_{AB} score calculation.

The downside of this approach is that is strictly coupled to the significance level α hyper-parameter. The setting of this parameter may not be simple to decide, other works explore this aspect [19,26].

4.3 Stat-weight

Let's present for simplicity the original Δ_{AB} score formula again (Equation 1):

$$\Delta_{AB} = \frac{wins(A) + \frac{1}{2}ties(A, B)}{wins(A) + wins(B) + ties(A, B)} - 0.5$$

The credit associated with each win or tie is a constant 1.

The idea is to assign a different credit to each win and tie. This credit is the estimated probability of the win/tie to have happened not by chance.

$$credit(q_x) = 1 - p-value(q_x)$$

The *p-value* for a query q_x that presents a tie is calculated with the Equation 2.

The *p-value* for a query q_x that presents a win is calculated with the Equation 3 and it is normalised with a min-max normalization ($min = 0$ and $max = 0.5$) to be between 0 and 1.

The proposed updates to the Δ_{AB} score formula are the following:

$$\begin{aligned} wins(A) &\Rightarrow \sum_{a=0}^{wins(A)} credit(q_a) \\ wins(B) &\Rightarrow \sum_{b=0}^{wins(B)} credit(q_b) \\ ties(A, B) &\Rightarrow \sum_{t=0}^{ties(A, B)} credit(q_t) \end{aligned}$$

q_a is in the query set showing a preference for the *ranker* A .

q_b is in the query set showing a preference for the *ranker* B .

q_t belongs to the query set showing a tie.

5 Results and Analysis

In this Section, we present and discuss the results of our experiments.

Table 5. Query distribution: uniform 1,000 queries, click model: *perfect model*, 136 rankers (9, 180 pairs)

<i>id</i>	<i>NDCG</i>	<i>clicks</i>	<i>accuracy</i>	<i>original-accuracy</i>
1	complete	top-10	0.852	0.898
2	complete	complete	0.825	0.898
3	top-10	top-10	0.902	0.898

5.1 Replication

run-1 follows the same experimental setup, dataset, and parameters from the original research, but it fails to replicate the originally recorded accuracy of TDI.

Also, the average ground truth NDCGs calculated for the rankers, don’t align with the ones reported by the original paper (Table 6):

Table 6. complete NDCG, averaged over the 1000 queries

<i>ranker</i>	<i>NDCG</i>	<i>original-paper NDCG</i>
1	0.550	0.231
14	0.536	0.201
64	0.600	0.301
77	0.570	0.262
84	0.574	0.256
96	0.549	0.219
97	0.564	0.303
106	0.606	0.253
108	0.614	0.306
134	0.614	0.341

We thought that the difference could have been caused by a dis alignment between the published paper and the original experiments NDCG and clicks generation parameters used at the time. For these reasons we executed two additional runs, trying to explain the possible causes of this failed replication. The closer we got to the original recorded accuracy is with *run-3*, but not exactly the same (see Table 5). Also, the monitored average NDCG@10 over the 1,000 queries are closer but not exactly matching the original work ones (Table 7). The random seed that drives the TDI interleaved-lists generation and the clicks can have a part, but it can’t affect the ground truth NDCG scores.

After discussing with the authors of the paper, we could ascertain that the NDCG formula used is the same as ours. However, we weren’t able to check the input parameters since we couldn’t have access to the original paper code. Our best guesses are therefore the following:

- **NDCG:** the published paper clearly specifies it is the complete NDCG, but the original experiments maybe initially used it and then were updated to

Table 7. NDCG@10, averaged over the 1,000 queries

<i>ranker</i>	<i>NDCG@10</i>	<i>original-paper NDCG</i>
1	0.195	0.231
14	0.179	0.201
64	0.294	0.301
77	0.246	0.262
84	0.239	0.256
96	0.194	0.219
97	0.234	0.303
106	0.295	0.253
108	0.306	0.306
134	0.333	0.341

use the NDCG@10 for the final reports. This could explain why NDCG@10 scores are much closer to the reported ones.

- **Queries:** the query set used is not the same as our runs i.e., not the first distinct 1,000 queries as occurring in the *MSLR* dataset rows. This could explain why NDCG@10 scores are closer but not exactly the same as the reported ones.

5.2 Uniform Query Distribution

Table 8. NDCG: top-10, clicks: top-10, click model: *perfect model*, 136 rankers (9,180 pairs)

			<i>accuracy</i>		
<i>id</i>	<i>queries</i>	<i>users</i>	<i>TDI</i>	<i>stat-pruning</i>	<i>stat-weight</i>
4	1 000	1	0.902	<i>N/A</i>	0.886
5	100	1	0.812	<i>N/A</i>	0.790
6	100	10	0.857	0.853	0.883

From Table 8, *run-4* and *run-5* show a better accuracy for the original TDI method with a uniform distribution of 100 or 1,000 queries if each query is executed once. In these scenarios there are very few clicks per query, the accuracy for *stat-pruning* is not available as it removes aggressively all the queries as deemed not significant. *stat-weight* doesn’t shine as well: it has too few clicks per query to work on. This is a little unrealistic for real-world use cases, even if uniform. So *run-6* explores what happens if the distribution is uniform and each query is executed 10 times.

In this scenario, we start to see the benefits of the *stat-weight* approach with a 2.6% increase in accuracy (it correctly guessed 239 additional pairs) in comparison to the classic TDI.

Comparing *run-5* and *run-6* we notice that by increasing the number of users running the queries uniformly, all the methods improve their accuracy and converge more quickly. This is expected as we get more clicks per query and it's interesting to notice that *stat-weight* is able to better handle the additional interactions discerning where they are reliable or not to identify the best ranker.

stat-weight was originally designed for long-tailed distributions, but the take away from this set of runs is that it is quicker to converge and can perform better than TDI also in uniform distributions, with the caveat that queries are repeated more than once by the users.

stat-pruning showed to be generally too aggressive and it's the worst in terms of accuracy (using $\alpha = 0.05$).

Table 9. NDCG: complete, clicks: complete, click model: *perfect model*, 136 rankers (9, 180 pairs)

<i>accuracy</i>					
<i>id</i>	<i>queries</i>	<i>users</i>	<i>TDI</i>	<i>stat-pruning</i>	<i>stat-weight</i>
7	100	10	0.828	0.839	0.857

From Table 9, *run-7* makes the task more difficult as the complete NDCG presents less difference between the rankers, so it's more challenging for the interleaving methods to guess correctly. See Table 3 and Table 4 for NDCG comparisons.

stat-weight demonstrated to be more sensitive in this difficult scenario identifying correctly 267 additional pairs.

5.3 Long-tailed Query Distribution

Table 10. NDCG: top-10, clicks: top-10, click model: *perfect model*

<i>accuracy</i>					
<i>id</i>	<i>u</i>	<i>rankers</i>	<i>TDI</i>	<i>stat-pruning</i>	<i>stat-weight</i>
8	0.020	136	0.880	0.860	0.897
9	0.125	136	0.892	0.900	0.904
10	0.250	70	0.904	0.910	0.911

From Table 10, *run-8*, *run-9* and *run-10* explore different long-tailed distributions. Due to computational limits, we had to restrict the number of rankers

the closer we were getting to the original *long-tailed-1* distribution. For this reason, it’s not fair to compare the executions against each other and each must be observed independently.

The steepest the long-tail, the better *stat-pruning* performs. This is expected as we assume the long part of the tail to add uncertainty for TDI, an uncertainty that is cut by the *stat-pruning* and mitigated by the *stat-weight* approach. This confirms the intuition that statistical hypothesis testing improves the classic TDI Δ_{AB} score accuracy in the long-tailed scenario.

Table 11. NDCG: complete, clicks: top-10, click model: *perfect model*, 136 rankers (9180 pairs)

<i>accuracy</i>					
<i>id</i>	<i>u</i>	<i>rankers</i>	<i>TDI</i>	<i>stat-pruning</i>	<i>stat-weight</i>
11	0.020	136	0.827	0.817	0.837

From Table 11, *run-11* explores again the harder problem of closer rankers with the complete NDCG. We can see that *stat-weight* confirms its sensitivity and it is able to identify correctly 91 additional pairs also in the long-tailed scenario.

5.4 Realistic Click Model

Table 12. Query distribution: long-tailed $u=0.020$, clicks: top-10, click model: *realistic model*, 136 rankers (9,180 pairs)

<i>accuracy</i>					
<i>id</i>	<i>NDCG</i>	<i>TDI</i>	<i>stat-pruning</i>	<i>stat-weight</i>	
12	top-10	0.818	0.708	0.833	
13	complete	0.782	0.693	0.795	

From Table 12, *run-12* introduces an additional challenge with the *realistic model* that produces noisier and fewer clicks, so it’s more difficult for the interleaving methods to guess correctly. *stat-weight* demonstrated to be robust to noise with a 1.5% increase (137 additional pairs) in comparison to the classic TDI. Finally, *run-13* tests the methods under an even more difficult situation with the complete NDCG. The overall scores across the three methods are smaller, but the *stat-weight* keeps consistently the lead.

6 Conclusions And Future Directions

RQ1 has not been satisfied.

Replicating the original research turns to be challenging from many angles: it was easy to align with the datasets but it required a substantial amount of work to figure out the exact parameters used in the original runs and design and develop from scratch the experiment code to cover all the necessary scenarios.

Unfortunately, it was not possible to exactly replicate the reported accuracy for TDI due to missing information and code unavailability. After a discussion with the authors of the paper, two hypotheses have been made: NDCG reported as a complete NDCG was instead using a cut-off; the query set is different from the one we used (which 1000 queries?).

RQ2 has been satisfied. We verified that it is possible to generalize the original TDI evaluation to long-tailed query distributions with good accuracy.

RQ3 has been satisfied. The reproducibility work brought many interesting insights and we developed a new set of statistical-based Δ_{AB} score methods: *stat-pruning* and *stat-weight*. Applying statistical hypothesis testing to assign a credit score to each of the queries in the evaluations has shown to be promising because adapts quite well to various real-world scenarios and doesn't add a substantial overhead in terms of performance.

stat-weight performs consistently well across realistic uniform and long-tailed query distributions, it's sensitive to small differences between the rankers and it is robust to noise.

stat-pruning performs well in some realistic scenarios, but it felt generally too aggressive and too coupled with the hyper-parameter α that can be tricky to tune.

We validated the intuitions of our analysis and our proposed methods using experiments based on a simulation framework developed from scratch.

Applying *stat-weight* to other interleaved methods in real-world scenarios is an interesting direction for future works. Also calculating the query credit with different statistical approaches and normalization could be explored. Finally, it would be interesting to run experiments with bigger numbers and many seeds to see how the different evaluation methods perform.

A Experiment framework code

A.1 Github Repository

<https://github.com/SeaseLtd/statistical-interleaving>

A.2 Datasets

MSLR long-tail-1

A.3 Runs

Instructions to replicate the runs are in the README.

The detailed output of each run is in: output

References

1. Chapelle, O., Joachims, T., Radlinski, F., Yue, Y.: Large-scale validation and analysis of interleaved search evaluation. *ACM Transactions on Information Systems (TOIS)* **30**(1), 1–41 (2012)
2. Chuklin, A., Serdyukov, P., De Rijke, M.: Click model-based information retrieval metrics. In: *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*. pp. 493–502 (2013)
3. Cleverdon, C.W., Mills, J., Keen, E.M.: Factors determining the performance of indexing systems, (volume 1: Design). Cranfield: College of Aeronautics **28** (1966)
4. Craswell, N., Zoeter, O., Taylor, M., Ramsey, B.: An experimental comparison of click position-bias models. In: *Proceedings of the 2008 international conference on web search and data mining*. pp. 87–94 (2008)
5. Dalgaard, P.: Power and the computation of sample size. In: *Introductory Statistics with R*, pp. 155–162. Springer (2008)
6. Guo, F., Li, L., Faloutsos, C.: Tailoring click models to user goals. In: *Proceedings of the 2009 workshop on Web Search Click Data*. pp. 88–92 (2009)
7. Guo, F., Liu, C., Wang, Y.M.: Efficient multiple-click models in web search. In: *Proceedings of the second acm international conference on web search and data mining*. pp. 124–131 (2009)
8. He, J., Zhai, C., Li, X.: Evaluation of methods for relative comparison of retrieval systems based on clickthroughs. In: *Proceedings of the 18th ACM conference on Information and knowledge management*. pp. 2029–2032 (2009)
9. Hofmann, K., Behr, F., Radlinski, F.: On caption bias in interleaving experiments. In: *Proceedings of the 21st ACM international conference on Information and knowledge management*. pp. 115–124 (2012)
10. Hofmann, K., Whiteson, S., De Rijke, M.: A probabilistic method for inferring preferences from clicks. In: *Proceedings of the 20th ACM international conference on Information and knowledge management*. pp. 249–258 (2011)
11. Hofmann, K., Whiteson, S., Rijke, M.D.: Fidelity, soundness, and efficiency of interleaved comparison methods. *ACM Transactions on Information Systems (TOIS)* **31**(4), 1–43 (2013)
12. Järvelin, K., Kekäläinen, J.: Cumulated gain-based evaluation of ir techniques. *ACM Transactions on Information Systems (TOIS)* **20**(4), 422–446 (2002)
13. Järvelin, K., Kekäläinen, J.: Ir evaluation methods for retrieving highly relevant documents. In: *ACM SIGIR Forum*. vol. 51, pp. 243–250. ACM New York, NY, USA (2017)
14. Joachims, T.: Optimizing search engines using clickthrough data. In: *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*. pp. 133–142 (2002)
15. Joachims, T., et al.: Evaluating retrieval performance using clickthrough data. (2003)
16. Kharitonov, E., Macdonald, C., Serdyukov, P., Ounis, I.: Using historical click data to increase interleaving sensitivity. In: *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*. pp. 679–688 (2013)
17. Kharitonov, E., Macdonald, C., Serdyukov, P., Ounis, I.: Generalized team draft interleaving. In: *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. pp. 773–782 (2015)
18. Myers, J.L., Well, A.D., Lorch, J.: Developing the fundamentals of hypothesis testing using the binomial distribution. *Research Design and Statistical Analysis* pp. 65–90 (2010)

19. Queen, J.P., Quinn, G.P., Keough, M.J.: Experimental design and data analysis for biologists. Cambridge university press (2002)
20. Radlinski, F., Craswell, N.: Optimized interleaving for online retrieval evaluation. In: Proceedings of the sixth ACM international conference on Web search and data mining. pp. 245–254 (2013)
21. Radlinski, F., Kurup, M., Joachims, T.: How does clickthrough data reflect retrieval quality? In: Proceedings of the 17th ACM conference on Information and knowledge management. pp. 43–52 (2008)
22. Schuth, A., Brintjes, R.J., Buüttner, F., van Doorn, J., Groenland, C., Oosterhuis, H., Tran, C.N., Veeling, B., van der Velde, J., Wechsler, R., et al.: Probabilistic multileave for online retrieval evaluation. In: Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval. pp. 955–958 (2015)
23. Schuth, A., Hofmann, K., Radlinski, F.: Predicting search satisfaction metrics with interleaved comparisons proceedings of the 38th international acm sigir conference on research and development in information retrieval, santiago, chile, august 9–13, 2015, ricardo a. ACM (2015)
24. Schuth, A., Sietsma, F., Whiteson, S., Lefortier, D., de Rijke, M.: Multileaved comparisons for fast online evaluation. In: Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management. pp. 71–80 (2014)
25. Sirkin, R.M.: Statistics for the social sciences. Sage (2006)
26. Sproull, N.L.: Handbook of research methods: A guide for practitioners and students in the social sciences. Scarecrow press (2002)
27. Yue, Y., Gao, Y., Chapelle, O., Zhang, Y., Joachims, T.: Learning more powerful test statistics for click-based retrieval evaluation. In: Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval. pp. 507–514 (2010)