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

Alessandro Benedetti(✉) and Anna Ruggero

Sease Ltd., London, UK
{a.benedetti,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) [11] has evaluated the most promising interleaved methods at the time, on uniform distributions of queries. In the real world, usually, there is an unbalanced distribution of repeated queries that follows a long-tailed users' search demand curve. This paper first aims to reproduce the Team Draft Interleaving accuracy evaluation on uniform query distributions [11] and then focuses on assessing how this method generalises to long-tailed real-world scenarios. The replicability 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 reproduce, replicate 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: Interleaved comparison · Interleaving · Implicit feedback · Online evaluation · Academia-Industry collaborations

1 Introduction

In information retrieval, online evaluation estimates the best ranking function for a system, targeting a live instance with real users and data. In previous works interleaved methods have been evaluated on a uniform distribution of queries [11]. In real-world applications, the same query is executed multiple times by different users and in different sessions, leading to a distribution of collected implicit feedback that is not uniform across the query set.

This paper aims to reproduce and then replicate the Team Draft Interleaving experiments from Hofmann et al. (2011) [11], investigating the effect that

J. Kamps et al. (Eds.): ECIR 2023, LNCS 13982, pp. 20–34, 2023.
https://doi.org/10.1007/978-3-031-28241-6_2

different query distributions have on the accuracy of this approach. The reason we chose this work [11] is that it is one of the most prominent surveys on interleaved methods and presents an experimental setup that felt perfect to evaluate the long-tailed real-world scenario. Specifically, three research questions arise:

– *RQ*1: Is it possible to reproduce the original paper experiments?
– *RQ*2: How does the original work generalise in the real-world scenario where queries have a long-tailed distribution?
– *RQ*3: Does applying statistical hypothesis testing improve the evaluation accuracy in such a scenario?

Thanks to the insights collected during the replicability work, we designed two novel methods that enrich the interleaving 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. We present a set of experiments to show interesting perspectives on the original work's reproducibility, confirm that the original work generalises quite accurately to the considered real-world scenario, and validate the intuition that our statistical analysis-based methods can improve the accuracy. The concepts 'ranking function', 'ranking model', and 'ranker' are used interchangeably.

The paper is organized as follows: Sect. 2 presents the related work. Section 3 details the experimental setup, datasets, and runs used for reproduction and replication. Section 4 introduces the theory behind the proposed improvements and describe our *stat-pruning* and *stat-weight* implementations. Section 5 discusses the experiments' runs and the obtained results. The paper's conclusions and future directions are listed in Sect. 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 [4]: an evaluation method based on explicit relevance judgments. 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 elaboration [2,3,21,24].

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 the commonly used AB testing, without losing accuracy in the obtained result [2,16,23]. Interleaving was introduced by Joachims [14,15] and from then, many improvements followed [2,20–22,24].

Team-Draft Interleaving 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 has also been shown to overcome issues of a previously implemented approach, Balanced interleaving, in determining the winning model [2]. Other types of interleaved methods are Document Constraint [9], Probabilistic Interleaving [11], 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 proved 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 is more complex but 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.

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 [10] and studies on the fidelity, soundness, and efficiency of interleaved methods [11].

In the studies mentioned so far, ad hoc uniform query distributions are generally used when evaluating ranking models. Balog et al. [1] address the issue of real-world long-tailed query distributions by reducing the evaluation to the most popular queries, the ones that effectively provide a good amount of data to work on, reserving an analysis of the totality of queries for the future.

3 Experiments

This paper aims to reproduce and then replicate under different scenarios experiment 1 from one of the most prominent surveys on interleaved methods [11].

In the first set of experiments, we address $RQ1$ with the same settings and data as the original work (uniform query distribution). Despite the original paper showing the probabilistic interleaving to be superior, as a baseline, we evaluate the Team Draft Interleaving (TDI) method only. The choice has been made because it is much simpler to implement and more popular in the industry. Furthermore, our focus is on the Δ_{AB} score estimator which is in common between the two approaches. We examine if the accuracy of the TDI method matches the published results and discuss the 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.

In the second set of experiments, we address $RQ2$ and $RQ3$ introducing a long-tailed query distribution. The aim is to examine how well the traditional TDI method generalises to real-world query distributions extracted from anonymized query logs 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 Sect. 3.1. The experimental setup is described in Sect. 3.2 and the experiment runs are explained in Sect. 3.3.

3.1 Datasets

All experiments make use of the MSLR-WEB30k Microsoft learning to rank ($MSLR$) dataset[1]. It consists of feature vectors extracted from query-document pairs along with relevance judgment labels. The relevance judgments take 5 values from 0 (irrelevant) to 4 (perfectly relevant). 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. The average number of judged documents differs from what has been written by Hofmann [11] and the authors confirmed it was a mistake in the original paper. For 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 use an industrial dataset we call *long-tail-1*. It consists of a list of anonymised query executions extracted from the query log of an e-commerce search engine, over a period of time. Each query is associated with the number of times it was executed. The amount of users collected per query is capped to $1,000$. This threshold has been chosen to maintain a realistic long-tailed distribution while keeping a sustainable experimental cost.

From this dataset, we derive the long tail in Fig. 1.

3.2 Experimental Setup

To reproduce the original experiments we designed and developed a system[2] that simulates a search engine with users submitting queries and interacting with the results (clicks). 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.
[2] https://github.com/SeaseLtd/statistical-interleaving.

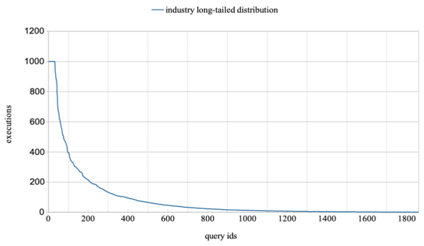

Fig. 1. Total unique queries: 1 861, total executions:156 550

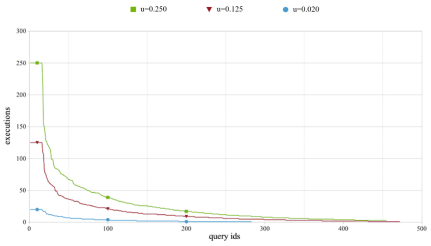

Fig. 2. Long-tailed query distributions used in the experiments

Each experiment run repeats a number of simulations. 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 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 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. The ground truth winner is calculated as the ranking function with the best Normalised Discounted Cumulative Gain (NDCG) [12,13] averaged over the query set, using the explicit relevance judgments provided with the dataset $MSLR$. The winning ranker identified by the Δ_{AB} score is compared to the ground truth winner: 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 submitted 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.

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 Fig. 2, Table 1). $u \leq 1$ has been introduced to experiment with different instances of realistic long-tailed distributions and to act within our computational limits.

Table 1. Scaling the query executions

u	Unique queries	Total executions
0.020	283	1 247
0.125	472	7 681
0.250	455	14 449

Click Models. The click model simulates user interactions according to the Dependent Click Model (DCM) [7,8], an extension of the cascade model [5]. It establishes the probability of a search result being clicked given its explicit relevance label (ground truth).

We use two models proposed by the original research [11]: the perfect and the realistic model.

NDCG. The NDCG metrics we use 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. 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.

3.3 Runs

We divided the runs into four groups: reproduction, uniform query distribution, long-tailed query distribution, and realistic click model. 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. We define a ranker for each of the 136 individual features provided with the $MSLR$ dataset. The results report the percentage of pairs for which the methods correctly identified the better ranker.

Reproduction. The scope of this set of runs is to reproduce *experiment 1* from the original research and answer $RQ1$. We exhaustively compare all $9,180$ distinct pairs derived from the 136 rankers. For each ranker pair, the user submits $1,000$ queries. The click model used is the *perfect model.*

– **Run 1**: exactly reproduce the original experiment, users click on the top-10 results for each query, to determine NDCG for the ground truth we use the complete NDCG.

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 NDCG.
– **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 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.

- **Run 4**: the query set consists of $1,000$ queries. Each query is executed once.
- **Run 5**: the query set consists of 100 queries. Each query is executed once.
- **Run 6**: the query set consists of 100 queries. Each query is executed 10 times.
- **Run 7**: the query set consists of 100 queries. Each query is executed 10 times. To determine NDCG for the ground truth we use the complete NDCG.

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 $RQ3$. Unless stated otherwise, we exhaustively compare all $9,180$ distinct pairs derived from the 136 rankers and calculate NDCG@10 to determine NDCG for the ground truth. The click model used is the *perfect model.* Each run uses a different long-tailed query distribution. Users click on the top-10 results for each query.

- **Run 8**: the query set consists of 283 unique queries repeated following the long-tailed distribution with $u = 0.020$.
- **Run 9**: the query set consists of 472 unique queries repeated following the long-tailed distribution with $u = 0.125$.
- **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 $u = 0.020$. To determine NDCG for the ground truth we use the complete NDCG.

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 $u = 0.020$. Users click on the top-10 results for each query.

- **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 NDCG.

4 Improving the Overall Winner Decision

In TDI, all the winners (i.e., all the queries) are considered equal when aggregating the results to establish the overall winning ranker (Eq. 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 assign 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.

4.1 Statistical Hypothesis Testing

In classic TDI, to assess the overall winner between $rankerA$ and $rankerB$, the Δ_{AB} score is computed as [2]:

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

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 have a tie

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. While performing our replicability research we observed two problems:

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

The overall winner decision may be polluted by the aforementioned queries. 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 happens after the computation of the clicks per ranker (h_a and h_b [2]) 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

$$p\text{-}value <= \alpha$$

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 [6].

To run 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 i.e. the probability of each ranking function 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 since each ranker collected 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:

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

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, a single-tailed test is used because the winner is known already.

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:

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

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 before the Δ_{AB} score calculation. The downside of this approach is that is strictly coupled to the significance level α hyper-parameter. Other works explore this aspect [19,26].

4.3 Stat-Weight

The credit associated with each win or tie in the original Δ_{AB} score formula (Eq. 1) 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.

$$credit(q_x) = 1 - p\text{-}value(q_x)$$

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

The *p-value* for a query q_x that presents a win is calculated with the Eq. 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:

$$wins(A) \Rightarrow \sum_{a=0}^{wins(A)} credit(q_a)$$

$$wins(B) \Rightarrow \sum_{b=0}^{wins(B)} credit(q_b)$$

$$ties(A, B) \Rightarrow \sum_{t=0}^{ties(A,B)} credit(q_t)$$

q_a is in the query set showing a preference for the *rankerA*.
q_b is in the query set showing a preference for the *rankerB*.
q_t belongs to the query set showing a tie.

5 Results and Analysis

5.1 Reproduction

Table 2. Query distribution: uniform 1,000 queries, click model: *perfect model*, 136 rankers (9,180 pairs)

id	NDCG	Clicks	Accuracy	Original-accuracy
1	complete	top-10	0.852	0.898
2	complete	complete	0.825	0.898
3	top-10	top-10	**0.902**	0.898

run-1 follows the same experimental setup, dataset, and parameters from the original research, but it fails to reproduce the originally recorded accuracy of TDI (see Table 2). We think that the difference can be caused by a dis alignment between the published paper and the NDCG and clicks generation parameters

Table 3. Complete NDCG and NDCG@10 averaged over the 1, 000 queries vs original-paper NDCG

ranker	NDCG	NDCG@10	Original-paper NDCG
1	0.550	0.195	0.231
14	0.536	0.179	0.201
64	0.600	0.294	0.301
84	0.574	0.239	0.256
97	0.564	0.234	0.303
106	0.606	0.295	0.253
134	0.614	0.333	0.341

used at run-time. For these reasons we executed two additional runs, trying to explain the possible causes of this failed reproduction. The closer we got to the originally recorded accuracy is with *run-3*, but not exactly the same (see Table 2).

The average ground truth NDCGs calculated for the rankers, do not align with the ones reported by the original paper (Table 3).

Also, the average NDCG@10 over the 1, 000 queries are closer but not exactly matching the original work ones (Table 3).

After long discussions with the original authors, we could ascertain that the NDCG formula used is the same as ours. However, we weren't able to check if the query set or NDCG cut-off correspond since it has not been possible to access 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 used a different cut-off. 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 4. Clicks: top-10, click model: *perfect model*, 136 rankers (9, 180 pairs)

				Accuracy		
id	*NDCG*	*Queries*	*Users*	*TDI*	*stat-pruning*	*stat-weight*
4	top-10	1 000	1	**0.902**	*N/A*	0.886
5	top-10	100	1	**0.812**	*N/A*	0.790
6	top-10	100	10	0.857	0.853	**0.883**
7	complete	100	10	0.828	0.839	**0.857**

From Table 4, *run-4* and *run-5* show a better accuracy for the original TDI method. 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* does not shine as well: it has too few clicks per query to work on. So *run-6* explores what happens if the distribution is uniform and each query is executed 10 times.

In this scenario, *stat-weight* does better than the baseline, with a 3% increase in accuracy (it correctly guessed 239 additional pairs). 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.

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

5.3 Long-Tailed Query Distribution

Table 5. Clicks: top-10, click model: *perfect model*

				Accuracy		
id	*NDCG*	*u*	*rankers*	*TDI*	*stat-pruning*	*stat-weight*
8	top-10	0.020	136	0.880	0.860	**0.897**
9	top-10	0.125	136	0.892	0.900	**0.904**
10	top-10	0.250	70	0.904	0.910	**0.911**
11	complete	0.020	136	0.827	0.817	**0.837**

From Table 5, *run-8*, *run-9* and *run-10* explore different long-tailed distributions. Due to computational limits we had to limit the amount of rankers, the closer we were getting to the original *long-tailed-1* distribution.

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.

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.

Table 6. Query distribution: long-tailed $u = 0.020$, clicks: top-10, click model: *realistic model*, 136 rankers (9, 180 pairs)

		Accuracy		
id	*NDCG*	*TDI*	*stat-pruning*	*stat-weight*
12	top-10	0.818	0.708	**0.833**
13	complete	0.782	0.693	**0.795**

5.4 Realistic Click Model

From Table 6, *run-12* introduces noisier and fewer clicks. *stat-weight* demonstrated to be robust to noise with a 1.8% increase (137 additional pairs) in comparison to the classic TDI. Finally, *run-13* tests the methods 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. Reproducing the original research turned out to be challenging from many angles: it was easy to align with the datasets but it required a substantial amount of work and discussions with the original authors to try to figure out the exact parameters and code used in the original runs. We had to design and develop from scratch the experiment code to cover all the necessary scenarios. Unfortunately, it was not possible to exactly reproduce the reported accuracy for TDI due to missing information and code unavailability.

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

$RQ3$ has been satisfied. Applying statistical hypothesis testing has shown to be promising: it adapts quite well to various real-world scenarios and doesn't add any big 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 normalizations 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.

References

1. Balog, K., Kelly, L., Schuth, A.: Head first: living labs for ad-hoc search evaluation. In: Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management, pp. 1815–1818 (2014)
2. Chapelle, O., Joachims, T., Radlinski, F., Yue, Y.: Large-scale validation and analysis of interleaved search evaluation. ACM Trans. Inf. Syst. (TOIS) **30**(1), 1–41 (2012)
3. 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)
4. Cleverdon, C.W., Mills, J., Keen, E.M.: Factors determining the performance of indexing systems, (Volume 1: Design), p. 28. College of Aeronautics, Cranfield (1966)
5. 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)
6. Dalgaard, P.: Power and the computation of sample size. In: Introductory Statistics with R, pp. 155–162. Springer, Cham (2008). https://doi.org/10.1007/0-387-22632-X_8
7. 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)
8. 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)
9. 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)
10. 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)
11. 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)
12. Järvelin, K., Kekäläinen, J.: Cumulated gain-based evaluation of IR techniques. ACM Trans. Inf. Syst. (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, Cambridge (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., 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, 9–13 August 2015, Ricardo. 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, London (2006)
26. Sproull, N.L.: Handbook of Research Methods: A Guide for Practitioners and Students in the Social Sciences. Scarecrow Press, Metuchen (2002)